

LATTE

v1.2.0

Generated by Doxygen 1.8.11

Contents

1	README	1
2	Todo List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	15
6.1	constants_mod Module Reference	15
6.1.1	Variable Documentation	17
6.1.1.1	allfiton	17
6.1.1.2	basistype	17
6.1.1.3	binfitstep	17
6.1.1.4	bint2fit	17
6.1.1.5	blksz	18
6.1.1.6	bndfil	18
6.1.1.7	box	18
6.1.1.8	box_old	18
6.1.1.9	boxdims	18
6.1.1.10	breaktol	18

6.1.1.11	cgorlib	18
6.1.1.12	charge	18
6.1.1.13	chempot	18
6.1.1.14	chtol	18
6.1.1.15	compforce	19
6.1.1.16	control	19
6.1.1.17	coordsfile	19
6.1.1.18	cove	19
6.1.1.19	debugon	19
6.1.1.20	dosfiton	19
6.1.1.21	ecoul	19
6.1.1.22	egap	19
6.1.1.23	ehomo	19
6.1.1.24	elec_etol	19
6.1.1.25	elec_qtol	20
6.1.1.26	elecmeth	20
6.1.1.27	electro	20
6.1.1.28	elumo	20
6.1.1.29	emeth	20
6.1.1.30	ente	20
6.1.1.31	entropykind	20
6.1.1.32	eps	20
6.1.1.33	erep	20
6.1.1.34	espin	20
6.1.1.35	espin_zero	21
6.1.1.36	existerror	21
6.1.1.37	f2v	21
6.1.1.38	fiton	21
6.1.1.39	freeze	21
6.1.1.40	fullqconv	21

6.1.1.41	hdim	21
6.1.1.42	int2fit	21
6.1.1.43	job	21
6.1.1.44	kbt	21
6.1.1.45	ke2t	22
6.1.1.46	kee	22
6.1.1.47	kon	22
6.1.1.48	latteinexists	22
6.1.1.49	lcniter	22
6.1.1.50	lcnon	22
6.1.1.51	libcalls	22
6.1.1.52	libinit	22
6.1.1.53	librun	22
6.1.1.54	massden	22
6.1.1.55	maxeval	23
6.1.1.56	maxminusmin	23
6.1.1.57	maxscf	23
6.1.1.58	mcbeta	23
6.1.1.59	mcsigma	23
6.1.1.60	mdadapt	23
6.1.1.61	mdmix	23
6.1.1.62	mdon	23
6.1.1.63	mineval	23
6.1.1.64	minsp2iter	23
6.1.1.65	mixer	24
6.1.1.66	mvv2ke	24
6.1.1.67	mvv2t	24
6.1.1.68	nats	24
6.1.1.69	nfitstep	24
6.1.1.70	ngpu	24

6.1.1.71	nmat	24
6.1.1.72	noelem	24
6.1.1.73	noint	24
6.1.1.74	norecs	24
6.1.1.75	numscf	25
6.1.1.76	occerrlimit	25
6.1.1.77	occsteps	25
6.1.1.78	ordernmol	25
6.1.1.79	parampath	25
6.1.1.80	parrep	25
6.1.1.81	pbcon	25
6.1.1.82	pi	25
6.1.1.83	pp2fit	25
6.1.1.84	ppbeta	25
6.1.1.85	ppfiton	26
6.1.1.86	ppnfitstep	26
6.1.1.87	ppngeom	26
6.1.1.88	ppnmol	26
6.1.1.89	ppoton	26
6.1.1.90	ppsigma	26
6.1.1.91	qfit	26
6.1.1.92	qiter	26
6.1.1.93	qmix	26
6.1.1.94	relaxme	26
6.1.1.95	restart	27
6.1.1.96	restartlib	27
6.1.1.97	rslevel	27
6.1.1.98	scfs	27
6.1.1.99	scfs_ii	27
6.1.1.100	scfstep	27

6.1.1.101 sp2conv	27
6.1.1.102 sparseon	27
6.1.1.103 spinmix	27
6.1.1.104 spinon	27
6.1.1.105 spintol	28
6.1.1.106 sponly	28
6.1.1.107 summass	28
6.1.1.108 togpa	28
6.1.1.109 tote	28
6.1.1.110 totne	28
6.1.1.111 tracelimit	28
6.1.1.112 trrhoh	28
6.1.1.113 tscale	28
6.1.1.114 vardt	28
6.1.1.115 vdwon	29
6.1.1.116 verbose	29
6.1.1.117 xbodison	29
6.1.1.118 xbodisorder	29
6.1.1.119 xboon	29
6.2 constraints_mod Module Reference	29
6.2.1 Detailed Description	29
6.2.2 Function/Subroutine Documentation	29
6.2.2.1 freeze_atoms(FTOT, VEL)	29
6.3 coulombarray Module Reference	31
6.3.1 Variable Documentation	31
6.3.1.1 calpha	31
6.3.1.2 calpha2	31
6.3.1.3 coslist	31
6.3.1.4 coulacc	32
6.3.1.5 coul b	32

6.3.1.6	coulcut	32
6.3.1.7	coulcut2	32
6.3.1.8	coulr1	32
6.3.1.9	coulvol	32
6.3.1.10	eightpi	32
6.3.1.11	fourcalpha2	32
6.3.1.12	k1_list	32
6.3.1.13	k2_list	32
6.3.1.14	k3_list	33
6.3.1.15	kcutoff	33
6.3.1.16	kcutoff2	33
6.3.1.17	keconst	33
6.3.1.18	ksq_list	33
6.3.1.19	latticevecs	33
6.3.1.20	lmax	33
6.3.1.21	mmax	33
6.3.1.22	nk	33
6.3.1.23	nmax	33
6.3.1.24	olddeltaqs	34
6.3.1.25	pi2	34
6.3.1.26	recipvecs	34
6.3.1.27	relperm	34
6.3.1.28	sinlist	34
6.3.1.29	sqrtpi	34
6.3.1.30	tfact	34
6.3.1.31	twopi	34
6.4	dbcsr_var_mod Module Reference	34
6.4.1	Function/Subroutine Documentation	35
6.4.1.1	myset_dist(dist_array, dist_size, nbins)	35
6.4.2	Variable Documentation	36

6.4.2.1	cbs	36
6.4.2.2	chksum	36
6.4.2.3	chksum2	37
6.4.2.4	col_blk_sizes	37
6.4.2.5	col_dist_a	37
6.4.2.6	diag	37
6.4.2.7	dist_a	37
6.4.2.8	dist_b	37
6.4.2.9	dist_c	37
6.4.2.10	error	37
6.4.2.11	found	37
6.4.2.12	grid_dist	37
6.4.2.13	group	38
6.4.2.14	ierr	38
6.4.2.15	iter	38
6.4.2.16	matrix_a	38
6.4.2.17	matrix_b	38
6.4.2.18	mp_comm	38
6.4.2.19	mp_env	38
6.4.2.20	mp_group	38
6.4.2.21	my_block	38
6.4.2.22	mynode	38
6.4.2.23	myploc	39
6.4.2.24	n	39
6.4.2.25	nblkcols_total	39
6.4.2.26	nblkrows_total	39
6.4.2.27	npdims	39
6.4.2.28	numnodes	39
6.4.2.29	pcol	39
6.4.2.30	pgrid	39

6.4.2.31	<code>proc_holds_blk</code>	39
6.4.2.32	<code>prow</code>	39
6.4.2.33	<code>rbs</code>	40
6.4.2.34	<code>row_blk_sizes</code>	40
6.4.2.35	<code>row_dist_a</code>	40
6.4.2.36	<code>temp</code>	40
6.4.2.37	<code>tr</code>	40
6.5	<code>diagarray</code> Module Reference	40
6.5.1	Variable Documentation	41
6.5.1.1	<code>cplist</code>	41
6.5.1.2	<code>diag_iwork</code>	41
6.5.1.3	<code>diag_liwork</code>	41
6.5.1.4	<code>diag_lrwork</code>	41
6.5.1.5	<code>diag_lwork</code>	41
6.5.1.6	<code>diag_lzwork</code>	41
6.5.1.7	<code>diag_rwork</code>	41
6.5.1.8	<code>diag_work</code>	41
6.5.1.9	<code>diag_zwork</code>	41
6.5.1.10	<code>downevals</code>	41
6.5.1.11	<code>downvecs</code>	42
6.5.1.12	<code>evals</code>	42
6.5.1.13	<code>evecs</code>	42
6.5.1.14	<code>exptol</code>	42
6.5.1.15	<code>ifail</code>	42
6.5.1.16	<code>kevals</code>	42
6.5.1.17	<code>kevecs</code>	42
6.5.1.18	<code>khtmlp</code>	42
6.5.1.19	<code>numlimit</code>	42
6.5.1.20	<code>upevals</code>	42
6.5.1.21	<code>upevecs</code>	43

6.5.1.22	zbo	43
6.5.1.23	zheevd_iwork	43
6.5.1.24	zheevd_liwork	43
6.5.1.25	zheevd_lrwork	43
6.5.1.26	zheevd_lwork	43
6.5.1.27	zheevd_rwork	43
6.5.1.28	zheevd_work	43
6.6	fermicommon Module Reference	43
6.6.1	Variable Documentation	44
6.6.1.1	a	44
6.6.1.2	cgtol	44
6.6.1.3	cgtol2	44
6.6.1.4	fermim	44
6.6.1.5	p0	44
6.6.1.6	r0	44
6.6.1.7	tmpmat	44
6.6.1.8	vala	44
6.6.1.9	valp0	44
6.6.1.10	valr0	44
6.6.1.11	valrho	45
6.6.1.12	valtmp	45
6.6.1.13	x2	45
6.7	genxprogress Module Reference	45
6.7.1	Detailed Description	45
6.7.2	Function/Subroutine Documentation	46
6.7.2.1	genxbml	46
6.7.3	Variable Documentation	46
6.7.3.1	igenx	46
6.7.3.2	over_bml	47
6.7.3.3	zk1_bml	47

6.7.3.4	zk2_bml	47
6.7.3.5	zk3_bml	47
6.7.3.6	zk4_bml	47
6.7.3.7	zk5_bml	47
6.7.3.8	zk6_bml	47
6.7.3.9	zmat_bml	47
6.7.3.10	zsp	47
6.8	get_end_scope Namespace Reference	47
6.8.1	Function Documentation	48
6.8.1.1	get_end(fin)	48
6.8.1.2	main()	48
6.9	homolumo Module Reference	48
6.9.1	Function/Subroutine Documentation	49
6.9.1.1	homolumogap(ITERZ)	49
6.9.1.2	sp2sequence()	49
6.10	kernelparser_mod Module Reference	50
6.10.1	Detailed Description	51
6.10.2	Function/Subroutine Documentation	51
6.10.2.1	parsing_kernel(KEYVECTOR_CHAR, VALVECTOR_CHAR, KEYVECTOR_IN↔ T, VALVECTOR_INT, KEYVECTOR_RE, VALVECTOR_RE, KEYVECTOR_L↔ OG, VALVECTOR_LOG, FILENAME, STARTSTOP)	51
6.11	kspacarray Module Reference	52
6.11.1	Variable Documentation	53
6.11.1.1	hk	53
6.11.1.2	hk0	53
6.11.1.3	hkdiag	53
6.11.1.4	kbo	53
6.11.1.5	kf	53
6.11.1.6	korthoh	53
6.11.1.7	kshift	53
6.11.1.8	kxmat	53

6.11.1.9	nktot	54
6.11.1.10	nkx	54
6.11.1.11	nky	54
6.11.1.12	nkz	54
6.11.1.13	sk	54
6.11.1.14	virbondk	54
6.11.1.15	zhjj	54
6.12	latte_lib Module Reference	54
6.12.1	Function/Subroutine Documentation	55
6.12.1.1	latte(NTYPES, TYPES, CR_IN, MASSES_IN, XLO, XHI, XY, XZ, YZ, FTOT_↵ OUT, MAXITER_IN, VENERG, VEL_IN, DT_IN, VIRIAL_INOUT, NEWSYSTEM, EXISTERROR_INOUT)	55
6.12.2	Variable Documentation	55
6.12.2.1	latte_abiversion	55
6.13	latteparser_latte_mod Module Reference	55
6.13.1	Detailed Description	56
6.13.2	Function/Subroutine Documentation	56
6.13.2.1	parse_control(FILENAME)	56
6.13.2.2	parse_kmesh(FILENAME)	58
6.13.2.3	parse_md(FILENAME)	60
6.13.3	Variable Documentation	61
6.13.3.1	lt	61
6.14	matrixio Module Reference	62
6.14.1	Function/Subroutine Documentation	62
6.14.1.1	writedmatrix(HSIZE, DARRAY)	62
6.14.1.2	wrotehmatrix(HSIZE, MSIZE, HARRAY, NITER, PVEC)	63
6.14.1.3	writemtx(ITER, HSIZE, II, JJ, VAL)	63
6.15	mdarray Module Reference	64
6.15.1	Variable Documentation	65
6.15.1.1	aveper	65
6.15.1.2	avet	65

6.15.1.3	c0	65
6.15.1.4	consmot	65
6.15.1.5	contiter	65
6.15.1.6	cumdt	65
6.15.1.7	dgamma	65
6.15.1.8	dt	65
6.15.1.9	dtzero	65
6.15.1.10	dumpfreq	65
6.15.1.11	e0	66
6.15.1.12	ehist	66
6.15.1.13	entropyiter	66
6.15.1.14	franprev	66
6.15.1.15	friction	66
6.15.1.16	gamma	66
6.15.1.17	gethug	66
6.15.1.18	hg	66
6.15.1.19	iset	66
6.15.1.20	mass	66
6.15.1.21	maxiter	67
6.15.1.22	npton	67
6.15.1.23	npttype	67
6.15.1.24	nvton	67
6.15.1.25	p0	67
6.15.1.26	phist	67
6.15.1.27	phistx	67
6.15.1.28	phisty	67
6.15.1.29	phistz	67
6.15.1.30	ptarget	67
6.15.1.31	rndist	68
6.15.1.32	rsfreq	68

6.15.1.33 seedinit	68
6.15.1.34 seedth	68
6.15.1.35 setth	68
6.15.1.36 shockdir	68
6.15.1.37 shockon	68
6.15.1.38 shockstart	68
6.15.1.39 shockstop	68
6.15.1.40 temperature	68
6.15.1.41 thermper	69
6.15.1.42 thermrun	69
6.15.1.43 thist	69
6.15.1.44 toinittemp	69
6.15.1.45 ttarget	69
6.15.1.46 tzero	69
6.15.1.47 uparticle	69
6.15.1.48 ushock	69
6.15.1.49 v	69
6.15.1.50 v0	69
6.15.1.51 vhist	70
6.15.1.52 wrtfreq	70
6.16 mixer_mod Module Reference	70
6.16.1 Detailed Description	70
6.16.2 Function/Subroutine Documentation	70
6.16.2.1 qmixprg(PITER)	70
6.16.3 Variable Documentation	72
6.16.3.1 dqin	72
6.16.3.2 dqout	73
6.16.3.3 mixinit	73
6.16.3.4 mx	73
6.16.3.5 scferror	73

6.17 myprecision Module Reference	73
6.17.1 Variable Documentation	74
6.17.1.1 eight	74
6.17.1.2 eleven	74
6.17.1.3 fifteen	74
6.17.1.4 five	74
6.17.1.5 fortyeight	74
6.17.1.6 four	74
6.17.1.7 fourteen	74
6.17.1.8 half	74
6.17.1.9 im	74
6.17.1.10 latteprec	74
6.17.1.11 minusone	75
6.17.1.12 nine	75
6.17.1.13 one	75
6.17.1.14 quarter	75
6.17.1.15 re	75
6.17.1.16 seven	75
6.17.1.17 six	75
6.17.1.18 sixteen	75
6.17.1.19 sqrt2	75
6.17.1.20 ten	75
6.17.1.21 third	76
6.17.1.22 thousand	76
6.17.1.23 three	76
6.17.1.24 threequart	76
6.17.1.25 twelve	76
6.17.1.26 twenty	76
6.17.1.27 twentyfour	76
6.17.1.28 twentysix	76

6.17.1.29 two	76
6.17.1.30 zero	76
6.18 neblistarray Module Reference	77
6.18.1 Variable Documentation	77
6.18.1.1 allocest	77
6.18.1.2 dimlist	77
6.18.1.3 maxcut	77
6.18.1.4 maxdimcoul	77
6.18.1.5 maxdimpp	77
6.18.1.6 maxdimtb	78
6.18.1.7 molid	78
6.18.1.8 nebcoul	78
6.18.1.9 nebpp	78
6.18.1.10 nebtb	78
6.18.1.11 nomol	78
6.18.1.12 pppmax2	78
6.18.1.13 rcutcoul2	78
6.18.1.14 rcuttb2	78
6.18.1.15 skin	78
6.18.1.16 totnebcoul	79
6.18.1.17 totnebpp	79
6.18.1.18 totnebtb	79
6.18.1.19 udneigh	79
6.19 nonoarray Module Reference	79
6.19.1 Variable Documentation	79
6.19.1.1 hjj	79
6.19.1.2 nono_evals	80
6.19.1.3 nono_iwork	80
6.19.1.4 nono_liwork	80
6.19.1.5 nono_lwork	80

6.19.1.6	nono_work	80
6.19.1.7	nonotmp	80
6.19.1.8	nonzero	80
6.19.1.9	orthoh	80
6.19.1.10	orthohdown	80
6.19.1.11	orthohup	80
6.19.1.12	sh2	81
6.19.1.13	smat	81
6.19.1.14	spintmp	81
6.19.1.15	umat	81
6.19.1.16	x2hrho	81
6.19.1.17	xmat	81
6.20	openfiles_mod Module Reference	81
6.20.1	Detailed Description	81
6.20.2	Function/Subroutine Documentation	81
6.20.2.1	get_file_unit(IO_MAX)	81
6.20.2.2	open_file(IO, NAME)	82
6.20.2.3	open_file_to_read(IO, NAME)	83
6.21	ppotarray Module Reference	84
6.21.1	Variable Documentation	85
6.21.1.1	nopps	85
6.21.1.2	potcoef	85
6.21.1.3	ppak	85
6.21.1.4	ppele1	85
6.21.1.5	ppele2	85
6.21.1.6	ppnk	85
6.21.1.7	ppr	85
6.21.1.8	pprk	85
6.21.1.9	ppspl	85
6.21.1.10	pptablength	85

6.21.1.11 ppval	86
6.22 purearray Module Reference	86
6.22.1 Variable Documentation	86
6.22.1.1 beta0	86
6.22.1.2 maxdim	86
6.22.1.3 nr_sp2_iter	86
6.22.1.4 pp	86
6.22.1.5 signlist	86
6.22.1.6 twox2	86
6.22.1.7 vv	87
6.22.1.8 x2	87
6.22.1.9 x2down	87
6.22.1.10 x2up	87
6.23 relaxcommon Module Reference	87
6.23.1 Variable Documentation	87
6.23.1.1 d1	87
6.23.1.2 mxrlx	87
6.23.1.3 oldd	87
6.23.1.4 oldf	88
6.23.1.5 prevf	88
6.23.1.6 relconst	88
6.23.1.7 reltype	88
6.23.1.8 rxftol	88
6.24 restartarray Module Reference	88
6.24.1 Variable Documentation	88
6.24.1.1 tmpbodiag	88
6.24.1.2 tmphdim	88
6.24.1.3 tmprhdown	88
6.24.1.4 tmprhoup	89
6.25 setuparray Module Reference	89

6.25.1	Variable Documentation	89
6.25.1.1	atele	89
6.25.1.2	atocc	90
6.25.1.3	basis	90
6.25.1.4	bo	90
6.25.1.5	bozero	90
6.25.1.6	btype	90
6.25.1.7	coulombv	90
6.25.1.8	cr	90
6.25.1.9	deltaq	90
6.25.1.10	ele	90
6.25.1.11	ele1	90
6.25.1.12	ele2	91
6.25.1.13	elempointer	91
6.25.1.14	f	91
6.25.1.15	fcoul	91
6.25.1.16	fpp	91
6.25.1.17	fpul	91
6.25.1.18	fscoul	91
6.25.1.19	fsspin	91
6.25.1.20	ftot	91
6.25.1.21	h	91
6.25.1.22	h0	92
6.25.1.23	hdiag	92
6.25.1.24	hed	92
6.25.1.25	hef	92
6.25.1.26	hep	92
6.25.1.27	hes	92
6.25.1.28	hr0	92
6.25.1.29	hubbardu	92

6.25.1.30	lcnshtft	92
6.25.1.31	matindlist	92
6.25.1.32	mycharge	93
6.25.1.33	orthorho	93
6.25.1.34	qlist	93
6.25.1.35	respchi	93
6.25.1.36	spinindlist	93
6.26	sp2progress Module Reference	93
6.26.1	Detailed Description	93
6.26.2	Function/Subroutine Documentation	94
6.26.2.1	sp2prg()	94
6.26.3	Variable Documentation	95
6.26.3.1	sp2d	95
6.26.3.2	sp2init	96
6.27	sparsearray Module Reference	96
6.27.1	Variable Documentation	96
6.27.1.1	bo_padded	96
6.27.1.2	fillinstop	96
6.27.1.3	hthresh	96
6.27.1.4	msparse	96
6.27.1.5	nnz	96
6.27.1.6	nnz_max	97
6.27.1.7	nnz_pad	97
6.27.1.8	numthresh	97
6.27.1.9	rx	97
6.27.1.10	rxtmp	97
6.27.1.11	thresholdon	97
6.27.1.12	work	97
6.27.1.13	xb	97
6.28	sparsemath Module Reference	97

6.28.1	Function/Subroutine Documentation	98
6.28.1.1	<code>sparseadd(TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)</code>	98
6.28.1.2	<code>sparsesetx2(TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)</code>	99
6.28.1.3	<code>sparsex2(TTRX, TTRX2, II, JJ, VAL, II2, JJ2, VAL2)</code>	99
6.28.2	Variable Documentation	101
6.28.2.1	<code>ix</code>	101
6.28.2.2	<code>jjb</code>	101
6.28.2.3	<code>x</code>	101
6.28.2.4	<code>y</code>	102
6.29	<code>sparsesp2</code> Module Reference	102
6.29.1	Function/Subroutine Documentation	102
6.29.1.1	<code>dense2sparse(HARRAY, HSIZE, II, JJ, VAL)</code>	102
6.29.1.2	<code>sp2loop(MSIZE, ITER, II, JJ, VAL)</code>	103
6.29.1.3	<code>sp2sequenceloop(MSIZE, ITER, II, JJ, VAL)</code>	105
6.29.1.4	<code>sparse2dense(SCALAR, II, JJ, VAL, DARRAY, HSIZE)</code>	107
6.30	<code>spinarray</code> Module Reference	108
6.30.1	Variable Documentation	109
6.30.1.1	<code>deltadim</code>	109
6.30.1.2	<code>deltaspin</code>	109
6.30.1.3	<code>h2vect</code>	109
6.30.1.4	<code>hdown</code>	109
6.30.1.5	<code>hup</code>	109
6.30.1.6	<code>olddeltaspin</code>	109
6.30.1.7	<code>rhodown</code>	109
6.30.1.8	<code>rhodownzero</code>	109
6.30.1.9	<code>rhoup</code>	110
6.30.1.10	<code>rhoupzero</code>	110
6.30.1.11	<code>spinlist</code>	110
6.30.1.12	<code>wdd</code>	110
6.30.1.13	<code>wff</code>	110

6.30.1.14 wpp	110
6.30.1.15 wss	110
6.31 subgraph Module Reference	110
6.31.1 Function/Subroutine Documentation	111
6.31.1.1 dense2sparsegraph(IIG, JJG, GGN)	111
6.31.1.2 partitiongraph	112
6.31.1.3 thresholdgraph	113
6.31.1.4 tracegraph(HDIM)	114
6.31.1.5 trnormgraph(ITERZ)	115
6.31.2 Variable Documentation	116
6.31.2.1 first_step	116
6.31.2.2 g	116
6.31.2.3 geps	117
6.31.2.4 node_in_part	117
6.31.2.5 nr_nodes	117
6.31.2.6 nr_of_nodes_in_part	117
6.31.2.7 nr_part	117
6.31.2.8 vvx	117
6.32 subgraphsp2 Module Reference	117
6.32.1 Function/Subroutine Documentation	118
6.32.1.1 extractsubgraph(I, IIH, JJH, HHN, IIG, JJG, IX, JJN, JJP, LG, L, LL)	118
6.32.1.2 normalizesubgraph(XS, JJN, L)	118
6.32.1.3 progressloop(IIH, JJH, HHN, IIG, JJG)	119
6.32.1.4 subgraphsp2loop(I, XS, JJP, L, LL)	121
6.32.1.5 subgraphsparse2dense(SCALAR, L, LL, JJN, JJP, DARRAY, XS)	122
6.32.2 Variable Documentation	123
6.32.2.1 ix	123
6.32.2.2 jjn	123
6.32.2.3 jjp	123
6.32.2.4 lg	124

6.32.2.5	xs	124
6.33	timer_mod Module Reference	124
6.33.1	Function/Subroutine Documentation	125
6.33.1.1	init_timer()	125
6.33.1.2	shutdown_timer()	126
6.33.1.3	start_timer(ITIMER)	126
6.33.1.4	stop_timer(ITIMER)	127
6.33.1.5	time_mls()	128
6.33.1.6	timedate_tag(TAG)	129
6.33.1.7	timer_results()	129
6.33.2	Variable Documentation	130
6.33.2.1	dense2sparse_timer	130
6.33.2.2	dmbuild_timer	131
6.33.2.3	latte_timer	131
6.33.2.4	num_timers	131
6.33.2.5	sp2all_timer	131
6.33.2.6	sp2sparse_timer	131
6.33.2.7	sparse2dense_timer	131
6.33.2.8	tavg	131
6.33.2.9	tclock_max	131
6.33.2.10	tclock_rate	131
6.33.2.11	tcount	131
6.33.2.12	tname	132
6.33.2.13	tpercent	132
6.33.2.14	tstart	132
6.33.2.15	tstart_clock	132
6.33.2.16	tstop_clock	132
6.33.2.17	tsum	132
6.33.2.18	total	132
6.33.2.19	tx	132

6.34 univarray Module Reference	132
6.34.1 Variable Documentation	133
6.34.1.1 bond	133
6.34.1.2 overl	133
6.34.1.3 pair	133
6.35 virialarray Module Reference	133
6.35.1 Variable Documentation	133
6.35.1.1 keten	133
6.35.1.2 pressure	133
6.35.1.3 strten	133
6.35.1.4 sysvol	134
6.35.1.5 virbond	134
6.35.1.6 vircoul	134
6.35.1.7 virial	134
6.35.1.8 virpair	134
6.35.1.9 virpul	134
6.35.1.10 virscoul	134
6.35.1.11 virsspin	134
6.36 xboarray Module Reference	134
6.36.1 Variable Documentation	135
6.36.1.1 alpha_xbo	135
6.36.1.2 chempot_pnk	135
6.36.1.3 cnk	135
6.36.1.4 kappa_scale	135
6.36.1.5 kappa_xbo	135
6.36.1.6 lastguess_chempot	135
6.36.1.7 pnk	135
6.36.1.8 prevlastguess_chempot	135
6.36.1.9 spin_pnk	135

7	Class Documentation	137
7.1	constraints_mod::constrains_type Type Reference	137
7.1.1	Detailed Description	137
7.1.2	Member Data Documentation	137
7.1.2.1	method	137
7.1.2.2	verbose	138
7.2	latteparser_latte_mod::latte_type Type Reference	138
7.2.1	Detailed Description	139
7.2.2	Member Data Documentation	139
7.2.2.1	bml_dmode	139
7.2.2.2	bml_type	139
7.2.2.3	coordsfile	139
7.2.2.4	coul_acc	139
7.2.2.5	efermi	140
7.2.2.6	jobname	140
7.2.2.7	maxscf	140
7.2.2.8	mdim	140
7.2.2.9	mdsteps	140
7.2.2.10	method	140
7.2.2.11	mixcoeff	140
7.2.2.12	mpulay	140
7.2.2.13	nlisteach	141
7.2.2.14	parampath	141
7.2.2.15	pulaycoeff	141
7.2.2.16	restart	141
7.2.2.17	scftol	141
7.2.2.18	threshold	141
7.2.2.19	timeratio	141
7.2.2.20	timestep	141
7.2.2.21	verbose	142
7.2.2.22	zmat	142

8 File Documentation	143
8.1 /home/christian/LATTE/README.md File Reference	143
8.2 /home/christian/LATTE/README.md	143
8.3 addqdep.f90 File Reference	144
8.3.1 Function/Subroutine Documentation	144
8.3.1.1 addqdep	144
8.4 addqdep.f90	145
8.5 algo.f90 File Reference	151
8.6 algo.f90	151
8.7 allfit.f90 File Reference	151
8.7.1 Function/Subroutine Documentation	151
8.7.1.1 allfit	151
8.8 allfit.f90	153
8.9 allocatcoulomb.f90 File Reference	161
8.9.1 Function/Subroutine Documentation	161
8.9.1.1 allocatcoulomb	161
8.10 allocatcoulomb.f90	162
8.11 allocateddiag.f90 File Reference	163
8.11.1 Function/Subroutine Documentation	163
8.11.1.1 allocateddiag	163
8.12 allocateddiag.f90	164
8.13 allocatenebarrays.f90 File Reference	166
8.13.1 Function/Subroutine Documentation	166
8.13.1.1 allocatenebarrays	166
8.14 allocatenebarrays.f90	167
8.15 allocatenono.f90 File Reference	168
8.15.1 Function/Subroutine Documentation	168
8.15.1.1 allocatenono	168
8.16 allocatenono.f90	169
8.17 allocatepure.f90 File Reference	170

8.17.1 Function/Subroutine Documentation	171
8.17.1.1 allocatpure	171
8.18 allocatpure.f90	172
8.19 allocatesubgraph.f90 File Reference	172
8.19.1 Function/Subroutine Documentation	173
8.19.1.1 allocatesubgraph	173
8.20 allocatesubgraph.f90	173
8.21 allocatexbo.f90 File Reference	173
8.21.1 Function/Subroutine Documentation	173
8.21.1.1 allocatexbo	173
8.22 allocatexbo.f90	174
8.23 am.f90 File Reference	177
8.23.1 Function/Subroutine Documentation	177
8.23.1.1 am(M, ALPHA)	177
8.24 am.f90	177
8.25 assessocc.f90 File Reference	177
8.25.1 Function/Subroutine Documentation	178
8.25.1.1 assessocc	178
8.26 assessocc.f90	178
8.27 atomcharge.f90 File Reference	179
8.27.1 Function/Subroutine Documentation	179
8.27.1.1 atomcharge(SWITCH)	179
8.28 atomcharge.f90	180
8.29 avepress.f90 File Reference	184
8.29.1 Function/Subroutine Documentation	184
8.29.1.1 avepress	184
8.30 avepress.f90	186
8.31 avesforhug.f90 File Reference	187
8.31.1 Function/Subroutine Documentation	187
8.31.1.1 avesforhug(MYPRESSURE, MYENERGY, MYTEMPERATURE, MYVOL)	187

8.32	avesforhug.f90	188
8.33	avetemp.f90 File Reference	189
8.33.1	Function/Subroutine Documentation	189
8.33.1.1	avetemp	189
8.34	avetemp.f90	190
8.35	bldnewH.f90 File Reference	191
8.35.1	Function/Subroutine Documentation	191
8.35.1.1	bldnewhs	191
8.36	bldnewH.f90	193
8.37	bldnewHS_sp.f90 File Reference	201
8.37.1	Function/Subroutine Documentation	201
8.37.1.1	bldnewhs_sp	201
8.38	bldnewHS_sp.f90	203
8.39	bldspinH.f90 File Reference	211
8.39.1	Function/Subroutine Documentation	211
8.39.1.1	bldspinH	211
8.40	bldspinH.f90	212
8.41	bm.f90 File Reference	217
8.41.1	Function/Subroutine Documentation	217
8.41.1.1	bm(M, ALPHA)	217
8.42	bm.f90	217
8.43	bodirect.f90 File Reference	217
8.43.1	Function/Subroutine Documentation	218
8.43.1.1	bovecs	218
8.44	bodirect.f90	219
8.45	bodirectprogress.f90 File Reference	222
8.45.1	Function/Subroutine Documentation	222
8.45.1.1	bovecsprg	222
8.46	bodirectprogress.f90	223
8.47	caselist.f90 File Reference	225

8.48	caselist.f90	225
8.49	conjgradient.f90 File Reference	225
8.49.1	Function/Subroutine Documentation	226
8.49.1.1	conjgradient(ITER, DELTAENERGY)	226
8.50	conjgradient.f90	227
8.51	constants_mod.f90 File Reference	228
8.52	constants_mod.f90	230
8.53	constraints_mod.f90 File Reference	231
8.54	constraints_mod.f90	232
8.55	coulomb_ewald.f90 File Reference	233
8.55.1	Function/Subroutine Documentation	233
8.55.1.1	coulombewald	233
8.56	coulomb_ewald.f90	234
8.57	coulomb_oldskool.f90 File Reference	236
8.57.1	Function/Subroutine Documentation	236
8.57.1.1	gascoulomb	236
8.58	coulomb_oldskool.f90	237
8.59	coulomb_rspace.f90 File Reference	240
8.59.1	Function/Subroutine Documentation	240
8.59.1.1	coulombrspace	240
8.60	coulomb_rspace.f90	241
8.61	coulombarray.f90 File Reference	244
8.62	coulombarray.f90	245
8.63	coultailcoef.f90 File Reference	246
8.63.1	Function/Subroutine Documentation	246
8.63.1.1	coultailcoef	246
8.64	coultailcoef.f90	247
8.65	dbcsr_var_mod.f90 File Reference	248
8.66	dbcsr_var_mod.f90	249
8.67	deallocateall.f90 File Reference	251

8.67.1 Function/Subroutine Documentation	251
8.67.1.1 deallocateall	251
8.68 deallocateall.f90	252
8.69 deallocatecoulomb.f90 File Reference	255
8.69.1 Function/Subroutine Documentation	255
8.69.1.1 deallocatecoulomb	255
8.70 deallocatecoulomb.f90	256
8.71 deallocatediag.f90 File Reference	257
8.71.1 Function/Subroutine Documentation	257
8.71.1.1 deallocatediag	257
8.72 deallocatediag.f90	258
8.73 deallocatenebarrays.f90 File Reference	259
8.73.1 Function/Subroutine Documentation	259
8.73.1.1 deallocatenebarrays	259
8.74 deallocatenebarrays.f90	260
8.75 deallocatenono.f90 File Reference	261
8.75.1 Function/Subroutine Documentation	261
8.75.1.1 deallocatenono	261
8.76 deallocatenono.f90	262
8.77 deallocatepure.f90 File Reference	263
8.77.1 Function/Subroutine Documentation	263
8.77.1.1 deallocatepure	263
8.78 deallocatepure.f90	264
8.79 deallocatesubgraph.f90 File Reference	265
8.79.1 Function/Subroutine Documentation	265
8.79.1.1 deallocatesubgraph	265
8.80 deallocatesubgraph.f90	266
8.81 deallocatexbo.f90 File Reference	266
8.81.1 Function/Subroutine Documentation	266
8.81.1.1 deallocatexbo	266

8.82 deallocatexbo.f90	267
8.83 deorthomyrho.f90 File Reference	268
8.83.1 Function/Subroutine Documentation	268
8.83.1.1 deorthomyrho	268
8.84 deorthomyrho.f90	269
8.85 deorthomyrho.h90 File Reference	271
8.85.1 Function/Subroutine Documentation	271
8.85.1.1 deorthomyrho.h90	271
8.86 deorthomyrho.h90	272
8.87 dfda.f90 File Reference	274
8.87.1 Function/Subroutine Documentation	274
8.87.1.1 dfda(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)	274
8.88 dfda.f90	274
8.89 dfdb.f90 File Reference	275
8.89.1 Function/Subroutine Documentation	275
8.89.1.1 dfdb(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)	275
8.90 dfdb.f90	275
8.91 dfdr.f90 File Reference	276
8.91.1 Function/Subroutine Documentation	276
8.91.1.1 dfdr(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)	276
8.92 dfdr.f90	276
8.93 dgspdr.f90 File Reference	277
8.93.1 Function/Subroutine Documentation	277
8.93.1.1 dgspdr(I, J, L1, L2, M, MAGR)	277
8.94 dgspdr.f90	277
8.95 diagarray.f90 File Reference	279
8.96 diagarray.f90	280
8.97 diagmyh.f90 File Reference	280
8.97.1 Function/Subroutine Documentation	280
8.97.1.1 diagmyh()	280

8.98 diagmyh.f90	281
8.99 dosfit.f90 File Reference	283
8.99.1 Function/Subroutine Documentation	283
8.99.1.1 dosfit	283
8.100dosfit.f90	285
8.101dslmmpda.f90 File Reference	291
8.101.1 Function/Subroutine Documentation	291
8.101.1.1 dslmmpda(L, M, MP, ALPHA, COSBETA)	291
8.102dslmmpda.f90	291
8.103dslmmpdb.f90 File Reference	292
8.103.1 Function/Subroutine Documentation	292
8.103.1.1 dslmmpdb(L, M, MP, ALPHA, COSBETA)	292
8.104dslmmpdb.f90	292
8.105dtlmpda.f90 File Reference	293
8.105.1 Function/Subroutine Documentation	293
8.105.1.1 dtlmpda(L, M, MP, ALPHA, COSBETA)	293
8.106dtlmpda.f90	293
8.107dtlmpdb.f90 File Reference	294
8.107.1 Function/Subroutine Documentation	294
8.107.1.1 dtlmpdb(L, M, MP, ALPHA, COSBETA)	294
8.108dtlmpdb.f90	294
8.109dunivscaling.f90 File Reference	295
8.109.1 Function/Subroutine Documentation	295
8.109.1.1 dunivscale_sub(R, A, DC, X, DXDR)	295
8.110dunivscaling.f90	296
8.111dunivscaling_function.f90 File Reference	297
8.111.1 Function/Subroutine Documentation	297
8.111.1.1 dunivscale(I, J, L1, L2, MP, R, WHICHINT)	297
8.112dunivscaling_function.f90	298
8.113dwignerddb.f90 File Reference	300

8.113.1 Function/Subroutine Documentation	300
8.113.1.1 dwignerddb(L, M, MP, COSBETA)	300
8.114dwignerddb.f90	301
8.115entropy.f90 File Reference	301
8.115.1 Function/Subroutine Documentation	301
8.115.1.1 entropy	301
8.116entropy.f90	303
8.117errors.f90 File Reference	311
8.117.1 Function/Subroutine Documentation	311
8.117.1.1 errors(SUB, TAG)	311
8.118errors.f90	313
8.119factorial.f90 File Reference	314
8.119.1 Function/Subroutine Documentation	314
8.119.1.1 factorial(l1)	314
8.120factorial.f90	314
8.121fcoulnono.f90 File Reference	315
8.121.1 Function/Subroutine Documentation	315
8.121.1.1 fcoulnono	315
8.122fcoulnono.f90	316
8.123fcoulnono_sp.f90 File Reference	321
8.123.1 Function/Subroutine Documentation	322
8.123.1.1 fcoulnono_sp	322
8.124fcoulnono_sp.f90	323
8.125fermiallocate.f90 File Reference	335
8.125.1 Function/Subroutine Documentation	335
8.125.1.1 fermiallocate	335
8.126fermiallocate.f90	336
8.127fermicommon.f90 File Reference	337
8.128fermicommon.f90	338
8.129fermideallocate.f90 File Reference	338

8.129.1 Function/Subroutine Documentation	338
8.129.1.1 fermideallocate	338
8.130fermideallocate.f90	339
8.131fermiexpans.f90 File Reference	340
8.131.1 Function/Subroutine Documentation	340
8.131.1.1 fermiexpans	340
8.132fermiexpans.f90	342
8.133fittingoutput.f90 File Reference	345
8.133.1 Function/Subroutine Documentation	345
8.133.1.1 fittingoutput(BECLEAN)	345
8.134fittingoutput.f90	346
8.135fspinnono.f90 File Reference	347
8.135.1 Function/Subroutine Documentation	348
8.135.1.1 fspinnono	348
8.136fspinnono.f90	349
8.137fspinnono_sp.f90 File Reference	354
8.137.1 Function/Subroutine Documentation	354
8.137.1.1 fspinnono_sp	354
8.138fspinnono_sp.f90	355
8.139gaussrn.f90 File Reference	363
8.139.1 Function/Subroutine Documentation	363
8.139.1.1 gaussrn(MEAN, STDDEV)	363
8.140gaussrn.f90	364
8.141gendiag.f90 File Reference	364
8.141.1 Function/Subroutine Documentation	364
8.141.1.1 gendiag	364
8.142gendiag.f90	365
8.143genX.f90 File Reference	366
8.143.1 Function/Subroutine Documentation	366
8.143.1.1 genx	366

8.144genX.f90	368
8.145genXprogress.f90 File Reference	370
8.146genXprogress.f90	371
8.147gershgorin.f90 File Reference	372
8.147.1 Function/Subroutine Documentation	372
8.147.1.1 gershgorin	372
8.148gershgorin.f90	373
8.149get_end_scope.py File Reference	375
8.150get_end_scope.py	376
8.151getbndfil.f90 File Reference	376
8.151.1 Function/Subroutine Documentation	376
8.151.1.1 getbndfil()	376
8.152getbndfil.f90	377
8.153getcoule.f90 File Reference	379
8.153.1 Function/Subroutine Documentation	379
8.153.1.1 getcoule	379
8.154getcoule.f90	380
8.155getdeltaq.f90 File Reference	381
8.155.1 Function/Subroutine Documentation	381
8.155.1.1 getdeltaq	381
8.156getdeltaq.f90	382
8.157getdeltaspin.f90 File Reference	385
8.157.1 Function/Subroutine Documentation	385
8.157.1.1 getdeltaspin	385
8.158getdeltaspin.f90	386
8.159getdensity.f90 File Reference	395
8.159.1 Function/Subroutine Documentation	395
8.159.1.1 getdensity	395
8.160getdensity.f90	396
8.161getdipole.f90 File Reference	397

8.161.1 Function/Subroutine Documentation	397
8.161.1.1 getdipole(DIPOLEMAG)	397
8.162getdipole.f90	398
8.163getforce.f90 File Reference	399
8.163.1 Function/Subroutine Documentation	399
8.163.1.1 getforce	399
8.164getforce.f90	401
8.165gethdim.f90 File Reference	403
8.165.1 Function/Subroutine Documentation	403
8.165.1.1 gethdim	403
8.166gethdim.f90	404
8.167getke.f90 File Reference	407
8.167.1 Function/Subroutine Documentation	407
8.167.1.1 getke	407
8.168getke.f90	408
8.169getmatindlist.f90 File Reference	409
8.169.1 Function/Subroutine Documentation	409
8.169.1.1 getmatindlist	409
8.170getmatindlist.f90	410
8.171getmaxf.f90 File Reference	412
8.171.1 Function/Subroutine Documentation	412
8.171.1.1 getmaxf(MAXF)	412
8.172getmaxf.f90	413
8.173getmdf.f90 File Reference	414
8.173.1 Function/Subroutine Documentation	414
8.173.1.1 getmdf(SWITCH, CURRITER)	414
8.174getmdf.f90	416
8.175getpressure.f90 File Reference	419
8.175.1 Function/Subroutine Documentation	419
8.175.1.1 getpressure	419

8.176getpressure.f90	420
8.177getrespf.f90 File Reference	422
8.177.1 Function/Subroutine Documentation	422
8.177.1.1 getrespf	422
8.178getrespf.f90	422
8.179getrho.f90 File Reference	424
8.179.1 Function/Subroutine Documentation	424
8.179.1.1 getrho(MDITER)	424
8.180getrho.f90	426
8.181getspinE.f90 File Reference	428
8.181.1 Function/Subroutine Documentation	428
8.181.1.1 getspine	428
8.182getspinE.f90	429
8.183getspinE_zero.f90 File Reference	432
8.183.1 Function/Subroutine Documentation	432
8.183.1.1 getspinezero	432
8.184getspinE_zero.f90	432
8.185getsplinenr.f90 File Reference	433
8.185.1 Function/Subroutine Documentation	433
8.185.1.1 spline	433
8.185.1.2 splint(X, Y, Y2, R, NEWY, GRAD)	434
8.186getsplinenr.f90	434
8.187gradH.f90 File Reference	435
8.187.1 Function/Subroutine Documentation	435
8.187.1.1 gradh	435
8.188gradH.f90	436
8.189gradH_sp.f90 File Reference	441
8.189.1 Function/Subroutine Documentation	441
8.189.1.1 gradhsp	441
8.190gradH_sp.f90	443

8.191homolumo.f90 File Reference	454
8.192homolumo.f90	455
8.193hugrescale.f90 File Reference	457
8.193.1 Function/Subroutine Documentation	457
8.193.1.1 hugrescale	457
8.194hugrescale.f90	458
8.195ifrestart.f90 File Reference	460
8.195.1 Function/Subroutine Documentation	460
8.195.1.1 ifrestart	460
8.196ifrestart.f90	462
8.197init_dbcsr.f90 File Reference	463
8.197.1 Function/Subroutine Documentation	463
8.197.1.1 init_dbcsr	464
8.198init_dbcsr.f90	465
8.199initcoulomb.f90 File Reference	467
8.199.1 Function/Subroutine Documentation	467
8.199.1.1 initcoulomb	467
8.200initcoulomb.f90	468
8.201initcoulombklist.f90 File Reference	470
8.201.1 Function/Subroutine Documentation	471
8.201.1.1 get_k_lists(RECIP_VECTORS)	471
8.202initcoulombklist.f90	472
8.203initialv.f90 File Reference	474
8.203.1 Function/Subroutine Documentation	474
8.203.1.1 initialv	474
8.204initialv.f90	476
8.205initrng.f90 File Reference	478
8.205.1 Function/Subroutine Documentation	478
8.205.1.1 initrng	478
8.206initrng.f90	479

8.207initshockcomp.f90 File Reference	480
8.207.1 Function/Subroutine Documentation	480
8.207.1.1 initshockcomp	480
8.208initshockcomp.f90	481
8.209kbldnewh.f90 File Reference	482
8.209.1 Function/Subroutine Documentation	482
8.209.1.1 kbldnewh	482
8.210kbldnewh.f90	484
8.211kbodirect.f90 File Reference	493
8.211.1 Function/Subroutine Documentation	493
8.211.1.1 kboevects	493
8.212kbodirect.f90	495
8.213kdeorthomyrho.f90 File Reference	500
8.213.1 Function/Subroutine Documentation	500
8.213.1.1 kdeorthomyrho	500
8.214kdeorthomyrho.f90	501
8.215kdiagmyh.f90 File Reference	502
8.215.1 Function/Subroutine Documentation	502
8.215.1.1 kdiagmyh	502
8.216kdiagmyh.f90	503
8.217kernelparser_mod.f90 File Reference	506
8.218kernelparser_mod.f90	506
8.219kfcoulnono.f90 File Reference	511
8.219.1 Function/Subroutine Documentation	511
8.219.1.1 kfcoulnono	511
8.220kfcoulnono.f90	513
8.221kgenX.f90 File Reference	519
8.221.1 Function/Subroutine Documentation	520
8.221.1.1 kgenx	520
8.222kgenX.f90	521

8.223kgetdos.f90 File Reference	522
8.223.1 Function/Subroutine Documentation	523
8.223.1.1 kgetdos	523
8.224kgetdos.f90	524
8.225kgetrho.f90 File Reference	525
8.225.1 Function/Subroutine Documentation	525
8.225.1.1 kgetrho	525
8.226kgetrho.f90	527
8.227kgradH.f90 File Reference	528
8.227.1 Function/Subroutine Documentation	528
8.227.1.1 kgradh	528
8.228kgradH.f90	530
8.229korthomyH.f90 File Reference	536
8.229.1 Function/Subroutine Documentation	537
8.229.1.1 korthomyh	537
8.230korthomyH.f90	538
8.231kpulay.f90 File Reference	538
8.231.1 Function/Subroutine Documentation	538
8.231.1.1 kpulay	538
8.232kpulay.f90	540
8.233kspacearray.f90 File Reference	547
8.234kspacearray.f90	547
8.235latte.f90 File Reference	548
8.235.1 Function/Subroutine Documentation	548
8.235.1.1 latte	548
8.236latte.f90	549
8.237latte_c_bind.f90 File Reference	555
8.237.1 Function/Subroutine Documentation	555
8.237.1.1 latte_c_abiversion()	555

8.237.1.2 latte_c_bind(FLAGS, NATS, COORDS, TYPES, NTYPES, MASSES, XLO, XHI, XY, XZ, YZ, FORCES, MAXITER, VENERG, VEL, DT, VIRIAL_INOUT, NEWS← YSTEM, EXISTERROR)	555
8.238latte_c_bind.f90	557
8.239latte_lib.f90 File Reference	558
8.240latte_lib.f90	559
8.241latteparser_latte_mod.f90 File Reference	569
8.242latteparser_latte_mod.f90	570
8.243masses2symbols.f90 File Reference	578
8.243.1 Function/Subroutine Documentation	578
8.243.1.1 masses2symbols(TYPES, NTYPES, MASSES_IN, NATSIN, SYMBOLS)	578
8.244masses2symbols.f90	580
8.245matrixio.f90 File Reference	581
8.246matrixio.f90	581
8.247mdarray.f90 File Reference	583
8.248mdarray.f90	584
8.249mixer_mod.f90 File Reference	585
8.250mixer_mod.f90	585
8.251mofit.f90 File Reference	586
8.251.1 Function/Subroutine Documentation	586
8.251.1.1 mofit	586
8.252mofit.f90	588
8.253mofit_plato.f90 File Reference	592
8.253.1 Function/Subroutine Documentation	592
8.253.1.1 mofitplato	592
8.254mofit_plato.f90	594
8.255msrelax.f90 File Reference	600
8.255.1 Function/Subroutine Documentation	600
8.255.1.1 msrelax	600
8.256msrelax.f90	602
8.257myprecision.f90 File Reference	607

8.258myprecision.f90	608
8.259neblast_cell.f90 File Reference	608
8.259.1 Function/Subroutine Documentation	608
8.259.1.1 neblasts(AMIALLO)	608
8.260neblast_cell.f90	610
8.261neblastarray.f90 File Reference	625
8.262neblastarray.f90	626
8.263neblasts.f90 File Reference	627
8.263.1 Function/Subroutine Documentation	627
8.263.1.1 neblasts(AMIALLO)	627
8.264neblasts.f90	628
8.265newkbldnewh.f90 File Reference	634
8.265.1 Function/Subroutine Documentation	634
8.265.1.1 kbldnewh	634
8.266newkbldnewh.f90	634
8.267nnz.f90 File Reference	643
8.267.1 Function/Subroutine Documentation	643
8.267.1.1 nnzend(M, HDIM)	643
8.267.1.2 nnzstart(MSPARSE, HDIM)	643
8.268nnz.f90	643
8.269noelec.f90 File Reference	644
8.269.1 Function/Subroutine Documentation	644
8.269.1.1 noelec(NUMEL)	644
8.270noelec.f90	644
8.271nonoarray.f90 File Reference	645
8.272nonoarray.f90	646
8.273norms.f90 File Reference	646
8.273.1 Function/Subroutine Documentation	646
8.273.1.1 norms	646
8.274norms.f90	647

8.275nptrescale.f90 File Reference	649
8.275.1 Function/Subroutine Documentation	649
8.275.1.1 nptrescale	649
8.276nptrescale.f90	650
8.277nvtandersen.f90 File Reference	652
8.277.1 Function/Subroutine Documentation	652
8.277.1.1 nvtandersen	652
8.278nvtandersen.f90	653
8.279nvtlangevin.f90 File Reference	654
8.279.1 Function/Subroutine Documentation	654
8.279.1.1 nvtlangevin(ITER)	654
8.280nvtlangevin.f90	656
8.281nvtNH.f90 File Reference	658
8.281.1 Function/Subroutine Documentation	658
8.281.1.1 nvtnh	658
8.282nvtNH.f90	660
8.283nvtrescale.f90 File Reference	662
8.283.1 Function/Subroutine Documentation	662
8.283.1.1 nvtrescale	662
8.284nvtrescale.f90	664
8.285openfiles_mod.f90 File Reference	666
8.286openfiles_mod.f90	666
8.287orthomyH.f90 File Reference	667
8.287.1 Function/Subroutine Documentation	667
8.287.1.1 orthomyh	667
8.288orthomyH.f90	668
8.289orthomyHprogress.f90 File Reference	670
8.289.1 Function/Subroutine Documentation	670
8.289.1.1 orthomyhprg	670
8.290orthomyHprogress.f90	671

8.291orthomyrho.f90 File Reference	673
8.291.1 Function/Subroutine Documentation	673
8.291.1.1 orthomyrho	673
8.292orthomyrho.f90	673
8.293pairpot.f90 File Reference	674
8.293.1 Function/Subroutine Documentation	675
8.293.1.1 pairpot	675
8.294pairpot.f90	675
8.295pairpot_noneb.f90 File Reference	678
8.295.1 Function/Subroutine Documentation	678
8.295.1.1 pairpotnoneb	678
8.296pairpot_noneb.f90	679
8.297pairpotspline.f90 File Reference	682
8.297.1 Function/Subroutine Documentation	682
8.297.1.1 pairpotspline	682
8.297.1.2 ppheavi(RK, R)	683
8.298pairpotspline.f90	683
8.299pairpottab.f90 File Reference	685
8.299.1 Function/Subroutine Documentation	685
8.299.1.1 pairpottab	685
8.300pairpottab.f90	686
8.301panic.f90 File Reference	688
8.301.1 Function/Subroutine Documentation	688
8.301.1.1 panic	688
8.302panic.f90	690
8.303parafileopen.f90 File Reference	693
8.303.1 Function/Subroutine Documentation	694
8.303.1.1 parafileopen	694
8.304parafileopen.f90	695
8.305parawrite.f90 File Reference	695

8.305.1 Function/Subroutine Documentation	695
8.305.1.1 parawrite(THETIME)	695
8.306parawrite.f90	696
8.307pbc.f90 File Reference	697
8.307.1 Function/Subroutine Documentation	697
8.307.1.1 pbc	697
8.308pbc.f90	698
8.309plot_ppot.f90 File Reference	699
8.309.1 Function/Subroutine Documentation	700
8.309.1.1 plotppot	700
8.310plot_ppot.f90	701
8.311plot_univ.f90 File Reference	702
8.311.1 Function/Subroutine Documentation	702
8.311.1.1 plotuniv	702
8.312plot_univ.f90	703
8.313ppfit.f90 File Reference	705
8.313.1 Function/Subroutine Documentation	705
8.313.1.1 ppfit	705
8.314ppfit.f90	707
8.315ppotarray.f90 File Reference	715
8.316ppotarray.f90	716
8.317printsparse.f90 File Reference	716
8.317.1 Function/Subroutine Documentation	716
8.317.1.1 printsparse	716
8.318printsparse.f90	716
8.319propchempot_xbo.f90 File Reference	717
8.319.1 Function/Subroutine Documentation	717
8.319.1.1 propchempot(ITER)	717
8.320propchempot_xbo.f90	718
8.321prospins_xbo.f90 File Reference	720

8.321.1 Function/Subroutine Documentation	720
8.321.1.1 propspins(ITER)	720
8.322propspins_xbo.f90	721
8.323pulay.f90 File Reference	723
8.323.1 Function/Subroutine Documentation	723
8.323.1.1 pulay	723
8.324pulay.f90	724
8.325pulay_sp.f90 File Reference	731
8.325.1 Function/Subroutine Documentation	731
8.325.1.1 pulay_sp	731
8.326pulay_sp.f90	733
8.327pulay_spprogress.f90 File Reference	741
8.327.1 Function/Subroutine Documentation	742
8.327.1.1 pulay_spprogress	742
8.328pulay_spprogress.f90	743
8.329purearray.f90 File Reference	751
8.330purearray.f90	752
8.331qconsistency.f90 File Reference	752
8.331.1 Function/Subroutine Documentation	753
8.331.1.1 qconsistency(SWITCH, MDITER)	753
8.332qconsistency.f90	754
8.333qneutral.f90 File Reference	760
8.333.1 Function/Subroutine Documentation	761
8.333.1.1 qneutral(SWITCH, MDITER)	761
8.334qneutral.f90	762
8.335readcontrols.f90 File Reference	766
8.335.1 Function/Subroutine Documentation	766
8.335.1.1 readcontrols	766
8.336readcontrols.f90	768
8.337readcr.f90 File Reference	772

8.337.1 Function/Subroutine Documentation	772
8.337.1.1 readcr	772
8.338readcr.f90	773
8.339readmdcontroller.f90 File Reference	775
8.339.1 Function/Subroutine Documentation	775
8.339.1.1 readmdcontroller	775
8.340readmdcontroller.f90	777
8.341readppot.f90 File Reference	779
8.341.1 Function/Subroutine Documentation	780
8.341.1.1 readppot()	780
8.342readppot.f90	781
8.343readppotspline.f90 File Reference	782
8.343.1 Function/Subroutine Documentation	783
8.343.1.1 readppotspline	783
8.344readppotspline.f90	784
8.345readppottab.f90 File Reference	785
8.345.1 Function/Subroutine Documentation	785
8.345.1.1 readppottab	785
8.346readppottab.f90	786
8.347readrestart.f90 File Reference	788
8.347.1 Function/Subroutine Documentation	788
8.347.1.1 readrestart	788
8.348readrestart.f90	789
8.349readrestartlib.f90 File Reference	792
8.349.1 Function/Subroutine Documentation	792
8.349.1.1 readrestartlib(ITER)	792
8.350readrestartlib.f90	792
8.351readtb.f90 File Reference	803
8.351.1 Function/Subroutine Documentation	803
8.351.1.1 readtb	803

8.352readtb.f90	804
8.353relaxcommon.f90 File Reference	806
8.354relaxcommon.f90	806
8.355resetprohd.f90 File Reference	807
8.355.1 Function/Subroutine Documentation	807
8.355.1.1 resetprohd	807
8.356resetprohd.f90	808
8.357restartarray.f90 File Reference	810
8.358restartarray.f90	810
8.359rhozero.f90 File Reference	811
8.359.1 Function/Subroutine Documentation	811
8.359.1.1 rhozero	811
8.360rhozero.f90	813
8.361setuparray.f90 File Reference	834
8.362setuparray.f90	835
8.363setuptbmd.f90 File Reference	836
8.363.1 Function/Subroutine Documentation	836
8.363.1.1 setuptbmd(NEWSYSTEM)	836
8.364setuptbmd.f90	837
8.365shiftH.f90 File Reference	838
8.365.1 Function/Subroutine Documentation	838
8.365.1.1 shifh(CHI)	838
8.366shiftH.f90	839
8.367shockcomp.f90 File Reference	843
8.367.1 Function/Subroutine Documentation	843
8.367.1.1 shockcomp	843
8.368shockcomp.f90	844
8.369shutdown_dbcsr.f90 File Reference	845
8.369.1 Function/Subroutine Documentation	845
8.369.1.1 shutdown_dbcsr	845

8.370shutdown_dbcsr.f90	846
8.371slmmp.f90 File Reference	847
8.371.1 Function/Subroutine Documentation	847
8.371.1.1 slmmp(L, M, MP, ALPHA, COSBETA)	847
8.372slmmp.f90	848
8.373solvmatcg.f90 File Reference	848
8.373.1 Function/Subroutine Documentation	848
8.373.1.1 solvmatcg	848
8.374solvmatcg.f90	850
8.375solvmatcg_sparse.f90 File Reference	855
8.375.1 Function/Subroutine Documentation	855
8.375.1.1 solvmatcg	855
8.376solvmatcg_sparse.f90	856
8.377solvmatlapack.f90 File Reference	861
8.377.1 Function/Subroutine Documentation	861
8.377.1.1 solvmatlapack	861
8.378solvmatlapack.f90	862
8.379sp2dbcsr.f90 File Reference	864
8.379.1 Function/Subroutine Documentation	864
8.379.1.1 sp2dbcsr	864
8.380sp2dbcsr.f90	865
8.381sp2fermi.f90 File Reference	867
8.381.1 Function/Subroutine Documentation	867
8.381.1.1 sp2fermi	867
8.382sp2fermi.f90	869
8.383sp2fermi_init.f90 File Reference	874
8.383.1 Function/Subroutine Documentation	874
8.383.1.1 sp2fermiinit	874
8.384sp2fermi_init.f90	876
8.385sp2gap.f90 File Reference	883

8.385.1 Function/Subroutine Documentation	883
8.385.1.1 sp2gap	883
8.386sp2gap.f90	884
8.387sp2gap_setup.f90 File Reference	885
8.387.1 Function/Subroutine Documentation	885
8.387.1.1 sp2gap_setup	885
8.388sp2gap_setup.f90	886
8.389sp2progress.f90 File Reference	889
8.390sp2progress.f90	889
8.391sp2pure.f90 File Reference	890
8.391.1 Function/Subroutine Documentation	890
8.391.1.1 sp2pure	890
8.392sp2pure.f90	892
8.393sp2pure_sparse.f90 File Reference	896
8.393.1 Function/Subroutine Documentation	896
8.393.1.1 sp2pure_sparse	896
8.394sp2pure_sparse.f90	898
8.395sp2pure_sparse_parallel.f90 File Reference	906
8.395.1 Function/Subroutine Documentation	906
8.395.1.1 sp2pure_sparse_parallel(MDITER)	906
8.396sp2pure_sparse_parallel.f90	908
8.397sp2pure_sparse_parallel_simple.f90 File Reference	910
8.397.1 Function/Subroutine Documentation	910
8.397.1.1 sp2pure_sparse_parallel_simple(MDITER)	910
8.398sp2pure_sparse_parallel_simple.f90	912
8.399sp2pure_subgraph_parallel.f90 File Reference	914
8.399.1 Function/Subroutine Documentation	914
8.399.1.1 sp2pure_subgraph_parallel(MDITER)	914
8.400sp2pure_subgraph_parallel.f90	915
8.401sp2T.f90 File Reference	917

8.401.1 Function/Subroutine Documentation	917
8.401.1.1 sp2t	917
8.402sp2T.f90	918
8.403sparsearray.f90 File Reference	922
8.404sparsearray.f90	922
8.405sparsemath.f90 File Reference	923
8.406sparsemath.f90	923
8.407sparsesp2.f90 File Reference	926
8.408sparsesp2.f90	926
8.409spinarray.f90 File Reference	931
8.410spinarray.f90	931
8.411spinrhodirect.f90 File Reference	932
8.411.1 Function/Subroutine Documentation	932
8.411.1.1 spinrhoevcs	932
8.412spinrhodirect.f90	934
8.413stdescent.f90 File Reference	938
8.413.1 Function/Subroutine Documentation	938
8.413.1.1 stdescent(ITER, DELTAENERGY, DELTAF)	938
8.414stdescent.f90	939
8.415subgraph.f90 File Reference	940
8.416subgraph.f90	941
8.417subgraphsp2.f90 File Reference	943
8.418subgraphsp2.f90	944
8.419summary.f90 File Reference	947
8.419.1 Function/Subroutine Documentation	947
8.419.1.1 summary	947
8.420summary.f90	948
8.421tabtest.f90 File Reference	956
8.421.1 Function/Subroutine Documentation	956
8.421.1.1 tabtest	956

8.422tabtest.f90	956
8.423tbmd.f90 File Reference	957
8.423.1 Function/Subroutine Documentation	958
8.423.1.1 tbmd	958
8.424tbmd.f90	959
8.425timer_mod.f90 File Reference	965
8.426timer_mod.f90	966
8.427tlmmp.f90 File Reference	968
8.427.1 Function/Subroutine Documentation	968
8.427.1.1 tlmmp(L, M, MP, ALPHA, COSBETA)	968
8.428tlmmp.f90	968
8.429toteng.f90 File Reference	969
8.429.1 Function/Subroutine Documentation	969
8.429.1.1 toteng	969
8.430toteng.f90	970
8.431univarray.f90 File Reference	972
8.432univarray.f90	972
8.433univscaling.f90 File Reference	973
8.433.1 Function/Subroutine Documentation	973
8.433.1.1 univscale_sub(R, A, X)	973
8.434univscaling.f90	974
8.435univscaling_function.f90 File Reference	975
8.435.1 Function/Subroutine Documentation	975
8.435.1.1 univscale(I, J, L1, L2, MP, R, WHICHINT)	975
8.436univscaling_function.f90	976
8.437univtailcoef.f90 File Reference	978
8.437.1 Function/Subroutine Documentation	978
8.437.1.1 univtailcoef(A)	978
8.438univtailcoef.f90	979
8.439vdwtailcoef.f90 File Reference	981

8.439.1 Function/Subroutine Documentation	981
8.439.1.1 vdwtailcoef	981
8.440vdwtailcoef.f90	982
8.441velverlet.f90 File Reference	984
8.441.1 Function/Subroutine Documentation	984
8.441.1.1 velverlet(CURRITER)	984
8.442velverlet.f90	986
8.443virialarray.f90 File Reference	987
8.444virialarray.f90	988
8.445wignerd.f90 File Reference	988
8.445.1 Function/Subroutine Documentation	989
8.445.1.1 wignerd(L, M, MP, COSBETA)	989
8.446wignerd.f90	989
8.447wrtcfgs.f90 File Reference	990
8.447.1 Function/Subroutine Documentation	990
8.447.1.1 wrtcfgs(ITER)	990
8.448wrtcfgs.f90	991
8.449wrtrestart.f90 File Reference	997
8.449.1 Function/Subroutine Documentation	997
8.449.1.1 wrtrestart(ITER)	997
8.450wrtrestart.f90	998
8.451wrtrestartlib.f90 File Reference	1002
8.451.1 Function/Subroutine Documentation	1002
8.451.1.1 wrtrestartlib(ITER)	1002
8.452wrtrestartlib.f90	1002
8.453xbo.f90 File Reference	1017
8.453.1 Function/Subroutine Documentation	1018
8.453.1.1 xbo(ITER)	1018
8.454xbo.f90	1019
8.455xboarray.f90 File Reference	1021
8.456xboarray.f90	1021
Bibliography	1023
Index	1025

Chapter 1

README

LATTE

Open source density functional tight binding molecular dynamics.

#LA-CC-10-004

This material was produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National Laboratory (LANL), which is operated by Los Alamos National Security, LLC for the U.S. Department of Energy. The U.S. Government has rights to use, reproduce, and distribute this software. NEITHER THE GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS SOFTWARE. If software is modified to produce derivative works, such modified software should be clearly marked, so as not to confuse it with the version available from LANL.

Additionally, this program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; version 2.0 of the License. Accordingly, this program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

Authors

Nicolas Bock (T-1) Marc J. Cawkwell (T-1) Josh D. Coe (T-1) Aditi Krishnapriyan (T-1) Matthew P. Kroonblawd (T-1) Adam Lang (T-1) Enrique Martinez Saez (MST-8) Susan M. Mniszewski (CCS-3) Christian F. A. Negre (T-1) Anders M. N. Niklasson (T-1) Edward Sanville (T-1) Mitchell A. Wood (T-1) Ping Yang (T-1)

Los Alamos National Laboratory

Citing

To cite the code, please proceed as follows:

with the following `bibtex` citation:

```
@misc{LATTE,  
  title = {LATTE},  
  url = {https://github.com/lanl/LATTE},  
  author = {N. Bock and M. J. Cawkwell and J. D. Coe and A. Krishnapriyan and M. P. Kroonblawd and A. Lang and  
  year = {2008}  
}
```


Chapter 2

Todo List

Module `constraints_mod`

Add constrains for orientation and mass center distances.

Type `constraints_mod::constrains_type`

construct a parser for the constrains

Subprogram `constraints_mod::freeze_atoms` (FTOT, VEL)

Generalize this to have constrains to different coordinates.

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

constants_mod	15
constraints_mod	
! To add constraints to the system. This module can be used to add constraints to the system.	
For example, fixing the position of certain atoms	29
coulombarray	31
dbcsr_var_mod	34
diagarray	40
fermicommon	43
genxprogress	
To produce a matrix Z which is needed to orthogonalize H	45
get_end_scope	47
homolumo	48
kernelparser_mod	
Some general parsing functions	50
kspacearray	52
latte_lib	54
latteparser_latte_mod	
LATTE parser	55
matrixio	62
mdarray	64
mixer_mod	
To implement mixing schemes from the progress library	70
myprecision	73
neblastarray	77
nonoarray	79
openfiles_mod	
Module to handle input output files	81
ppotarray	84
purearray	86
relaxcommon	87
restartarray	88
setuparray	89
sp2progress	
To apply the sp2 method using the progress library	93
sparsearray	96

sparsemath	97
sparsesp2	102
spinarray	108
subgraph	110
subgraphsp2	117
timer_mod	124
univarray	132
virialarray	133
xboarray	134

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

constraints_mod::constrains_type	
General input variables for constrains	137
latteparser_latte_mod::latte_type	
General latte input variables type	138

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

addqdep.f90	144
algo.f90	151
allfit.f90	151
allocatcoulomb.f90	161
allocatediag.f90	163
allocatenebarrays.f90	166
allocatenono.f90	168
allocatpure.f90	170
allocatesubgraph.f90	172
allocatexbo.f90	173
am.f90	177
assessocc.f90	177
atomcharge.f90	179
avepress.f90	184
avesforhug.f90	187
avetemp.f90	189
bldnewH.f90	191
bldnewHS_sp.f90	201
bldspinH.f90	211
bm.f90	217
bodirect.f90	217
bodirectprogress.f90	222
caselist.f90	225
conjgradient.f90	225
constants_mod.f90	228
constraints_mod.f90	231
coulomb_ewald.f90	233
coulomb_oldskool.f90	236
coulomb_rspace.f90	240
coulombarray.f90	244
coultailcoef.f90	246
dbcsr_var_mod.f90	248
deallocateall.f90	251
deallocatecoulomb.f90	255
deallocatediag.f90	257

deallocatenebarrays.f90	259
deallocatenono.f90	261
deallocatepure.f90	263
deallocatesubgraph.f90	265
deallocatexbo.f90	266
deorthomyrho.f90	268
deorthomyrhoprogress.f90	271
dfda.f90	274
dfdb.f90	275
dfdr.f90	276
dgspdr.f90	277
diagarray.f90	279
diagmyh.f90	280
dosfit.f90	283
dslmmpda.f90	291
dslmmpdb.f90	292
dtlmpda.f90	293
dtlmpdb.f90	294
dunivscaling.f90	295
dunivscaling_function.f90	297
dwignerddb.f90	300
entropy.f90	301
errors.f90	311
factorial.f90	314
fcoulnono.f90	315
fcoulnono_sp.f90	321
fermiallocate.f90	335
fermicommon.f90	337
fermideallocate.f90	338
fermiexpans.f90	340
fittingoutput.f90	345
fspinnono.f90	347
fspinnono_sp.f90	354
gaussrn.f90	363
gendiag.f90	364
genX.f90	366
genXprogress.f90	370
gershgorin.f90	372
get_end_scope.py	375
getbndfil.f90	376
getcoule.f90	379
getdeltaq.f90	381
getdeltaspin.f90	385
getdensity.f90	395
getdipole.f90	397
getforce.f90	399
gethdim.f90	403
getke.f90	407
getmatindlist.f90	409
getmaxf.f90	412
getmdf.f90	414
getpressure.f90	419
getrespf.f90	422
getrho.f90	424
getspinE.f90	428
getspinE_zero.f90	432
getsplinennr.f90	433
gradH.f90	435

gradH_sp.f90	441
homolumo.f90	454
hugrescale.f90	457
ifrestart.f90	460
init_dbcsr.f90	463
initcoulomb.f90	467
initcoulombklist.f90	470
initialv.f90	474
initrng.f90	478
initshockcomp.f90	480
kbldnewh.f90	482
kbodirect.f90	493
kdeorthomyrho.f90	500
kdiagmyh.f90	502
kernelparser_mod.f90	506
kfcoulnono.f90	511
kgenX.f90	519
kgetdos.f90	522
kgetrho.f90	525
kgradH.f90	528
korthomyH.f90	536
kpulay.f90	538
kspacearray.f90	547
latte.f90	548
latte_c_bind.f90	555
latte_lib.f90	558
latteparser_latte_mod.f90	569
masses2symbols.f90	578
matrixio.f90	581
mdarray.f90	583
mixer_mod.f90	585
mofit.f90	586
mofit_plato.f90	592
msrelax.f90	600
myprecision.f90	607
neblast_cell.f90	608
neblastarray.f90	625
neblasts.f90	627
newkbldnewh.f90	634
nnz.f90	643
noelec.f90	644
nonoarray.f90	645
norms.f90	646
nptrescale.f90	649
nvtandersen.f90	652
nvtlangevin.f90	654
nvtNH.f90	658
nvtrescale.f90	662
openfiles_mod.f90	666
orthomyH.f90	667
orthomyHprogress.f90	670
orthomyrho.f90	673
pairpot.f90	674
pairpot_noneb.f90	678
pairpotspline.f90	682
pairpottab.f90	685
panic.f90	688
parafileopen.f90	693

parawrite.f90	695
pbcs.f90	697
plot_ppot.f90	699
plot_univ.f90	702
ppfit.f90	705
ppotarray.f90	715
printsparse.f90	716
propchempot_xbo.f90	717
prospins_xbo.f90	720
pulay.f90	723
pulay_sp.f90	731
pulay_spprogress.f90	741
purearray.f90	751
qconsistency.f90	752
qneutral.f90	760
readcontrols.f90	766
readcr.f90	772
readmdcontroller.f90	775
readppot.f90	779
readppotspline.f90	782
readppottab.f90	785
readrestart.f90	788
readrestartlib.f90	792
readtb.f90	803
relaxcommon.f90	806
resetprohd.f90	807
restartarray.f90	810
rhozero.f90	811
setuparray.f90	834
setuptbmd.f90	836
shiftH.f90	838
shockcomp.f90	843
shutdown_dbcsr.f90	845
slmmp.f90	847
solvematcg.f90	848
solvematcg_sparse.f90	855
solvematlapack.f90	861
sp2dbcsr.f90	864
sp2fermi.f90	867
sp2fermi_init.f90	874
sp2gap.f90	883
sp2gap_setup.f90	885
sp2progress.f90	889
sp2pure.f90	890
sp2pure_sparse.f90	896
sp2pure_sparse_parallel.f90	906
sp2pure_sparse_parallel_simple.f90	910
sp2pure_subgraph_parallel.f90	914
sp2T.f90	917
sparsearray.f90	922
sparsemath.f90	923
sparsesp2.f90	926
spinarray.f90	931
spinrhodirect.f90	932
stdescent.f90	938
subgraph.f90	940
subgraphsp2.f90	943
summary.f90	947

tabtest.f90	956
tbmd.f90	957
timer_mod.f90	965
tlmmp.f90	968
toteng.f90	969
univarray.f90	972
univscaling.f90	973
univscaling_function.f90	975
univtailcoef.f90	978
vdwtailcoef.f90	981
velverlet.f90	984
virialarray.f90	987
wignerd.f90	988
wrtcfgs.f90	990
wrtrestart.f90	997
wrtrestartlib.f90	1002
xbo.f90	1017
xboarray.f90	1021

Chapter 6

Namespace Documentation

6.1 constants_mod Module Reference

Variables

- integer [allfiton](#)
- character(len=20) [basistype](#)
- integer [binfitstep](#)
- integer [bint2fit](#)
- integer [blksz](#)
- real(latteprec) [bndfil](#)
- real(latteprec), dimension(3, 3) [box](#)
- real(latteprec), dimension(3, 3) [box_old](#)
- real(latteprec), dimension(3) [boxdims](#)
- real(latteprec) [breaktol](#)
- integer [cgorlib](#)
- integer [charge](#)
- real(latteprec) [chempot](#)
- real(latteprec) [chtol](#)
- integer [compforce](#)
- integer [control](#)
- character(len=100) [coordsfile](#) = `"/bl/inputblock.dat"`
- real(latteprec) [cove](#)
- integer [debugon](#)
- integer [dosfiton](#)
- real(latteprec) [ecoul](#)
- real(latteprec) [egap](#)
- real(latteprec) [ehomo](#)
- real(latteprec) [elec_etol](#)
- real(latteprec) [elec_qtol](#)
- integer [elecmeth](#)
- integer [electro](#)
- real(latteprec) [elumo](#)
- integer [emeth](#)
- real(latteprec) [ente](#)
- integer [entropykind](#)
- real(latteprec) [eps](#)
- real(latteprec) [erep](#)

- real(latteprec) [espin](#)
- real(latteprec) [espin_zero](#)
- logical(1) [existerror](#)
- real(latteprec), parameter [f2v](#) = 9.6484504393669415d-003
- integer [fiton](#)
- integer [freeze](#)
- integer [fullqconv](#)
- integer [hdim](#)
- integer [int2fit](#)
- character(len=20) [job](#)
- real(latteprec) [kbt](#)
- real(latteprec), parameter [ke2t](#) = 1.0/0.000086173435
- real(latteprec) [kee](#)
- integer [kon](#)
- logical [latteinexists](#)
- integer [lcniter](#)
- integer [lcnon](#)
- integer [libcalls](#) = 0
- logical [libinit](#) = .FALSE.
- logical [librun](#) = .FALSE.
- real(latteprec) [massden](#)
- real(latteprec) [maxeval](#)
- real(latteprec) [maxminusmin](#)
- integer [maxscf](#)
- real(latteprec) [mcbeta](#)
- real(latteprec) [mcsigma](#)
- integer [mdadapt](#)
- real(latteprec) [mdmix](#)
- integer [mdon](#)
- real(latteprec) [mineval](#)
- integer [minsp2iter](#)
- integer [mixer](#)
- real(latteprec), parameter [mvv2ke](#) = 166.0538782/1.602176487
- real(latteprec), parameter [mvv2t](#) = 1660538.782/1.3806504
- integer [nats](#)
- integer [nfitstep](#)
- integer [ngpu](#)
- integer [nmat](#)
- integer [noelem](#)
- integer [noint](#)
- integer [norecs](#)
- integer [numscf](#)
- real(latteprec) [occerrlimit](#)
- integer [occsteps](#)
- integer [ordernmol](#)
- character(len=100) [parampath](#) = "./TBparam"
- integer [parrep](#)
- integer [pbcon](#)
- real(latteprec), parameter [pi](#) = TWO*ACOS(ZERO)
- integer [pp2fit](#)
- real(latteprec) [ppbeta](#)
- integer [ppfiton](#)
- integer [ppnfitstep](#)
- integer [ppngeom](#)
- integer [ppnmol](#)

- integer [ppoton](#)
- real(latteprec) [ppsigma](#)
- integer [qfit](#)
- integer [qiter](#)
- real(latteprec) [qmix](#)
- integer [relaxme](#)
- integer [restart](#)
- integer [restartlib](#)
- integer [rslevel](#)
- integer [scfs](#)
- integer [scfs_i](#)
- integer [scfstep](#) = 0
- character(len=3) [sp2conv](#)
- integer [sparseon](#)
- real(latteprec) [spinmix](#)
- integer [spinon](#)
- real(latteprec) [spintol](#)
- integer [sponly](#)
- real(latteprec) [summass](#)
- real(latteprec), parameter [togpa](#) = 160.2176487
- real(latteprec) [tote](#)
- real(latteprec) [totne](#)
- real(latteprec) [tracelimit](#)
- real(latteprec) [trrhoh](#)
- real(latteprec) [tscale](#)
- integer [vardt](#)
- integer [vdwon](#)
- integer [verbose](#) = 1
- integer [xbodison](#)
- integer [xbodisorder](#)
- integer [xboon](#)

6.1.1 Variable Documentation

6.1.1.1 integer constants_mod::allfiton

Definition at line [53](#) of file [constants_mod.f90](#).

6.1.1.2 character(len = 20) constants_mod::basistype

Definition at line [76](#) of file [constants_mod.f90](#).

6.1.1.3 integer constants_mod::binfitstep

Definition at line [52](#) of file [constants_mod.f90](#).

6.1.1.4 integer constants_mod::bint2fit

Definition at line [52](#) of file [constants_mod.f90](#).

6.1.1.5 integer constants_mod::blksz

Definition at line 44 of file [constants_mod.f90](#).

6.1.1.6 real(latteprec) constants_mod::bndfil

Definition at line 62 of file [constants_mod.f90](#).

6.1.1.7 real(latteprec), dimension(3,3) constants_mod::box

Definition at line 61 of file [constants_mod.f90](#).

6.1.1.8 real(latteprec), dimension(3,3) constants_mod::box_old

Definition at line 61 of file [constants_mod.f90](#).

6.1.1.9 real(latteprec), dimension(3) constants_mod::boxdims

Definition at line 61 of file [constants_mod.f90](#).

6.1.1.10 real(latteprec) constants_mod::breaktol

Definition at line 71 of file [constants_mod.f90](#).

6.1.1.11 integer constants_mod::cgorlib

Definition at line 37 of file [constants_mod.f90](#).

6.1.1.12 integer constants_mod::charge

Definition at line 47 of file [constants_mod.f90](#).

6.1.1.13 real(latteprec) constants_mod::chempot

Definition at line 66 of file [constants_mod.f90](#).

6.1.1.14 real(latteprec) constants_mod::chtol

Definition at line 68 of file [constants_mod.f90](#).

6.1.1.15 integer constants_mod::compforce

Definition at line 49 of file [constants_mod.f90](#).

6.1.1.16 integer constants_mod::control

Definition at line 30 of file [constants_mod.f90](#).

6.1.1.17 character(len = 100) constants_mod::coordsfile = "/bl/inputblock.dat"

Definition at line 79 of file [constants_mod.f90](#).

6.1.1.18 real(latteprec) constants_mod::cove

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.19 integer constants_mod::debugon

Definition at line 43 of file [constants_mod.f90](#).

6.1.1.20 integer constants_mod::dosfiton

Definition at line 51 of file [constants_mod.f90](#).

6.1.1.21 real(latteprec) constants_mod::ecoul

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.22 real(latteprec) constants_mod::egap

Definition at line 67 of file [constants_mod.f90](#).

6.1.1.23 real(latteprec) constants_mod::ehomo

Definition at line 67 of file [constants_mod.f90](#).

6.1.1.24 real(latteprec) constants_mod::elec_etol

Definition at line 69 of file [constants_mod.f90](#).

6.1.1.25 `real(latteprec) constants_mod::elec_qtol`

Definition at line 69 of file [constants_mod.f90](#).

6.1.1.26 `integer constants_mod::elecmeth`

Definition at line 34 of file [constants_mod.f90](#).

6.1.1.27 `integer constants_mod::electro`

Definition at line 34 of file [constants_mod.f90](#).

6.1.1.28 `real(latteprec) constants_mod::elum0`

Definition at line 67 of file [constants_mod.f90](#).

6.1.1.29 `integer constants_mod::emeth`

Definition at line 37 of file [constants_mod.f90](#).

6.1.1.30 `real(latteprec) constants_mod::ente`

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.31 `integer constants_mod::entropykind`

Definition at line 42 of file [constants_mod.f90](#).

6.1.1.32 `real(latteprec) constants_mod::eps`

Definition at line 95 of file [constants_mod.f90](#).

6.1.1.33 `real(latteprec) constants_mod::erep`

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.34 `real(latteprec) constants_mod::espin`

Definition at line 64 of file [constants_mod.f90](#).

6.1.1.35 `real(latteprec) constants_mod::espin_zero`

Definition at line 64 of file [constants_mod.f90](#).

6.1.1.36 `logical(1) constants_mod::existerror`

Definition at line 85 of file [constants_mod.f90](#).

6.1.1.37 `real(latteprec), parameter constants_mod::f2v = 9.6484504393669415d-003`

Definition at line 101 of file [constants_mod.f90](#).

6.1.1.38 `integer constants_mod::fiton`

Definition at line 43 of file [constants_mod.f90](#).

6.1.1.39 `integer constants_mod::freeze`

Definition at line 60 of file [constants_mod.f90](#).

6.1.1.40 `integer constants_mod::fullqconv`

Definition at line 36 of file [constants_mod.f90](#).

6.1.1.41 `integer constants_mod::hdim`

Definition at line 29 of file [constants_mod.f90](#).

6.1.1.42 `integer constants_mod::int2fit`

Definition at line 51 of file [constants_mod.f90](#).

6.1.1.43 `character(len = 20) constants_mod::job`

Definition at line 82 of file [constants_mod.f90](#).

6.1.1.44 `real(latteprec) constants_mod::kbt`

Definition at line 66 of file [constants_mod.f90](#).

6.1.1.45 `real(latteprec), parameter constants_mod::ke2t = 1.0/0.000086173435`

Definition at line 100 of file [constants_mod.f90](#).

6.1.1.46 `real(latteprec) constants_mod::kee`

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.47 `integer constants_mod::kon`

Definition at line 48 of file [constants_mod.f90](#).

6.1.1.48 `logical constants_mod::latteinexists`

Definition at line 89 of file [constants_mod.f90](#).

6.1.1.49 `integer constants_mod::lcniter`

Definition at line 33 of file [constants_mod.f90](#).

6.1.1.50 `integer constants_mod::lcnon`

Definition at line 33 of file [constants_mod.f90](#).

6.1.1.51 `integer constants_mod::libcalls = 0`

Definition at line 84 of file [constants_mod.f90](#).

6.1.1.52 `logical constants_mod::libinit = .FALSE.`

Definition at line 83 of file [constants_mod.f90](#).

6.1.1.53 `logical constants_mod::librun = .FALSE.`

Definition at line 86 of file [constants_mod.f90](#).

6.1.1.54 `real(latteprec) constants_mod::massden`

Definition at line 72 of file [constants_mod.f90](#).

6.1.1.55 `real(latteprec) constants_mod::maxeval`

Definition at line 65 of file [constants_mod.f90](#).

6.1.1.56 `real(latteprec) constants_mod::maxminusmin`

Definition at line 65 of file [constants_mod.f90](#).

6.1.1.57 `integer constants_mod::maxscf`

Definition at line 40 of file [constants_mod.f90](#).

6.1.1.58 `real(latteprec) constants_mod::mcbeta`

Definition at line 73 of file [constants_mod.f90](#).

6.1.1.59 `real(latteprec) constants_mod::mcsigma`

Definition at line 73 of file [constants_mod.f90](#).

6.1.1.60 `integer constants_mod::mdadapt`

Definition at line 54 of file [constants_mod.f90](#).

6.1.1.61 `real(latteprec) constants_mod::mdmix`

Definition at line 70 of file [constants_mod.f90](#).

6.1.1.62 `integer constants_mod::mdon`

Definition at line 30 of file [constants_mod.f90](#).

6.1.1.63 `real(latteprec) constants_mod::mineval`

Definition at line 65 of file [constants_mod.f90](#).

6.1.1.64 `integer constants_mod::minsp2iter`

Definition at line 40 of file [constants_mod.f90](#).

6.1.1.65 integer constants_mod::mixer

Definition at line 57 of file [constants_mod.f90](#).

6.1.1.66 real(latteprec), parameter constants_mod::mvv2ke = 166.0538782/1.602176487

Definition at line 99 of file [constants_mod.f90](#).

6.1.1.67 real(latteprec), parameter constants_mod::mvv2t = 1660538.782/1.3806504

Definition at line 103 of file [constants_mod.f90](#).

6.1.1.68 integer constants_mod::nats

Definition at line 29 of file [constants_mod.f90](#).

6.1.1.69 integer constants_mod::nfitstep

Definition at line 51 of file [constants_mod.f90](#).

6.1.1.70 integer constants_mod::ngpu

Definition at line 45 of file [constants_mod.f90](#).

6.1.1.71 integer constants_mod::nmat

Definition at line 29 of file [constants_mod.f90](#).

6.1.1.72 integer constants_mod::noelem

Definition at line 29 of file [constants_mod.f90](#).

6.1.1.73 integer constants_mod::noint

Definition at line 29 of file [constants_mod.f90](#).

6.1.1.74 integer constants_mod::norecs

Definition at line 42 of file [constants_mod.f90](#).

6.1.1.75 integer constants_mod::numscf

Definition at line 46 of file [constants_mod.f90](#).

6.1.1.76 real(latteprec) constants_mod::occerrlimit

Definition at line 95 of file [constants_mod.f90](#).

6.1.1.77 integer constants_mod::occsteps

Definition at line 93 of file [constants_mod.f90](#).

6.1.1.78 integer constants_mod::ordernmol

Definition at line 39 of file [constants_mod.f90](#).

6.1.1.79 character(len = 100) constants_mod::parampath = "/TBparam"

Definition at line 78 of file [constants_mod.f90](#).

6.1.1.80 integer constants_mod::parrep

Definition at line 55 of file [constants_mod.f90](#).

6.1.1.81 integer constants_mod::pbcon

Definition at line 30 of file [constants_mod.f90](#).

6.1.1.82 real(latteprec), parameter constants_mod::pi = TWO*ACOS(ZERO)

Definition at line 105 of file [constants_mod.f90](#).

6.1.1.83 integer constants_mod::pp2fit

Definition at line 52 of file [constants_mod.f90](#).

6.1.1.84 real(latteprec) constants_mod::ppbeta

Definition at line 74 of file [constants_mod.f90](#).

6.1.1.85 integer constants_mod::ppfiton

Definition at line 53 of file [constants_mod.f90](#).

6.1.1.86 integer constants_mod::ppnfitstep

Definition at line 52 of file [constants_mod.f90](#).

6.1.1.87 integer constants_mod::ppngeom

Definition at line 52 of file [constants_mod.f90](#).

6.1.1.88 integer constants_mod::ppnmol

Definition at line 52 of file [constants_mod.f90](#).

6.1.1.89 integer constants_mod::ppoton

Definition at line 35 of file [constants_mod.f90](#).

6.1.1.90 real(latteprec) constants_mod::ppsigma

Definition at line 74 of file [constants_mod.f90](#).

6.1.1.91 integer constants_mod::qfit

Definition at line 51 of file [constants_mod.f90](#).

6.1.1.92 integer constants_mod::qiter

Definition at line 36 of file [constants_mod.f90](#).

6.1.1.93 real(latteprec) constants_mod::qmix

Definition at line 70 of file [constants_mod.f90](#).

6.1.1.94 integer constants_mod::relaxme

Definition at line 30 of file [constants_mod.f90](#).

6.1.1.95 integer constants_mod::restart

Definition at line 30 of file [constants_mod.f90](#).

6.1.1.96 integer constants_mod::restartlib

Definition at line 59 of file [constants_mod.f90](#).

6.1.1.97 integer constants_mod::rslevel

Definition at line 58 of file [constants_mod.f90](#).

6.1.1.98 integer constants_mod::scfs

Definition at line 32 of file [constants_mod.f90](#).

6.1.1.99 integer constants_mod::scfs_ii

Definition at line 32 of file [constants_mod.f90](#).

6.1.1.100 integer constants_mod::scfstep = 0

Definition at line 92 of file [constants_mod.f90](#).

6.1.1.101 character(len = 3) constants_mod::sp2conv

Definition at line 75 of file [constants_mod.f90](#).

6.1.1.102 integer constants_mod::sparseon

Definition at line 38 of file [constants_mod.f90](#).

6.1.1.103 real(latteprec) constants_mod::spinmix

Definition at line 70 of file [constants_mod.f90](#).

6.1.1.104 integer constants_mod::spinon

Definition at line 38 of file [constants_mod.f90](#).

6.1.1.105 `real(latteprec) constants_mod::spintol`

Definition at line 68 of file [constants_mod.f90](#).

6.1.1.106 `integer constants_mod::sponly`

Definition at line 50 of file [constants_mod.f90](#).

6.1.1.107 `real(latteprec) constants_mod::summass`

Definition at line 72 of file [constants_mod.f90](#).

6.1.1.108 `real(latteprec), parameter constants_mod::togpa = 160.2176487`

Definition at line 102 of file [constants_mod.f90](#).

6.1.1.109 `real(latteprec) constants_mod::tote`

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.110 `real(latteprec) constants_mod::totne`

Definition at line 62 of file [constants_mod.f90](#).

6.1.1.111 `real(latteprec) constants_mod::tracelimit`

Definition at line 95 of file [constants_mod.f90](#).

6.1.1.112 `real(latteprec) constants_mod::trrhoh`

Definition at line 63 of file [constants_mod.f90](#).

6.1.1.113 `real(latteprec) constants_mod::tscale`

Definition at line 94 of file [constants_mod.f90](#).

6.1.1.114 `integer constants_mod::vardt`

Definition at line 41 of file [constants_mod.f90](#).

6.1.1.115 integer constants_mod::vdwon

Definition at line 35 of file [constants_mod.f90](#).

6.1.1.116 integer constants_mod::verbose = 1

Definition at line 56 of file [constants_mod.f90](#).

6.1.1.117 integer constants_mod::xbodison

Definition at line 31 of file [constants_mod.f90](#).

6.1.1.118 integer constants_mod::xbodisorder

Definition at line 31 of file [constants_mod.f90](#).

6.1.1.119 integer constants_mod::xboon

Definition at line 31 of file [constants_mod.f90](#).

6.2 constraints_mod Module Reference

! To add constrains to the system. This module can be used to add constrains to the system. For example, fixing the position of certain atoms.

Data Types

- type [constrains_type](#)
General input variables for constrains.

Functions/Subroutines

- subroutine, public [freeze_atoms](#) (FTOT, VEL)
To freeze a group of atoms. The atoms indices to be frozen are read from a file. The file is formatted as follows:

6.2.1 Detailed Description

! To add constrains to the system. This module can be used to add constrains to the system. For example, fixing the position of certain atoms.

Todo Add constrains for orientation and mass center distances.

6.2.2 Function/Subroutine Documentation

6.2.2.1 subroutine, public constraints_mod::freeze_atoms (real(dp), dimension(:,:), intent(inout) FTOT, real(dp), dimension(:,:), intent(inout), optional VEL)

To freeze a group of atoms. The atoms indices to be frozen are read from a file. The file is formatted as follows:

```
3 #Number of freezed atoms 10 #We are freezing all the coordinates of atom 10 20 #We are freezing all the
coordinates of atom 10 40 #We are freezing all the coordinates of atom 10
```

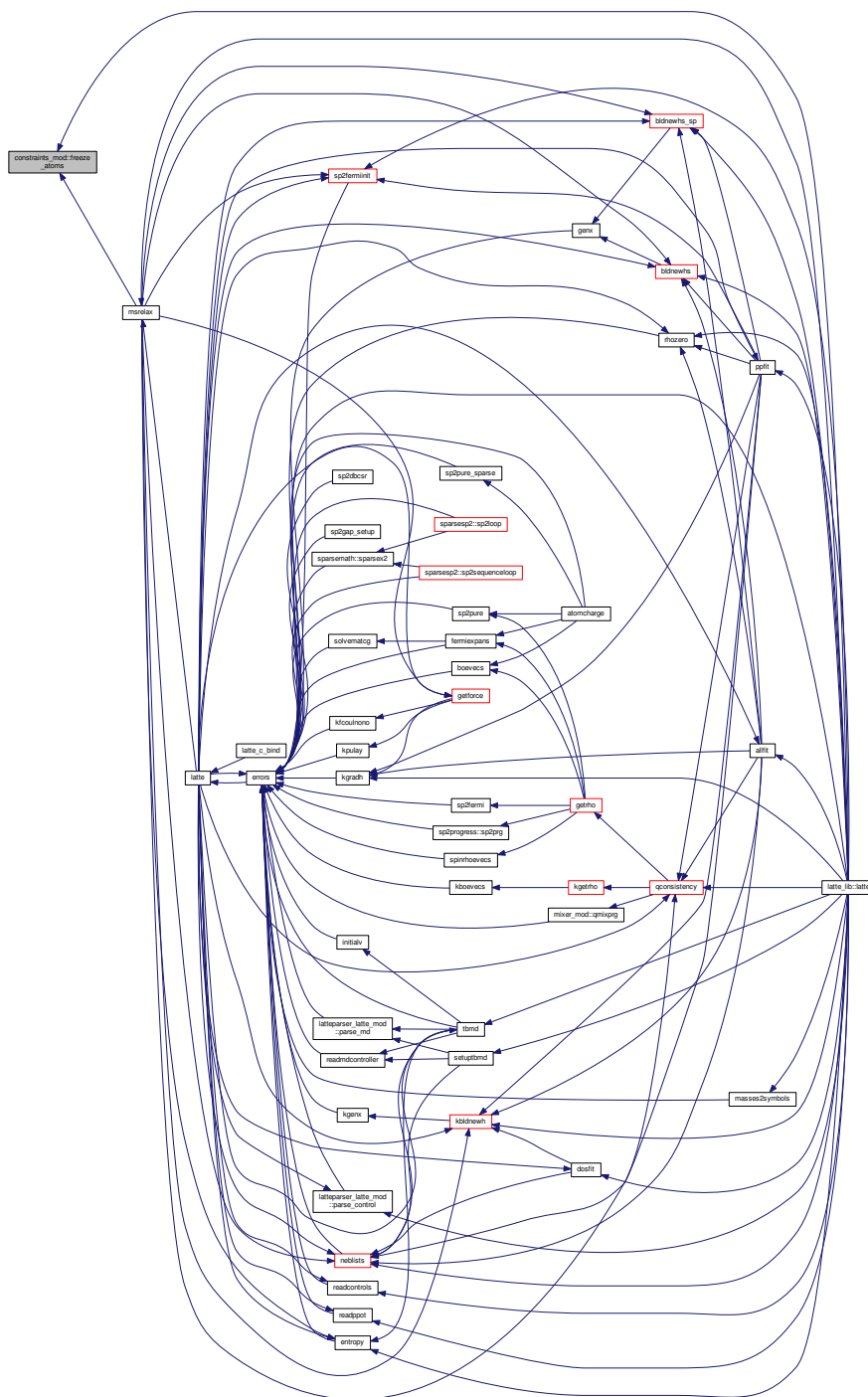
Todo Generalize this to have constrains to different coordinates.

Parameters

<i>FTOT</i>	Total forces. FTOT(1,3) gives the force on x direction for atom 3.
<i>VEL</i>	Velocities. VEL(1,3) gives the velocity on x direction for atom 3.

Definition at line 76 of file constraints_mod.f90.

Here is the caller graph for this function:



6.3 coulombarray Module Reference

Variables

- real(latteprec) [calpha](#)
- real(latteprec) [calpha2](#)
- real(latteprec), dimension(:), allocatable [coslist](#)
- real(latteprec) [coulacc](#)
- real(latteprec), dimension(6) [coulb](#)
- real(latteprec) [coulcut](#)
- real(latteprec) [coulcut2](#)
- real(latteprec) [coulr1](#)
- real(latteprec) [coulvol](#)
- real(latteprec) [eightpi](#)
- real(latteprec) [fourcalpha2](#)
- real(latteprec), dimension(:), allocatable [k1_list](#)
- real(latteprec), dimension(:), allocatable [k2_list](#)
- real(latteprec), dimension(:), allocatable [k3_list](#)
- real(latteprec) [kcutoff](#)
- real(latteprec) [kcutoff2](#)
- real(latteprec) [keconst](#)
- real(latteprec), dimension(:), allocatable [ksq_list](#)
- real(latteprec), dimension(3, 3) [latticevecs](#)
- integer [lmax](#)
- integer [mmax](#)
- integer [nk](#)
- integer [nmax](#)
- real(latteprec), dimension(:), allocatable [olddeltaqs](#)
- real(latteprec) [pi2](#)
- real(latteprec), dimension(3, 3) [recipvecs](#)
- real(latteprec) [relperm](#)
- real(latteprec), dimension(:), allocatable [sinlist](#)
- real(latteprec) [sqrtpi](#)
- real(latteprec) [tfact](#)
- real(latteprec) [twopi](#)

6.3.1 Variable Documentation

6.3.1.1 real(latteprec) coulombarray::calpha

Definition at line 38 of file [coulombarray.f90](#).

6.3.1.2 real(latteprec) coulombarray::calpha2

Definition at line 38 of file [coulombarray.f90](#).

6.3.1.3 real(latteprec), dimension(:), allocatable coulombarray::coslist

Definition at line 45 of file [coulombarray.f90](#).

6.3.1.4 `real(latteprec) coulombarray::coulacc`

Definition at line 32 of file [coulombarray.f90](#).

6.3.1.5 `real(latteprec), dimension(6) coulombarray::coulb`

Definition at line 39 of file [coulombarray.f90](#).

6.3.1.6 `real(latteprec) coulombarray::coulcut`

Definition at line 31 of file [coulombarray.f90](#).

6.3.1.7 `real(latteprec) coulombarray::coulcut2`

Definition at line 37 of file [coulombarray.f90](#).

6.3.1.8 `real(latteprec) coulombarray::coulr1`

Definition at line 33 of file [coulombarray.f90](#).

6.3.1.9 `real(latteprec) coulombarray::coulvol`

Definition at line 38 of file [coulombarray.f90](#).

6.3.1.10 `real(latteprec) coulombarray::eightpi`

Definition at line 40 of file [coulombarray.f90](#).

6.3.1.11 `real(latteprec) coulombarray::fourcalpha2`

Definition at line 38 of file [coulombarray.f90](#).

6.3.1.12 `real(latteprec), dimension(:), allocatable coulombarray::k1_list`

Definition at line 35 of file [coulombarray.f90](#).

6.3.1.13 `real(latteprec), dimension(:), allocatable coulombarray::k2_list`

Definition at line 35 of file [coulombarray.f90](#).

6.3.1.14 `real(latteprec), dimension(:), allocatable coulombarray::k3_list`

Definition at line 35 of file [coulombarray.f90](#).

6.3.1.15 `real(latteprec) coulombarray::kcutoff`

Definition at line 37 of file [coulombarray.f90](#).

6.3.1.16 `real(latteprec) coulombarray::kcutoff2`

Definition at line 37 of file [coulombarray.f90](#).

6.3.1.17 `real(latteprec) coulombarray::keconst`

Definition at line 42 of file [coulombarray.f90](#).

6.3.1.18 `real(latteprec), dimension(:), allocatable coulombarray::ksq_list`

Definition at line 36 of file [coulombarray.f90](#).

6.3.1.19 `real(latteprec), dimension(3,3) coulombarray::latticevecs`

Definition at line 34 of file [coulombarray.f90](#).

6.3.1.20 `integer coulombarray::lmax`

Definition at line 30 of file [coulombarray.f90](#).

6.3.1.21 `integer coulombarray::mmax`

Definition at line 30 of file [coulombarray.f90](#).

6.3.1.22 `integer coulombarray::nk`

Definition at line 30 of file [coulombarray.f90](#).

6.3.1.23 `integer coulombarray::nmax`

Definition at line 30 of file [coulombarray.f90](#).

6.3.1.24 `real(latteprec), dimension(:), allocatable coulombarray::olddeltaqs`

Definition at line 44 of file [coulombarray.f90](#).

6.3.1.25 `real(latteprec) coulombarray::pi2`

Definition at line 40 of file [coulombarray.f90](#).

6.3.1.26 `real(latteprec), dimension(3,3) coulombarray::recipvecs`

Definition at line 34 of file [coulombarray.f90](#).

6.3.1.27 `real(latteprec) coulombarray::relperm`

Definition at line 41 of file [coulombarray.f90](#).

6.3.1.28 `real(latteprec), dimension(:), allocatable coulombarray::sinlist`

Definition at line 45 of file [coulombarray.f90](#).

6.3.1.29 `real(latteprec) coulombarray::sqrtpi`

Definition at line 40 of file [coulombarray.f90](#).

6.3.1.30 `real(latteprec) coulombarray::tfact`

Definition at line 43 of file [coulombarray.f90](#).

6.3.1.31 `real(latteprec) coulombarray::twopi`

Definition at line 40 of file [coulombarray.f90](#).

6.4 `dbcsr_var_mod` Module Reference

Functions/Subroutines

- subroutine [myset_dist](#) (`dist_array`, `dist_size`, `nbins`)

Variables

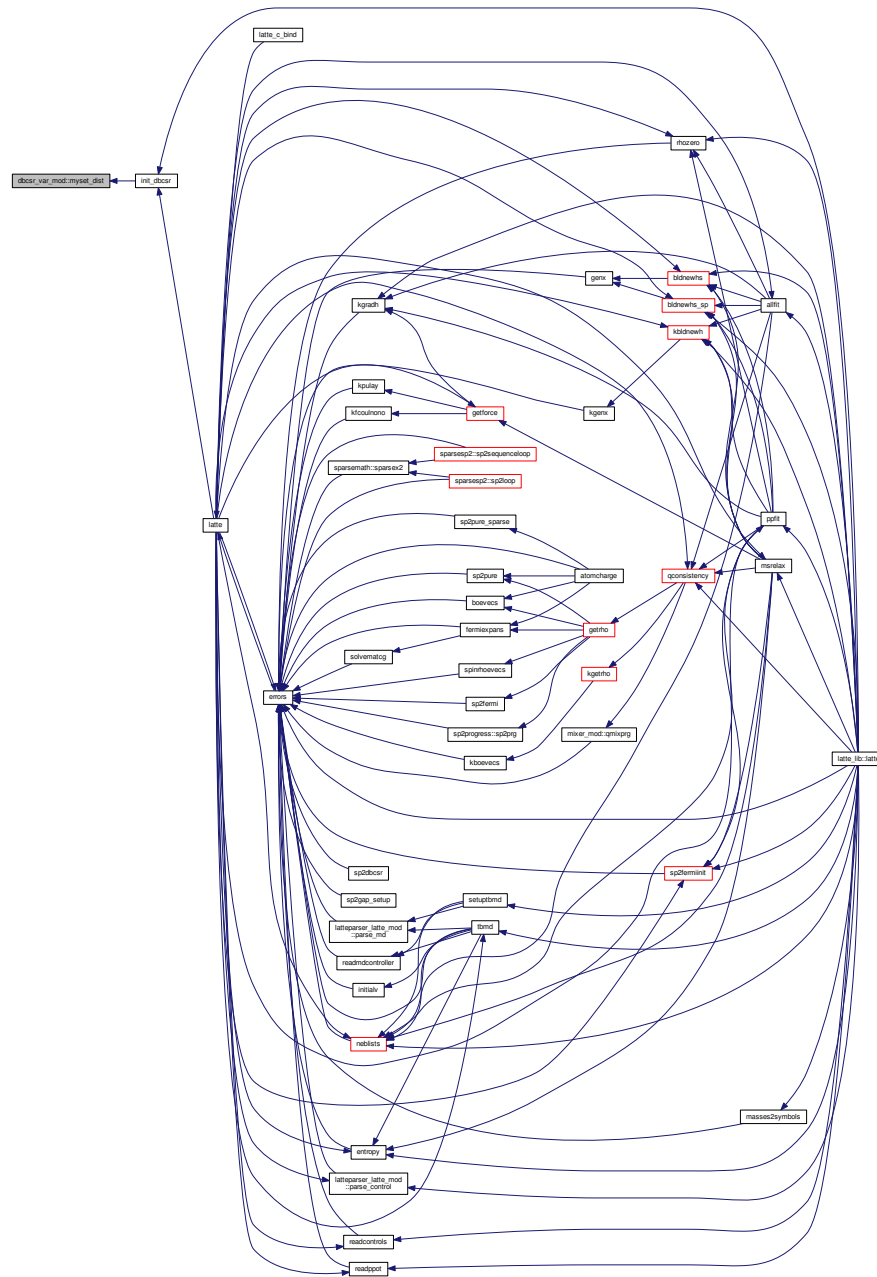
- integer, dimension(:), pointer [cbs](#)
- real(latteprec) [chksum](#)
- real(latteprec) [chksum2](#)
- type(array_i1d_obj) [col_blk_sizes](#)
- type(array_i1d_obj) [col_dist_a](#)
- real(latteprec), dimension(:), allocatable [diag](#)
- type(dbcsr_distribution_obj) [dist_a](#)
- type(dbcsr_distribution_obj) [dist_b](#)
- type(dbcsr_distribution_obj) [dist_c](#)
- type(dbcsr_error_type) [error](#)
- logical [found](#)
- integer, dimension(:), allocatable [grid_dist](#)
- integer [group](#)
- integer [ierr](#)
- type(dbcsr_iterator) [iter](#)
- type(dbcsr_obj) [matrix_a](#)
- type(dbcsr_obj) [matrix_b](#)
- integer [mp_comm](#)
- type(dbcsr_mp_obj) [mp_env](#)
- integer [mp_group](#)
- real(latteprec), dimension(2:2) [my_block](#)
- integer [mynode](#)
- integer, dimension(2) [myploc](#)
- integer [n](#)
- integer [nblkcols_total](#)
- integer [nblkrows_total](#)
- integer, dimension(2) [npdims](#)
- integer [numnodes](#)
- integer [pcol](#)
- integer, dimension(:, :), allocatable [pgrid](#)
- integer [proc_holds_blk](#)
- integer [prow](#)
- integer, dimension(:), pointer [rbs](#)
- type(array_i1d_obj) [row_blk_sizes](#)
- type(array_i1d_obj) [row_dist_a](#)
- integer [temp](#)
- logical [tr](#)

6.4.1 Function/Subroutine Documentation

- 6.4.1.1 subroutine `dbcsr_var_mod::myset_dist` (type(array_i1d_obj), intent(out) *dist_array*, integer, intent(in) *dist_size*, integer, intent(in) *nbins*)

Definition at line 84 of file `dbcsr_var_mod.f90`.

Here is the caller graph for this function:



6.4.2 Variable Documentation

6.4.2.1 integer, dimension(:), pointer dbcsr_var_mod::cbs

Definition at line 62 of file dbcsr_var_mod.f90.

6.4.2.2 real(latteprec) dbcsr_var_mod::chksum

Definition at line 75 of file `dbcsr_var_mod.f90`.

6.4.2.3 `real(latteprec) dbcsr_var_mod::chksum2`

Definition at line 75 of file [dbcsr_var_mod.f90](#).

6.4.2.4 `type(array_i1d_obj) dbcsr_var_mod::col_blk_sizes`

Definition at line 63 of file [dbcsr_var_mod.f90](#).

6.4.2.5 `type(array_i1d_obj) dbcsr_var_mod::col_dist_a`

Definition at line 63 of file [dbcsr_var_mod.f90](#).

6.4.2.6 `real(latteprec), dimension(:), allocatable dbcsr_var_mod::diag`

Definition at line 70 of file [dbcsr_var_mod.f90](#).

6.4.2.7 `type(dbcsr_distribution_obj) dbcsr_var_mod::dist_a`

Definition at line 69 of file [dbcsr_var_mod.f90](#).

6.4.2.8 `type(dbcsr_distribution_obj) dbcsr_var_mod::dist_b`

Definition at line 69 of file [dbcsr_var_mod.f90](#).

6.4.2.9 `type(dbcsr_distribution_obj) dbcsr_var_mod::dist_c`

Definition at line 69 of file [dbcsr_var_mod.f90](#).

6.4.2.10 `type(dbcsr_error_type) dbcsr_var_mod::error`

Definition at line 59 of file [dbcsr_var_mod.f90](#).

6.4.2.11 `logical dbcsr_var_mod::found`

Definition at line 72 of file [dbcsr_var_mod.f90](#).

6.4.2.12 `integer, dimension(:), allocatable dbcsr_var_mod::grid_dist`

Definition at line 73 of file [dbcsr_var_mod.f90](#).

6.4.2.13 integer dbcsr_var_mod::group

Definition at line 68 of file [dbcsr_var_mod.f90](#).

6.4.2.14 integer dbcsr_var_mod::ierr

Definition at line 64 of file [dbcsr_var_mod.f90](#).

6.4.2.15 type(dbcsr_iterator) dbcsr_var_mod::iter

Definition at line 74 of file [dbcsr_var_mod.f90](#).

6.4.2.16 type(dbcsr_obj) dbcsr_var_mod::matrix_a

Definition at line 57 of file [dbcsr_var_mod.f90](#).

6.4.2.17 type(dbcsr_obj) dbcsr_var_mod::matrix_b

Definition at line 57 of file [dbcsr_var_mod.f90](#).

6.4.2.18 integer dbcsr_var_mod::mp_comm

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.19 type(dbcsr_mp_obj) dbcsr_var_mod::mp_env

Definition at line 67 of file [dbcsr_var_mod.f90](#).

6.4.2.20 integer dbcsr_var_mod::mp_group

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.21 real(latteprec), dimension(2:2) dbcsr_var_mod::my_block

Definition at line 71 of file [dbcsr_var_mod.f90](#).

6.4.2.22 integer dbcsr_var_mod::mynode

Definition at line 68 of file [dbcsr_var_mod.f90](#).

6.4.2.23 integer, dimension(2) dbcsr_var_mod::myploc

Definition at line 68 of file [dbcsr_var_mod.f90](#).

6.4.2.24 integer dbcsr_var_mod::n

Definition at line 68 of file [dbcsr_var_mod.f90](#).

6.4.2.25 integer dbcsr_var_mod::nblkcols_total

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.26 integer dbcsr_var_mod::nblkrows_total

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.27 integer, dimension(2) dbcsr_var_mod::npdims

Definition at line 64 of file [dbcsr_var_mod.f90](#).

6.4.2.28 integer dbcsr_var_mod::numnodes

Definition at line 68 of file [dbcsr_var_mod.f90](#).

6.4.2.29 integer dbcsr_var_mod::pcol

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.30 integer, dimension(:,:), allocatable dbcsr_var_mod::pgrid

Definition at line 65 of file [dbcsr_var_mod.f90](#).

6.4.2.31 integer dbcsr_var_mod::proc_holds_blk

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.32 integer dbcsr_var_mod::prow

Definition at line 66 of file [dbcsr_var_mod.f90](#).

6.4.2.33 integer, dimension(:), pointer dbcsr_var_mod::rbs

Definition at line 62 of file [dbcsr_var_mod.f90](#).

6.4.2.34 type(array_i1d_obj) dbcsr_var_mod::row_blk_sizes

Definition at line 63 of file [dbcsr_var_mod.f90](#).

6.4.2.35 type(array_i1d_obj) dbcsr_var_mod::row_dist_a

Definition at line 63 of file [dbcsr_var_mod.f90](#).

6.4.2.36 integer dbcsr_var_mod::temp

Definition at line 76 of file [dbcsr_var_mod.f90](#).

6.4.2.37 logical dbcsr_var_mod::tr

Definition at line 72 of file [dbcsr_var_mod.f90](#).

6.5 diagarray Module Reference

Variables

- real(latteprec), dimension(:), allocatable [cplist](#)
- integer, dimension(:), allocatable [diag_iwork](#)
- integer [diag_liwork](#)
- integer [diag_lrwork](#)
- integer [diag_lwork](#)
- integer [diag_lzwork](#)
- real(latteprec), dimension(:), allocatable [diag_rwork](#)
- real(latteprec), dimension(:), allocatable [diag_work](#)
- complex(latteprec), dimension(:), allocatable [diag_zwork](#)
- real(latteprec), dimension(:), allocatable [downevals](#)
- real(latteprec), dimension(:, :), allocatable [downevecs](#)
- real(latteprec), dimension(:), allocatable [evals](#)
- real(latteprec), dimension(:, :), allocatable [evecs](#)
- real(latteprec), parameter [exptol](#) = 30.0
- integer, dimension(:), allocatable [ifail](#)
- real(latteprec), dimension(:, :), allocatable [kevals](#)
- complex(latteprec), dimension(:, :, :), allocatable [kevecs](#)
- complex(latteprec), dimension(:, :), allocatable [khtmp](#)
- real(latteprec) [numlimit](#)
- real(latteprec), dimension(:), allocatable [upevals](#)
- real(latteprec), dimension(:, :), allocatable [upevecs](#)
- complex(latteprec), dimension(:, :), allocatable [zbo](#)
- integer, dimension(:), allocatable [zheevd_iwork](#)
- integer [zheevd_liwork](#)
- integer [zheevd_lrwork](#)
- integer [zheevd_lwork](#)
- real(latteprec), dimension(:), allocatable [zheevd_rwork](#)
- complex(latteprec), dimension(:), allocatable [zheevd_work](#)

6.5.1 Variable Documentation

6.5.1.1 `real(latteprec), dimension(:), allocatable diagarray::cplist`

Definition at line 41 of file [diagarray.f90](#).

6.5.1.2 `integer, dimension(:), allocatable diagarray::diag_iwork`

Definition at line 31 of file [diagarray.f90](#).

6.5.1.3 `integer diagarray::diag_liwork`

Definition at line 29 of file [diagarray.f90](#).

6.5.1.4 `integer diagarray::diag_lrwork`

Definition at line 29 of file [diagarray.f90](#).

6.5.1.5 `integer diagarray::diag_lwork`

Definition at line 29 of file [diagarray.f90](#).

6.5.1.6 `integer diagarray::diag_lzwork`

Definition at line 29 of file [diagarray.f90](#).

6.5.1.7 `real(latteprec), dimension(:), allocatable diagarray::diag_rwork`

Definition at line 33 of file [diagarray.f90](#).

6.5.1.8 `real(latteprec), dimension(:), allocatable diagarray::diag_work`

Definition at line 33 of file [diagarray.f90](#).

6.5.1.9 `complex(latteprec), dimension(:), allocatable diagarray::diag_zwork`

Definition at line 35 of file [diagarray.f90](#).

6.5.1.10 `real(latteprec), dimension(:), allocatable diagarray::downvals`

Definition at line 39 of file [diagarray.f90](#).

6.5.1.11 `real(latteprec), dimension(:, :), allocatable diagarray::downvecs`

Definition at line 39 of file [diagarray.f90](#).

6.5.1.12 `real(latteprec), dimension(:), allocatable diagarray::evals`

Definition at line 37 of file [diagarray.f90](#).

6.5.1.13 `real(latteprec), dimension(:, :), allocatable diagarray::evecs`

Definition at line 37 of file [diagarray.f90](#).

6.5.1.14 `real(latteprec), parameter diagarray::exptol = 30.0`

Definition at line 43 of file [diagarray.f90](#).

6.5.1.15 `integer, dimension(:), allocatable diagarray::ifail`

Definition at line 31 of file [diagarray.f90](#).

6.5.1.16 `real(latteprec), dimension(:, :), allocatable diagarray::kevals`

Definition at line 41 of file [diagarray.f90](#).

6.5.1.17 `complex(latteprec), dimension(:, :, :), allocatable diagarray::kevecs`

Definition at line 42 of file [diagarray.f90](#).

6.5.1.18 `complex(latteprec), dimension(:, :, :), allocatable diagarray::khtmp`

Definition at line 40 of file [diagarray.f90](#).

6.5.1.19 `real(latteprec) diagarray::numlimit`

Definition at line 44 of file [diagarray.f90](#).

6.5.1.20 `real(latteprec), dimension(:), allocatable diagarray::upevals`

Definition at line 38 of file [diagarray.f90](#).

6.5.1.21 `real(latteprec), dimension(:, :), allocatable diagarray::upevecs`

Definition at line 38 of file [diagarray.f90](#).

6.5.1.22 `complex(latteprec), dimension(:, :), allocatable diagarray::zbo`

Definition at line 40 of file [diagarray.f90](#).

6.5.1.23 `integer, dimension(:), allocatable diagarray::zheevd_iwork`

Definition at line 32 of file [diagarray.f90](#).

6.5.1.24 `integer diagarray::zheevd_liwork`

Definition at line 30 of file [diagarray.f90](#).

6.5.1.25 `integer diagarray::zheevd_lrwork`

Definition at line 30 of file [diagarray.f90](#).

6.5.1.26 `integer diagarray::zheevd_lwork`

Definition at line 30 of file [diagarray.f90](#).

6.5.1.27 `real(latteprec), dimension(:), allocatable diagarray::zheevd_rwork`

Definition at line 34 of file [diagarray.f90](#).

6.5.1.28 `complex(latteprec), dimension(:), allocatable diagarray::zheevd_work`

Definition at line 36 of file [diagarray.f90](#).

6.6 fermicommon Module Reference

Variables

- `real(latteprec), dimension(:, :), allocatable a`
- `real(latteprec) cgtol`
- `real(latteprec) cgtol2`
- `integer fermim`
- `real(latteprec), dimension(:, :), allocatable p0`
- `real(latteprec), dimension(:, :), allocatable r0`
- `real(latteprec), dimension(:, :), allocatable tmpmat`
- `real(latteprec), dimension(:), allocatable vala`
- `real(latteprec), dimension(:), allocatable valp0`
- `real(latteprec), dimension(:), allocatable valr0`
- `real(latteprec), dimension(:), allocatable valrho`
- `real(latteprec), dimension(:), allocatable valtmp`
- `real(latteprec), dimension(:, :), allocatable x2`

6.6.1 Variable Documentation

6.6.1.1 `real(latteprec), dimension(:, :), allocatable fermicommon::a`

Definition at line 37 of file [fermicommon.f90](#).

6.6.1.2 `real(latteprec) fermicommon::cgtol`

Definition at line 31 of file [fermicommon.f90](#).

6.6.1.3 `real(latteprec) fermicommon::cgtol2`

Definition at line 31 of file [fermicommon.f90](#).

6.6.1.4 `integer fermicommon::fermim`

Definition at line 30 of file [fermicommon.f90](#).

6.6.1.5 `real(latteprec), dimension(:, :), allocatable fermicommon::p0`

Definition at line 36 of file [fermicommon.f90](#).

6.6.1.6 `real(latteprec), dimension(:, :), allocatable fermicommon::r0`

Definition at line 36 of file [fermicommon.f90](#).

6.6.1.7 `real(latteprec), dimension(:, :), allocatable fermicommon::tmpmat`

Definition at line 38 of file [fermicommon.f90](#).

6.6.1.8 `real(latteprec), dimension(:), allocatable fermicommon::vala`

Definition at line 43 of file [fermicommon.f90](#).

6.6.1.9 `real(latteprec), dimension(:), allocatable fermicommon::valp0`

Definition at line 42 of file [fermicommon.f90](#).

6.6.1.10 `real(latteprec), dimension(:), allocatable fermicommon::valr0`

Definition at line 42 of file [fermicommon.f90](#).

6.6.1.11 `real(latteprec), dimension(:), allocatable fermicommon::valrho`

Definition at line 42 of file [fermicommon.f90](#).

6.6.1.12 `real(latteprec), dimension(:), allocatable fermicommon::valtmp`

Definition at line 43 of file [fermicommon.f90](#).

6.6.1.13 `real(latteprec), dimension(:, :), allocatable fermicommon::x2`

Definition at line 38 of file [fermicommon.f90](#).

6.7 genxprogress Module Reference

To produce a matrix Z which is needed to orthogonalize H .

Functions/Subroutines

- subroutine, public [genxbml](#)

Variables

- integer, public [igenx](#) = 0
- type(bml_matrix_t), public [over_bml](#)
- type(bml_matrix_t), public [zk1_bml](#)
- type(bml_matrix_t), public [zk2_bml](#)
- type(bml_matrix_t), public [zk3_bml](#)
- type(bml_matrix_t), public [zk4_bml](#)
- type(bml_matrix_t), public [zk5_bml](#)
- type(bml_matrix_t), public [zk6_bml](#)
- type(bml_matrix_t), public [zmat_bml](#)
- type(genzspinp), public [zsp](#)

6.7.1 Detailed Description

To produce a matrix Z which is needed to orthogonalize H .

$$H_{orth} = Z^\dagger H Z \text{ See Negre 2016 [2]}$$

6.7.3.2 `type(bml_matrix_t), public genxprogress::over_bml`

Definition at line 42 of file [genXprogress.f90](#).

6.7.3.3 `type(bml_matrix_t), public genxprogress::zk1_bml`

Definition at line 43 of file [genXprogress.f90](#).

6.7.3.4 `type(bml_matrix_t), public genxprogress::zk2_bml`

Definition at line 43 of file [genXprogress.f90](#).

6.7.3.5 `type(bml_matrix_t), public genxprogress::zk3_bml`

Definition at line 43 of file [genXprogress.f90](#).

6.7.3.6 `type(bml_matrix_t), public genxprogress::zk4_bml`

Definition at line 44 of file [genXprogress.f90](#).

6.7.3.7 `type(bml_matrix_t), public genxprogress::zk5_bml`

Definition at line 44 of file [genXprogress.f90](#).

6.7.3.8 `type(bml_matrix_t), public genxprogress::zk6_bml`

Definition at line 44 of file [genXprogress.f90](#).

6.7.3.9 `type(bml_matrix_t), public genxprogress::zmat_bml`

Definition at line 44 of file [genXprogress.f90](#).

6.7.3.10 `type(genzspinp), public genxprogress::zsp`

Definition at line 46 of file [genXprogress.f90](#).

6.8 get_end_scope Namespace Reference

Functions

- def [get_end](#) (fin)
- def [main](#) ()

6.8.1 Function Documentation

6.8.1.1 `def get_end_scope.get_end (fin)`

Definition at line 4 of file [get_end_scope.py](#).

Here is the caller graph for this function:



6.8.1.2 `def get_end_scope.main ()`

The main function.

Definition at line 28 of file [get_end_scope.py](#).

Here is the call graph for this function:

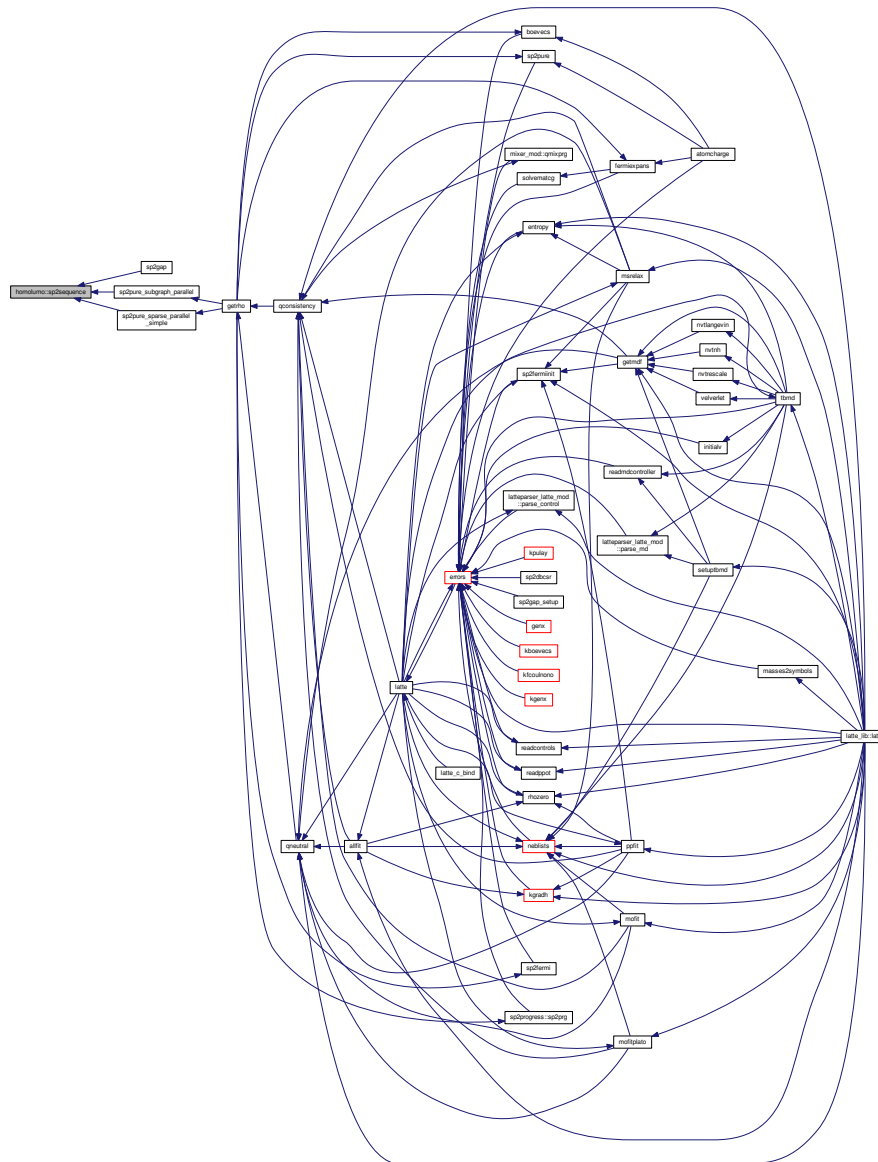


6.9 homolumo Module Reference

Functions/Subroutines

- subroutine [homolumogap](#) (ITERZ)
- subroutine [sp2sequence](#) ()

Here is the caller graph for this function:



6.10 kernelparser_mod Module Reference

Some general parsing functions.

Functions/Subroutines

- subroutine, public `parsing_kernel` (KEYVECTOR_CHAR, VALVECTOR_CHAR, KEYVECTOR_INT, VALVECTOR_INT, KEYVECTOR_RE, VALVECTOR_RE, KEYVECTOR_LOG, VALVECTOR_LOG, FILENAME, STARTSTOP)

The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.

6.10.1 Detailed Description

Some general parsing functions.

Author

C. F. A. Negre (cnegre@lanl.gov)

6.10.2 Function/Subroutine Documentation

6.10.2.1 subroutine, public kernelparser_mod::parsing_kernel (character(50), dimension(:) *KEYVECTOR_CHAR*, character(100), dimension(:) *VALVECTOR_CHAR*, character(50), dimension(:) *KEYVECTOR_INT*, integer, dimension(:) *VALVECTOR_INT*, character(50), dimension(:) *KEYVECTOR_RE*, real(dp), dimension(:) *VALVECTOR_RE*, character(50), dimension(:) *KEYVECTOR_LOG*, logical, dimension(:) *VALVECTOR_LOG*, character(len=*) *FILENAME*, character(len=*), dimension(2), intent(in), optional *STARTSTOP*)

The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.

Note

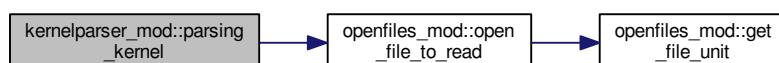
This parsing strategy can only parse a file of 500 lines by 500 words.

Warning

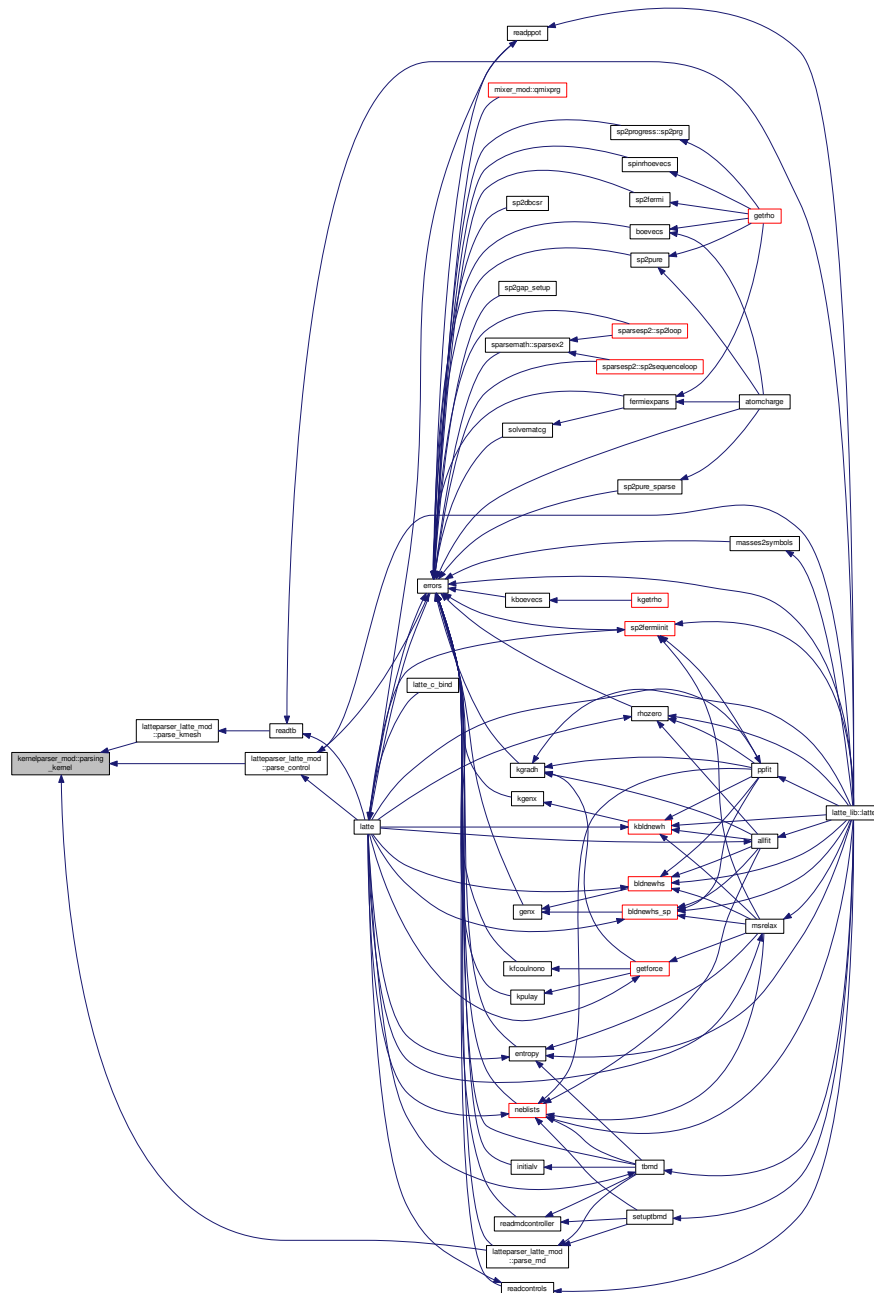
If the length of variable vect is changed, this could produce a segmentation fault.

Definition at line 34 of file [kernelparser_mod.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.11 kspacearray Module Reference

Variables

- `complex(latteprec, dimension(:, :, :), allocatable` [hk](#)
- `complex(latteprec, dimension(:, :, :), allocatable` [hk0](#)
- `complex(latteprec, dimension(:, :), allocatable` [hkdiag](#)
- `complex(latteprec, dimension(:, :, :), allocatable` [kbo](#)
- `complex(latteprec, dimension(:, :), allocatable` [kf](#)

- `complex(latteprec)`, `dimension(:, :, :)`, allocatable [korthoh](#)
- `real(latteprec)`, `dimension(3)` [kshift](#)
- `complex(latteprec)`, `dimension(:, :, :)`, allocatable [kxmat](#)
- integer [nktot](#)
- integer [nkx](#)
- integer [nky](#)
- integer [nkz](#)
- `complex(latteprec)`, `dimension(:, :, :)`, allocatable [sk](#)
- `complex(latteprec)`, `dimension(6)` [virbondk](#)
- `complex(latteprec)`, `dimension(:)`, allocatable [zhjj](#)

6.11.1 Variable Documentation

6.11.1.1 `complex(latteprec)`, `dimension(:, :, :)`, allocatable `kspacearray::hk`

Definition at line 32 of file [kspacearray.f90](#).

6.11.1.2 `complex(latteprec)`, `dimension(:, :, :)`, allocatable `kspacearray::hk0`

Definition at line 35 of file [kspacearray.f90](#).

6.11.1.3 `complex(latteprec)`, `dimension(:, :)`, allocatable `kspacearray::hkdiag`

Definition at line 32 of file [kspacearray.f90](#).

6.11.1.4 `complex(latteprec)`, `dimension(:, :, :)`, allocatable `kspacearray::kbo`

Definition at line 32 of file [kspacearray.f90](#).

6.11.1.5 `complex(latteprec)`, `dimension(:, :)`, allocatable `kspacearray::kf`

Definition at line 33 of file [kspacearray.f90](#).

6.11.1.6 `complex(latteprec)`, `dimension(:, :, :)`, allocatable `kspacearray::korthoh`

Definition at line 34 of file [kspacearray.f90](#).

6.11.1.7 `real(latteprec)`, `dimension(3)` `kspacearray::kshift`

Definition at line 30 of file [kspacearray.f90](#).

6.11.1.8 `complex(latteprec)`, `dimension(:, :, :)`, allocatable `kspacearray::kxmat`

Definition at line 33 of file [kspacearray.f90](#).

6.11.1.9 integer kspacearray::nktot

Definition at line 29 of file [kspacearray.f90](#).

6.11.1.10 integer kspacearray::nkx

Definition at line 29 of file [kspacearray.f90](#).

6.11.1.11 integer kspacearray::nky

Definition at line 29 of file [kspacearray.f90](#).

6.11.1.12 integer kspacearray::nkz

Definition at line 29 of file [kspacearray.f90](#).

6.11.1.13 complex(latteprec), dimension(:, :, :), allocatable kspacearray::sk

Definition at line 33 of file [kspacearray.f90](#).

6.11.1.14 complex(latteprec), dimension(6) kspacearray::virbondk

Definition at line 31 of file [kspacearray.f90](#).

6.11.1.15 complex(latteprec), dimension(:), allocatable kspacearray::zhjj

Definition at line 34 of file [kspacearray.f90](#).

6.12 latte_lib Module Reference

Functions/Subroutines

- subroutine, public [latte](#) (NTYPES, TYPES, CR_IN, MASSES_IN, XLO, XHI, XY, XZ, YZ, FTOT_OUT, MAX←
ITER_IN, VENERG, VEL_IN, DT_IN, VIRIAL_INOUT, NEWSYSTEM, EXISTERROR_INOUT)

Variables

- integer, parameter, public [latte_abiversion](#) = 20180207

6.12.1 Function/Subroutine Documentation

6.12.1.1 subroutine, public latte_lib::latte (integer, intent(in) *NTYPES*, integer, dimension(:), intent(in) *TYPES*, real(latteprec), dimension(:, :), intent(in) *CR_IN*, real(latteprec), dimension(:), intent(in) *MASSES_IN*, real(latteprec), dimension(3), intent(in) *XLO*, real(latteprec), dimension(3), intent(in) *XHI*, real(latteprec), intent(in) *XY*, real(latteprec), intent(in) *XZ*, real(latteprec), intent(in) *YZ*, real(latteprec), dimension(:, :), intent(out) *FTOT_OUT*, integer, intent(in) *MAXITER_IN*, real(latteprec), intent(out) *VENERG*, real(latteprec), dimension(:, :), intent(in) *VEL_IN*, real(latteprec), intent(in) *DT_IN*, real(latteprec), dimension(6), intent(out) *VIRIAL_INOUT*, integer, intent(inout) *NEWSYSTEM*, logical(1), intent(inout) *EXISTERROR_INOUT*)

Definition at line 111 of file [latte_lib.f90](#).

6.12.2 Variable Documentation

6.12.2.1 integer, parameter, public latte_lib::latte_abiversion = 20180207

Definition at line 65 of file [latte_lib.f90](#).

6.13 latteparser_latte_mod Module Reference

LATTE parser.

Data Types

- type [latte_type](#)
General latte input variables type.

Functions/Subroutines

- subroutine, public [parse_control](#) (FILENAME)
The parser for Latte General input variables.
- subroutine, public [parse_kmesh](#) (FILENAME)
The parser for K Mesh input variables.
- subroutine, public [parse_md](#) (FILENAME)
The parser for Latte General input variables.

Variables

- type([latte_type](#)), public [lt](#)

6.13.1 Detailed Description

LATTE parser.

This module is used to parse all the necessary input variables for a LATTE TB run (SCF/OPT/MD) Adding a new input keyword to the parser:

- If the variable is real, we have to increase nkey_re.
- Add the keyword (character type) in the keyvector_re vector.
- Add a default value (real type) in the valvector_re.
- Define a new variable in the latte type and pass the value through valvector_re(num) where num is the position of the new keyword in the vector.
- Use DUMMY= as a placeholder. This variable will be ignored by not searched by the parser.

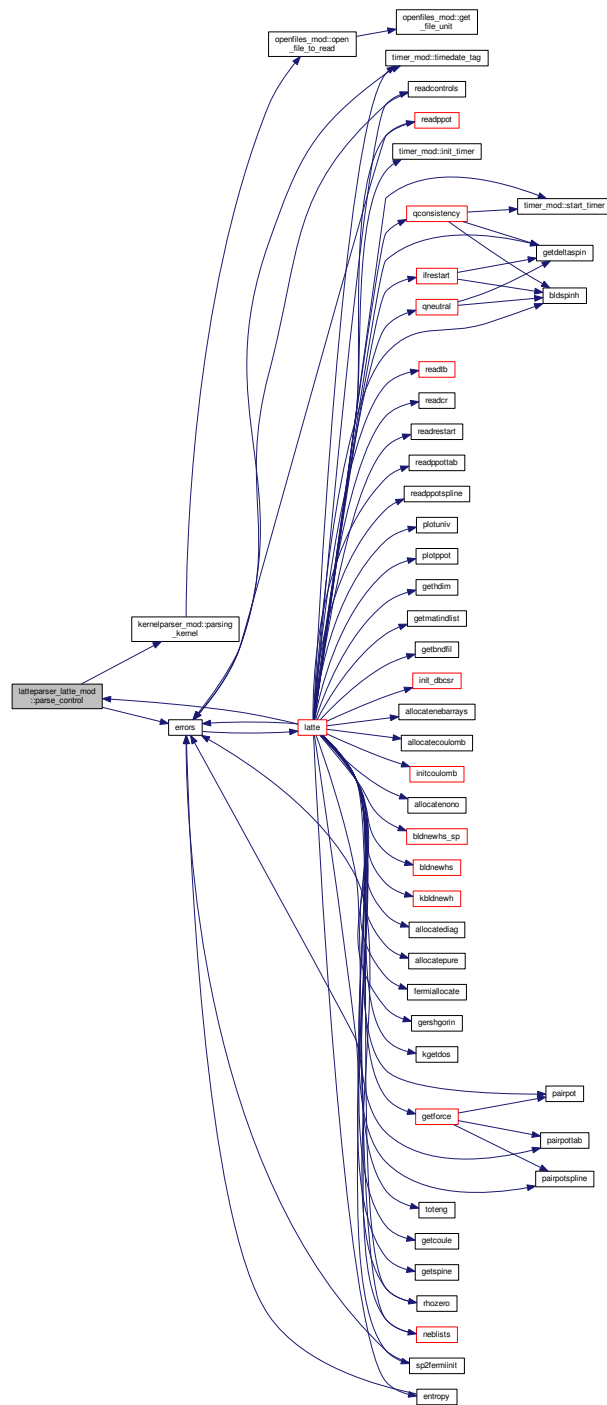
6.13.2 Function/Subroutine Documentation

6.13.2.1 subroutine, public latteparser_latte_mod::parse_control (character(len=*) *FILENAME*)

The parser for Latte General input variables.

Definition at line 122 of file [latteparser_latte_mod.f90](#).

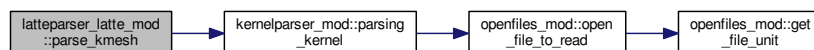
Here is the call graph for this function:



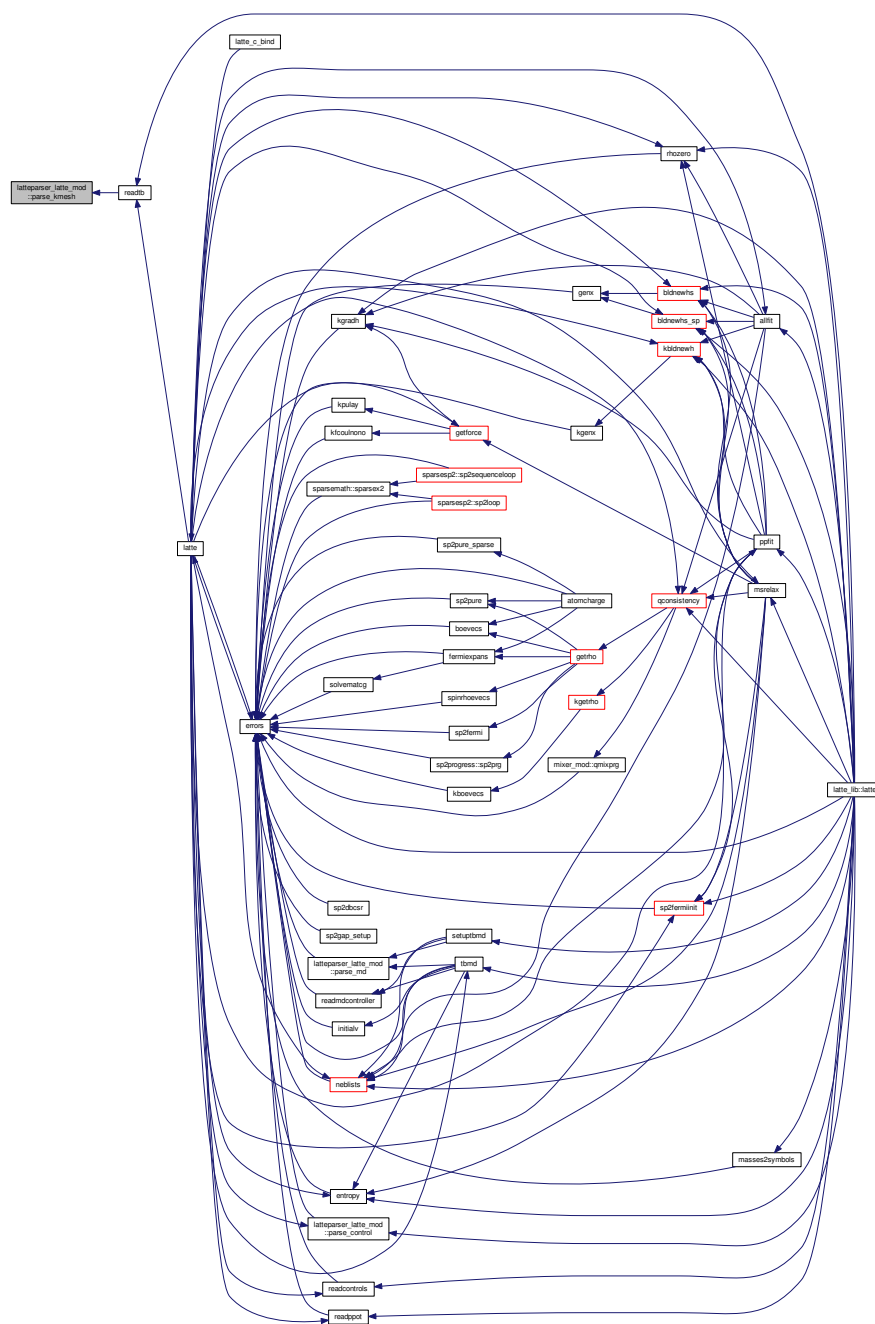
The parser for K Mesh input variables.



Here is the caller graph for this function:



Here is the caller graph for this function:

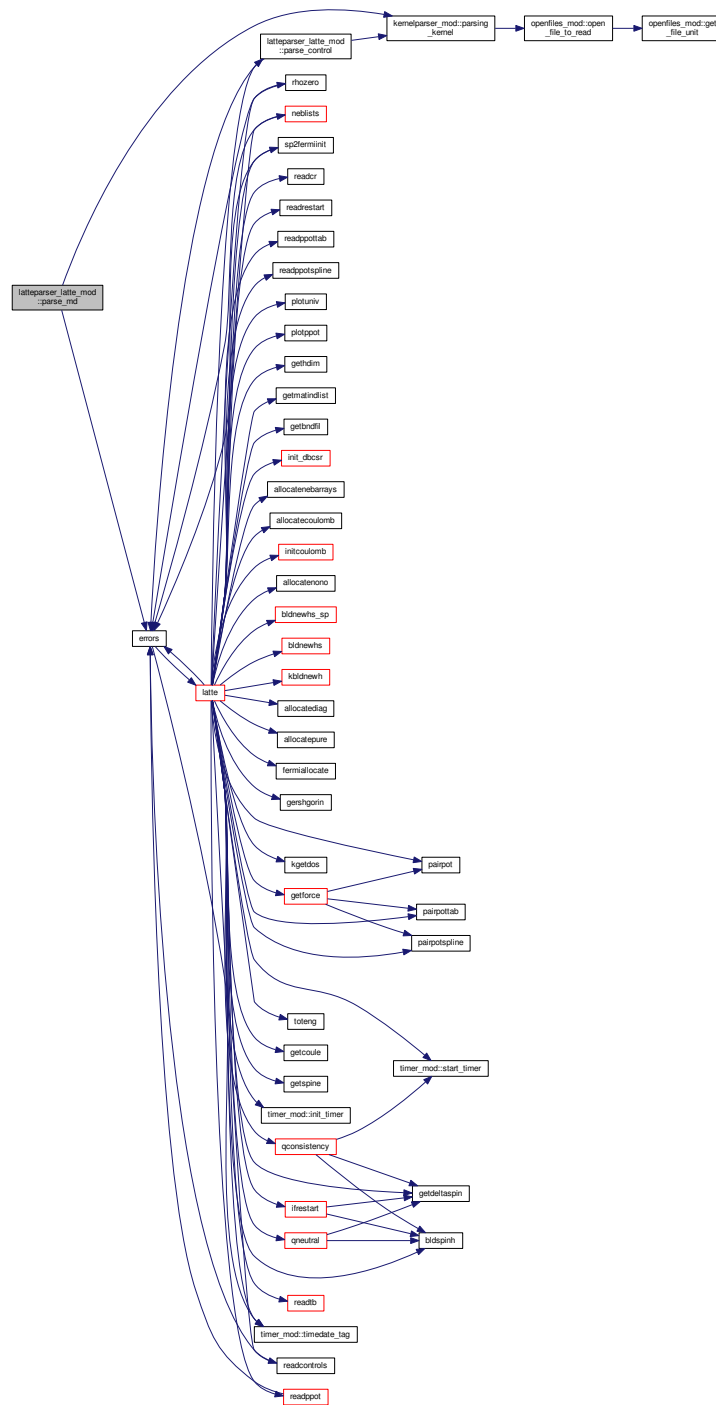


6.13.2.3 subroutine, public latteparser_latte_mod::parse_md (character(len=*) *FILENAME*)

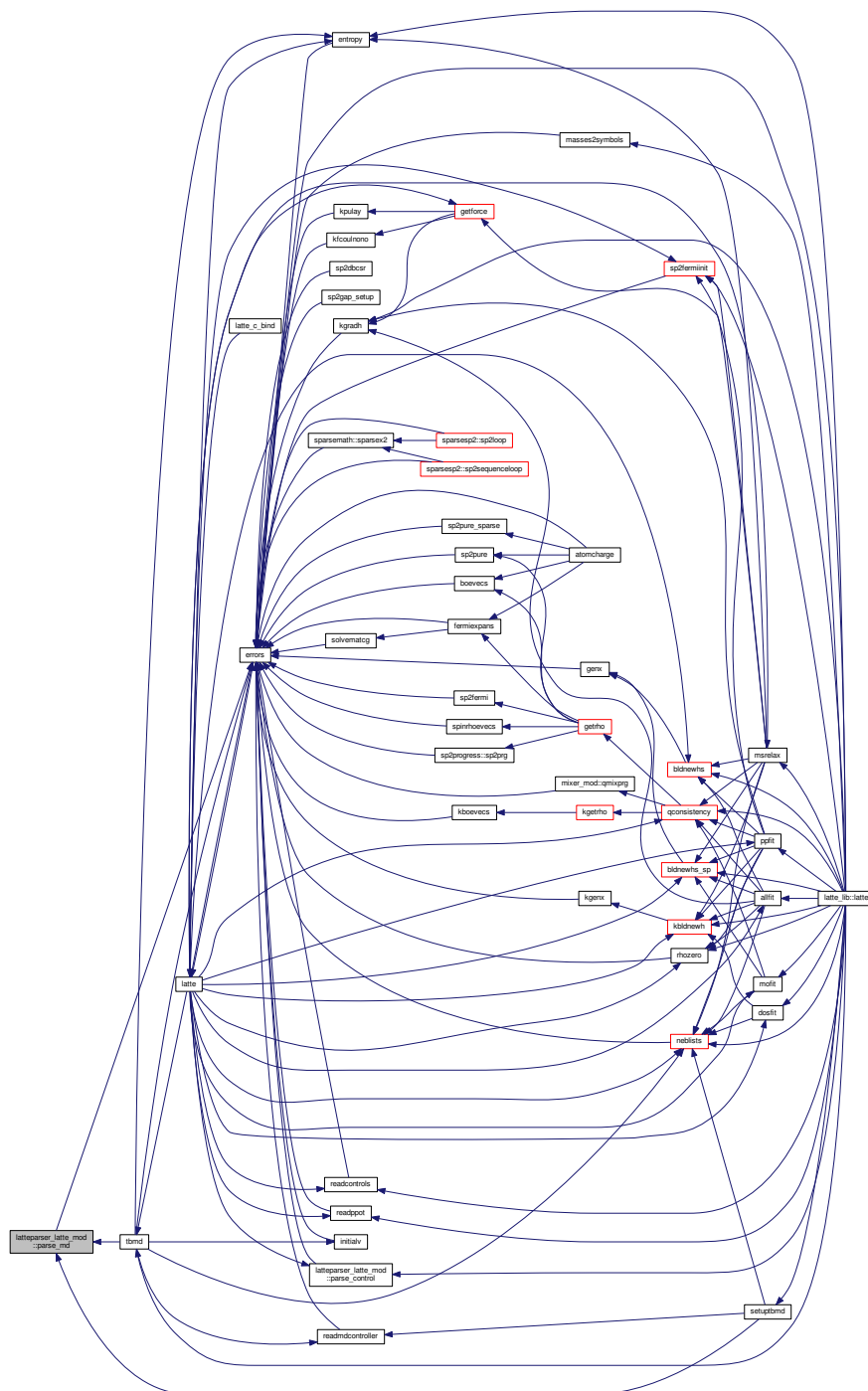
The parser for Latte General input variables.

Definition at line 514 of file [latteparser_latte_mod.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.13.3 Variable Documentation

6.13.3.1 type(latte_type), public latteparser_latte_mod::lt

Definition at line 113 of file latteparser_latte_mod.f90.

Functions/Subroutines

- ### 6.14.1 Function/Subroutine Documentation

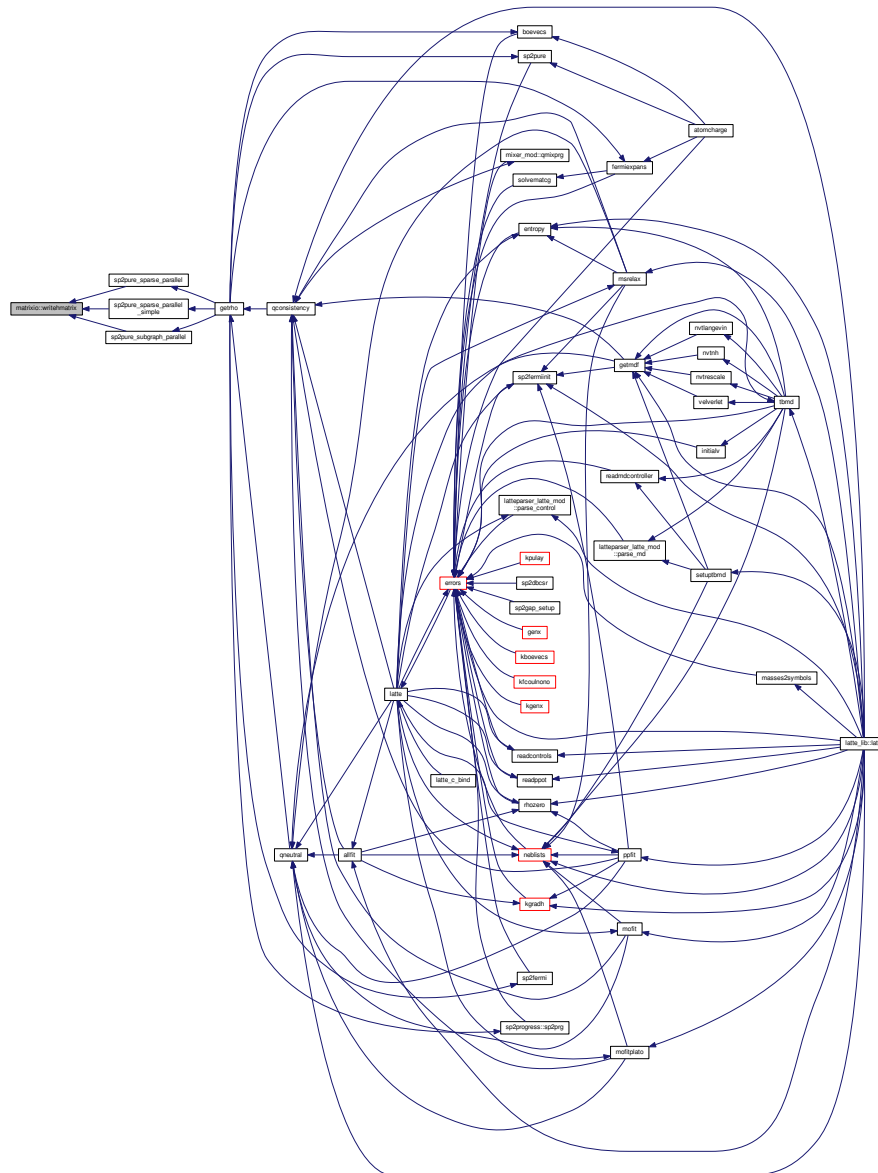
Definition at line 62 of file `matrixio.f90`.

[illegible]

6.14.1.2 subroutine `matrixio::writehmatrix` (integer, intent(in) *HSIZE*, integer, intent(in) *MSIZE*, real(latteprec), dimension(:, :), intent(in) *HARRAY*, integer, intent(in) *NITER*, integer, dimension(:), intent(in) *PVEC*)

Definition at line 32 of file [matrixio.f90](#).

Here is the caller graph for this function:



6.14.1.3 subroutine `matrixio::writemtx` (integer, intent(in) *ITER*, integer, intent(in) *HSIZE*, integer, dimension(:), intent(in) *II*, integer, dimension(:, :), intent(in) *JJ*, real(latteprec), dimension(:, :), intent(in) *VAL*)

Definition at line 85 of file [matrixio.f90](#).

6.15 mdarray Module Reference

Variables

- integer [aveper](#)
- real(latteprec) [avet](#)
- real(latteprec) [c0](#)
- real(latteprec) [consmot](#)
- integer [contiter](#)
- real(latteprec) [cumdt](#)
- real(latteprec) [dgamma](#)
- real(latteprec) [dt](#)
- real(latteprec) [dtzero](#)
- integer [dumpfreq](#)
- real(latteprec) [e0](#)
- real(latteprec), dimension(:), allocatable [ehist](#)
- integer [entropyiter](#)
- real(latteprec), dimension(:, :), allocatable [franprev](#)
- real(latteprec) [friction](#)
- real(latteprec) [gamma](#)
- integer [gethug](#)
- real(latteprec) [hg](#)
- integer [iset](#)
- real(latteprec), dimension(:), allocatable [mass](#)
- integer [maxiter](#)
- integer [npton](#)
- character(len=10) [nptype](#)
- integer [nvton](#)
- real(latteprec) [p0](#)
- real(latteprec), dimension(:), allocatable [phist](#)
- real(latteprec), dimension(:), allocatable [phistx](#)
- real(latteprec), dimension(:), allocatable [phisty](#)
- real(latteprec), dimension(:), allocatable [phistz](#)
- real(latteprec) [ptarget](#)
- character(len=10) [rndist](#)
- integer [rsfreq](#)
- character(len=10) [seedinit](#)
- integer [seedth](#)
- integer [setth](#)
- integer [shockdir](#)
- integer [shockon](#)
- integer [shockstart](#)
- integer [shockstop](#)
- real(latteprec) [temperature](#)
- integer [thermper](#)
- integer [thermrun](#)
- real(latteprec), dimension(:), allocatable [thist](#)
- integer [toinittemp](#)
- real(latteprec) [ttarget](#)
- real(latteprec) [tzero](#)
- real(latteprec) [uparticle](#)
- real(latteprec) [ushock](#)
- real(latteprec), dimension(:, :), allocatable [v](#)
- real(latteprec) [v0](#)
- real(latteprec), dimension(:), allocatable [vhist](#)
- integer [wrtfreq](#)

6.15.1 Variable Documentation

6.15.1.1 integer mdarray::aveper

Definition at line 35 of file [mdarray.f90](#).

6.15.1.2 real(latteprec) mdarray::avet

Definition at line 34 of file [mdarray.f90](#).

6.15.1.3 real(latteprec) mdarray::c0

Definition at line 43 of file [mdarray.f90](#).

6.15.1.4 real(latteprec) mdarray::consmot

Definition at line 63 of file [mdarray.f90](#).

6.15.1.5 integer mdarray::contiter

Definition at line 35 of file [mdarray.f90](#).

6.15.1.6 real(latteprec) mdarray::cumdt

Definition at line 56 of file [mdarray.f90](#).

6.15.1.7 real(latteprec) mdarray::dgamma

Definition at line 62 of file [mdarray.f90](#).

6.15.1.8 real(latteprec) mdarray::dt

Definition at line 33 of file [mdarray.f90](#).

6.15.1.9 real(latteprec) mdarray::dtzero

Definition at line 33 of file [mdarray.f90](#).

6.15.1.10 integer mdarray::dumpfreq

Definition at line 36 of file [mdarray.f90](#).

6.15.1.11 `real(latteprec) mdarray::e0`

Definition at line 34 of file [mdarray.f90](#).

6.15.1.12 `real(latteprec), dimension(:), allocatable mdarray::ehist`

Definition at line 31 of file [mdarray.f90](#).

6.15.1.13 `integer mdarray::entropyiter`

Definition at line 37 of file [mdarray.f90](#).

6.15.1.14 `real(latteprec), dimension(:, :), allocatable mdarray::franprev`

Definition at line 52 of file [mdarray.f90](#).

6.15.1.15 `real(latteprec) mdarray::friction`

Definition at line 48 of file [mdarray.f90](#).

6.15.1.16 `real(latteprec) mdarray::gamma`

Definition at line 62 of file [mdarray.f90](#).

6.15.1.17 `integer mdarray::gethug`

Definition at line 35 of file [mdarray.f90](#).

6.15.1.18 `real(latteprec) mdarray::hg`

Definition at line 34 of file [mdarray.f90](#).

6.15.1.19 `integer mdarray::iset`

Definition at line 29 of file [mdarray.f90](#).

6.15.1.20 `real(latteprec), dimension(:), allocatable mdarray::mass`

Definition at line 30 of file [mdarray.f90](#).

6.15.1.21 integer mdarray::maxiter

Definition at line 36 of file [mdarray.f90](#).

6.15.1.22 integer mdarray::npton

Definition at line 35 of file [mdarray.f90](#).

6.15.1.23 character(len=10) mdarray::npttype

Definition at line 38 of file [mdarray.f90](#).

6.15.1.24 integer mdarray::nvton

Definition at line 35 of file [mdarray.f90](#).

6.15.1.25 real(latteprec) mdarray::p0

Definition at line 34 of file [mdarray.f90](#).

6.15.1.26 real(latteprec), dimension(:), allocatable mdarray::phist

Definition at line 31 of file [mdarray.f90](#).

6.15.1.27 real(latteprec), dimension(:), allocatable mdarray::phistx

Definition at line 32 of file [mdarray.f90](#).

6.15.1.28 real(latteprec), dimension(:), allocatable mdarray::phisty

Definition at line 32 of file [mdarray.f90](#).

6.15.1.29 real(latteprec), dimension(:), allocatable mdarray::phistz

Definition at line 32 of file [mdarray.f90](#).

6.15.1.30 real(latteprec) mdarray::ptarget

Definition at line 33 of file [mdarray.f90](#).

6.15.1.31 `character(len=10) mdarray::rndist`

Definition at line 38 of file [mdarray.f90](#).

6.15.1.32 `integer mdarray::rsfreq`

Definition at line 36 of file [mdarray.f90](#).

6.15.1.33 `character(len=10) mdarray::seedinit`

Definition at line 38 of file [mdarray.f90](#).

6.15.1.34 `integer mdarray::seedth`

Definition at line 46 of file [mdarray.f90](#).

6.15.1.35 `integer mdarray::setth`

Definition at line 47 of file [mdarray.f90](#).

6.15.1.36 `integer mdarray::shockdir`

Definition at line 42 of file [mdarray.f90](#).

6.15.1.37 `integer mdarray::shockon`

Definition at line 42 of file [mdarray.f90](#).

6.15.1.38 `integer mdarray::shockstart`

Definition at line 42 of file [mdarray.f90](#).

6.15.1.39 `integer mdarray::shockstop`

Definition at line 42 of file [mdarray.f90](#).

6.15.1.40 `real(latteprec) mdarray::temperature`

Definition at line 33 of file [mdarray.f90](#).

6.15.1.41 integer mdarray::thermper

Definition at line 36 of file [mdarray.f90](#).

6.15.1.42 integer mdarray::thermrun

Definition at line 36 of file [mdarray.f90](#).

6.15.1.43 real(latteprec), dimension(:), allocatable mdarray::thist

Definition at line 31 of file [mdarray.f90](#).

6.15.1.44 integer mdarray::tointtemp

Definition at line 35 of file [mdarray.f90](#).

6.15.1.45 real(latteprec) mdarray::ttarget

Definition at line 33 of file [mdarray.f90](#).

6.15.1.46 real(latteprec) mdarray::tzero

Definition at line 33 of file [mdarray.f90](#).

6.15.1.47 real(latteprec) mdarray::uparticle

Definition at line 43 of file [mdarray.f90](#).

6.15.1.48 real(latteprec) mdarray::ushock

Definition at line 43 of file [mdarray.f90](#).

6.15.1.49 real(latteprec), dimension(:, :), allocatable mdarray::v

Definition at line 30 of file [mdarray.f90](#).

6.15.1.50 real(latteprec) mdarray::v0

Definition at line 34 of file [mdarray.f90](#).

6.15.1.51 `real(latteprec), dimension(:), allocatable mdarray::vhist`

Definition at line 31 of file [mdarray.f90](#).

6.15.1.52 `integer mdarray::wrtfreq`

Definition at line 36 of file [mdarray.f90](#).

6.16 mixer_mod Module Reference

To implement mixing schemes from the progress library.

Functions/Subroutines

- subroutine, public [qmixprg](#) (PITER)

Variables

- `real(latteprec), dimension(:, :), allocatable, public dqin`
- `real(latteprec), dimension(:, :), allocatable, public dqout`
- `logical, public mixinit = .FALSE.`
- `type(mx_type), public mx`
- `real(latteprec), public scferror`

6.16.1 Detailed Description

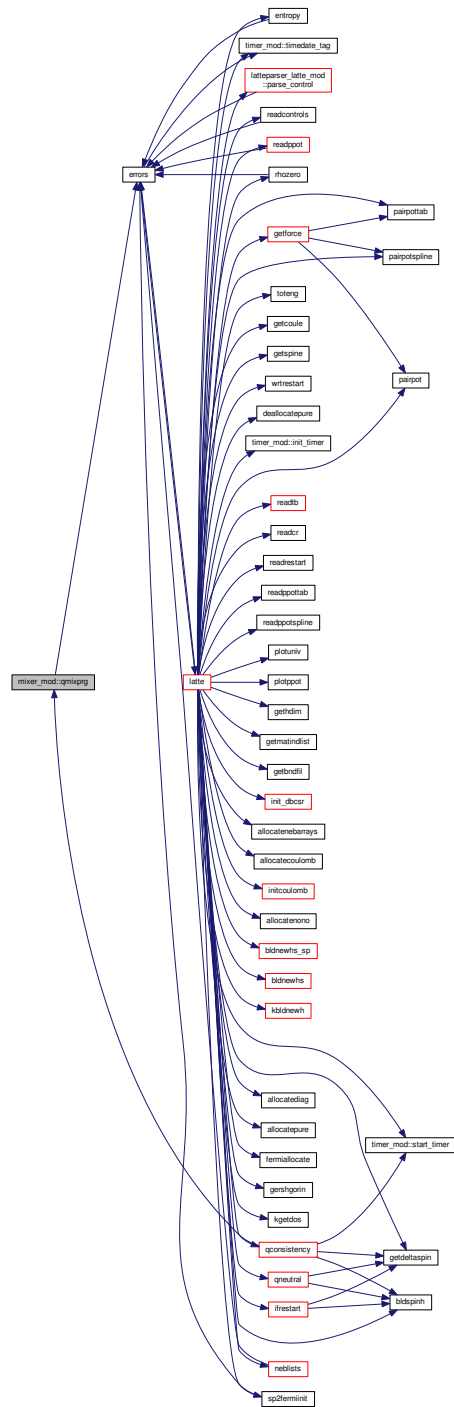
To implement mixing schemes from the progress library.

6.16.2 Function/Subroutine Documentation

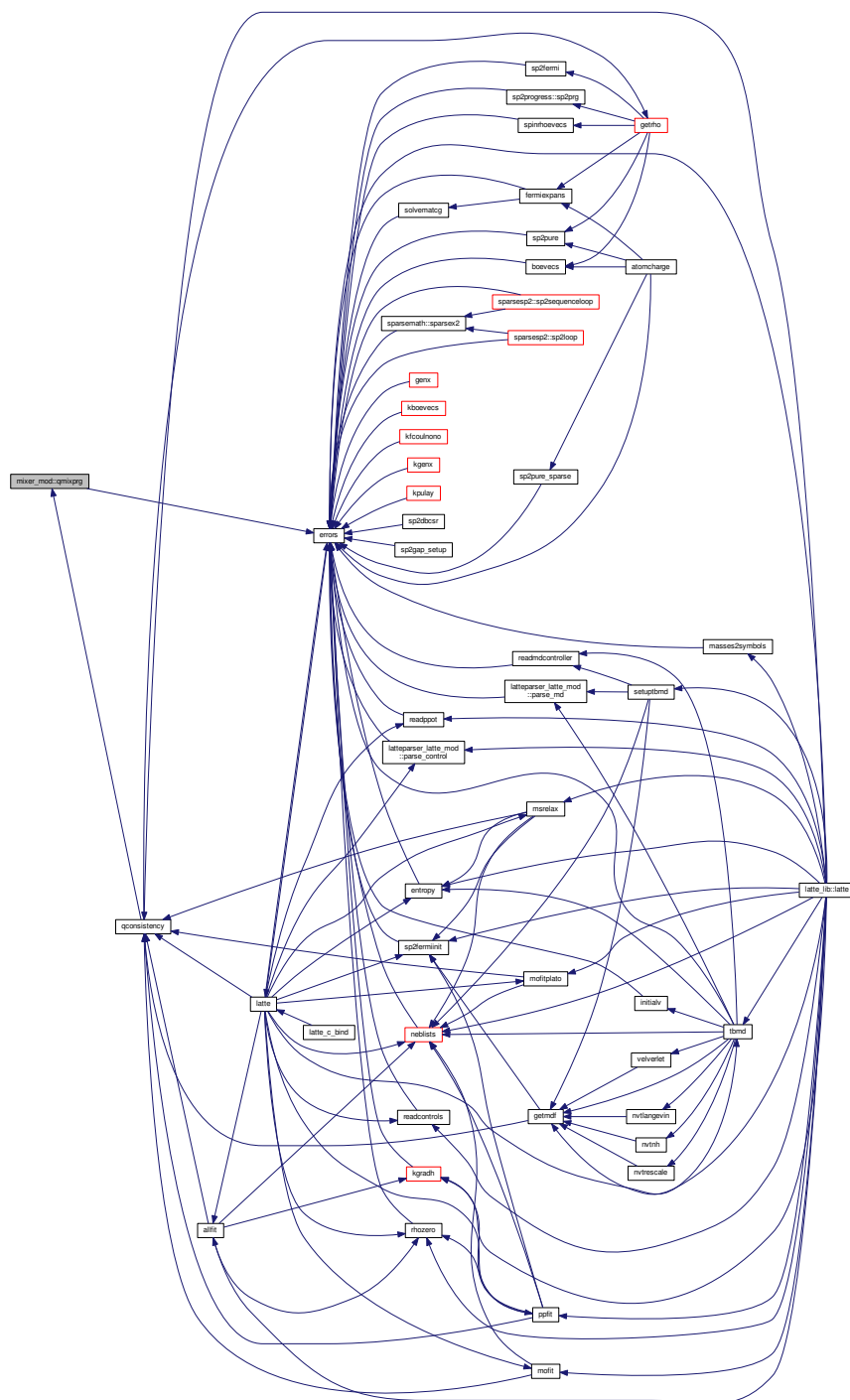
6.16.2.1 subroutine, public `mixer_mod::qmixprg (integer, intent(in) PITER)`

Definition at line 46 of file [mixer_mod.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.16.3 Variable Documentation

6.16.3.1 `real(latteprec)`, `dimension(:, :)`, `allocatable`, `public mixer_mod::dqin`

Definition at line 39 of file `mixer_mod.f90`.

6.16.3.2 `real(latteprec), dimension(:, :), allocatable, public mixer_mod::dqout`

Definition at line 39 of file [mixer_mod.f90](#).

6.16.3.3 `logical, public mixer_mod::mixinit = .FALSE.`

Definition at line 38 of file [mixer_mod.f90](#).

6.16.3.4 `type(mx_type), public mixer_mod::mx`

Definition at line 41 of file [mixer_mod.f90](#).

6.16.3.5 `real(latteprec), public mixer_mod::scferror`

Definition at line 40 of file [mixer_mod.f90](#).

6.17 myprecision Module Reference

Variables

- `real(latteprec)`, parameter `eight` = 8.0
- `real(latteprec)`, parameter `eleven` = 11.0
- `real(latteprec)`, parameter `fifteen` = 15.0
- `real(latteprec)`, parameter `five` = 5.0
- `real(latteprec)`, parameter `fortyeight` = 48.0
- `real(latteprec)`, parameter `four` = 4.0
- `real(latteprec)`, parameter `fourteen` = 14.0
- `real(latteprec)`, parameter `half` = 0.5
- `complex(latteprec)`, parameter `im` = (ZERO, ONE)
- `integer`, parameter `latteprec` = 8
- `real(latteprec)`, parameter `minusone` = -1.0
- `real(latteprec)`, parameter `nine` = 9.0
- `real(latteprec)`, parameter `one` = 1.0
- `real(latteprec)`, parameter `quarter` = 0.25
- `complex(latteprec)`, parameter `re` = (ONE, ZERO)
- `real(latteprec)`, parameter `seven` = 7.0
- `real(latteprec)`, parameter `six` = 6.0
- `real(latteprec)`, parameter `sixteen` = 16.0
- `real(latteprec)`, parameter `sqrt2` = SQRT(2.0D0)
- `real(latteprec)`, parameter `ten` = 10.0
- `real(latteprec)`, parameter `third` = 1.0D0/3.0D0
- `real(latteprec)`, parameter `thousand` = 1000.0
- `real(latteprec)`, parameter `three` = 3.0
- `real(latteprec)`, parameter `threequart` = 0.75
- `real(latteprec)`, parameter `twelve` = 12.0
- `real(latteprec)`, parameter `twenty` = 20.0
- `real(latteprec)`, parameter `twentyfour` = 24.0
- `real(latteprec)`, parameter `twentysix` = 26.0
- `real(latteprec)`, parameter `two` = 2.0
- `real(latteprec)`, parameter `zero` = 0.0

6.17.1 Variable Documentation

6.17.1.1 `real(latteprec)`, parameter `myprecision::eight = 8.0`

Definition at line 45 of file [myprecision.f90](#).

6.17.1.2 `real(latteprec)`, parameter `myprecision::eleven = 11.0`

Definition at line 46 of file [myprecision.f90](#).

6.17.1.3 `real(latteprec)`, parameter `myprecision::fifteen = 15.0`

Definition at line 47 of file [myprecision.f90](#).

6.17.1.4 `real(latteprec)`, parameter `myprecision::five = 5.0`

Definition at line 44 of file [myprecision.f90](#).

6.17.1.5 `real(latteprec)`, parameter `myprecision::fortyeight = 48.0`

Definition at line 50 of file [myprecision.f90](#).

6.17.1.6 `real(latteprec)`, parameter `myprecision::four = 4.0`

Definition at line 44 of file [myprecision.f90](#).

6.17.1.7 `real(latteprec)`, parameter `myprecision::fourteen = 14.0`

Definition at line 47 of file [myprecision.f90](#).

6.17.1.8 `real(latteprec)`, parameter `myprecision::half = 0.5`

Definition at line 51 of file [myprecision.f90](#).

6.17.1.9 `complex(latteprec)`, parameter `myprecision::im = (ZERO, ONE)`

Definition at line 55 of file [myprecision.f90](#).

6.17.1.10 `integer` parameter `myprecision::latteprec = 8`

Definition at line 33 of file [myprecision.f90](#).

6.17.1.11 `real(latteprec)`, parameter `myprecision::minusone = -1.0`

Definition at line 53 of file [myprecision.f90](#).

6.17.1.12 `real(latteprec)`, parameter `myprecision::nine = 9.0`

Definition at line 46 of file [myprecision.f90](#).

6.17.1.13 `real(latteprec)`, parameter `myprecision::one = 1.0`

Definition at line 43 of file [myprecision.f90](#).

6.17.1.14 `real(latteprec)`, parameter `myprecision::quarter = 0.25`

Definition at line 52 of file [myprecision.f90](#).

6.17.1.15 `complex(latteprec)`, parameter `myprecision::re = (ONE, ZERO)`

Definition at line 55 of file [myprecision.f90](#).

6.17.1.16 `real(latteprec)`, parameter `myprecision::seven = 7.0`

Definition at line 45 of file [myprecision.f90](#).

6.17.1.17 `real(latteprec)`, parameter `myprecision::six = 6.0`

Definition at line 45 of file [myprecision.f90](#).

6.17.1.18 `real(latteprec)`, parameter `myprecision::sixteen = 16.0`

Definition at line 48 of file [myprecision.f90](#).

6.17.1.19 `real(latteprec)`, parameter `myprecision::sqrt2 = SQRT(2.0D0)`

Definition at line 54 of file [myprecision.f90](#).

6.17.1.20 `real(latteprec)`, parameter `myprecision::ten = 10.0`

Definition at line 46 of file [myprecision.f90](#).

6.17.1.21 `real(latteprec)`, parameter `myprecision::third = 1.0D0/3.0D0`

Definition at line 51 of file [myprecision.f90](#).

6.17.1.22 `real(latteprec)`, parameter `myprecision::thousand = 1000.0`

Definition at line 53 of file [myprecision.f90](#).

6.17.1.23 `real(latteprec)`, parameter `myprecision::three = 3.0`

Definition at line 44 of file [myprecision.f90](#).

6.17.1.24 `real(latteprec)`, parameter `myprecision::threequart = 0.75`

Definition at line 52 of file [myprecision.f90](#).

6.17.1.25 `real(latteprec)`, parameter `myprecision::twelve = 12.0`

Definition at line 47 of file [myprecision.f90](#).

6.17.1.26 `real(latteprec)`, parameter `myprecision::twenty = 20.0`

Definition at line 48 of file [myprecision.f90](#).

6.17.1.27 `real(latteprec)`, parameter `myprecision::twentyfour = 24.0`

Definition at line 49 of file [myprecision.f90](#).

6.17.1.28 `real(latteprec)`, parameter `myprecision::twentysix = 26.0`

Definition at line 49 of file [myprecision.f90](#).

6.17.1.29 `real(latteprec)`, parameter `myprecision::two = 2.0`

Definition at line 43 of file [myprecision.f90](#).

6.17.1.30 `real(latteprec)`, parameter `myprecision::zero = 0.0`

Definition at line 43 of file [myprecision.f90](#).

6.18 neblistarray Module Reference

Variables

- integer [allocest](#)
- integer [dimlist](#)
- real(latteprec) [maxcut](#)
- integer [maxdimcoul](#)
- integer [maxdimpp](#)
- integer [maxdimtb](#)
- integer, dimension(:), allocatable [molid](#)
- integer, dimension(:, :, :), allocatable [nebcoul](#)
- integer, dimension(:, :, :), allocatable [nebpp](#)
- integer, dimension(:, :, :), allocatable [nebtb](#)
- integer [nomol](#)
- real(latteprec) [ppmax2](#)
- real(latteprec) [rcutcoul2](#)
- real(latteprec) [rcuttb2](#)
- real(latteprec) [skin](#)
- integer, dimension(:), allocatable [totnebcoul](#)
- integer, dimension(:), allocatable [totnebpp](#)
- integer, dimension(:), allocatable [totnebtb](#)
- integer [udneigh](#)

6.18.1 Variable Documentation

6.18.1.1 integer neblistarray::allocest

Definition at line 34 of file [neblistarray.f90](#).

6.18.1.2 integer neblistarray::dimlist

Definition at line 34 of file [neblistarray.f90](#).

6.18.1.3 real(latteprec) neblistarray::maxcut

Definition at line 36 of file [neblistarray.f90](#).

6.18.1.4 integer neblistarray::maxdimcoul

Definition at line 33 of file [neblistarray.f90](#).

6.18.1.5 integer neblistarray::maxdimpp

Definition at line 33 of file [neblistarray.f90](#).

6.18.1.6 integer neblistarray::maxdimtb

Definition at line 33 of file [neblistarray.f90](#).

6.18.1.7 integer, dimension(:), allocatable neblistarray::molid

Definition at line 31 of file [neblistarray.f90](#).

6.18.1.8 integer, dimension(:,:), allocatable neblistarray::nebcoul

Definition at line 30 of file [neblistarray.f90](#).

6.18.1.9 integer, dimension(:,:), allocatable neblistarray::nebpp

Definition at line 30 of file [neblistarray.f90](#).

6.18.1.10 integer, dimension(:,:), allocatable neblistarray::nebtb

Definition at line 30 of file [neblistarray.f90](#).

6.18.1.11 integer neblistarray::nomol

Definition at line 32 of file [neblistarray.f90](#).

6.18.1.12 real(latteprec) neblistarray::ppmax2

Definition at line 36 of file [neblistarray.f90](#).

6.18.1.13 real(latteprec) neblistarray::rcutcoul2

Definition at line 36 of file [neblistarray.f90](#).

6.18.1.14 real(latteprec) neblistarray::rcuttb2

Definition at line 36 of file [neblistarray.f90](#).

6.18.1.15 real(latteprec) neblistarray::skin

Definition at line 35 of file [neblistarray.f90](#).

6.18.1.16 integer, dimension(:), allocatable neblistarray::totnebcoul

Definition at line 29 of file [neblistarray.f90](#).

6.18.1.17 integer, dimension(:), allocatable neblistarray::totnebpp

Definition at line 29 of file [neblistarray.f90](#).

6.18.1.18 integer, dimension(:), allocatable neblistarray::totnebtb

Definition at line 29 of file [neblistarray.f90](#).

6.18.1.19 integer neblistarray::udneigh

Definition at line 32 of file [neblistarray.f90](#).

6.19 nonoarray Module Reference

Variables

- real(latteprec), dimension(:), allocatable [hjj](#)
- real(latteprec), dimension(:), allocatable [nono_evals](#)
- integer, dimension(:), allocatable [nono_iwork](#)
- integer [nono_liwork](#)
- integer [nono_lwork](#)
- real(latteprec), dimension(:), allocatable [nono_work](#)
- real(latteprec), dimension(:, :), allocatable [nonotmp](#)
- integer [nonzero](#)
- real(latteprec), dimension(:, :), allocatable [orthoh](#)
- real(latteprec), dimension(:, :), allocatable [orthohdown](#)
- real(latteprec), dimension(:, :), allocatable [orthohup](#)
- real(latteprec), dimension(:, :), allocatable [sh2](#)
- real(latteprec), dimension(:, :), allocatable [smat](#)
- real(latteprec), dimension(:, :), allocatable [spintmp](#)
- real(latteprec), dimension(:, :), allocatable [umat](#)
- real(latteprec), dimension(:, :), allocatable [x2hrho](#)
- real(latteprec), dimension(:, :), allocatable [xmat](#)

6.19.1 Variable Documentation

6.19.1.1 real(latteprec), dimension(:), allocatable nonoarray::hjj

Definition at line 44 of file [nonoarray.f90](#).

6.19.1.2 `real(latteprec), dimension(:), allocatable nonoarray::nono_evals`

Definition at line 40 of file [nonoarray.f90](#).

6.19.1.3 `integer, dimension(:), allocatable nonoarray::nono_iwork`

Definition at line 38 of file [nonoarray.f90](#).

6.19.1.4 `integer nonoarray::nono_liwork`

Definition at line 36 of file [nonoarray.f90](#).

6.19.1.5 `integer nonoarray::nono_lwork`

Definition at line 36 of file [nonoarray.f90](#).

6.19.1.6 `real(latteprec), dimension(:), allocatable nonoarray::nono_work`

Definition at line 39 of file [nonoarray.f90](#).

6.19.1.7 `real(latteprec), dimension(:, :), allocatable nonoarray::nonotmp`

Definition at line 41 of file [nonoarray.f90](#).

6.19.1.8 `integer nonoarray::nonzero`

Definition at line 37 of file [nonoarray.f90](#).

6.19.1.9 `real(latteprec), dimension(:, :), allocatable nonoarray::orthoh`

Definition at line 42 of file [nonoarray.f90](#).

6.19.1.10 `real(latteprec), dimension(:, :), allocatable nonoarray::orthohdown`

Definition at line 43 of file [nonoarray.f90](#).

6.19.1.11 `real(latteprec), dimension(:, :), allocatable nonoarray::orthohup`

Definition at line 43 of file [nonoarray.f90](#).

6.19.1.12 `real(latteprec), dimension(:, :), allocatable nonoarray::sh2`

Definition at line 44 of file [nonoarray.f90](#).

6.19.1.13 `real(latteprec), dimension(:, :), allocatable nonoarray::smat`

Definition at line 41 of file [nonoarray.f90](#).

6.19.1.14 `real(latteprec), dimension(:, :), allocatable nonoarray::spintmp`

Definition at line 48 of file [nonoarray.f90](#).

6.19.1.15 `real(latteprec), dimension(:, :), allocatable nonoarray::umat`

Definition at line 40 of file [nonoarray.f90](#).

6.19.1.16 `real(latteprec), dimension(:, :), allocatable nonoarray::x2hrho`

Definition at line 48 of file [nonoarray.f90](#).

6.19.1.17 `real(latteprec), dimension(:, :), allocatable nonoarray::xmat`

Definition at line 41 of file [nonoarray.f90](#).

6.20 openfiles_mod Module Reference

Module to handle input output files.

Functions/Subroutines

- integer function, public [get_file_unit](#) (IO_MAX)
Returns a unit number that is not in use.
- subroutine, public [open_file](#) (IO, NAME)
Opens a file to write.
- subroutine, public [open_file_to_read](#) (IO, NAME)
Opens a file to read.

6.20.1 Detailed Description

Module to handle input output files.

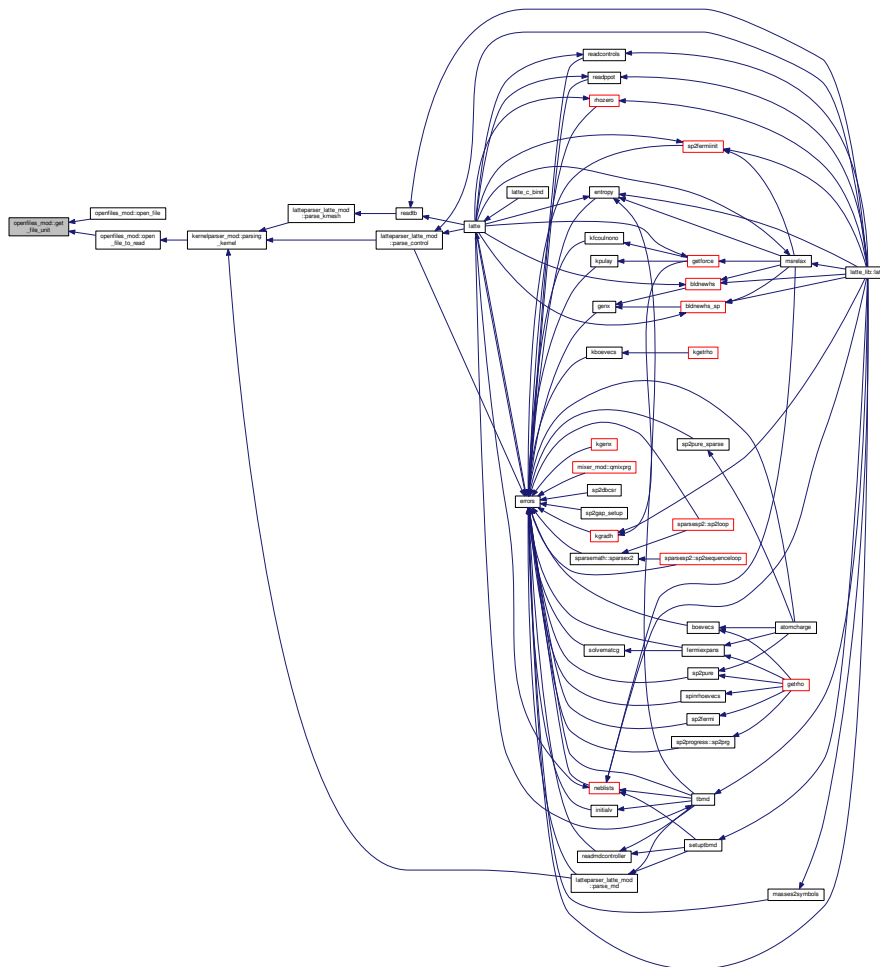
6.20.2 Function/Subroutine Documentation

6.20.2.1 integer function, public `openfiles_mod::get_file_unit (integer IO_MAX)`

Returns a unit number that is not in use.

<i>io_max</i>	Maximum units to search.
<i>get_file_unit</i>	Unit return to use for the file.

Here is the caller graph for this function:

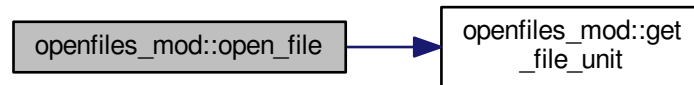


Opens a file to write.

<i>io</i>	Unit for the file.
<i>name</i>	Name of the file.

Generated by Doxygen

Here is the call graph for this function:



6.20.2.3 subroutine, public openfiles_mod::open_file_to_read (integer *IO*, character(len=*) *NAME*)

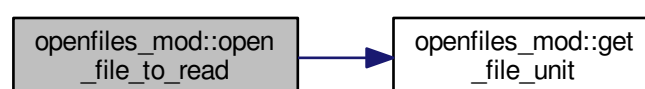
Opens a file to read.

Parameters

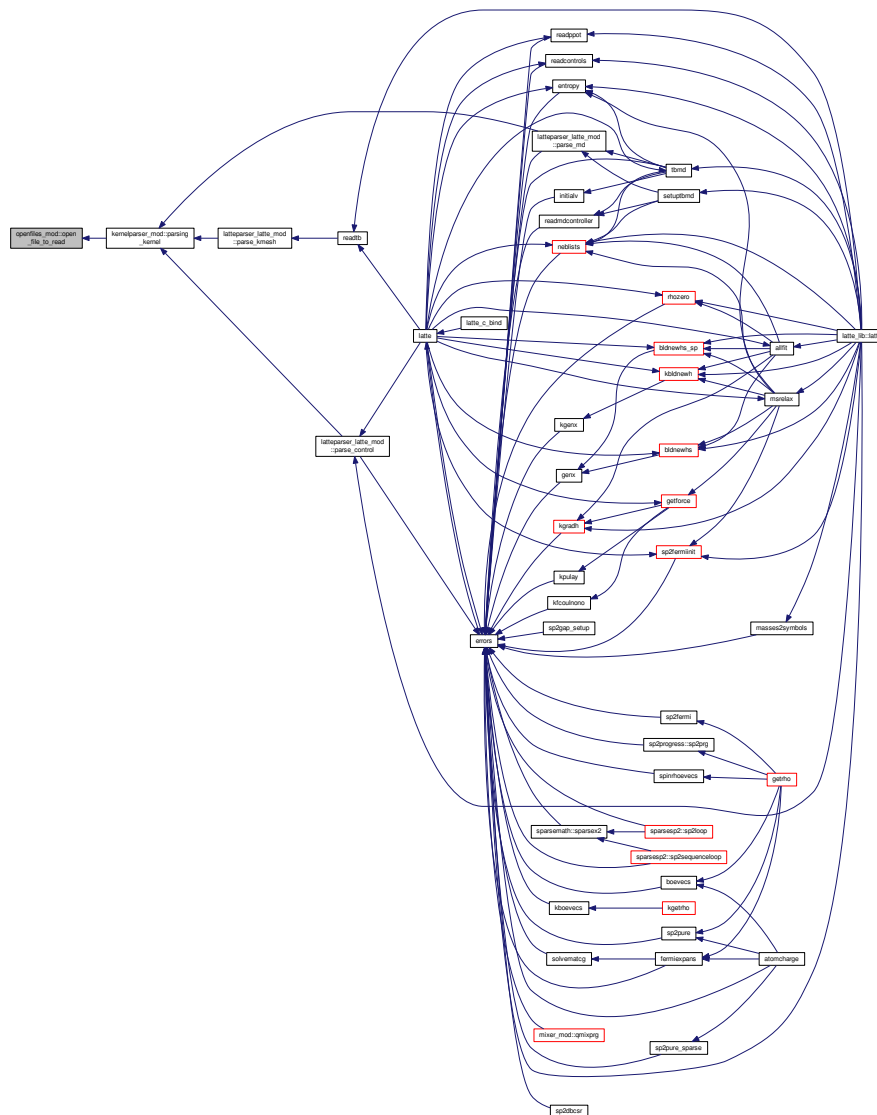
<i>io</i>	Unit for the file.
<i>name</i>	Name of the file.

Definition at line 53 of file [openfiles_mod.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.21 ppotarray Module Reference

Variables

- integer [nopps](#)
- real(latteprec), dimension(:,:), allocatable [potcoef](#)
- real(latteprec), dimension(:,:), allocatable [ppak](#)
- character(len=2), dimension(:), allocatable [ppele1](#)
- character(len=2), dimension(:), allocatable [ppele2](#)
- integer, dimension(:), allocatable [ppnk](#)
- real(latteprec), dimension(:,:), allocatable [ppr](#)
- real(latteprec), dimension(:,:), allocatable [pprk](#)
- real(latteprec), dimension(:,:), allocatable [ppspl](#)
- integer, dimension(:), allocatable [pptablength](#)
- real(latteprec), dimension(:,:), allocatable [ppval](#)

6.21.1 Variable Documentation

6.21.1.1 integer ppotarray::nopps

Definition at line 30 of file [ppotarray.f90](#).

6.21.1.2 real(latteprec), dimension(:,:), allocatable ppotarray::potcoef

Definition at line 32 of file [ppotarray.f90](#).

6.21.1.3 real(latteprec), dimension(:,:), allocatable ppotarray::ppak

Definition at line 36 of file [ppotarray.f90](#).

6.21.1.4 character(len=2), dimension(:), allocatable ppotarray::ppele1

Definition at line 33 of file [ppotarray.f90](#).

6.21.1.5 character(len=2), dimension(:), allocatable ppotarray::ppele2

Definition at line 33 of file [ppotarray.f90](#).

6.21.1.6 integer, dimension(:), allocatable ppotarray::ppnk

Definition at line 31 of file [ppotarray.f90](#).

6.21.1.7 real(latteprec), dimension(:,:), allocatable ppotarray::ppr

Definition at line 34 of file [ppotarray.f90](#).

6.21.1.8 real(latteprec), dimension(:,:), allocatable ppotarray::pprk

Definition at line 36 of file [ppotarray.f90](#).

6.21.1.9 real(latteprec), dimension(:,:), allocatable ppotarray::ppspl

Definition at line 34 of file [ppotarray.f90](#).

6.21.1.10 integer, dimension(:), allocatable ppotarray::pptablength

Definition at line 31 of file [ppotarray.f90](#).

6.21.1.11 `real(latteprec), dimension(:, :), allocatable ppotarray::ppval`

Definition at line 34 of file [ppotarray.f90](#).

6.22 purearray Module Reference

Variables

- `real(latteprec)` [beta0](#)
- `integer` [maxdim](#)
- `integer` [nr_sp2_iter](#)
- `integer, dimension(100)` [pp](#)
- `integer, dimension(:, :), allocatable` [signlist](#)
- `real(latteprec), dimension(:, :), allocatable` [twoxx2](#)
- `real(latteprec), dimension(100)` [vv](#)
- `real(latteprec), dimension(:, :), allocatable` [x2](#)
- `real(latteprec), dimension(:, :), allocatable` [x2down](#)
- `real(latteprec), dimension(:, :), allocatable` [x2up](#)

6.22.1 Variable Documentation

6.22.1.1 `real(latteprec) purearray::beta0`

Definition at line 31 of file [purearray.f90](#).

6.22.1.2 `integer purearray::maxdim`

Definition at line 29 of file [purearray.f90](#).

6.22.1.3 `integer purearray::nr_sp2_iter`

Definition at line 37 of file [purearray.f90](#).

6.22.1.4 `integer, dimension(100) purearray::pp`

Definition at line 37 of file [purearray.f90](#).

6.22.1.5 `integer, dimension(:, :), allocatable purearray::signlist`

Definition at line 30 of file [purearray.f90](#).

6.22.1.6 `real(latteprec), dimension(:, :), allocatable purearray::twoxx2`

Definition at line 32 of file [purearray.f90](#).

6.22.1.7 `real(latteprec), dimension(100) purearray::vv`

Definition at line 38 of file [purearray.f90](#).

6.22.1.8 `real(latteprec), dimension(:, :), allocatable purearray::x2`

Definition at line 32 of file [purearray.f90](#).

6.22.1.9 `real(latteprec), dimension(:, :), allocatable purearray::x2down`

Definition at line 33 of file [purearray.f90](#).

6.22.1.10 `real(latteprec), dimension(:, :), allocatable purearray::x2up`

Definition at line 33 of file [purearray.f90](#).

6.23 relaxcommon Module Reference

Variables

- `real(latteprec), dimension(:, :), allocatable d1`
- `real(latteprec) mxrlx`
- `real(latteprec), dimension(:, :), allocatable oldd`
- `real(latteprec), dimension(:, :), allocatable oldf`
- `real(latteprec) prevf`
- `real(latteprec) relconst`
- `character(len=2) reltype`
- `real(latteprec) rxftol`

6.23.1 Variable Documentation

6.23.1.1 `real(latteprec), dimension(:, :), allocatable relaxcommon::d1`

Definition at line 36 of file [relaxcommon.f90](#).

6.23.1.2 `real(latteprec) relaxcommon::mxrlx`

Definition at line 30 of file [relaxcommon.f90](#).

6.23.1.3 `real(latteprec), dimension(:, :), allocatable relaxcommon::oldd`

Definition at line 36 of file [relaxcommon.f90](#).

6.23.1.4 `real(latteprec), dimension(:,:), allocatable relaxcommon::oldf`

Definition at line 36 of file [relaxcommon.f90](#).

6.23.1.5 `real(latteprec) relaxcommon::prevf`

Definition at line 31 of file [relaxcommon.f90](#).

6.23.1.6 `real(latteprec) relaxcommon::relconst`

Definition at line 31 of file [relaxcommon.f90](#).

6.23.1.7 `character(len=2) relaxcommon::reltype`

Definition at line 29 of file [relaxcommon.f90](#).

6.23.1.8 `real(latteprec) relaxcommon::rlxftol`

Definition at line 30 of file [relaxcommon.f90](#).

6.24 restartarray Module Reference

Variables

- `real(latteprec), dimension(:), allocatable tmpbodiag`
- `integer tmphdim`
- `real(latteprec), dimension(:), allocatable tmprhodown`
- `real(latteprec), dimension(:), allocatable tmprhoup`

6.24.1 Variable Documentation

6.24.1.1 `real(latteprec), dimension(:), allocatable restartarray::tmpbodiag`

Definition at line 30 of file [restartarray.f90](#).

6.24.1.2 `integer restartarray::tmphdim`

Definition at line 29 of file [restartarray.f90](#).

6.24.1.3 `real(latteprec), dimension(:), allocatable restartarray::tmprhodown`

Definition at line 30 of file [restartarray.f90](#).

6.24.1.4 `real(latteprec)`, `dimension(:)`, allocatable `restartarray::tmprhoup`

Definition at line 30 of file [restartarray.f90](#).

6.25 setuparray Module Reference

Variables

- `character(len=2)`, `dimension(:)`, allocatable [atele](#)
- `real(latteprec)`, `dimension(:)`, allocatable [atocc](#)
- `character(len=4)`, `dimension(:)`, allocatable [basis](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [bo](#)
- `real(latteprec)`, `dimension(:)`, allocatable [bozero](#)
- `character(len=3)`, `dimension(:)`, allocatable [btype](#)
- `real(latteprec)`, `dimension(:)`, allocatable [coulombv](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [cr](#)
- `real(latteprec)`, `dimension(:)`, allocatable [deltaq](#)
- `character(len=2)`, `dimension(:)`, allocatable [ele](#)
- `character(len=2)`, `dimension(:)`, allocatable [ele1](#)
- `character(len=2)`, `dimension(:)`, allocatable [ele2](#)
- `integer`, `dimension(:)`, allocatable [elmpointer](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [f](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [fcoul](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [fpp](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [fpul](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [fscoul](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [fsspin](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [ftot](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [h](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [h0](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hdiag](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hed](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hef](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hep](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hes](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hr0](#)
- `real(latteprec)`, `dimension(:)`, allocatable [hubbardu](#)
- `real(latteprec)`, `dimension(:)`, allocatable [lcnsht](#)
- `integer`, `dimension(:)`, allocatable [matindlist](#)
- `real(latteprec)`, `dimension(:)`, allocatable [mycharge](#)
- `real(latteprec)`, `dimension(:, :)`, allocatable [orthorho](#)
- `real(latteprec)`, `dimension(:)`, allocatable [qlist](#)
- `real(latteprec)`, `dimension(:)`, allocatable [respchi](#)
- `integer`, `dimension(:)`, allocatable [spinindlist](#)

6.25.1 Variable Documentation

6.25.1.1 `character(len=2)`, `dimension(:)`, allocatable `setuparray::atele`

Definition at line 42 of file [setuparray.f90](#).

6.25.1.2 `real(latteprec), dimension(:), allocatable setuparray::atocc`

Definition at line 33 of file [setuparray.f90](#).

6.25.1.3 `character(len=4), dimension(:), allocatable setuparray::basis`

Definition at line 44 of file [setuparray.f90](#).

6.25.1.4 `real(latteprec), dimension(:, :), allocatable setuparray::bo`

Definition at line 34 of file [setuparray.f90](#).

6.25.1.5 `real(latteprec), dimension(:), allocatable setuparray::bozero`

Definition at line 34 of file [setuparray.f90](#).

6.25.1.6 `character(len=3), dimension(:), allocatable setuparray::btype`

Definition at line 43 of file [setuparray.f90](#).

6.25.1.7 `real(latteprec), dimension(:), allocatable setuparray::coulombv`

Definition at line 48 of file [setuparray.f90](#).

6.25.1.8 `real(latteprec), dimension(:, :), allocatable setuparray::cr`

Definition at line 31 of file [setuparray.f90](#).

6.25.1.9 `real(latteprec), dimension(:), allocatable setuparray::deltaq`

Definition at line 38 of file [setuparray.f90](#).

6.25.1.10 `character(len=2), dimension(:), allocatable setuparray::ele`

Definition at line 42 of file [setuparray.f90](#).

6.25.1.11 `character(len=2), dimension(:), allocatable setuparray::ele1`

Definition at line 42 of file [setuparray.f90](#).

6.25.1.12 `character(len=2), dimension(:), allocatable setuparray::ele2`

Definition at line 42 of file [setuparray.f90](#).

6.25.1.13 `integer, dimension(:), allocatable setuparray::elempointer`

Definition at line 29 of file [setuparray.f90](#).

6.25.1.14 `real(latteprec), dimension(:, :), allocatable setuparray::f`

Definition at line 36 of file [setuparray.f90](#).

6.25.1.15 `real(latteprec), dimension(:, :), allocatable setuparray::fcoul`

Definition at line 36 of file [setuparray.f90](#).

6.25.1.16 `real(latteprec), dimension(:, :), allocatable setuparray::fpp`

Definition at line 36 of file [setuparray.f90](#).

6.25.1.17 `real(latteprec), dimension(:, :), allocatable setuparray::fpul`

Definition at line 37 of file [setuparray.f90](#).

6.25.1.18 `real(latteprec), dimension(:, :), allocatable setuparray::fscoul`

Definition at line 37 of file [setuparray.f90](#).

6.25.1.19 `real(latteprec), dimension(:, :), allocatable setuparray::fsspin`

Definition at line 37 of file [setuparray.f90](#).

6.25.1.20 `real(latteprec), dimension(:, :), allocatable setuparray::ftot`

Definition at line 36 of file [setuparray.f90](#).

6.25.1.21 `real(latteprec), dimension(:, :), allocatable setuparray::h`

Definition at line 34 of file [setuparray.f90](#).

6.25.1.22 `real(latteprec), dimension(:, :), allocatable setuparray::h0`

Definition at line 34 of file [setuparray.f90](#).

6.25.1.23 `real(latteprec), dimension(:), allocatable setuparray::hdiag`

Definition at line 34 of file [setuparray.f90](#).

6.25.1.24 `real(latteprec), dimension(:), allocatable setuparray::hed`

Definition at line 33 of file [setuparray.f90](#).

6.25.1.25 `real(latteprec), dimension(:), allocatable setuparray::hef`

Definition at line 33 of file [setuparray.f90](#).

6.25.1.26 `real(latteprec), dimension(:), allocatable setuparray::hep`

Definition at line 33 of file [setuparray.f90](#).

6.25.1.27 `real(latteprec), dimension(:), allocatable setuparray::hes`

Definition at line 33 of file [setuparray.f90](#).

6.25.1.28 `real(latteprec), dimension(:), allocatable setuparray::hr0`

Definition at line 32 of file [setuparray.f90](#).

6.25.1.29 `real(latteprec), dimension(:), allocatable setuparray::hubbardu`

Definition at line 40 of file [setuparray.f90](#).

6.25.1.30 `real(latteprec), dimension(:), allocatable setuparray::lcnsht`

Definition at line 39 of file [setuparray.f90](#).

6.25.1.31 `integer, dimension(:), allocatable setuparray::matindlist`

Definition at line 30 of file [setuparray.f90](#).

6.25.1.32 `real(latteprec), dimension(:), allocatable setuparray::mycharge`

Definition at line 38 of file [setuparray.f90](#).

6.25.1.33 `real(latteprec), dimension(:, :), allocatable setuparray::orthorho`

Definition at line 35 of file [setuparray.f90](#).

6.25.1.34 `real(latteprec), dimension(:), allocatable setuparray::qlist`

Definition at line 38 of file [setuparray.f90](#).

6.25.1.35 `real(latteprec), dimension(:), allocatable setuparray::respchi`

Definition at line 41 of file [setuparray.f90](#).

6.25.1.36 `integer, dimension(:), allocatable setuparray::spinindlist`

Definition at line 30 of file [setuparray.f90](#).

6.26 sp2progress Module Reference

To apply the sp2 method using the progress library.

Functions/Subroutines

- subroutine, public [sp2prg](#) ()

This routine implements the sp2 technique with the routines of the progress library.

Variables

- type(sp2data_type), public [sp2d](#)
- logical, public [sp2init](#) = .FALSE.

6.26.1 Detailed Description

To apply the sp2 method using the progress library.

This is a special interface to apply the sp2 solvers from the progress library. See <https://github.com/losalamos/qmd-progress>.

6.26.2 Function/Subroutine Documentation

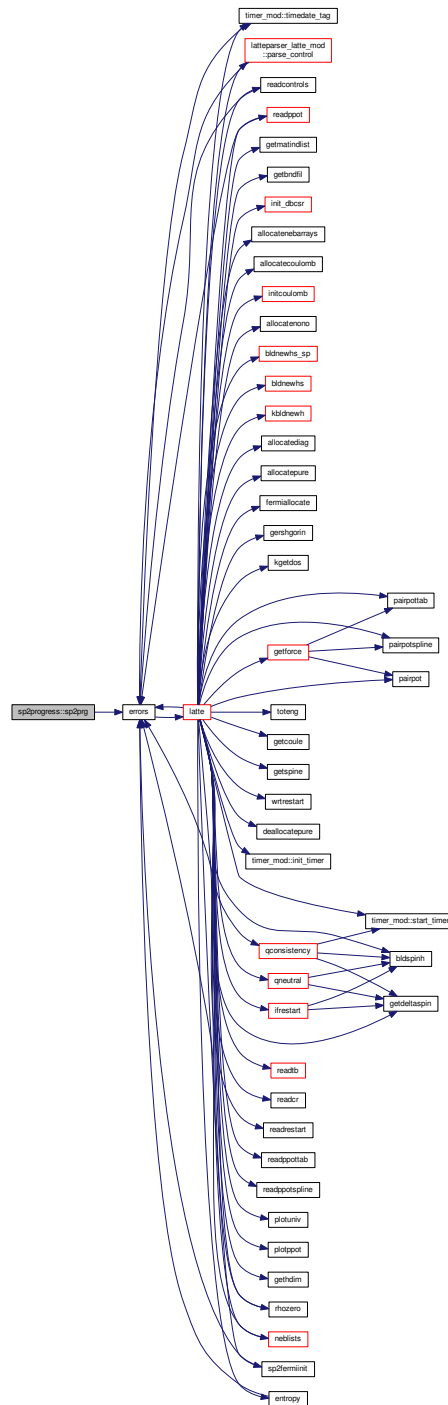
6.26.2.1 subroutine, public `sp2progress::sp2prg ()`

This routine implements the sp2 technique with the routines of the progress library.

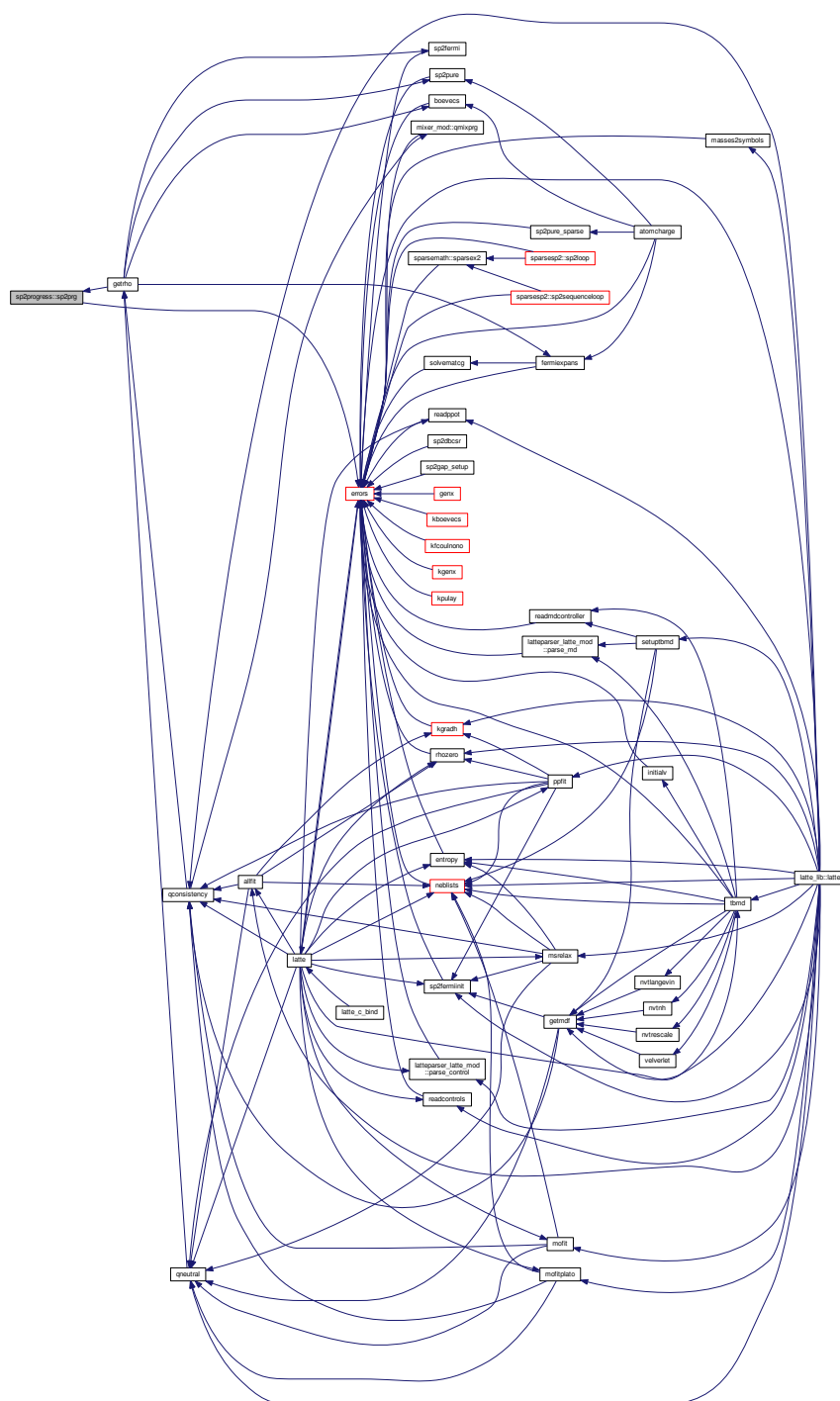
Parsing sp2 input parameters. this will read the variables in the input file.

Definition at line 59 of file [sp2progress.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.26.3 Variable Documentation

6.26.3.1 type(sp2data_type), public sp2progress::sp2d

Definition at line 50 of file sp2progress.f90.

6.26.3.2 logical, public `sp2progress::sp2init = .FALSE.`

Definition at line 48 of file [sp2progress.f90](#).

6.27 sparsearray Module Reference

Variables

- real(latteprec), dimension(:,:), allocatable [bo_padded](#)
- integer [fillinstop](#)
- real(latteprec), parameter [hthresh](#) = 1.0D-14
- integer [msparse](#)
- integer [nnz](#)
- integer [nnz_max](#)
- integer [nnz_pad](#) = 3
- real(latteprec) [numthresh](#)
- integer, dimension(:), allocatable [rx](#)
- integer, dimension(:), allocatable [rxtmp](#)
- integer [thresholdon](#)
- real(latteprec), dimension(:), allocatable [work](#)
- integer, dimension(:), allocatable [xb](#)

6.27.1 Variable Documentation

6.27.1.1 real(latteprec), dimension(:,:), allocatable `sparsearray::bo_padded`

Definition at line 38 of file [sparsearray.f90](#).

6.27.1.2 integer `sparsearray::fillinstop`

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.3 real(latteprec), parameter `sparsearray::hthresh` = 1.0D-14

Definition at line 37 of file [sparsearray.f90](#).

6.27.1.4 integer `sparsearray::msparse`

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.5 integer `sparsearray::nnz`

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.6 integer sparsearray::nnz_max

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.7 integer sparsearray::nnz_pad = 3

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.8 real(latteprec) sparsearray::numthresh

Definition at line 34 of file [sparsearray.f90](#).

6.27.1.9 integer, dimension(:), allocatable sparsearray::rx

Definition at line 32 of file [sparsearray.f90](#).

6.27.1.10 integer, dimension(:), allocatable sparsearray::rxtmp

Definition at line 32 of file [sparsearray.f90](#).

6.27.1.11 integer sparsearray::thresholdon

Definition at line 29 of file [sparsearray.f90](#).

6.27.1.12 real(latteprec), dimension(:), allocatable sparsearray::work

Definition at line 36 of file [sparsearray.f90](#).

6.27.1.13 integer, dimension(:), allocatable sparsearray::xb

Definition at line 32 of file [sparsearray.f90](#).

6.28 sparsemath Module Reference

Functions/Subroutines

- subroutine [sparseadd](#) (TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
- subroutine [sparsesetx2](#) (TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
- subroutine [sparsex2](#) (TTRX, TTRX2, II, JJ, VAL, II2, JJ2, VAL2)

Variables

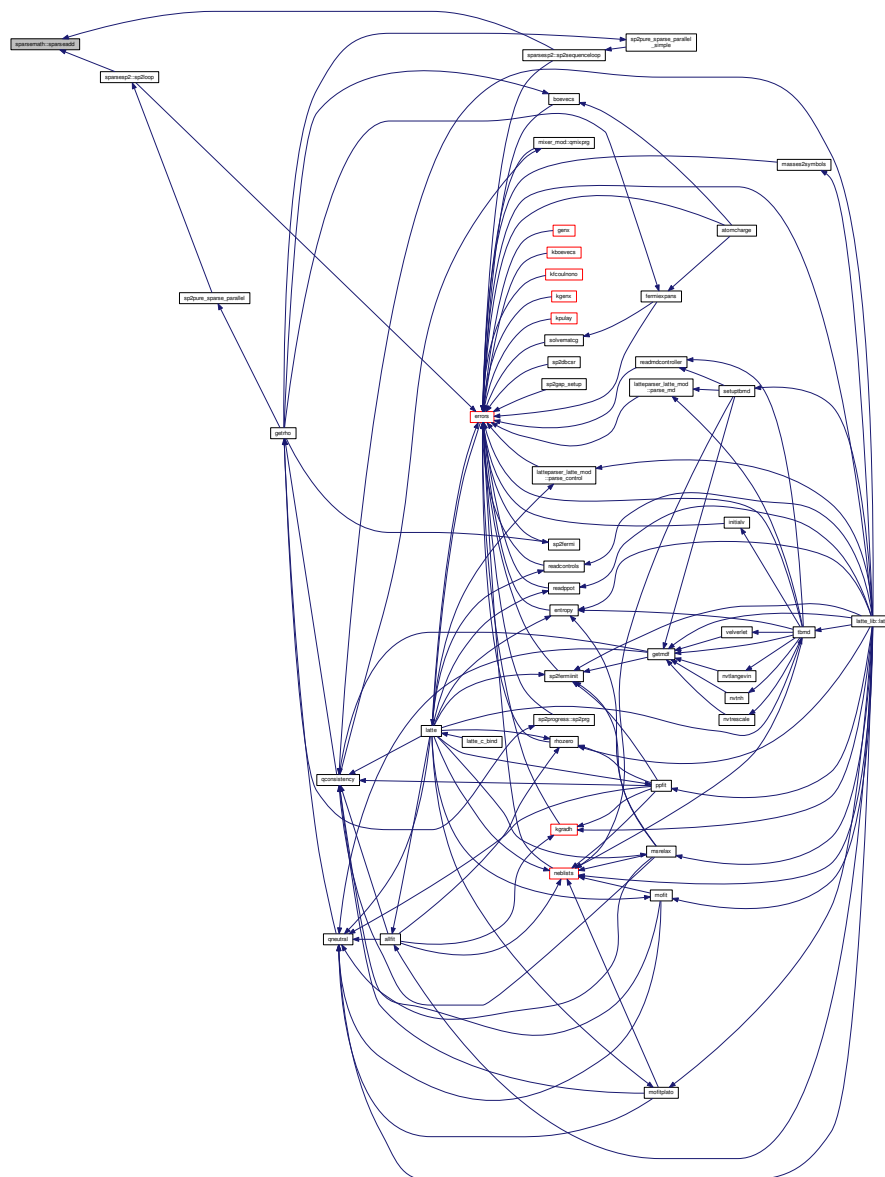
- integer, dimension(:), allocatable [ix](#)
- integer, dimension(:,:), allocatable [jjb](#)
- real(latteprec), dimension(:), allocatable [x](#)
- real(latteprec), dimension(:), allocatable [y](#)

6.28.1 Function/Subroutine Documentation

6.28.1.1 subroutine `sparsemath::sparseadd` (real(latteprec), intent(inout) *TTRNORM*, integer, dimension(:), intent(inout) *II*, integer, dimension(:,:), intent(inout) *JJ*, real(latteprec), dimension(:,:), intent(inout) *VAL*, integer, dimension(:), intent(in) *II2*, integer, dimension(:,:), intent(in) *JJ2*, real(latteprec), dimension(:,:), intent(in) *VAL2*)

Definition at line 115 of file [sparsemath.f90](#).

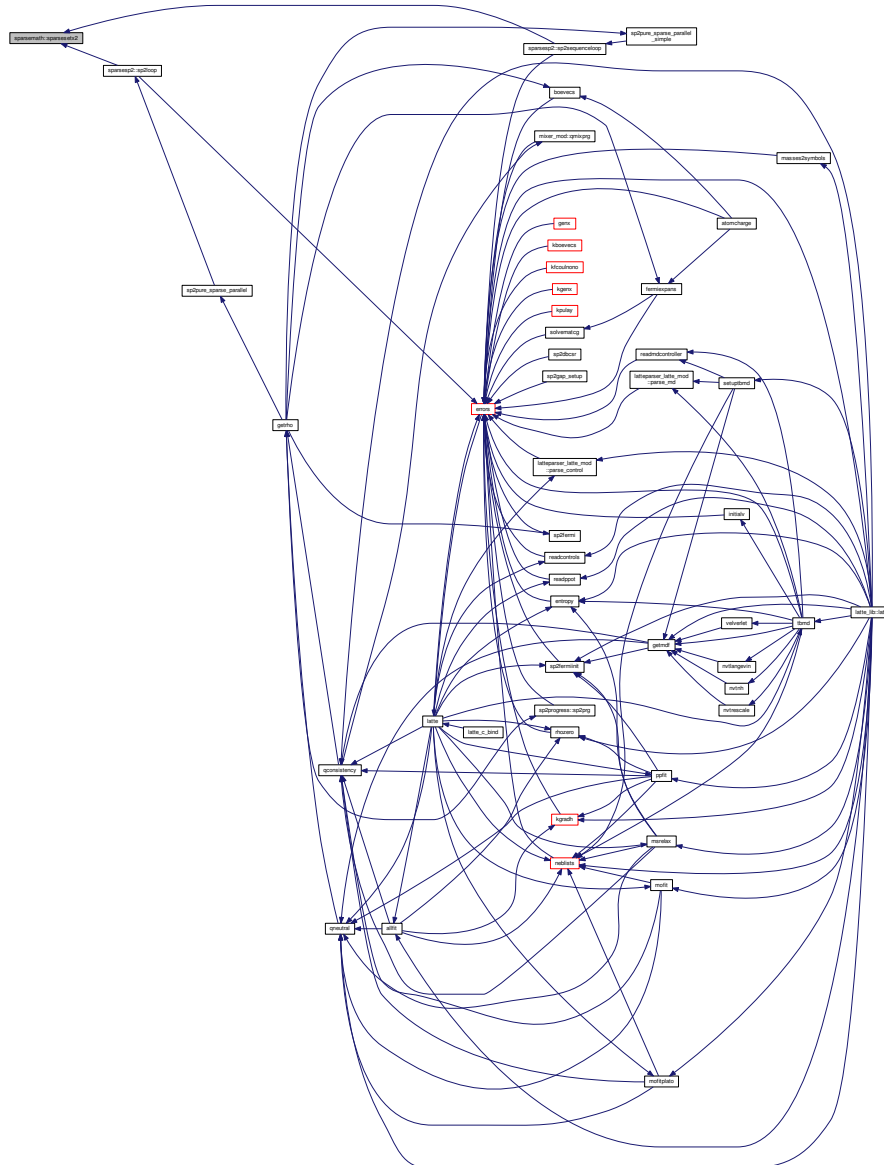
Here is the caller graph for this function:



6.28.1.2 subroutine `sparsemath::sparsesetx2` (`real(latteprec)`, `intent(inout) TTRNORM`, `integer`, `dimension(:)`, `intent(inout) JJ`, `integer`, `dimension(:, :)`, `intent(inout) VAL`, `integer`, `dimension(:)`, `intent(in) JJ2`, `real(latteprec)`, `dimension(:, :)`, `intent(in) VAL2`)

Definition at line 194 of file [sparsemath.f90](#).

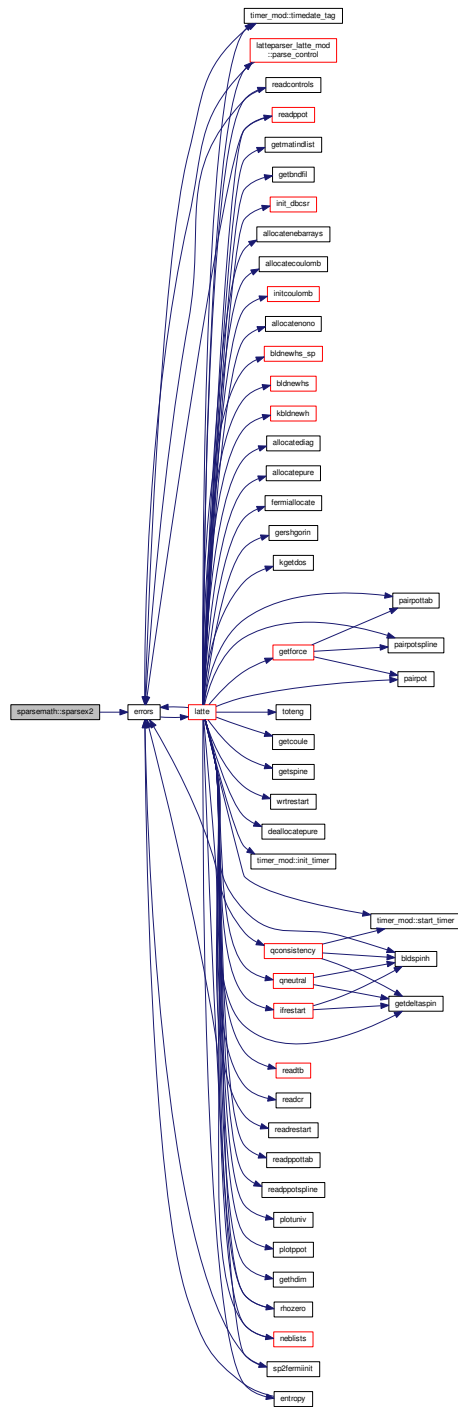
Here is the caller graph for this function:



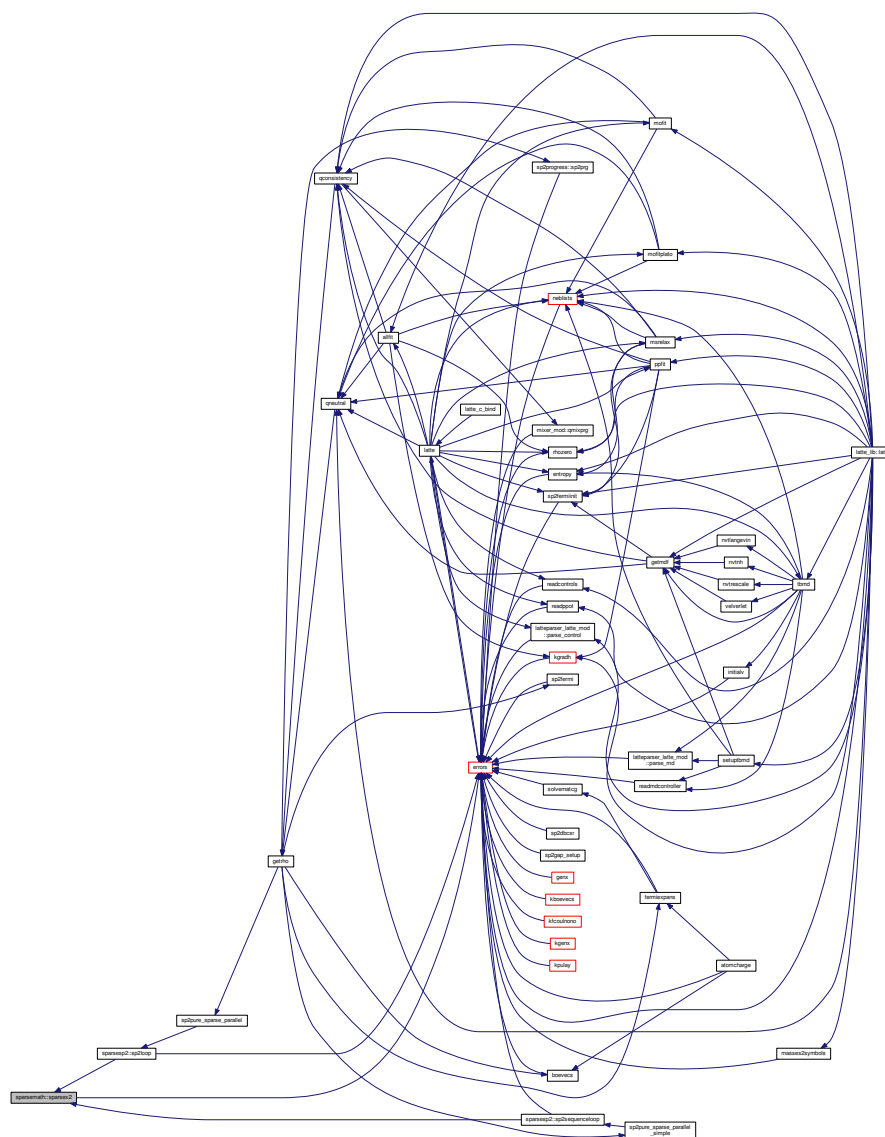
6.28.1.3 subroutine `sparsemath::sparseset2` (`real(latteprec)`, `intent(inout) TTRX`, `real(latteprec)`, `intent(inout) TTRX2`, `integer`, `dimension(:)`, `intent(in) JJ`, `integer`, `dimension(:, :)`, `intent(in) VAL`, `integer`, `dimension(:)`, `intent(inout) JJ2`, `real(latteprec)`, `dimension(:, :)`, `intent(inout) VAL2`)

Definition at line 39 of file [sparsemath.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



6.28.2 Variable Documentation

6.28.2.1 integer, dimension(:), allocatable sparsemath::ix

Definition at line 29 of file sparsemath.f90.

6.28.2.2 integer, dimension(:, :), allocatable sparsemath::jib

Definition at line 29 of file sparsemath.f90.

6.28.2.3 `real(latteprec)`, `dimension(:)`, `allocatable sparsemath::x`

Definition at line 31 of file sparsemath.f90.

6.28.2.4 `real(latteprec), dimension(:), allocatable sparsemath::y`

Definition at line 31 of file [sparsemath.f90](#).

6.29 `sparsesp2` Module Reference

Functions/Subroutines

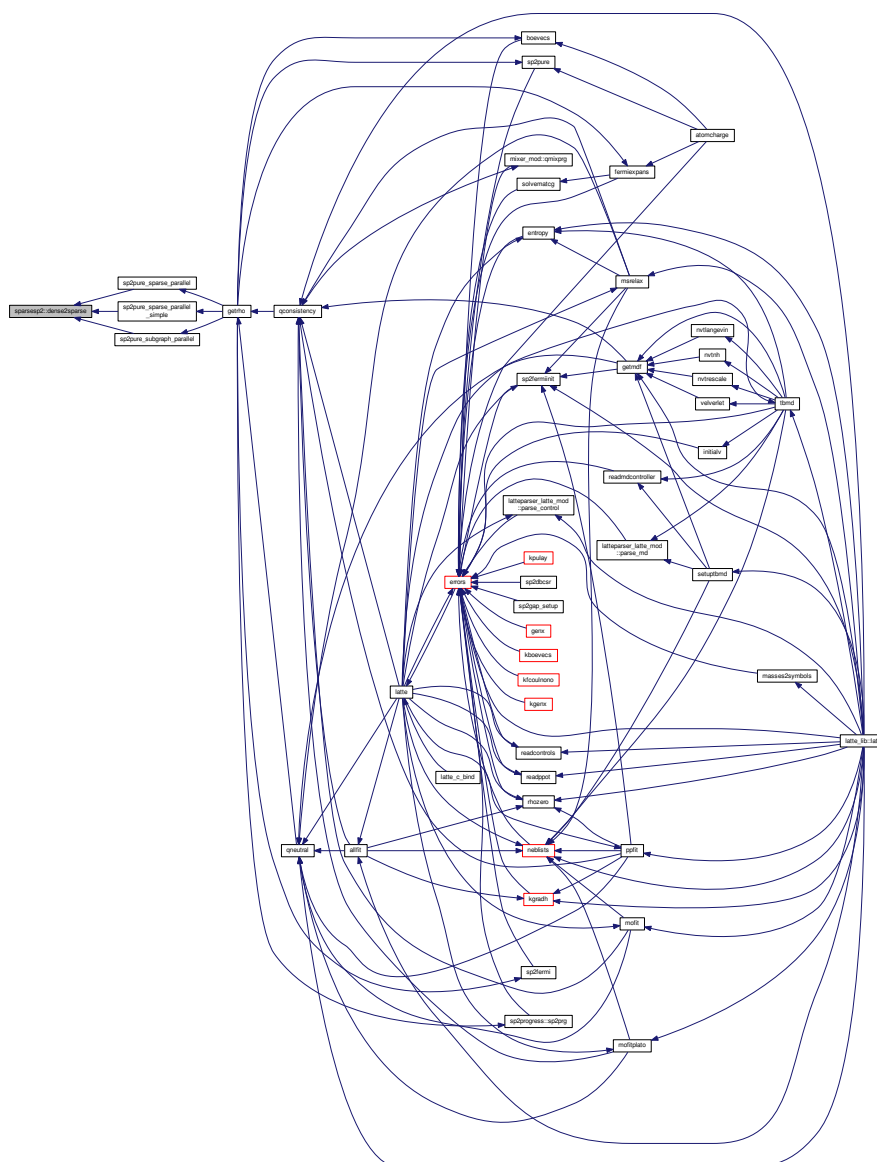
- subroutine [dense2sparse](#) (*HARRAY*, *HSIZE*, *II*, *JJ*, *VAL*)
- subroutine [sp2loop](#) (*MSIZE*, *ITER*, *II*, *JJ*, *VAL*)
- subroutine [sp2sequenceloop](#) (*MSIZE*, *ITER*, *II*, *JJ*, *VAL*)
- subroutine [sparse2dense](#) (*SCALAR*, *II*, *JJ*, *VAL*, *DARRAY*, *HSIZE*)

6.29.1 Function/Subroutine Documentation

6.29.1.1 subroutine `sparsesp2::dense2sparse` (*real(latteprec)*, *dimension(:, :)*, *intent(in) HARRAY*, *integer*, *intent(in) HSIZE*, *integer*, *dimension(:)*, *intent(inout) II*, *integer*, *dimension(:, :)*, *intent(inout) JJ*, *real(latteprec)*, *dimension(:, :)*, *intent(inout) VAL*)

Definition at line 29 of file [sparsesp2.f90](#).

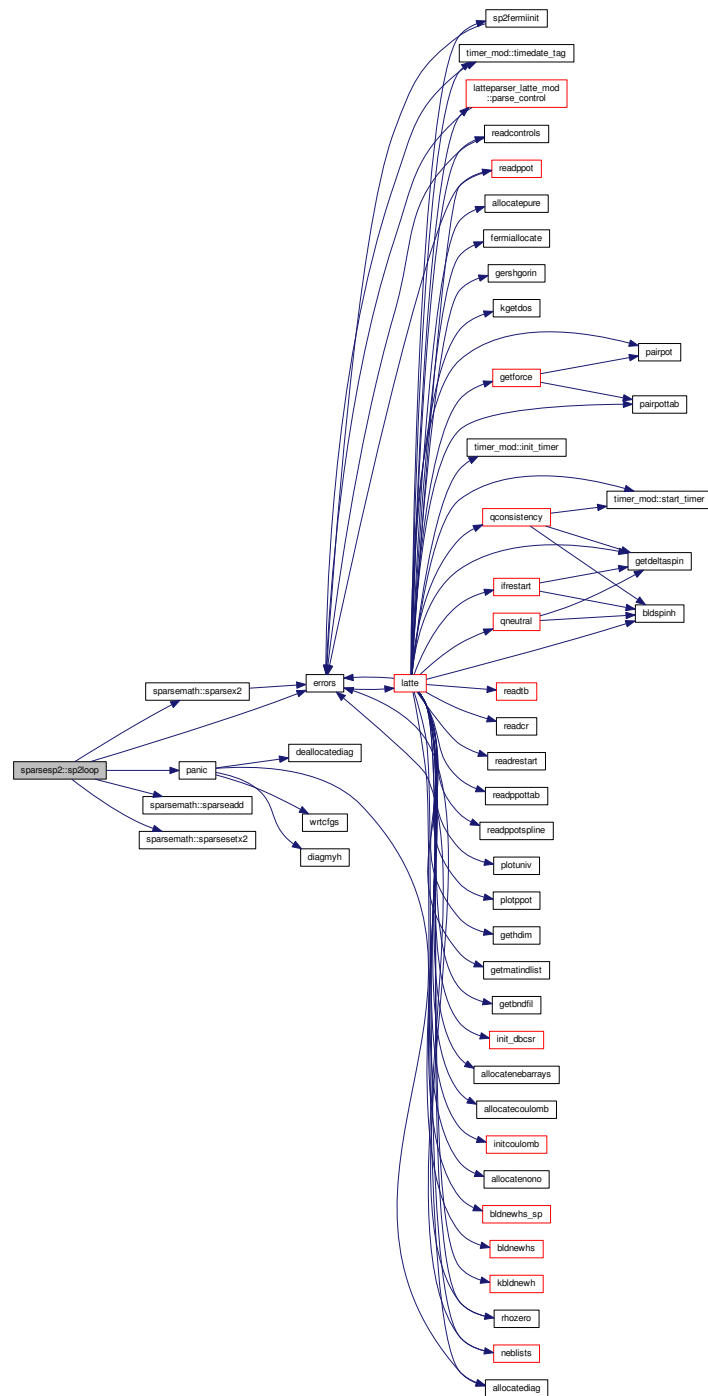
Here is the caller graph for this function:



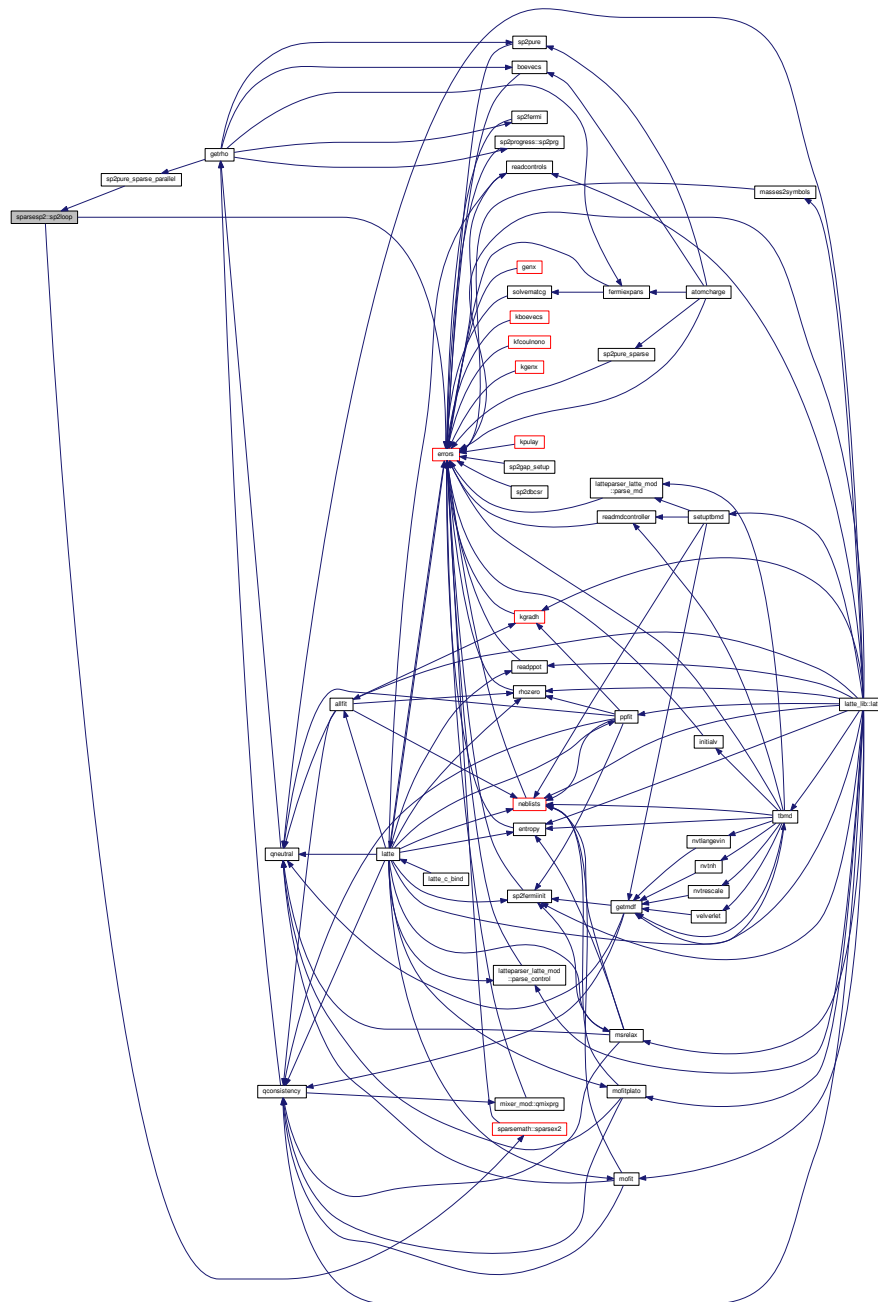
6.29.1.2 subroutine sparsesp2::sp2loop (integer, intent(inout) *MSIZE*, integer, intent(inout) *ITER*, integer, dimension(:), intent(inout) *II*, integer, dimension(:, :), intent(inout) *JJ*, real(latteprec), dimension(:, :), intent(inout) *VAL*)

Definition at line 127 of file sparsesp2.f90.

Here is the call graph for this function:



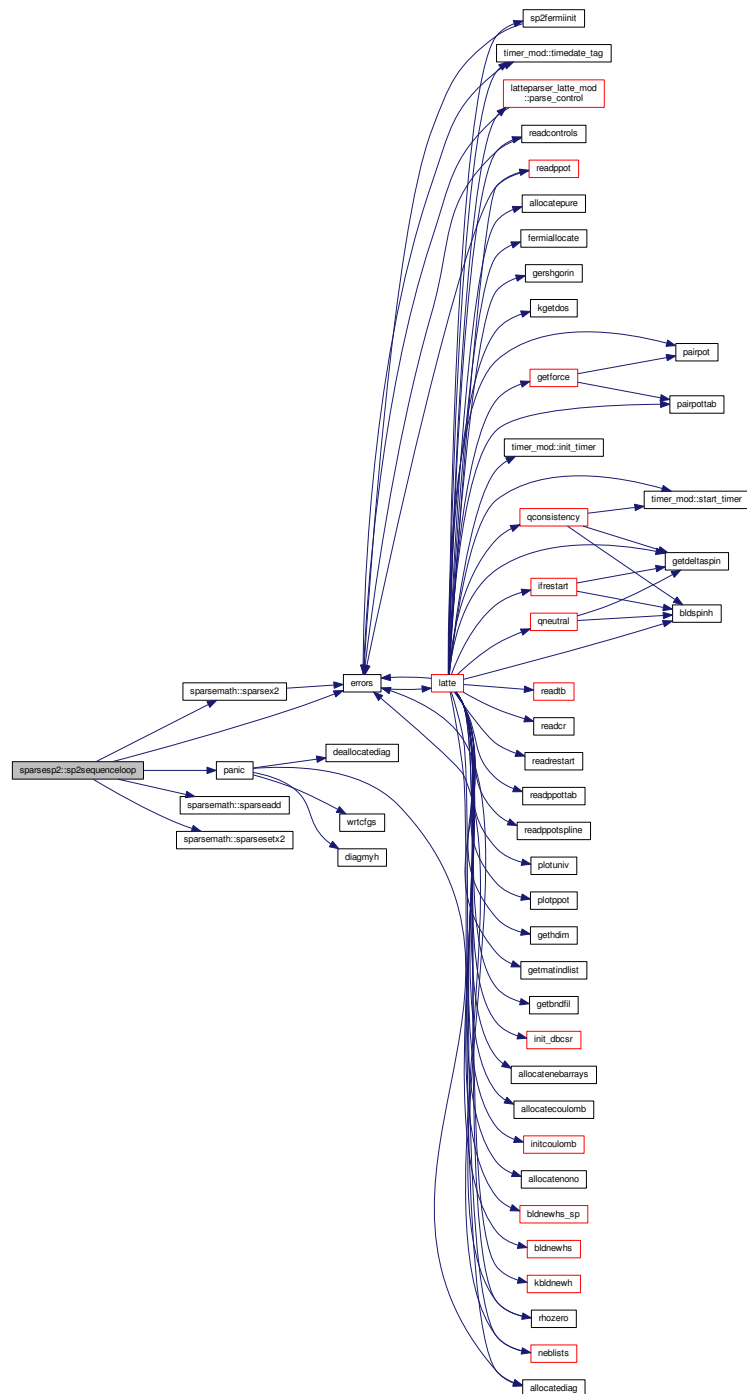
Here is the caller graph for this function:



6.29.1.3 subroutine sparsesp2::sp2sequenceloop (integer, intent(inout) *MSIZE*, integer, intent(inout) *ITER*, integer, dimension(:), intent(inout) *ll*, integer, dimension(:, :), intent(inout) *JJ*, real(latteprec), dimension(:, :), intent(inout) *VAL*)

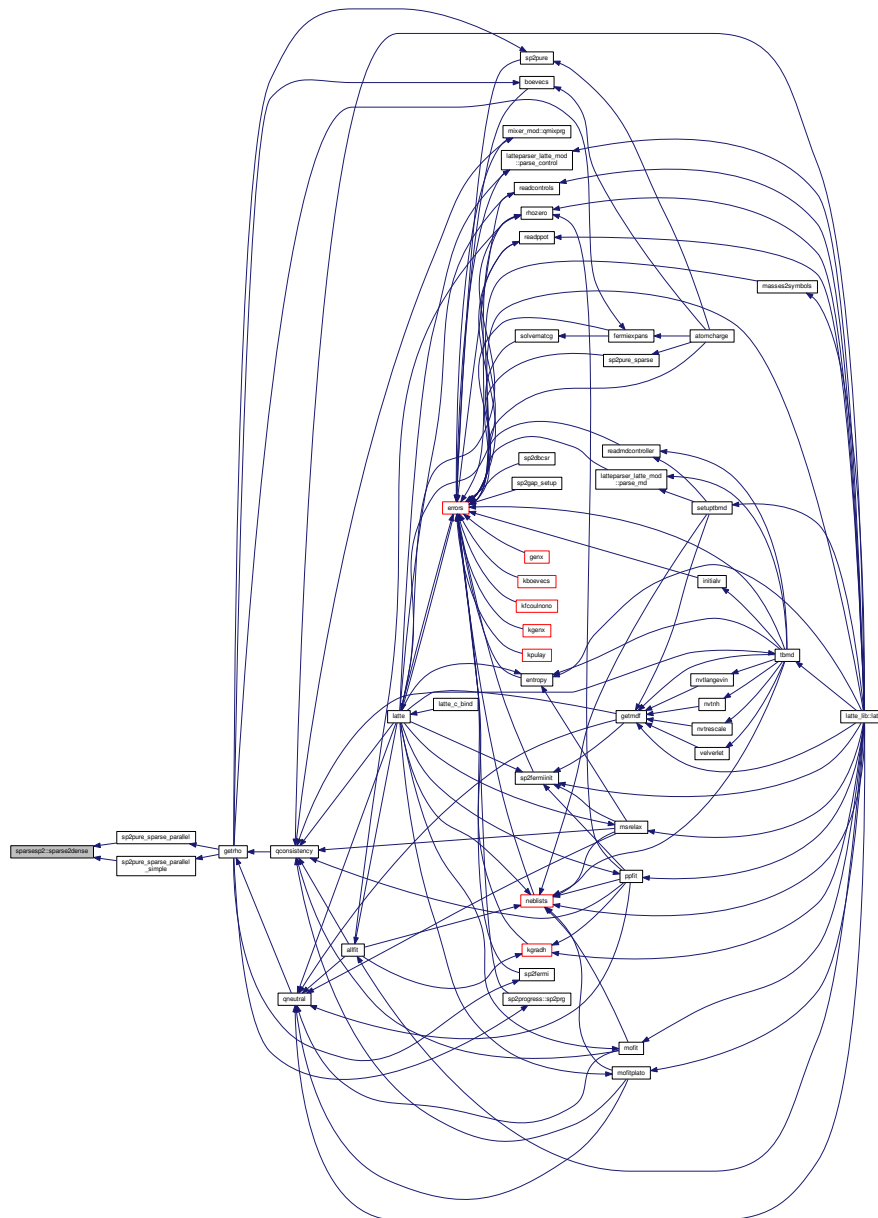
Definition at line 257 of file sparsesp2.f90.

Here is the call graph for this function:



Definition at line 103 of file sparsesp2.f90.

Here is the caller graph for this function:



6.30 spinarray Module Reference

Variables

- integer [deltadim](#)
- real(latteprec), dimension(:), allocatable [deltaspin](#)
- real(latteprec), dimension(:), allocatable [h2vect](#)
- real(latteprec), dimension(:, :), allocatable [hdown](#)
- real(latteprec), dimension(:, :), allocatable [hup](#)
- real(latteprec), dimension(:), allocatable [olddeltaspin](#)
- real(latteprec), dimension(:, :), allocatable [rhodown](#)

- `real(latteprec), dimension(:), allocatable` [rhdownzero](#)
- `real(latteprec), dimension(:, :), allocatable` [rhoup](#)
- `real(latteprec), dimension(:), allocatable` [rhoupzero](#)
- `real(latteprec), dimension(:), allocatable` [spinlist](#)
- `real(latteprec), dimension(:), allocatable` [wdd](#)
- `real(latteprec), dimension(:), allocatable` [wff](#)
- `real(latteprec), dimension(:), allocatable` [wpp](#)
- `real(latteprec), dimension(:), allocatable` [wss](#)

6.30.1 Variable Documentation

6.30.1.1 integer spinarray::deltadim

Definition at line 29 of file [spinarray.f90](#).

6.30.1.2 real(latteprec), dimension(:), allocatable spinarray::deltaspin

Definition at line 34 of file [spinarray.f90](#).

6.30.1.3 real(latteprec), dimension(:), allocatable spinarray::h2vect

Definition at line 35 of file [spinarray.f90](#).

6.30.1.4 real(latteprec), dimension(:, :), allocatable spinarray::hdown

Definition at line 31 of file [spinarray.f90](#).

6.30.1.5 real(latteprec), dimension(:, :), allocatable spinarray::hup

Definition at line 31 of file [spinarray.f90](#).

6.30.1.6 real(latteprec), dimension(:), allocatable spinarray::olddeltaspin

Definition at line 34 of file [spinarray.f90](#).

6.30.1.7 real(latteprec), dimension(:, :), allocatable spinarray::rhdown

Definition at line 32 of file [spinarray.f90](#).

6.30.1.8 real(latteprec), dimension(:), allocatable spinarray::rhdownzero

Definition at line 33 of file [spinarray.f90](#).

6.30.1.9 `real(latteprec), dimension(:, :), allocatable spinarray::rhoup`

Definition at line 32 of file [spinarray.f90](#).

6.30.1.10 `real(latteprec), dimension(:), allocatable spinarray::rhoupzero`

Definition at line 33 of file [spinarray.f90](#).

6.30.1.11 `real(latteprec), dimension(:), allocatable spinarray::spinlist`

Definition at line 35 of file [spinarray.f90](#).

6.30.1.12 `real(latteprec), dimension(:), allocatable spinarray::wdd`

Definition at line 30 of file [spinarray.f90](#).

6.30.1.13 `real(latteprec), dimension(:), allocatable spinarray::wff`

Definition at line 30 of file [spinarray.f90](#).

6.30.1.14 `real(latteprec), dimension(:), allocatable spinarray::wpp`

Definition at line 30 of file [spinarray.f90](#).

6.30.1.15 `real(latteprec), dimension(:), allocatable spinarray::wss`

Definition at line 30 of file [spinarray.f90](#).

6.31 `subgraph` Module Reference

Functions/Subroutines

- subroutine [dense2sparsegraph](#) (IIG, JJG, GGN)
- subroutine [partitiongraph](#)
- subroutine [thresholdgraph](#)
- `real(latteprec)` function [tracegraph](#) (HDIM)
- subroutine [tnormgraph](#) (ITERZ)

Variables

- integer [first_step](#)
- real(latteprec), dimension(:,:), allocatable [g](#)
- real(latteprec), parameter [geps](#) = 1.0E-3
- integer, dimension(:,:), allocatable [node_in_part](#)
- integer [nr_nodes](#)
- integer, dimension(:), allocatable [nr_of_nodes_in_part](#)
- integer [nr_part](#)
- real(latteprec), dimension(:,:), allocatable [vvx](#)

6.31.1 Function/Subroutine Documentation

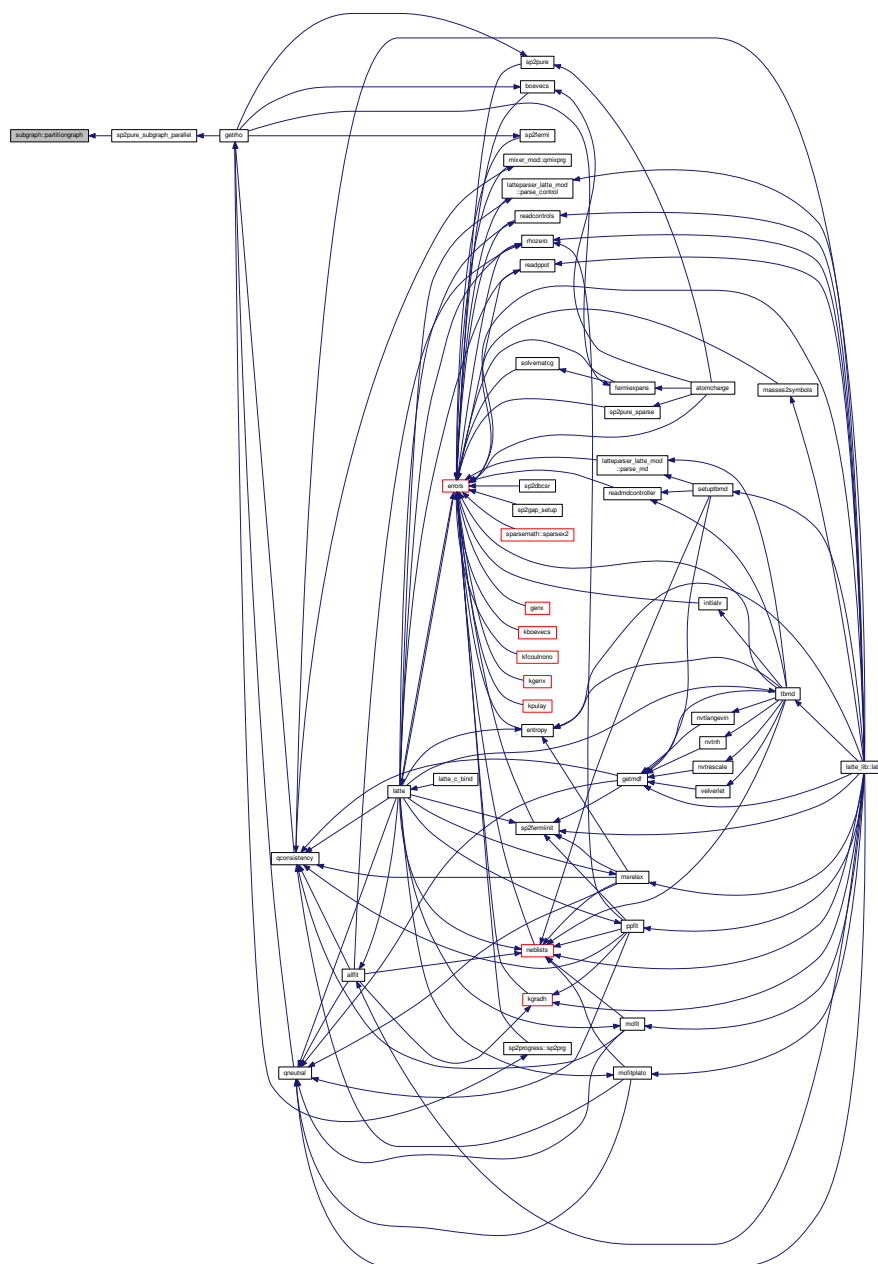
6.31.1.1 subroutine `subgraph::dense2sparsegraph` (integer, dimension(:), intent(inout) *IG*, integer, dimension(:,:), intent(inout) *JJG*, real(latteprec), dimension(:,:), intent(inout) *GGN*)

Definition at line 59 of file [subgraph.f90](#).

6.31.1.2 subroutine subgraph::partitiongraph ()

Generated by Doxygen

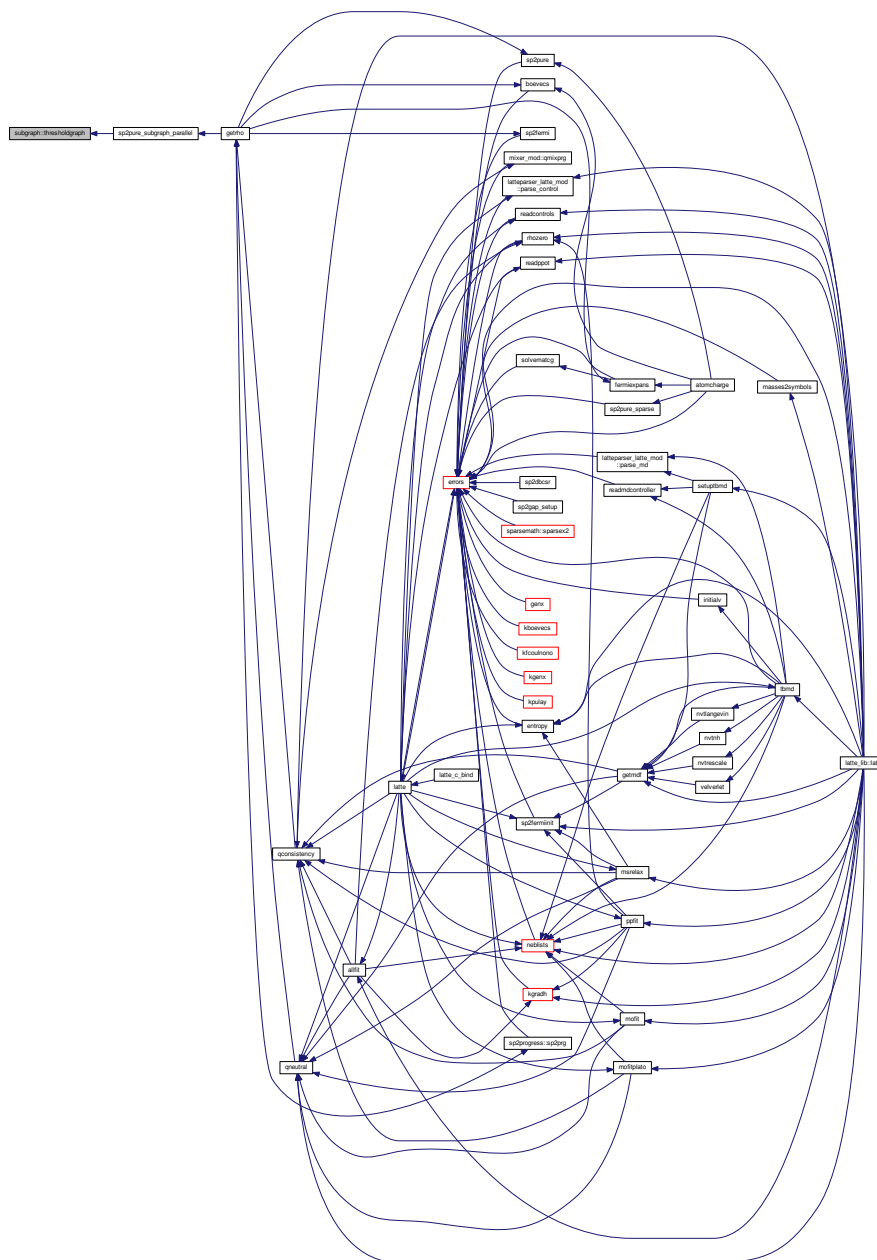
Here is the caller graph for this function:



6.31.1.3 subroutine subgraph::thresholdgraph ()

Definition at line 39 of file subgraph.f90.

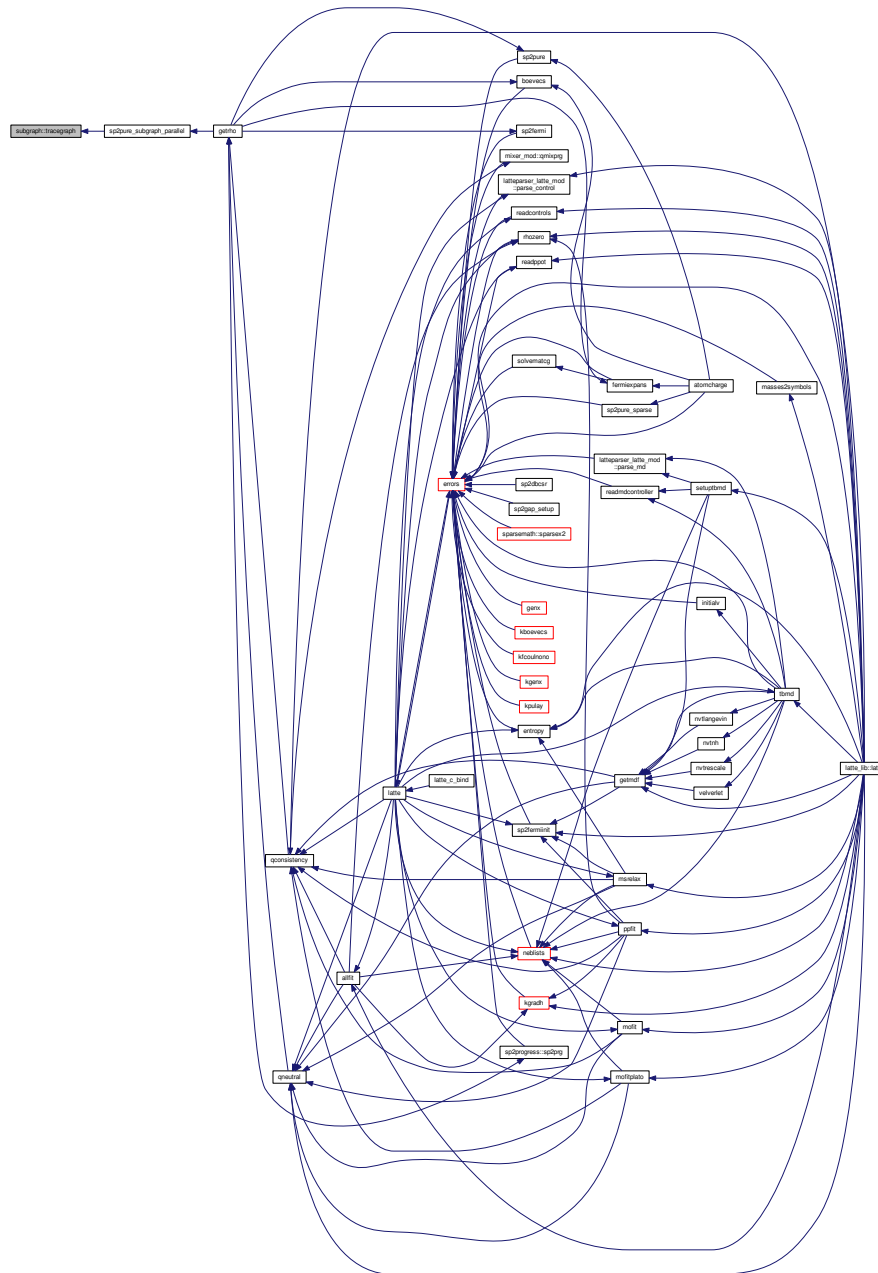
Here is the caller graph for this function:



6.31.1.4 real(latteprec) function subgraph::tracegraph (integer, intent(in) HDIM)

Definition at line 154 of file subgraph.f90.

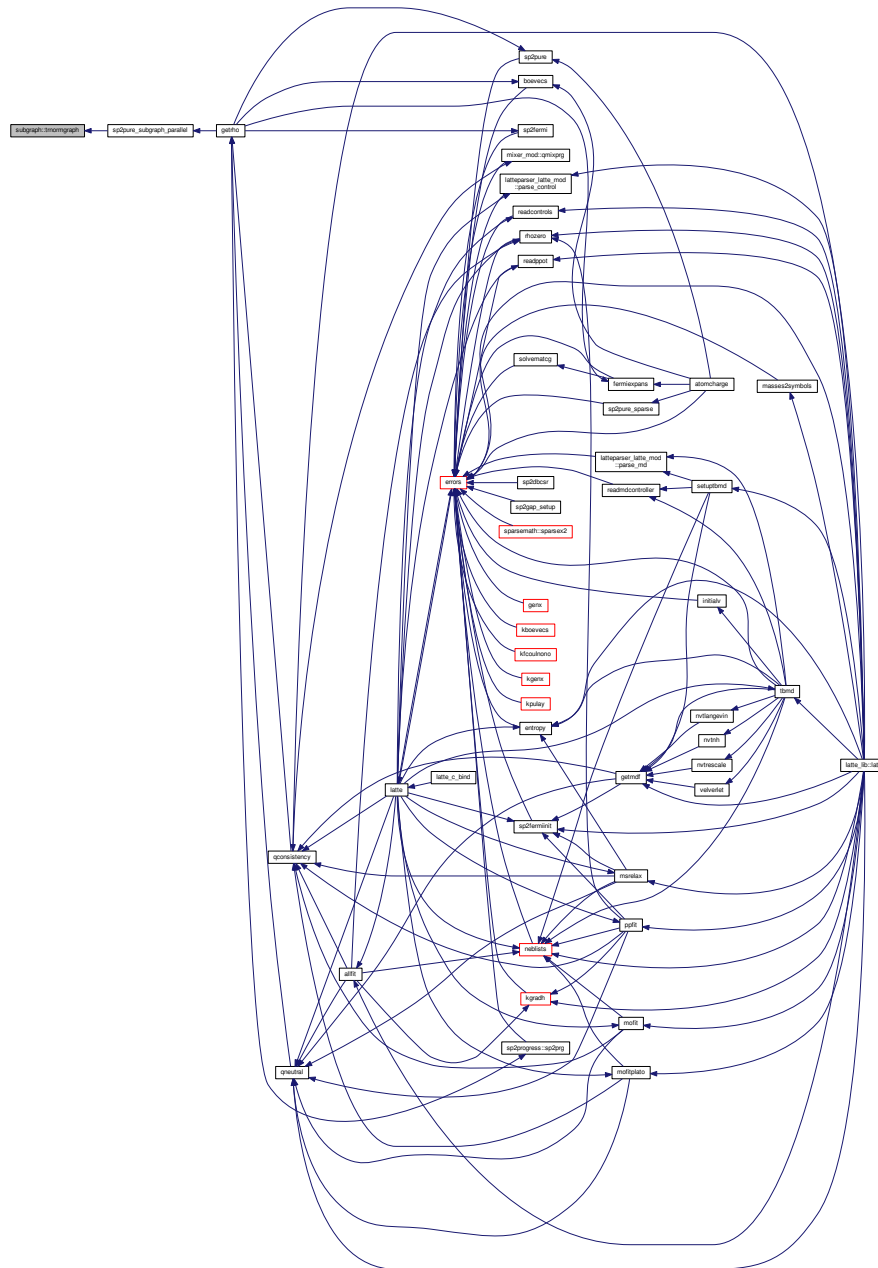
Here is the caller graph for this function:



6.31.1.5 subroutine subgraph::trnormgraph (integer, intent(inout) ITERZ)

Definition at line 127 of file [subgraph.f90](#).

Here is the caller graph for this function:



6.31.2 Variable Documentation

6.31.2.1 integer subgraph::first_step

Definition at line 30 of file subgraph.f90.

6.31.2.2 `real(latteprec)`, `dimension(:, :, allocatable subgraph::g`

Definition at line 32 of file subgraph.f90.

6.31.2.3 `real(latteprec)`, parameter `subgraph::geps = 1.0E-3`

Definition at line 34 of file [subgraph.f90](#).

6.31.2.4 `integer`, `dimension(:, :)`, allocatable `subgraph::node_in_part`

Definition at line 31 of file [subgraph.f90](#).

6.31.2.5 `integer` `subgraph::nr_nodes`

Definition at line 30 of file [subgraph.f90](#).

6.31.2.6 `integer`, `dimension(:)`, allocatable `subgraph::nr_of_nodes_in_part`

Definition at line 31 of file [subgraph.f90](#).

6.31.2.7 `integer` `subgraph::nr_part`

Definition at line 30 of file [subgraph.f90](#).

6.31.2.8 `real(latteprec)`, `dimension(:, :)`, allocatable `subgraph::vxx`

Definition at line 33 of file [subgraph.f90](#).

6.32 subgraphsp2 Module Reference

Functions/Subroutines

- subroutine [extractsubgraph](#) (`I`, `IIH`, `JJH`, `HHN`, `IIG`, `JJG`, `IX`, `JJN`, `JJP`, `LG`, `L`, `LL`)
- subroutine [normalizesubgraph](#) (`XS`, `JJN`, `L`)
- subroutine [progressloop](#) (`IIH`, `JJH`, `HHN`, `IIG`, `JJG`)
- subroutine [subgraphsp2loop](#) (`I`, `XS`, `JJP`, `L`, `LL`)
- subroutine [subgraphsp2dense](#) (`SCALAR`, `L`, `LL`, `JJN`, `JJP`, `DARRAY`, `XS`)

Variables

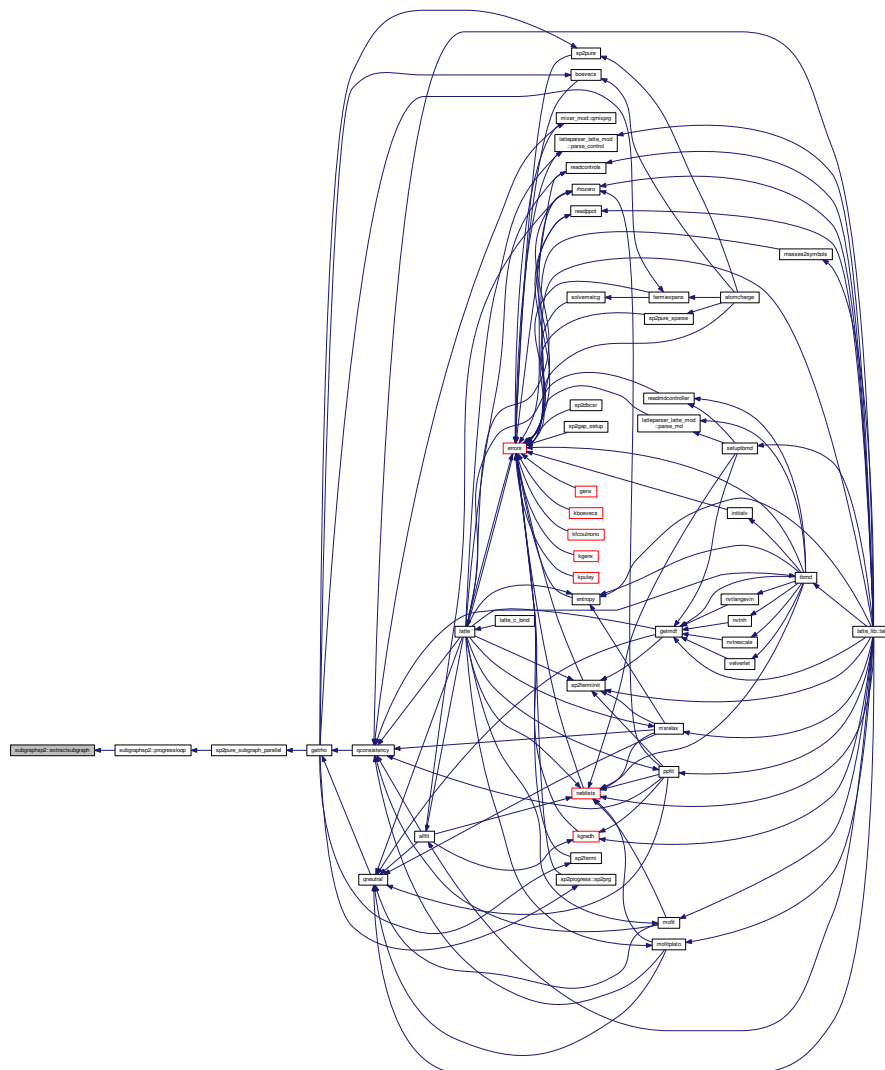
- `integer`, `dimension(:)`, allocatable `ix`
- `integer`, `dimension(:)`, allocatable `jjn`
- `integer`, `dimension(:)`, allocatable `jjp`
- `integer`, `dimension(:)`, allocatable `lg`
- `real(latteprec)`, `dimension(:, :)`, allocatable `xs`

6.32.1 Function/Subroutine Documentation

6.32.1.1 subroutine `subgraphsp2::extractsubgraph` (integer, intent(in) *I*, integer, dimension(:), intent(in) *IIH*, integer, dimension(:, :), intent(in) *JJH*, real(latteprec), dimension(:, :), intent(in) *HHN*, integer, dimension(:), intent(in) *IIG*, integer, dimension(:, :), intent(in) *JJG*, integer, dimension(:), intent(inout) *IX*, integer, dimension(:), intent(inout) *JJN*, integer, dimension(:), intent(inout) *JJP*, integer, dimension(:), intent(inout) *LG*, integer, intent(inout) *L*, integer, intent(inout) *LL*)

Definition at line 95 of file [subgraphsp2.f90](#).

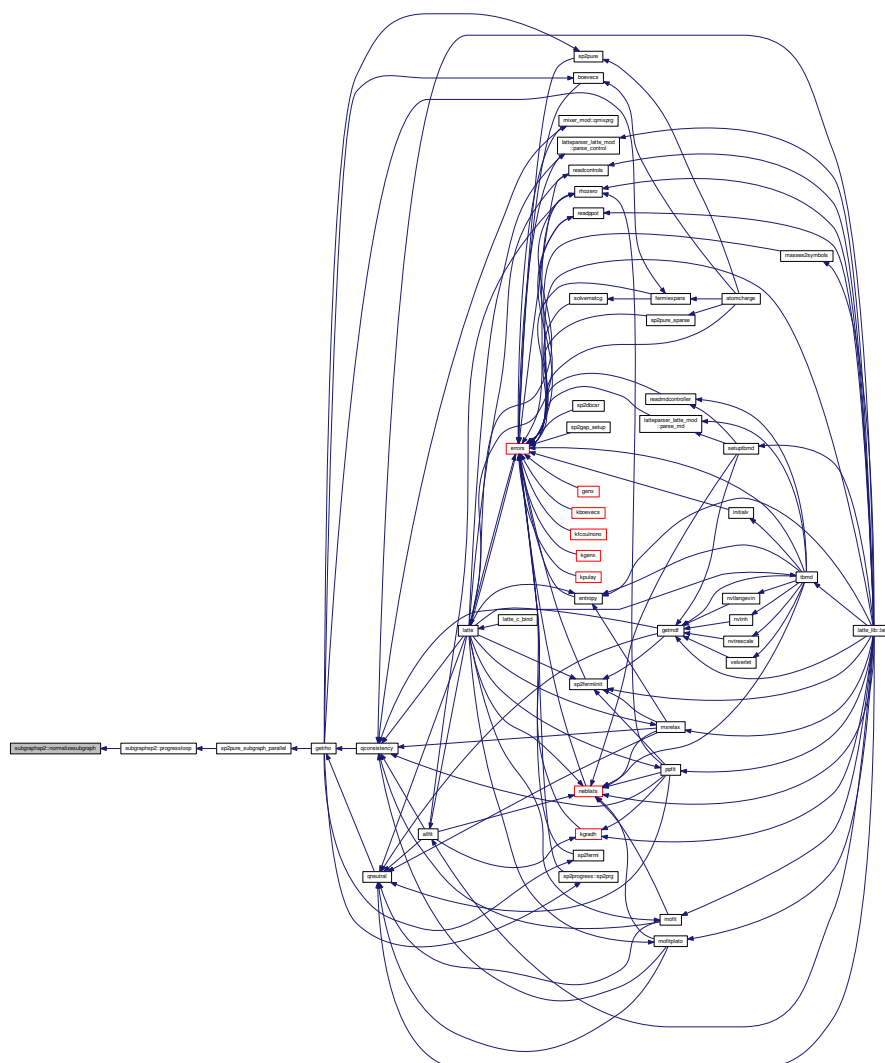
Here is the caller graph for this function:



6.32.1.2 subroutine `subgraphsp2::normalizesubgraph` (real(latteprec), dimension(:, :), intent(inout) *XS*, integer, dimension(:), intent(in) *JJN*, integer, intent(in) *L*)

Definition at line 184 of file [subgraphsp2.f90](#).

Here is the caller graph for this function:



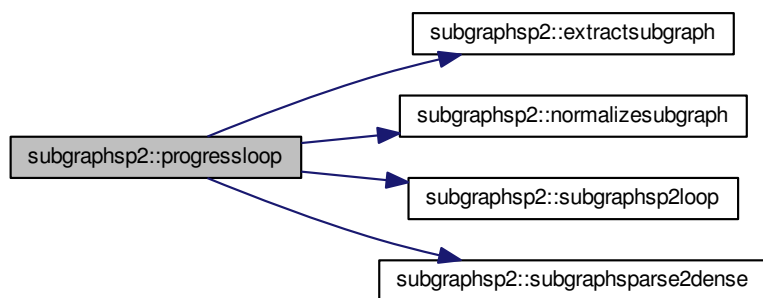
```

6.32.1.3 subroutine subgraphsp2::progressloop ( integer, dimension(:), intent(in) IIH, integer, dimension(:, :), intent(in) JJH,
real(latteprec), dimension(:, :), intent(in) HHN, integer, dimension(:), intent(in) IIG, integer, dimension(:, :), intent(in) JJG
)

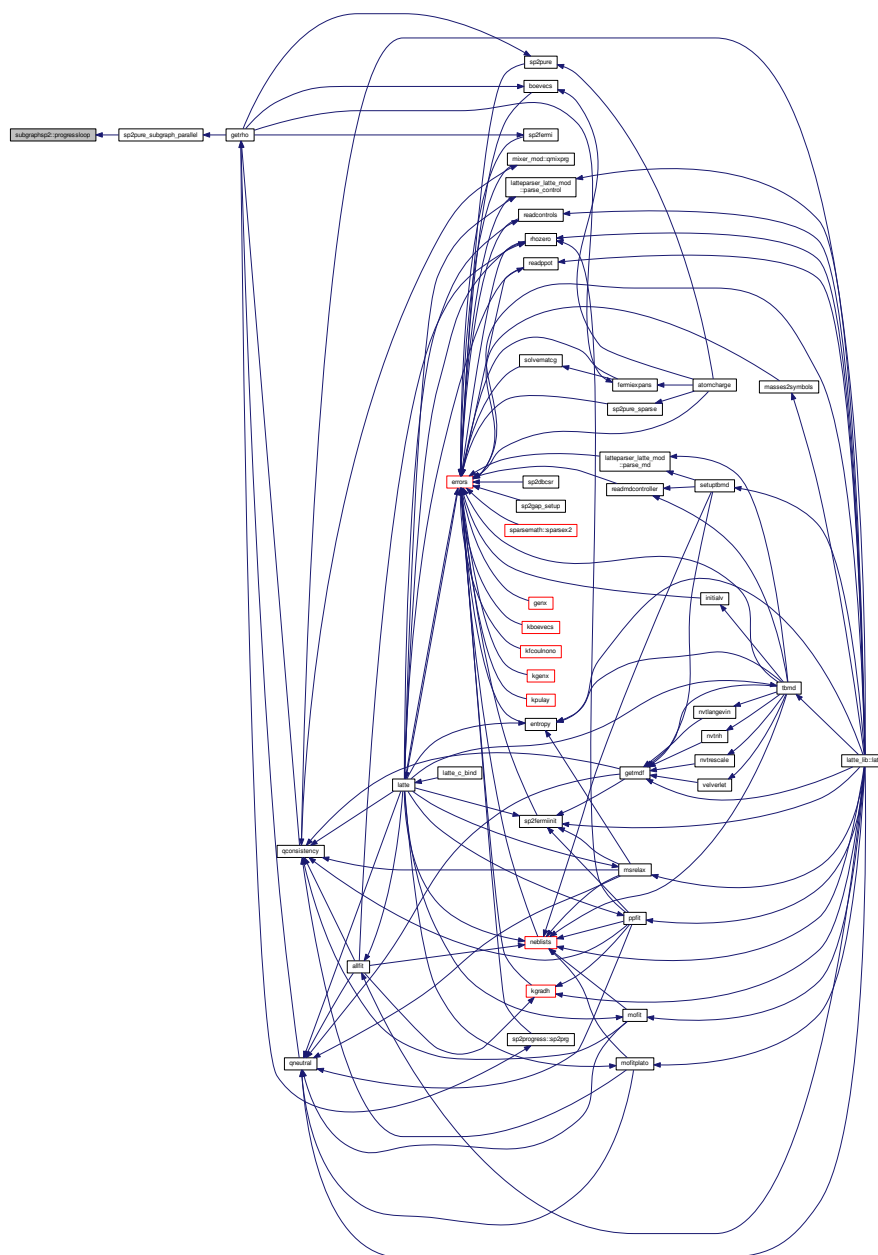
```

Definition at line 36 of file subgraphsp2.f90.

Here is the call graph for this function:



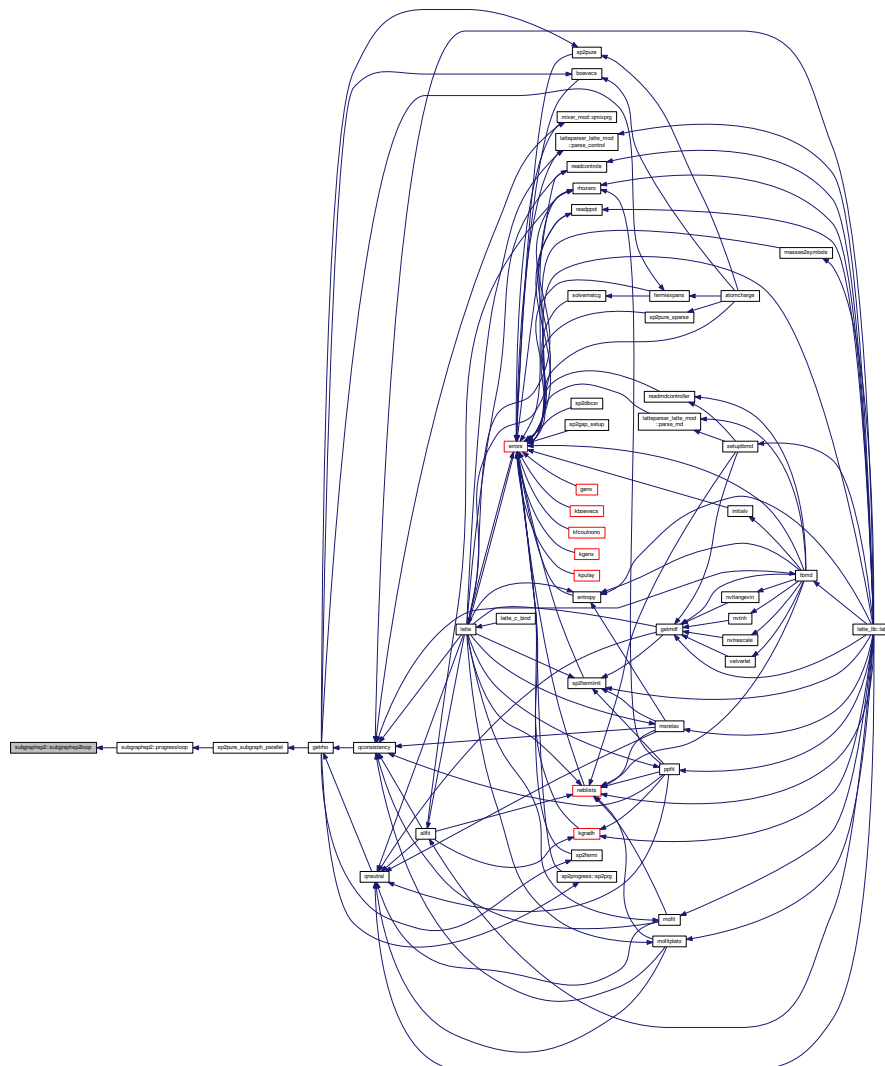
Here is the caller graph for this function:



6.32.1.4 subroutine subgraphsp2::subgraphsp2loop (integer,intent(in) *I*, real(latteprec), dimension(:,,:), intent(inout) *XS*, integer, dimension(:,), intent(in) *JJP*, integer, intent(in) *L*, integer, intent(in) *LL*)

Definition at line 212 of file subgraphsp2.f90.

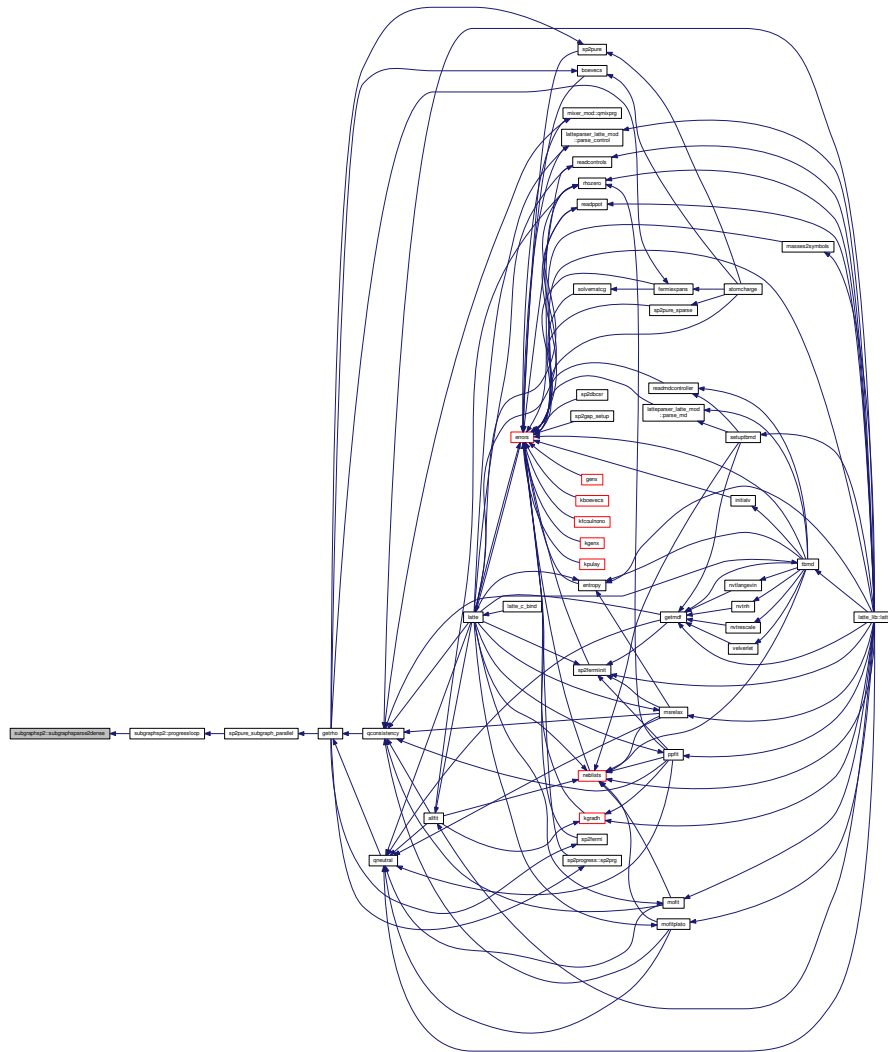
Here is the caller graph for this function:



6.32.1.5 subroutine subgraphsp2::subgraphsparse2dense (real(latteprec), intent(in) SCALAR, integer, intent(in) L, integer, intent(in) LL, integer, dimension(:), intent(in) JJN, integer, dimension(:), intent(in) JJP, real(latteprec), dimension(:,:), intent(inout) DARRAY, real(latteprec), dimension(:,:), intent(inout) XS)

Definition at line 266 of file subgraphsp2.f90.

Here is the caller graph for this function:



6.32.2 Variable Documentation

6.32.2.1 integer, dimension(:), allocatable subgraphsp2::ix

Definition at line 29 of file [subgraphsp2.f90](#).

6.32.2.2 integer, dimension(:), allocatable subgraphsp2::jjn

Definition at line 29 of file [subgraphsp2.f90](#).

6.32.2.3 integer, dimension(:), allocatable subgraphsp2::jjp

Definition at line 29 of file [subgraphsp2.f90](#).

6.32.2.4 integer, dimension(:), allocatable subgraphsp2::lg

Definition at line 29 of file [subgraphsp2.f90](#).

6.32.2.5 real(latteprec), dimension(:, :), allocatable subgraphsp2::xs

Definition at line 31 of file [subgraphsp2.f90](#).

6.33 timer_mod Module Reference

Functions/Subroutines

- integer function [init_timer](#) ()
- integer function [shutdown_timer](#) ()
- integer function [start_timer](#) (ITIMER)
- integer function [stop_timer](#) (ITIMER)
- real(8) function [time_mls](#) ()
- subroutine [timedate_tag](#) (TAG)
- integer function [timer_results](#) ()

Variables

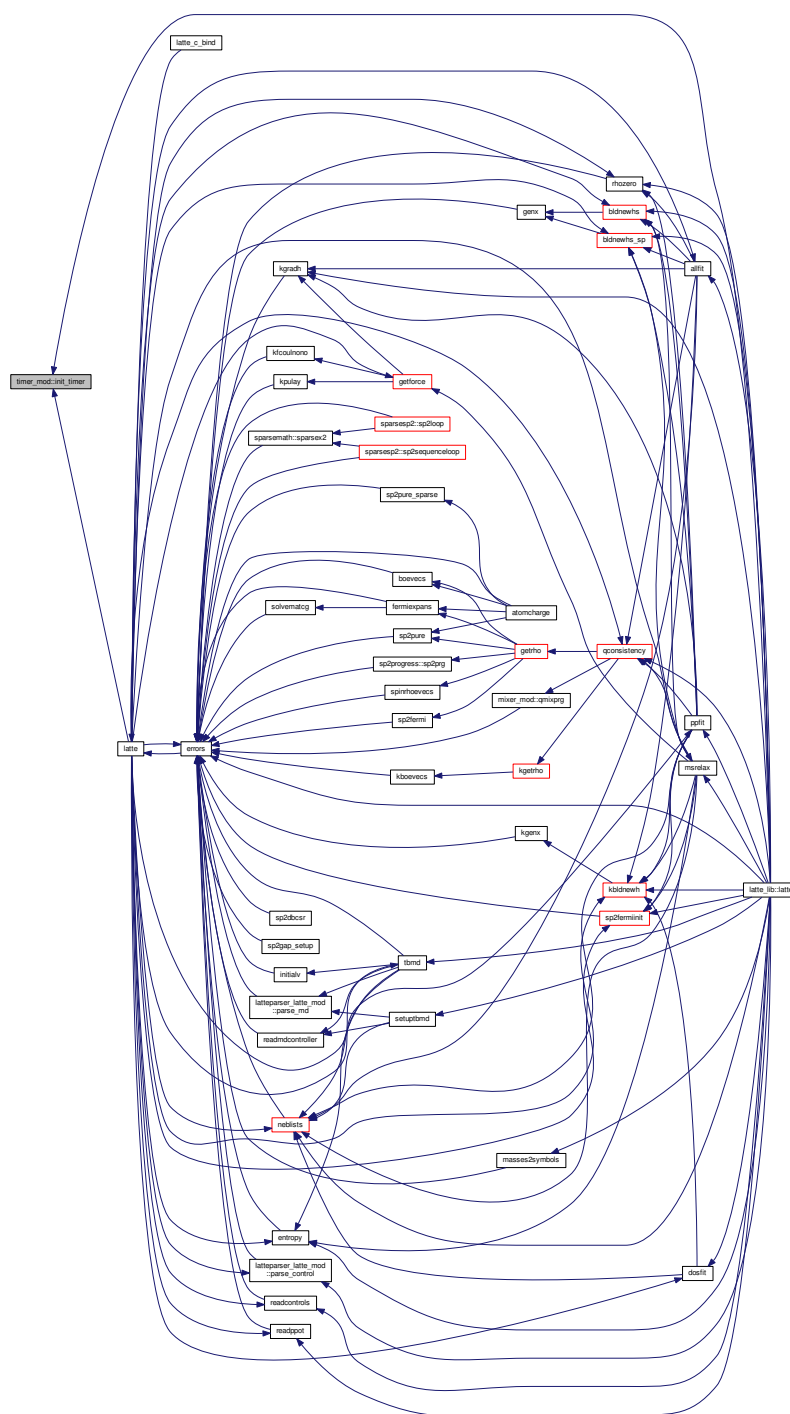
- integer [dense2sparse_timer](#)
- integer [dmbuild_timer](#)
- integer [latte_timer](#)
- integer [num_timers](#)
- integer [sp2all_timer](#)
- integer [sp2sparse_timer](#)
- integer [sparse2dense_timer](#)
- real, dimension(:), allocatable [tavg](#)
- integer [tclock_max](#)
- integer [tclock_rate](#)
- integer, dimension(:), allocatable [tcount](#)
- character(len=20), dimension(:), allocatable [tname](#)
- real, dimension(:), allocatable [tpercent](#)
- integer, dimension(:), allocatable [tstart](#)
- integer [tstart_clock](#)
- integer [tstop_clock](#)
- real, dimension(:), allocatable [tsum](#)
- integer, dimension(:), allocatable [ttotal](#)
- integer [tx](#)

6.33.1 Function/Subroutine Documentation

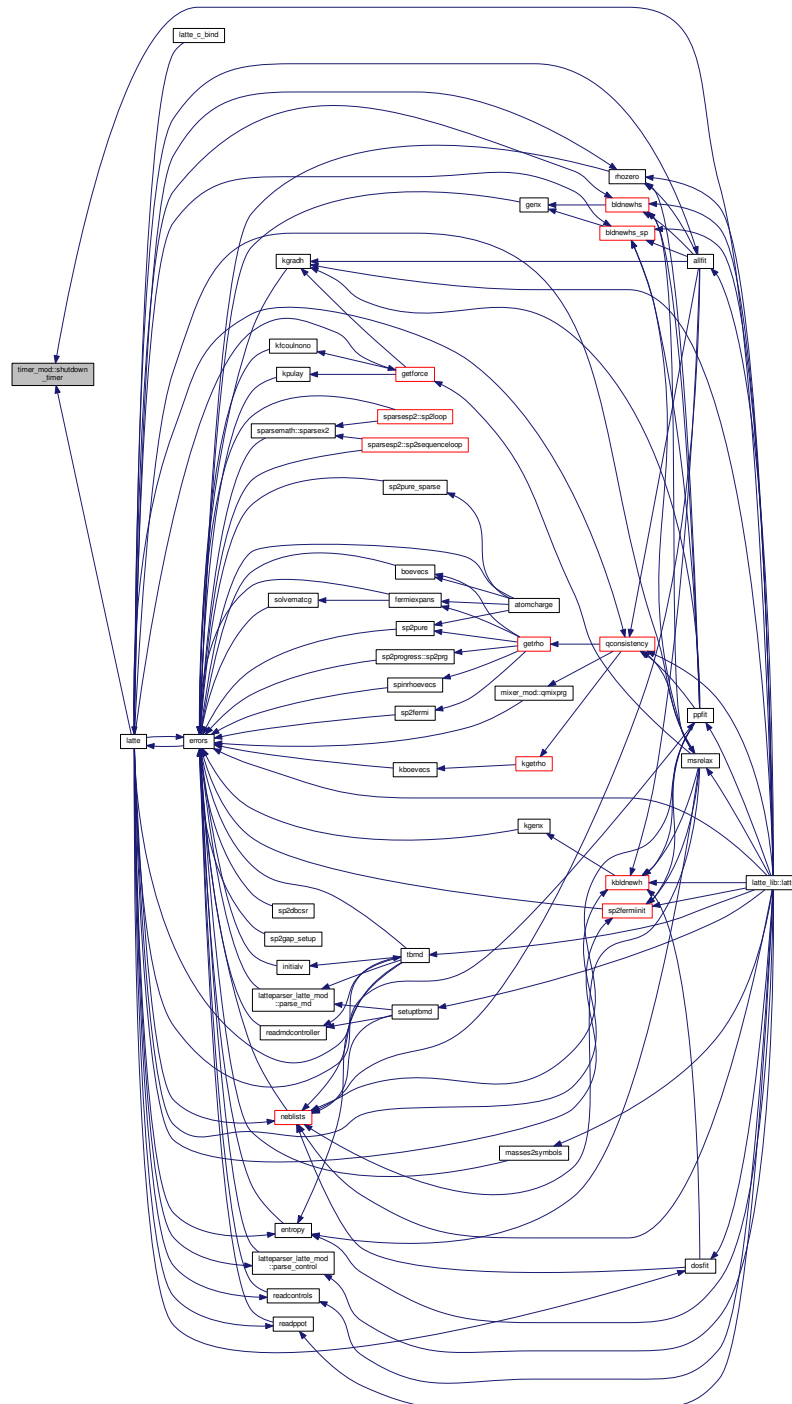
6.33.1.1 integer function timer_mod::init_timer ()

Definition at line 40 of file [timer_mod.f90](#).

Here is the caller graph for this function:

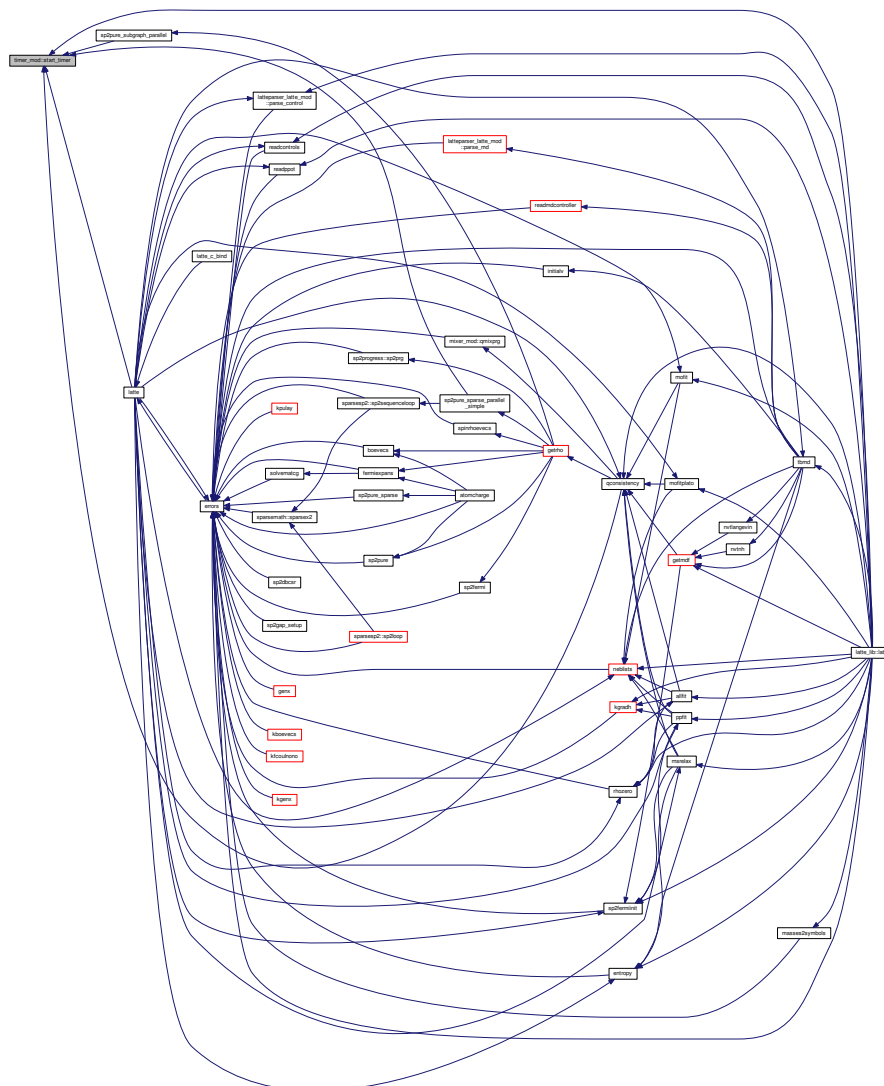


Definition at line 76 of file timer_mod.f90.



Definition at line 89 of file `timer_mod.f90`.

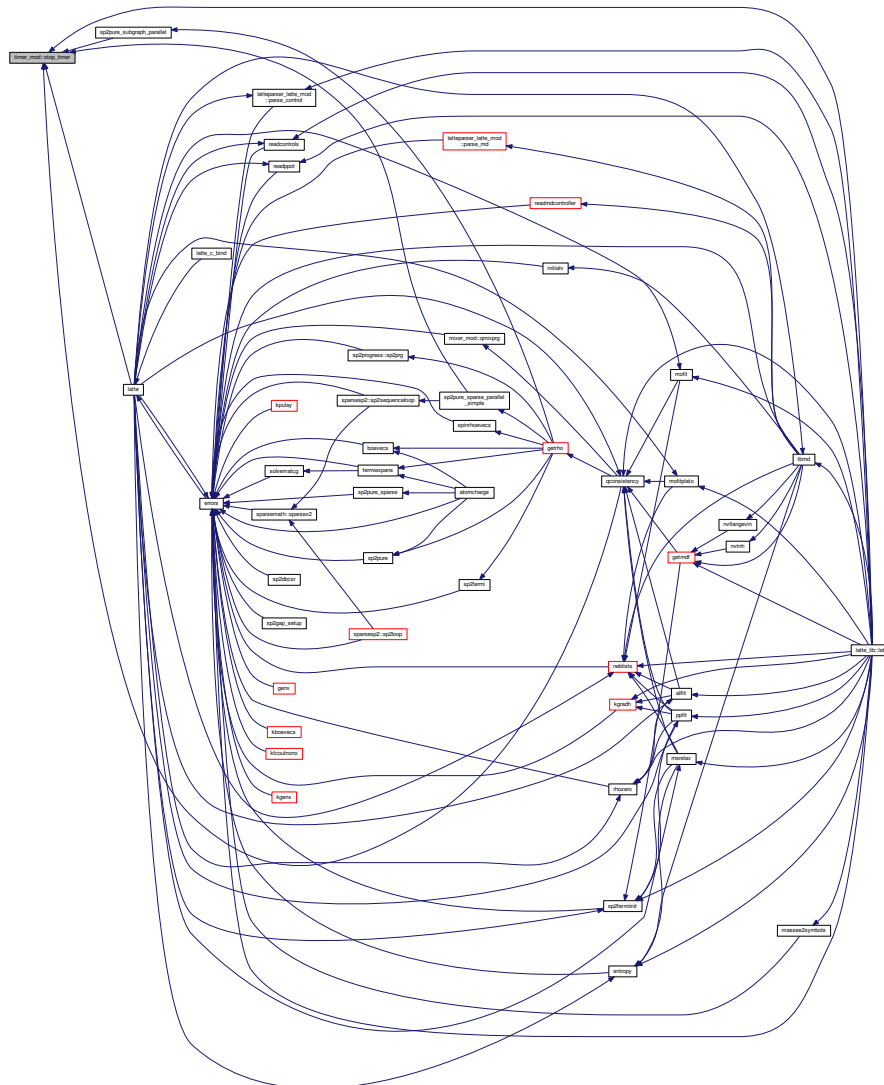
Here is the caller graph for this function:



6.33.1.4 integer function timer_mod::stop_timer (integer *ITIMER*)

Definition at line 102 of file [timer_mod.f90](#).

Here is the caller graph for this function:



6.33.1.5 real(8) function timer_mod::time_mls ()

Definition at line 159 of file timer_mod.f90.

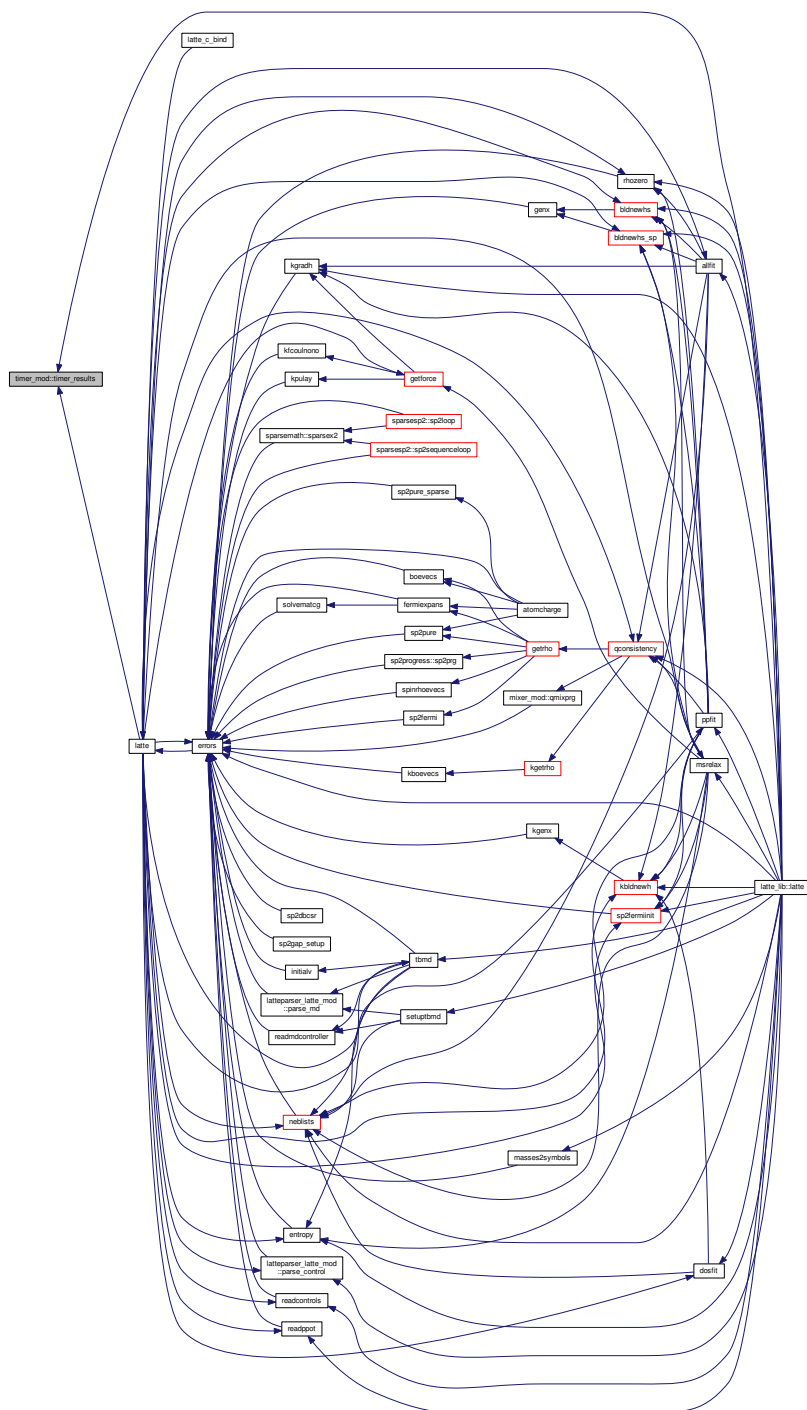
Here is the caller graph for this function:



Definition at line 145 of file timer_mod.f90.

Definition at line 117 of file timer_mod.f90.

Here is the caller graph for this function:



6.33.2 Variable Documentation

6.33.2.1 integer timer_mod::dense2sparse_timer

Definition at line 29 of file timer_mod.f90.

6.33.2.2 integer timer_mod::dmbuild_timer

Definition at line 29 of file [timer_mod.f90](#).

6.33.2.3 integer timer_mod::latte_timer

Definition at line 29 of file [timer_mod.f90](#).

6.33.2.4 integer timer_mod::num_timers

Definition at line 28 of file [timer_mod.f90](#).

6.33.2.5 integer timer_mod::sp2all_timer

Definition at line 30 of file [timer_mod.f90](#).

6.33.2.6 integer timer_mod::sp2sparse_timer

Definition at line 30 of file [timer_mod.f90](#).

6.33.2.7 integer timer_mod::sparse2dense_timer

Definition at line 29 of file [timer_mod.f90](#).

6.33.2.8 real, dimension(:), allocatable timer_mod::tavg

Definition at line 32 of file [timer_mod.f90](#).

6.33.2.9 integer timer_mod::tclock_max

Definition at line 27 of file [timer_mod.f90](#).

6.33.2.10 integer timer_mod::tclock_rate

Definition at line 27 of file [timer_mod.f90](#).

6.33.2.11 integer, dimension(:), allocatable timer_mod::tcount

Definition at line 31 of file [timer_mod.f90](#).

6.33.2.12 `character(len=20), dimension(:), allocatable timer_mod::tname`

Definition at line 33 of file [timer_mod.f90](#).

6.33.2.13 `real, dimension(:), allocatable timer_mod::tpercent`

Definition at line 32 of file [timer_mod.f90](#).

6.33.2.14 `integer, dimension(:), allocatable timer_mod::tstart`

Definition at line 31 of file [timer_mod.f90](#).

6.33.2.15 `integer timer_mod::tstart_clock`

Definition at line 27 of file [timer_mod.f90](#).

6.33.2.16 `integer timer_mod::tstop_clock`

Definition at line 27 of file [timer_mod.f90](#).

6.33.2.17 `real, dimension(:), allocatable timer_mod::tsum`

Definition at line 32 of file [timer_mod.f90](#).

6.33.2.18 `integer, dimension(:), allocatable timer_mod::total`

Definition at line 31 of file [timer_mod.f90](#).

6.33.2.19 `integer timer_mod::tx`

Definition at line 28 of file [timer_mod.f90](#).

6.34 univarray Module Reference

Variables

- `real(latteprec), dimension(:, :), allocatable bond`
- `real(latteprec), dimension(:, :), allocatable overl`
- `real(latteprec), dimension(:, :), allocatable pair`

6.34.1 Variable Documentation

6.34.1.1 `real(latteprec), dimension(:,:), allocatable univarray::bond`

Definition at line 29 of file [univarray.f90](#).

6.34.1.2 `real(latteprec), dimension(:,:), allocatable univarray::overl`

Definition at line 29 of file [univarray.f90](#).

6.34.1.3 `real(latteprec), dimension(:,:), allocatable univarray::pair`

Definition at line 29 of file [univarray.f90](#).

6.35 virialarray Module Reference

Variables

- `real(latteprec), dimension(6)` [keten](#)
- `real(latteprec)` [pressure](#)
- `real(latteprec), dimension(6)` [strten](#)
- `real(latteprec)` [sysvol](#)
- `real(latteprec), dimension(6)` [virbond](#)
- `real(latteprec), dimension(6)` [vircoul](#)
- `real(latteprec), dimension(6)` [virial](#)
- `real(latteprec), dimension(6)` [virpair](#)
- `real(latteprec), dimension(6)` [virpul](#)
- `real(latteprec), dimension(6)` [virscoul](#)
- `real(latteprec), dimension(6)` [virsspin](#)

6.35.1 Variable Documentation

6.35.1.1 `real(latteprec), dimension(6) virialarray::keten`

Definition at line 43 of file [virialarray.f90](#).

6.35.1.2 `real(latteprec) virialarray::pressure`

Definition at line 44 of file [virialarray.f90](#).

6.35.1.3 `real(latteprec), dimension(6) virialarray::strten`

Definition at line 43 of file [virialarray.f90](#).

6.35.1.4 `real(latteprec) virialarray::sysvol`

Definition at line 44 of file [virialarray.f90](#).

6.35.1.5 `real(latteprec), dimension(6) virialarray::virbond`

Definition at line 41 of file [virialarray.f90](#).

6.35.1.6 `real(latteprec), dimension(6) virialarray::vircoul`

Definition at line 41 of file [virialarray.f90](#).

6.35.1.7 `real(latteprec), dimension(6) virialarray::virial`

Definition at line 41 of file [virialarray.f90](#).

6.35.1.8 `real(latteprec), dimension(6) virialarray::virpair`

Definition at line 41 of file [virialarray.f90](#).

6.35.1.9 `real(latteprec), dimension(6) virialarray::virpul`

Definition at line 42 of file [virialarray.f90](#).

6.35.1.10 `real(latteprec), dimension(6) virialarray::virscoul`

Definition at line 42 of file [virialarray.f90](#).

6.35.1.11 `real(latteprec), dimension(6) virialarray::virsspin`

Definition at line 42 of file [virialarray.f90](#).

6.36 `xboarray` Module Reference

Variables

- `real(latteprec) alpha_xbo`
- `real(latteprec), dimension(:), allocatable chempot_pnk`
- `real(latteprec), dimension(10) cnk`
- `real(latteprec) kappa_scale`
- `real(latteprec) kappa_xbo`
- `real(latteprec) lastguess_chempot`
- `real(latteprec), dimension(:, :), allocatable pnk`
- `real(latteprec) prevlastguess_chempot`
- `real(latteprec), dimension(:, :), allocatable spin_pnk`

6.36.1 Variable Documentation

6.36.1.1 `real(latteprec) xboarray::alpha_xbo`

Definition at line 34 of file [xboarray.f90](#).

6.36.1.2 `real(latteprec), dimension(:), allocatable xboarray::chempot_pnk`

Definition at line 30 of file [xboarray.f90](#).

6.36.1.3 `real(latteprec), dimension(10) xboarray::cnk`

Definition at line 35 of file [xboarray.f90](#).

6.36.1.4 `real(latteprec) xboarray::kappa_scale`

Definition at line 34 of file [xboarray.f90](#).

6.36.1.5 `real(latteprec) xboarray::kappa_xbo`

Definition at line 34 of file [xboarray.f90](#).

6.36.1.6 `real(latteprec) xboarray::lastguess_chempot`

Definition at line 32 of file [xboarray.f90](#).

6.36.1.7 `real(latteprec), dimension(:, :), allocatable xboarray::pnk`

Definition at line 29 of file [xboarray.f90](#).

6.36.1.8 `real(latteprec) xboarray::prevlastguess_chempot`

Definition at line 32 of file [xboarray.f90](#).

6.36.1.9 `real(latteprec), dimension(:, :), allocatable xboarray::spin_pnk`

Definition at line 31 of file [xboarray.f90](#).

Chapter 7

Class Documentation

7.1 `constraints_mod::constrains_type` Type Reference

General input variables for constrains.

Public Attributes

- `character(20)` [method](#)
Constrain method.
- `integer` [verbose](#)
Verbosity level.

7.1.1 Detailed Description

General input variables for constrains.

Todo construct a parser for the constrains

Definition at line [50](#) of file [constraints_mod.f90](#).

7.1.2 Member Data Documentation

7.1.2.1 `character(20)` `constraints_mod::constrains_type::method`

Constrain method.

Definition at line [56](#) of file [constraints_mod.f90](#).

7.1.2.2 integer constraints_mod::constrains_type::verbose

Verbosity level.

Definition at line 53 of file [constraints_mod.f90](#).

The documentation for this type was generated from the following file:

- [constraints_mod.f90](#)

7.2 latteparser_latte_mod::latte_type Type Reference

General latte input variables type.

Public Attributes

- character(20) [bml_dmode](#)
Distribution mode (sequential, distributed, or graph_distributed).
- character(20) [bml_type](#)
Matrix format (Dense or Ellpack).
- character(100) [coordsfile](#)
File containing coordinates.
- real(dp) [coul_acc](#)
Coulomb Accuracy.
- real(dp) [efermi](#)
Restart calculation.
- character(20) [jobname](#)
Name of the current job.
- integer [maxscf](#)
Maximum SCF iterations.
- integer [mdim](#)
Max nonzero elements per row for every row see [1].
- integer [mdsteps](#)
Total number of steps for MD simulation.
- character(20) [method](#)
Solver method.
- real(dp) [mixcoeff](#)
Linear mixing coefficient.
- integer [mpulay](#)
Coulomb Accuracy.
- integer [nlisteach](#)
File containing coordinates.
- character(100) [parampath](#)
Total number of steps for MD simulation.
- real(dp) [pulaycoeff](#)
Pulay mixing coefficient.
- logical [restart](#)
Restart calculation.

- real(dp) [scftol](#)
SCF tolerance.
- real(dp) [threshold](#)
Threshold values for matrix elements.
- real(dp) [timeratio](#)
Estimated ration between real & k space time efficiency.
- real(dp) [timestep](#)
Total number of steps for MD simulation.
- integer [verbose](#)
Verbosity level.
- character(20) [zmat](#)
Z Matrix calculation type.

7.2.1 Detailed Description

General latte input variables type.

Definition at line 42 of file [latteparser_latte_mod.f90](#).

7.2.2 Member Data Documentation

7.2.2.1 character(20) latteparser_latte_mod::latte_type::bml_dmode

Distribution mode (sequential, distributed, or graph_distributed).

Definition at line 60 of file [latteparser_latte_mod.f90](#).

7.2.2.2 character(20) latteparser_latte_mod::latte_type::bml_type

Matrix format (Dense or Ellpack).

Definition at line 57 of file [latteparser_latte_mod.f90](#).

7.2.2.3 character(100) latteparser_latte_mod::latte_type::coordsfile

File containing coordinates.

Definition at line 99 of file [latteparser_latte_mod.f90](#).

7.2.2.4 real(dp) latteparser_latte_mod::latte_type::coul_acc

Coulomb Accuracy.

Definition at line 63 of file [latteparser_latte_mod.f90](#).

7.2.2.5 `real(dp) latteparser_latte_mod::latte_type::efermi`

Restart calculation.

Definition at line 108 of file [latteparser_latte_mod.f90](#).

7.2.2.6 `character(20) latteparser_latte_mod::latte_type::jobname`

Name of the current job.

Definition at line 45 of file [latteparser_latte_mod.f90](#).

7.2.2.7 `integer latteparser_latte_mod::latte_type::maxscf`

Maximum SCF iterations.

Definition at line 75 of file [latteparser_latte_mod.f90](#).

7.2.2.8 `integer latteparser_latte_mod::latte_type::mdim`

Max nonzero elements per row for every row see [1] .

Definition at line 54 of file [latteparser_latte_mod.f90](#).

7.2.2.9 `integer latteparser_latte_mod::latte_type::mdsteps`

Total number of steps for MD simulation.

Definition at line 90 of file [latteparser_latte_mod.f90](#).

7.2.2.10 `character(20) latteparser_latte_mod::latte_type::method`

Solver method.

Definition at line 84 of file [latteparser_latte_mod.f90](#).

7.2.2.11 `real(dp) latteparser_latte_mod::latte_type::mixcoeff`

Linear mixing coefficient.

Definition at line 69 of file [latteparser_latte_mod.f90](#).

7.2.2.12 `integer latteparser_latte_mod::latte_type::mpulay`

Coulomb Accuracy.

Definition at line 72 of file [latteparser_latte_mod.f90](#).

7.2.2.13 integer latteparser_latte_mod::latte_type::nlisteach

File containing coordinates.

Definition at line 102 of file [latteparser_latte_mod.f90](#).

7.2.2.14 character(100) latteparser_latte_mod::latte_type::parampath

Total number of steps for MD simulation.

Definition at line 96 of file [latteparser_latte_mod.f90](#).

7.2.2.15 real(dp) latteparser_latte_mod::latte_type::pulaycoeff

Pulay mixing coefficient.

Definition at line 66 of file [latteparser_latte_mod.f90](#).

7.2.2.16 logical latteparser_latte_mod::latte_type::restart

Restart calculation.

Definition at line 105 of file [latteparser_latte_mod.f90](#).

7.2.2.17 real(dp) latteparser_latte_mod::latte_type::scftol

SCF tolerance.

Definition at line 78 of file [latteparser_latte_mod.f90](#).

7.2.2.18 real(dp) latteparser_latte_mod::latte_type::threshold

Threshold values for matrix elements.

Definition at line 51 of file [latteparser_latte_mod.f90](#).

7.2.2.19 real(dp) latteparser_latte_mod::latte_type::timeratio

Estimated ration between real & k space time efficiency.

Definition at line 87 of file [latteparser_latte_mod.f90](#).

7.2.2.20 real(dp) latteparser_latte_mod::latte_type::timestep

Total number of steps for MD simulation.

Definition at line 93 of file [latteparser_latte_mod.f90](#).

7.2.2.21 integer lattparser_latte_mod::latte_type::verbose

Verbosity level.

Definition at line 48 of file [lattparser_latte_mod.f90](#).

7.2.2.22 character(20) lattparser_latte_mod::latte_type::zmat

Z Matrix calculation type.

Definition at line 81 of file [lattparser_latte_mod.f90](#).

The documentation for this type was generated from the following file:

- [lattparser_latte_mod.f90](#)

Chapter 8

File Documentation

8.1 /home/christian/LATTE/README.md File Reference

8.2 /home/christian/LATTE/README.md

```
00001 [![Build
      Status](https://travis-ci.org/lanl/LATTE.svg?branch=master)](https://travis-ci.org/lanl/LATTE)
00002
00003 [![codecov](https://codecov.io/gh/lanl/LATTE/branch/master/graph/badge.svg)](https://codecov.io/gh/lanl/LATTE)
00004 # LATTE
00005
00006 Open source density functional tight binding molecular dynamics.
00007
00008 #LA-CC-10-004
00009
00010 This material was produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
      Laboratory (LANL), which is operated by Los Alamos National Security, LLC for the U.S. Department of Energy.
      The U.S. Government has rights to use, reproduce, and distribute this software. NEITHER THE GOVERNMENT NOR
      LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE
      USE OF THIS SOFTWARE. If software is modified to produce derivative works, such modified software should be
      clearly marked, so as not to confuse it with the version available from LANL.
00011
00012 Additionally, this program is free software; you can redistribute it and/or modify it under the terms
      of the GNU General Public License as published by the Free Software Foundation; version 2.0 of the License.
      Accordingly, this program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
      without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
      Public License for more details.
00013
00014 # Authors
00015
00016 Nicolas Bock (T-1)
00017 Marc J. Cawkwell (T-1)
00018 Josh D. Coe (T-1)
00019 Aditi Krishnapriyan (T-1)
00020 Matthew P. Kroonblawd (T-1)
00021 Adam Lang (T-1)
00022 Enrique Martinez Saez (MST-8)
00023 Susan M. Mniszewski (CCS-3)
00024 Christian F. A. Negre (T-1)
00025 Anders M. N. Niklasson (T-1)
00026 Edward Sanville (T-1)
00027 Mitchell A. Wood (T-1)
00028 Ping Yang (T-1)
00029
00030 Los Alamos National Laboratory
00031
00032
00033 # Citing
00034
00035 To cite the code, please proceed as follows:
00036
00037 [![DOI](https://zenodo.org/badge/75976231.svg)](https://zenodo.org/badge/latestdoi/75976231)
00038
00039 with the following 'bibtex' citation:
00040
```

```
00041      @misc{LATTE,  
00042          title = {LATTE},  
00043          url = {https://github.com/lanl/LATTE},  
00044          author = {N. Bock and M. J. Cawkwell and J. D. Coe and A. Krishnapriyan and M. P. Kroonblawd and  
A. Lang and E. Martinez Saez and S. M. Mniszewski and C. F. A. Negre and A. M. N. Niklasson and E. Sanville  
and M. A. Wood and P. Yang},  
00045          year = {2008}  
00046      }
```

8.3 addqdep.f90 File Reference

Functions/Subroutines

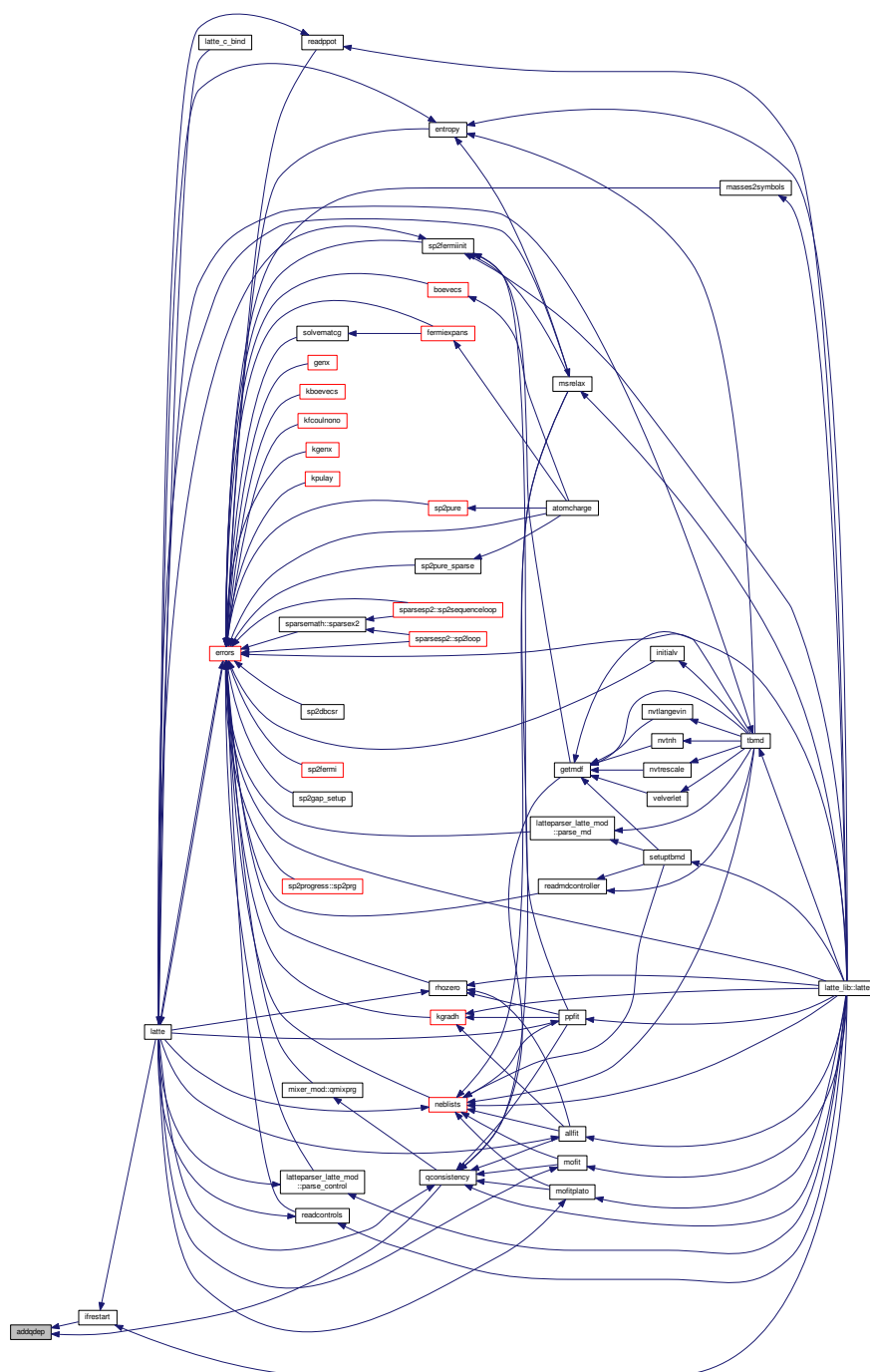
- subroutine [addqdep](#)

8.3.1 Function/Subroutine Documentation

8.3.1.1 subroutine `addqdep` ()

Definition at line 23 of file [addqdep.f90](#).

Here is the caller graph for this function:



8.4 addqdep.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,       !

```

```

00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS      !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such    !
00010 ! modified software should be clearly marked, so as not to confuse it     !
00011 ! with the version available from LANL.                                   !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it     !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be     !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of   !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                         !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE addqdep
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE coulombarray
00027     USE nonoarray
00028     USE kspacearray
00029     USE myprecision
00030
00031     IMPLICIT NONE
00032
00033     INTEGER :: I, J, K
00034     INTEGER :: INDEX, NUMORB
00035     REAL(LATTEPREC) :: ES, EP, ED, EF
00036     REAL(LATTEPREC) :: HMOD
00037     COMPLEX(LATTEPREC) :: ZHMOD
00038     IF (existerror) RETURN
00039
00040     index = 0
00041
00042     IF (kon .EQ. 0) THEN
00043
00044         IF ( basistype .EQ. "ORTHO") THEN
00045
00046             DO i = 1, nats
00047
00048                 hmod = hubbardu(elempointer(i))*deltaq(i) +
00049 coulombbv(i)
00049
00050                 SELECT CASE(basis(elempointer(i)))
00051
00052                     CASE("s")
00053
00054                         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00055                         index = index + 1
00056
00057                     CASE("p")
00058
00059                         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00060                         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00061                         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00062                         index = index + 3
00063
00064                     CASE("d")
00065
00066                         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00067                         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00068                         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00069                         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00070                         h(index + 5, index + 5) = hdiag(index + 5) + hmod
00071                         index = index + 5
00072
00073                     CASE("f")
00074
00075                         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00076                         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00077                         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00078                         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00079                         h(index + 5, index + 5) = hdiag(index + 5) + hmod
00080                         h(index + 6, index + 6) = hdiag(index + 6) + hmod
00081                         h(index + 7, index + 7) = hdiag(index + 7) + hmod
00082                         index = index + 7
00083
00084                     CASE("sp")
00085
00086                         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00087                         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00088                         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00089                         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00090                         index = index + 4
00091
00092                     CASE("sd")
00093

```



```

00094      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00095      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00096      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00097      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00098      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00099      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00100      index = index + 6
00101
00102      CASE("sf")
00103
00104      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00105      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00106      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00107      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00108      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00109      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00110      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00111      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00112      index = index + 8
00113
00114      CASE("pd")
00115
00116      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00117      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00118      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00119      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00120      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00121      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00122      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00123      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00124      index = index + 8
00125
00126      CASE("pf")
00127
00128      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00129      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00130      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00131      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00132      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00133      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00134      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00135      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00136      h(index + 9, index + 9) = hdiag(index + 9) + hmod
00137      h(index + 10, index + 10) = hdiag(index + 10) + hmod
00138      index = index + 10
00139
00140      CASE("df")
00141
00142      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00143      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00144      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00145      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00146      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00147      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00148      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00149      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00150      h(index + 9, index + 9) = hdiag(index + 9) + hmod
00151      h(index + 10, index + 10) = hdiag(index + 10) + hmod
00152      h(index + 11, index + 11) = hdiag(index + 11) + hmod
00153      h(index + 12, index + 12) = hdiag(index + 12) + hmod
00154      index = index + 12
00155
00156      CASE("spd")
00157
00158      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00159      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00160      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00161      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00162      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00163      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00164      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00165      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00166      h(index + 9, index + 9) = hdiag(index + 9) + hmod
00167      index = index + 9
00168
00169      CASE("spf")
00170
00171      h(index + 1, index + 1) = hdiag(index + 1) + hmod
00172      h(index + 2, index + 2) = hdiag(index + 2) + hmod
00173      h(index + 3, index + 3) = hdiag(index + 3) + hmod
00174      h(index + 4, index + 4) = hdiag(index + 4) + hmod
00175      h(index + 5, index + 5) = hdiag(index + 5) + hmod
00176      h(index + 6, index + 6) = hdiag(index + 6) + hmod
00177      h(index + 7, index + 7) = hdiag(index + 7) + hmod
00178      h(index + 8, index + 8) = hdiag(index + 8) + hmod
00179      h(index + 9, index + 9) = hdiag(index + 9) + hmod
00180      h(index + 10, index + 10) = hdiag(index + 10) + hmod

```

```

00181         h(index + 11, index + 11) = hdiag(index + 11) + hmod
00182         index = index + 11
00183
00184     CASE("sdf")
00185
00186         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00187         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00188         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00189         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00190         h(index + 5, index + 5) = hdiag(index + 5) + hmod
00191         h(index + 6, index + 6) = hdiag(index + 6) + hmod
00192         h(index + 7, index + 7) = hdiag(index + 7) + hmod
00193         h(index + 8, index + 8) = hdiag(index + 8) + hmod
00194         h(index + 9, index + 9) = hdiag(index + 9) + hmod
00195         h(index + 10, index + 10) = hdiag(index + 10) + hmod
00196         h(index + 11, index + 11) = hdiag(index + 11) + hmod
00197         h(index + 12, index + 12) = hdiag(index + 12) + hmod
00198         h(index + 13, index + 13) = hdiag(index + 13) + hmod
00199         index = index + 13
00200
00201     CASE("pdf")
00202
00203         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00204         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00205         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00206         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00207         h(index + 5, index + 5) = hdiag(index + 5) + hmod
00208         h(index + 6, index + 6) = hdiag(index + 6) + hmod
00209         h(index + 7, index + 7) = hdiag(index + 7) + hmod
00210         h(index + 8, index + 8) = hdiag(index + 8) + hmod
00211         h(index + 9, index + 9) = hdiag(index + 9) + hmod
00212         h(index + 10, index + 10) = hdiag(index + 10) + hmod
00213         h(index + 11, index + 11) = hdiag(index + 11) + hmod
00214         h(index + 12, index + 12) = hdiag(index + 12) + hmod
00215         h(index + 13, index + 13) = hdiag(index + 13) + hmod
00216         h(index + 14, index + 14) = hdiag(index + 14) + hmod
00217         h(index + 15, index + 15) = hdiag(index + 15) + hmod
00218         index = index + 15
00219
00220     CASE("spdf")
00221
00222         h(index + 1, index + 1) = hdiag(index + 1) + hmod
00223         h(index + 2, index + 2) = hdiag(index + 2) + hmod
00224         h(index + 3, index + 3) = hdiag(index + 3) + hmod
00225         h(index + 4, index + 4) = hdiag(index + 4) + hmod
00226         h(index + 5, index + 5) = hdiag(index + 5) + hmod
00227         h(index + 6, index + 6) = hdiag(index + 6) + hmod
00228         h(index + 7, index + 7) = hdiag(index + 7) + hmod
00229         h(index + 8, index + 8) = hdiag(index + 8) + hmod
00230         h(index + 9, index + 9) = hdiag(index + 9) + hmod
00231         h(index + 10, index + 10) = hdiag(index + 10) + hmod
00232         h(index + 11, index + 11) = hdiag(index + 11) + hmod
00233         h(index + 12, index + 12) = hdiag(index + 12) + hmod
00234         h(index + 13, index + 13) = hdiag(index + 13) + hmod
00235         h(index + 14, index + 14) = hdiag(index + 14) + hmod
00236         h(index + 15, index + 15) = hdiag(index + 15) + hmod
00237         h(index + 16, index + 16) = hdiag(index + 16) + hmod
00238         index = index + 16
00239
00240     END SELECT
00241
00242 ENDDO
00243
00244 ELSEIF ( basistype .EQ. "NONORTHO" ) THEN
00245
00246     !
00247     ! When we have a non-orthogonal basis, the electrostatic
00248     ! potential enters as SH_1
00249     !
00250
00251     DO i = 1, nats
00252
00253         hmod = hubbardu(elempointer(i))*deltaq(i) +
00254 coulombv(i)
00255
00256         SELECT CASE(basis(elempointer(i)))
00257
00258             CASE("s")
00259                 numorb = 1
00260             CASE("p")
00261                 numorb = 3
00262             CASE("d")
00263                 numorb = 5
00264             CASE("f")
00265                 numorb = 7
00266             CASE("sp")
00267                 numorb = 4

```

```

00267         CASE("sd")
00268             numorb = 6
00269         CASE("sf")
00270             numorb = 8
00271         CASE("pd")
00272             numorb = 8
00273         CASE("pf")
00274             numorb = 10
00275         CASE("df")
00276             numorb = 12
00277         CASE("spd")
00278             numorb = 9
00279         CASE("spf")
00280             numorb = 11
00281         CASE("sdf")
00282             numorb = 13
00283         CASE("pdf")
00284             numorb = 15
00285         CASE("spdf")
00286             numorb = 16
00287         END SELECT
00288
00289         DO j = 1, numorb
00290
00291             index = index + 1
00292             hjj(index) = hmod
00293
00294         ENDDO
00295
00296     ENDDO
00297
00298     ! H = H_0 + S*H_1
00299
00300     DO i = 1, hdim
00301         DO j = 1, hdim
00302
00303             h(j,i) = h0(j,i) + smat(j,i)*(hjj(i) + hjj(j))/two
00304
00305         ENDDO
00306     ENDDO
00307
00308     ENDIF
00309
00310     ELSE
00311         ! IF (KON .EQ. 1) THEN
00312         ! k-space - we have to add the potential to all NKTOT Hamiltonians
00313
00314         ! Orthogonal basis only at the moment
00315
00316         IF ( basistype .EQ. "ORTHO") THEN
00317
00318             DO k = 1, nktot
00319
00320                 index = 0
00321
00322                 DO i = 1, nats
00323
00324                     zhmod = cmplx(hubbardu(elempointer(i))*deltaq(i) &
00325                                + coulombv(i))
00326
00327                     SELECT CASE(basis(elempointer(i)))
00328
00329                         CASE("s")
00330                             numorb = 1
00331                         CASE("p")
00332                             numorb = 3
00333                         CASE("d")
00334                             numorb = 5
00335                         CASE("f")
00336                             numorb = 7
00337                         CASE("sp")
00338                             numorb = 4
00339                         CASE("sd")
00340                             numorb = 6
00341                         CASE("sf")
00342                             numorb = 8
00343                         CASE("pd")
00344                             numorb = 8
00345                         CASE("pf")
00346                             numorb = 10
00347                         CASE("df")
00348                             numorb = 12
00349                         CASE("spd")
00350                             numorb = 9
00351                         CASE("spf")
00352                             numorb = 11
00353                         CASE("sdf")

```

```

00354         numorb = 13
00355         CASE("pdf")
00356             numorb = 15
00357         CASE("spdf")
00358             numorb = 16
00359         END SELECT
00360
00361         DO j = 1, numorb
00362
00363             index = index + 1
00364             hk(index, index, k) = hkdiag(index, k) + zhmod
00365
00366         ENDDO
00367
00368     ENDDO
00369
00370 ENDDO
00371
00372 ELSEIF (basistype .EQ. "NONORTHO") THEN ! k-space and non-orthogonal basis
00373
00374     DO i = 1, nats
00375
00376         hmod = hubbardu(elempointer(i))*deltaq(i) +
coulombv(i)
00377
00378         SELECT CASE(basis(elempointer(i)))
00379
00380             CASE("s")
00381                 numorb = 1
00382             CASE("p")
00383                 numorb = 3
00384             CASE("d")
00385                 numorb = 5
00386             CASE("f")
00387                 numorb = 7
00388             CASE("sp")
00389                 numorb = 4
00390             CASE("sd")
00391                 numorb = 6
00392             CASE("sf")
00393                 numorb = 8
00394             CASE("pd")
00395                 numorb = 8
00396             CASE("pf")
00397                 numorb = 10
00398             CASE("df")
00399                 numorb = 12
00400             CASE("spd")
00401                 numorb = 9
00402             CASE("spf")
00403                 numorb = 11
00404             CASE("sdf")
00405                 numorb = 13
00406             CASE("pdf")
00407                 numorb = 15
00408             CASE("spdf")
00409                 numorb = 16
00410             END SELECT
00411
00412         DO j = 1, numorb
00413
00414             index = index + 1
00415             zhjj(index) = cmplx(hmod)
00416
00417         ENDDO
00418
00419     ENDDO
00420
00421     ! H = H_0 + S*H_1
00422
00423     DO k = 1, nktot
00424         DO i = 1, hdim
00425             DO j = 1, hdim
00426
00427                 hk(j,i,k) = hk0(j,i,k) + sk(j,i,k)*(zhjj(i) + zhjj(j))/
two
00428
00429             ENDDO
00430         ENDDO
00431     ENDDO
00432
00433 ENDIF
00434
00435
00436
00437 ENDIF
00438

```

```

00439  IF (debugon .EQ. 1) THEN
00440
00441      OPEN(unit=31, status="UNKNOWN", file="myH.dat")
00442
00443      print*, "Caution - the H0+H1 matrix is being written to file"
00444
00445      IF (kon .EQ. 0) THEN
00446
00447          DO i = 1, hdim
00448              WRITE(31,10) (h(i,j), j = 1, hdim)
00449          ENDDO
00450
00451      ELSE
00452
00453          DO k = 1, nktot
00454              WRITE(31,*) k
00455              DO i = 1, hdim
00456                  WRITE(31,12) (hk(i,j,k), j = 1, hdim)
00457              ENDDO
00458          ENDDO
00459
00460      ENDIF
00461
00462      CLOSE(31)
00463
00464  ENDIF
00465
00466 10 FORMAT(100g18.9)
00467 11 FORMAT(i5,100g18.9)
00468 12 FORMAT(100f8.3)
00469  RETURN
00470
00471 END SUBROUTINE addqdep
00472
00473

```

8.5 algo.f90 File Reference

8.6 algo.f90

```

00001 PROGRAM algo
00002   INTEGER :: i
00003   DO i = 1, n
00004       run = 1
00005       enndo
00006   END DO
00007

```

8.7 allfit.f90 File Reference

Functions/Subroutines

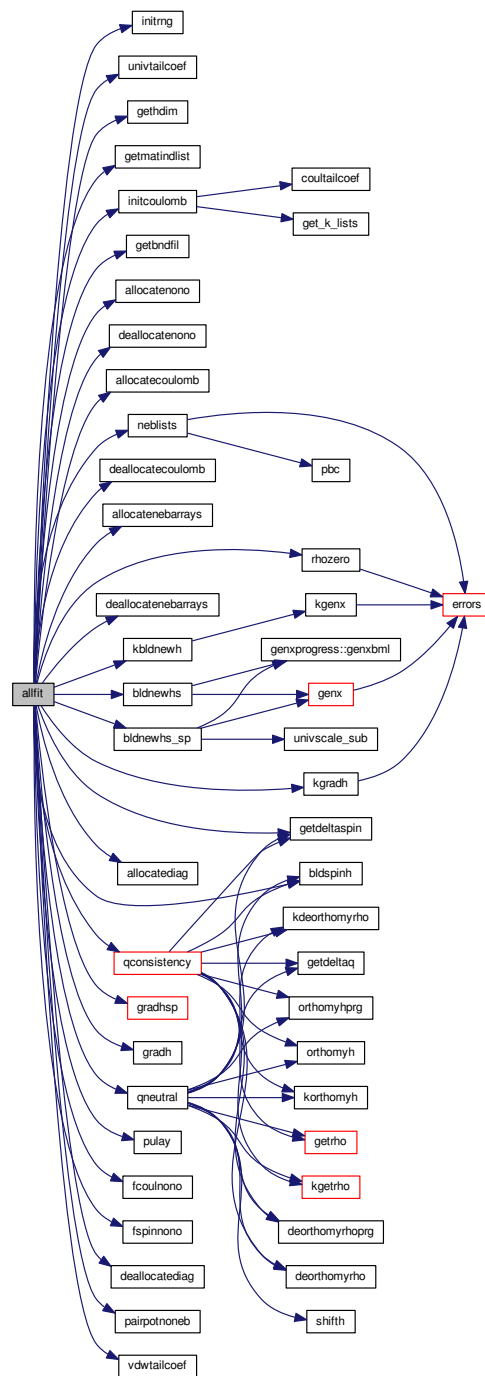
- subroutine [allfit](#)

8.7.1 Function/Subroutine Documentation

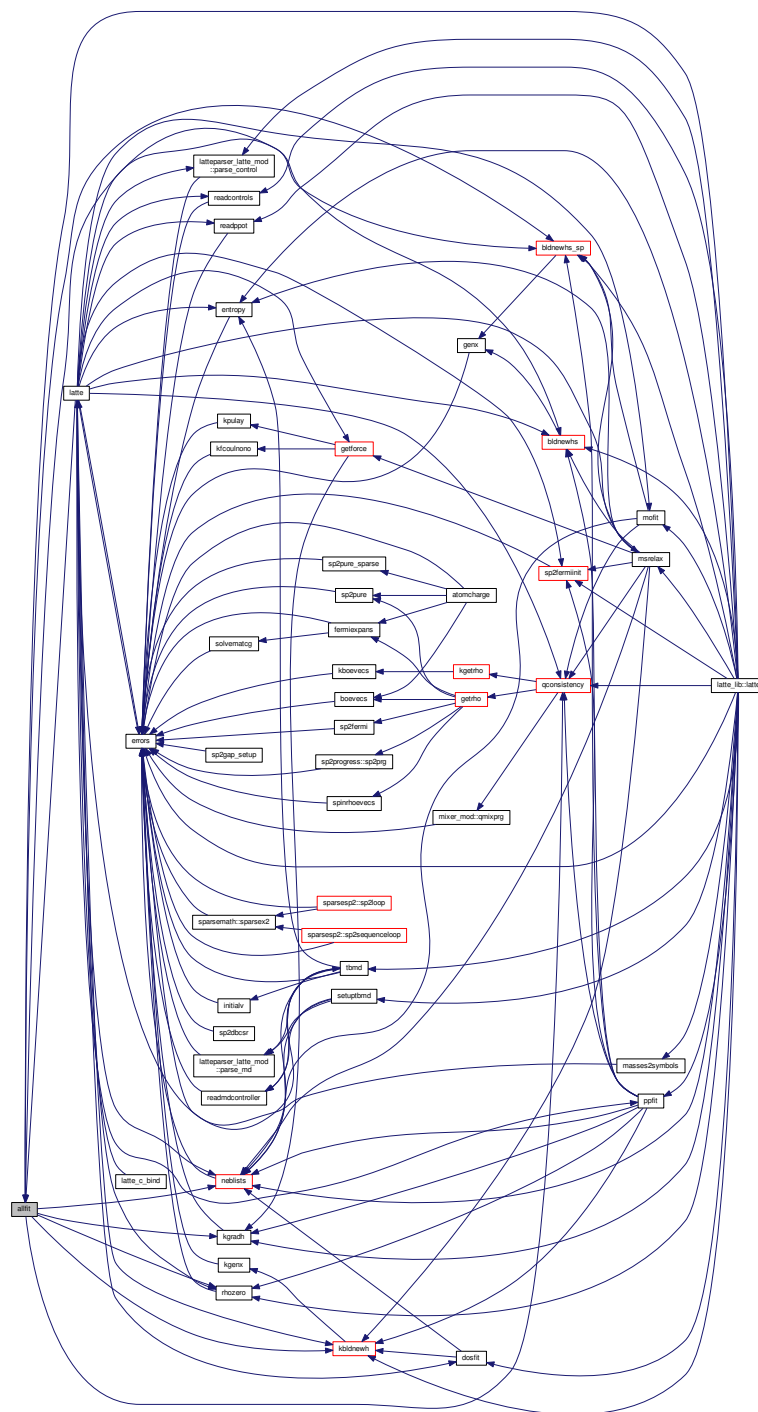
8.7.1.1 subroutine allfit ()

Definition at line 23 of file [allfit.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.8 allfit.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allfit
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE kspacearray
00029   USE neblistarray
00030   USE univarray
00031   USE ppotarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, Z, NSNAP, ATMOL, TOTAT, COUNT, II, COUNT2, ACC
00037   INTEGER :: PICK, PARAMPICK, OLOOP, BACC
00038   INTEGER, ALLOCATABLE :: ATINMOL(:)
00039   REAL(LATTEPREC), ALLOCATABLE :: COORDS(:,:), FORCES(:,:), FORCEIN(:,:)
00040   REAL(LATTEPREC), ALLOCATABLE :: BONDFORCES(:,:), FPAIR(:,:)
00041   REAL(LATTEPREC) :: Y, ENERGY, MYERR, MYERRINIT, MINERR, PREVERR
00042   REAL(LATTEPREC) :: RN, TMPERR
00043   REAL(LATTEPREC) :: BONDMINERR, BONDPREVERR
00044   REAL(LATTEPREC), ALLOCATABLE :: PPBEST(:,:), PPOLD(:,:), PPKEEP(:,:)
00045   REAL(LATTEPREC), ALLOCATABLE :: BINTBEST(:,:), BINTOLD(:,:)
00046   CHARACTER(LEN=2) :: X
00047   CHARACTER(LEN=2), ALLOCATABLE :: SPEC(:)
00048   IF (existerror) RETURN
00049
00050   ALLOCATE(ppbest(5,pp2fit), ppold(5,pp2fit), ppkeep(5,pp2fit))
00051   ALLOCATE(bintbest(2,bint2fit), bintold(2,bint2fit))
00052
00053   CALL initrng
00054
00055   ! Read the xyz files and gradients
00056
00057   OPEN(unit=60, status="OLD", file="forces")
00058
00059   nsnap = ppnmol*ppngeom
00060
00061   totat = 0
00062   DO i = 1, nsnap
00063     ! print*, I
00064     READ(60,*) atmol
00065     totat = totat+atmol
00066     READ(60,*) energy
00067     DO j = 1, atmol
00068       READ(60,*) x, y,y,y,y,y,y
00069     ENDDO
00070   ENDDO
00071
00072   rewind(60)
00073
00074   ALLOCATE(coords(3,totat), forces(3,totat), spec(totat), atinmol(nsnap))
00075   ALLOCATE(bondforces(3,totat), fpair(3,totat), forcein(3,totat))
00076
00077   count = 0
00078
00079   DO i = 1, nsnap
00080     READ(60,*) atinmol(i)
00081     READ(60,*) energy
00082
00083     DO j = 1, atinmol(i)
00084
00085       count = count + 1
00086
00087       READ(60,*) spec(count), coords(1,count), coords(2,count), &
00088         coords(3,count), forcein(1,count), forcein(2,count), &
00089         forcein(3,count)
00090
00091     ENDDO
00092   ENDDO
00093

```



```

00094 10 FORMAT(a1,6f12.6)
00095
00096   forcein = -forcein*27.211385/0.591772
00097
00098   ! First get the forces from the bond part and electrostatics
00099
00100   bacc = 0
00101
00102   DO oloop = 1, binfitstep
00103
00104       IF (oloop .GT. 1) THEN
00105
00106           DO i = 1, bint2fit
00107               DO j = 1, 2
00108                   bintold(j,i) = bond(j,i)
00109               ENDDO
00110           ENDDO
00111
00112           CALL random_number(rn)
00113
00114           pick = int(rn*REAL(bint2fit)) + 1
00115
00116           CALL random_number(rn)
00117
00118           parampick = int(rn*two) + 1
00119
00120           CALL random_number(rn)
00121
00122           bond(parampick,pick) = bond(parampick,pick) * &
00123               (one + ppsigma*(two*rn - one))
00124
00125           CALL univtailcoef(bond(:,pick))
00126
00127       ENDIF
00128
00129       count = 0
00130       count2 = 0
00131
00132       DO ii = 1, nsnap
00133
00134           DEALLOCATE(cr, atele, f, fpp, ftot)
00135           DEALLOCATE(deltaq, mycharge)
00136           DEALLOCATE(elempointer)
00137           IF (electro .EQ. 0) DEALLOCATE(lcnshift)
00138           IF (kon .EQ. 1) DEALLOCATE(kf)
00139           IF (basistype .EQ. "NONORTHO") THEN
00140               IF (spinon .EQ. 0) THEN
00141                   DEALLOCATE(fpul, fscoul)
00142               ELSE
00143                   DEALLOCATE(fpul, fscoul, fsspin)
00144               ENDIF
00145           ENDIF
00146
00147           ! Put the coordinates into CR
00148
00149           nats = atinmol(ii)
00150
00151           ALLOCATE(cr(3,nats), atele(nats), f(3,nats), fpp(3,
nats), ftot(3,nats))
00152           ALLOCATE(deltaq(nats), mycharge(nats))
00153           ALLOCATE(elempointer(nats))
00154
00155           IF (electro .EQ. 0) THEN
00156               ALLOCATE(lcnshift(nats))
00157               lcnshift = zero
00158           ENDIF
00159
00160           IF (kon .EQ. 1) ALLOCATE(kf(3,nats))
00161
00162           IF (basistype .EQ. "NONORTHO") THEN
00163               IF (spinon .EQ. 0) THEN
00164                   ALLOCATE(fpul(3,nats), fscoul(3,nats))
00165               ELSE
00166                   ALLOCATE(fpul(3,nats), fscoul(3,nats), fsspin(3,
nats))
00167               ENDIF
00168           ENDIF
00169
00170           DO j = 1, nats
00171               count = count + 1
00172               atele(j) = spec(count)
00173               cr(1,j) = coords(1,count)
00174               cr(2,j) = coords(2,count)
00175               cr(3,j) = coords(3,count)
00176           ENDDO
00177
00178           ! Set up pointer to the data in TBparam/electrons.dat

```

```

00179
00180      DO i = 1, nats
00181          DO j = 1, noelem
00182              IF (atele(i) .EQ. ele(j)) elempointer(i) = j
00183          ENDDO
00184      ENDDO
00185
00186      ! For use when getting the partial charges
00187
00188      DEALLOCATE(qlist)
00189
00190      ! Allocate the Hamiltonian matrix
00191
00192      IF (kon .EQ. 0) THEN
00193
00194          ! Real space
00195          DEALLOCATE(h, hdiag)
00196
00197      ELSE ! k-space
00198
00199          DEALLOCATE(hk, hkdiag)
00200
00201      ENDIF
00202
00203      IF (basistype .EQ. "NONORTHO") DEALLOCATE(h0)
00204
00205      IF (spinon .EQ. 0) THEN
00206
00207          ! No spins: allocate 1 double-occupied bond order matrix
00208
00209          IF (kon .EQ. 0) THEN
00210
00211              DEALLOCATE(bo)
00212
00213          ELSE
00214
00215              DEALLOCATE(kbo)
00216
00217          ENDIF
00218
00219      ELSEIF (spinon .EQ. 1) THEN
00220
00221          DEALLOCATE(hup, hdown)
00222          DEALLOCATE(rhoup, rhodown)
00223          DEALLOCATE(h2vect)
00224
00225          IF (basistype .EQ. "NONORTHO") DEALLOCATE(spinlist)
00226
00227          DEALLOCATE(deltaspin, olddeltaspin)
00228
00229      ENDIF
00230
00231      CALL gethdim
00232
00233      DEALLOCATE(matindlist)
00234
00235      IF (spinon .EQ. 1) DEALLOCATE(spinindlist)
00236
00237      CALL getmatindlist
00238
00239      IF (spinon .EQ. 0) THEN
00240          DEALLOCATE(bozero)
00241      ELSE
00242          DEALLOCATE(rhoupzero, rhodownzero)
00243      ENDIF
00244
00245      CALL rhozero
00246
00247      CALL getbndfil
00248
00249
00250      IF (basistype .EQ. "NONORTHO") THEN
00251          IF (ii .EQ. 1) THEN
00252              CALL allocatenono
00253          ELSE
00254              CALL deallocatenono
00255              CALL allocatenono
00256          ENDIF
00257      ENDIF
00258
00259      IF (ii .EQ. 1) THEN
00260
00261          CALL allocatcoulomb
00262          CALL initcoulomb
00263
00264      ELSE
00265

```

```

00266         CALL deallocatcoulomb
00267         CALL allocatcoulomb
00268         CALL initcoulomb
00269
00270     ENDIF
00271
00272     IF (ii .EQ. 1) THEN
00273
00274         CALL allocatenebarrays
00275         CALL neblists(0)
00276
00277     ELSE
00278
00279         CALL deallocatenebarrays
00280
00281         CALL allocatenebarrays
00282
00283         DEALLOCATE(nebtb, nebpp)
00284         IF (electro .EQ. 1) DEALLOCATE(nebcoul)
00285
00286         CALL neblists(0)
00287
00288     ENDIF
00289
00290     ! Now we have the arrays set up we can get the energy and forces
00291
00292     ! Build the charge independent H matrix
00293
00294     IF (kon .EQ. 0) THEN
00295
00296         IF (sponly .EQ. 0) THEN
00297             CALL bldnewhs_sp
00298         ELSE
00299             CALL bldnewhs
00300         ENDIF
00301
00302     ELSE
00303
00304         CALL kbldnewh
00305
00306     ENDIF
00307
00308     IF (spinon .EQ. 1) THEN
00309         CALL getdeltaspin
00310         CALL bldspinh
00311     ENDIF
00312
00313     CALL allocateddiag
00314
00315     IF (electro .EQ. 0) CALL qneutral(0,1) ! Local charge neutrality
00316
00317     IF (electro .EQ. 1) CALL qconsistency(0,1) ! Self-consistent charges
00318
00319     IF (kon .EQ. 0) THEN
00320
00321         IF (sponly .EQ. 0) THEN
00322             CALL gradhsp
00323         ELSE
00324             CALL gradh
00325         ENDIF
00326
00327     ELSE
00328
00329         CALL kgradh
00330
00331     ENDIF
00332
00333     ftot = two * f
00334
00335     IF (electro .EQ. 1) ftot = ftot + fcoul
00336
00337     IF (basistype .EQ. "NONORTHO") THEN
00338
00339         CALL pulay
00340
00341         CALL fcoulnono
00342
00343         ftot = ftot - two*fpul + fscoul
00344
00345         IF (spinon .EQ. 1) THEN
00346             CALL fspinnono
00347             ftot = ftot + fsspin
00348         ENDIF
00349
00350     ENDIF
00351
00352     DO i = 1, nats

```

```

00353         count2 = count2 + 1
00354         bondforces(1,count2) = ftot(1,i)
00355         bondforces(2,count2) = ftot(2,i)
00356         bondforces(3,count2) = ftot(3,i)
00357     ENDDO
00358
00359     CALL deallocatediag
00360
00361 ENDDO
00362
00363 IF (basistype .EQ. "NONORTHO") CALL deallocatenono
00364
00365 IF (electro .EQ. 1) CALL deallocatecoulomb
00366
00367 CALL deallocatenebarrays
00368
00369 DEALLOCATE(nebtb, nebpp)
00370 IF (electro .EQ. 1) DEALLOCATE(nebcoul)
00371
00372
00373 forces = forcein - bondforces
00374
00375
00376 ! Now for the optimization
00377
00378 ! Initial error
00379
00380 count = 0
00381 count2 = 0
00382
00383 DO ii = 1, nsnap
00384
00385     DEALLOCATE(cr, atele, fpp)
00386
00387     nats = atinmol(ii)
00388
00389     ALLOCATE(cr(3,nats), atele(nats), fpp(3,nats))
00390
00391     DO i = 1, nats
00392         count = count+1
00393         cr(1,i) = coords(1,count)
00394         cr(2,i) = coords(2,count)
00395         cr(3,i) = coords(3,count)
00396         atele(i) = spec(count)
00397     ENDDO
00398
00399     CALL pairpotnoneb
00400
00401     DO i = 1, nats
00402         count2 = count2 + 1
00403         fpair(1,count2) = fpp(1,i)
00404         fpair(2,count2) = fpp(2,i)
00405         fpair(3,count2) = fpp(3,i)
00406     ENDDO
00407
00408 ENDDO
00409
00410 myerrinit = zero
00411 DO i = 1, nsnap
00412
00413     tmperr = zero
00414     DO j = 1, atinmol(i)
00415
00416         tmperr = tmperr + &
00417             (forces(1,j) - fpair(1,j))*(forces(1,j) - fpair(1,j)) + &
00418             (forces(2,j) - fpair(2,j))*(forces(2,j) - fpair(2,j)) + &
00419             (forces(3,j) - fpair(3,j))*(forces(3,j) - fpair(3,j))
00420
00421     ENDDO
00422
00423     myerrinit = myerrinit + tmperr/REAL(atinmol(i))
00424
00425 ENDDO
00426
00427 myerrinit = myerrinit/REAL(nsnap)
00428 ! PRINT*, MYERRINIT
00429
00430 ! ALLOCATE(PPORIG(5,PP2FIT), PPBEST(5,PP2FIT), PPOLD(5,PP2FIT))
00431
00432 DO i = 1, pp2fit
00433     DO j = 1, 5
00434         ppbest(j,i) = potcoef(j,i)
00435     ENDDO
00436 ENDDO
00437
00438 ! PPBEST = PPORIG
00439

```

```

00440      acc = 0
00441      DO z = 1, ppnfitstep
00442
00443          ! Change the pair potential and the cut-offs too
00444
00445          DO i = 1, pp2fit
00446              DO j = 1, 5
00447                  ppold(j,i) = potcoef(j,i)
00448              ENDDO
00449          ENDDO
00450
00451          CALL random_number(rn)
00452
00453          pick = int(rn*REAL(pp2fit)) + 1
00454
00455          CALL random_number(rn)
00456
00457          parampick = int(rn*five) + 1
00458
00459          CALL random_number(rn)
00460
00461          potcoef(parampick,pick) = potcoef(parampick,pick) * &
00462              (one + ppsigma*(two*rn - one))
00463
00464          CALL vdwtailcoef
00465
00466          ! Here we get the contribution to the forces from the
00467          ! pair potential
00468
00469          count = 0
00470          count2 = 0
00471
00472          IF (z .EQ. 1) preverr = myerrinit
00473          IF (z .EQ. 1) minerr = myerrinit
00474
00475          DO ii = 1, nsnap
00476
00477              DEALLOCATE(cr, atele, fpp)
00478
00479              nats = atinmol(ii)
00480
00481              ALLOCATE(cr(3,nats), atele(nats), fpp(3,nats))
00482
00483              DO i = 1, nats
00484                  count = count+1
00485                  cr(1,i) = coords(1,count)
00486                  cr(2,i) = coords(2,count)
00487                  cr(3,i) = coords(3,count)
00488                  atele(i) = spec(count)
00489              ENDDO
00490
00491              CALL pairpotnoneb
00492
00493              DO i = 1, nats
00494                  count2 = count2 + 1
00495                  fpair(1,count2) = fpp(1,i)
00496                  fpair(2,count2) = fpp(2,i)
00497                  fpair(3,count2) = fpp(3,i)
00498              ENDDO
00499          ENDDO
00500
00501          myerr = zero
00502          DO i = 1, nsnap
00503
00504              tmperr = zero
00505              DO j = 1, atinmol(i)
00506
00507                  tmperr = tmperr + &
00508                      (forces(1,j) - fpair(1,j))*(forces(1,j) - fpair(1,j)) + &
00509                      (forces(2,j) - fpair(2,j))*(forces(2,j) - fpair(2,j)) + &
00510                      (forces(3,j) - fpair(3,j))*(forces(3,j) - fpair(3,j))
00511
00512              ENDDO
00513
00514              myerr = myerr + tmperr/REAL(atinmol(i))
00515          ENDDO
00516
00517          myerr = myerr/REAL(nsnap)
00518
00519          !      PRINT*, MYERR
00520
00521          IF (myerr .LT. preverr) THEN
00522
00523              acc = acc + 1
00524              preverr = myerr
00525          ENDIF
00526      END DO

```

```

00527
00528         !               PRINT*, ACC, MYERR
00529
00530     IF (myerr .LT. minerr) THEN
00531         minerr = myerr
00532
00533         DO i = 1, pp2fit
00534             DO j = 1, 5
00535                 ppbest(j,i) = potcoef(j,i)
00536             ENDDO
00537         ENDDO
00538
00539     ENDIF
00540
00541 ELSE
00542
00543     CALL random_number(rn)
00544
00545     IF (exp(-(myerr - preverr)*ppbeta) .GT. rn) THEN
00546         acc = acc+1
00547         preverr = myerr
00548
00549         !               PRINT*, ACC, MYERR
00550
00551     ELSE
00552
00553         ! Put the original coefficients back
00554
00555         DO i = 1, pp2fit
00556             DO j = 1, 5
00557                 potcoef(j,i) = ppold(j,i)
00558             ENDDO
00559         ENDDO
00560
00561     ENDIF
00562
00563 ENDIF
00564
00565 ENDDO
00566
00567 IF (oloop .EQ. 1) THEN
00568     bondpreverr = minerr
00569     bondminerr = minerr
00570
00571     DO i = 1, bint2fit
00572         DO j = 1, 2
00573             bintbest(j,i) = bond(j,i)
00574         ENDDO
00575     ENDDO
00576
00577 ELSE
00578     IF (minerr .LT. bondpreverr) THEN
00579         bacc = bacc + 1
00580         bondpreverr = minerr
00581         print*, bacc, minerr
00582
00583         IF (minerr .LT. bondminerr) THEN
00584             bondminerr = minerr
00585
00586             DO i = 1, bint2fit
00587                 DO j = 1, 2
00588                     bintbest(j,i) = bond(j,i)
00589                 ENDDO
00590             ENDDO
00591
00592             DO i = 1, pp2fit
00593                 DO j = 1, 5
00594                     ppkeep(j,i) = potcoef(j,i)
00595                 ENDDO
00596             ENDDO
00597
00598             DO i = 1, pp2fit
00599                 DO j = 1, 5
00600                     ppkeep(j,i) = potcoef(j,i)
00601                 ENDDO
00602             ENDDO
00603
00604         ENDIF
00605
00606     ELSE
00607
00608         CALL random_number(rn)
00609
00610         IF (exp(-(minerr - bondpreverr)*ppbeta) .GT. rn) THEN
00611             bacc = bacc+1
00612
00613

```

```

00614         bondpreverr = minerr
00615
00616         print*, bacc, minerr
00617
00618         ELSE
00619
00620             ! Put the original coefficients back
00621
00622             DO i = 1, bint2fit
00623                 DO j = 1, 2
00624                     bond(j,i) = bintold(j,i)
00625                 ENDDO
00626             ENDDO
00627
00628         ENDIF
00629
00630     ENDIF
00631
00632 ENDIF
00633
00634 ENDDO
00635
00636 DO i = 1, bint2fit
00637     WRITE(6,20) ele1(i), ele2(i), bintbest(1,i), bintbest(2,i)
00638 ENDDO
00639
00640
00641 DO i = 1, pp2fit
00642     WRITE(6,20) ppele1(i), ppele2(i), ppkeep(1,i), ppkeep(2,i), &
00643         ppkeep(3,i), ppkeep(4,i), ppkeep(5,i), potcoef(6,i), potcoef(7,i), &
00644         potcoef(8,i), potcoef(9,i), potcoef(10,i)
00645
00646 ENDDO
00647
00648 CALL allocatenebarrays
00649
00650 CALL neblists(0)
00651
00652 DEALLOCATE(ppkeep, ppold, ppbest)
00653 DEALLOCATE(bintbest, bintold)
00654
00655 20 FORMAT(a2, 1x, a2, 1x, 10f16.8)
00656
00657 RETURN
00658
00659 END SUBROUTINE allfit

```

8.9 allocatcoulomb.f90 File Reference

Functions/Subroutines

- subroutine [allocatcoulomb](#)

8.9.1 Function/Subroutine Documentation

8.9.1.1 subroutine allocatcoulomb ()

Definition at line 23 of file [allocatcoulomb.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.     !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of     !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatcoulomb
00023
00024   USE constants_mod
00025   USE coulombarray
00026   USE setuparray
00027
00028   IMPLICIT NONE
00029   IF (existerror) RETURN
00030
00031   ALLOCATE(olddeltaqs(nats))
00032   ALLOCATE(coulombv(nats))
00033   ALLOCATE(fcoul(3,nats))
00034   ALLOCATE(sinlist(nats), coslist(nats))
00035
00036   olddeltaqs = zero
00037   coulombv = zero
00038   fcoul = zero
00039
00040
00041 END SUBROUTINE allocatcoulomb

```

8.11 allocatediag.f90 File Reference

Functions/Subroutines

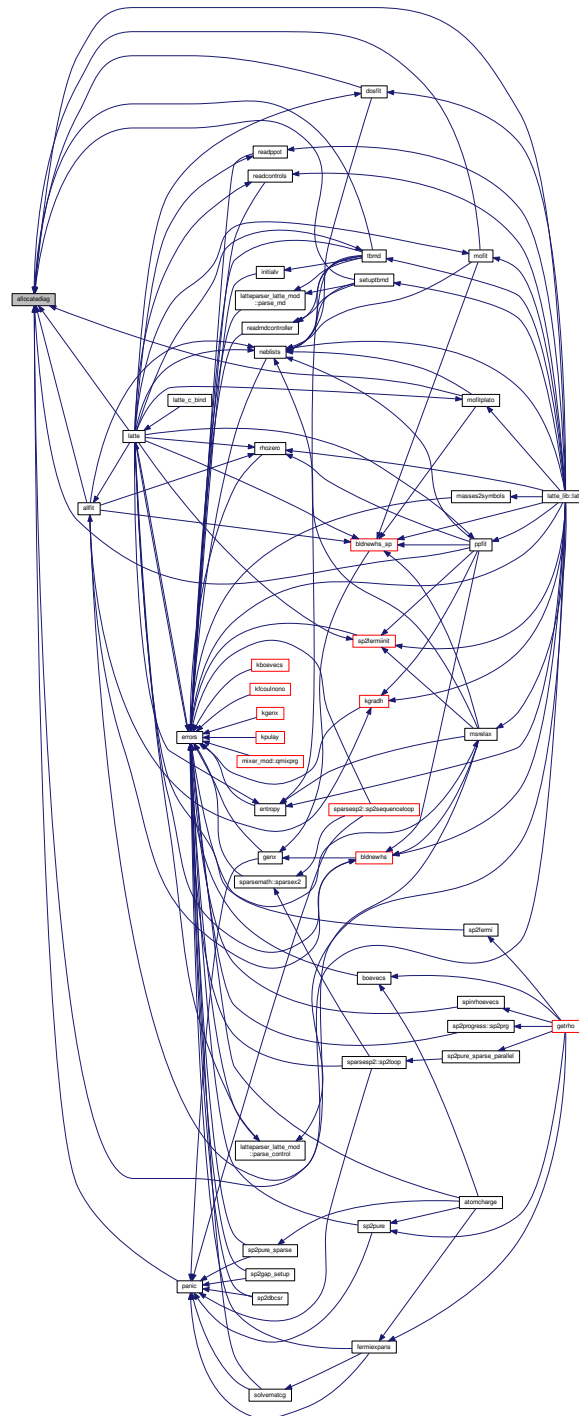
- subroutine [allocatediag](#)

8.11.1 Function/Subroutine Documentation

8.11.1.1 subroutine [allocatediag](#) ()

Definition at line 23 of file [allocatediag.f90](#).

Here is the caller graph for this function:



8.12 allocatediag.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatediag
00023
00024   USE constants_mod
00025   USE diagarray
00026   USE kspacearray
00027
00028   IMPLICIT NONE
00029   IF (existerror) RETURN
00030
00031   IF (kon .EQ. 0) THEN
00032
00033
00034   #ifdef XSYEV
00035
00036       diag_lwork = 3*hdim - 1
00037       ALLOCATE(diag_work(diag_lwork))
00038
00039   #elif defined(XSYEVD)
00040
00041       diag_lwork = 1 + 6*hdim + 2*hdim*hdim
00042       diag_liwork = 3 + 5*hdim
00043
00044       ALLOCATE(diag_work(diag_lwork), diag_iwork(
00045         diag_liwork))
00046   #endif
00047
00048
00049   IF (spinon .EQ. 0) THEN
00050       ALLOCATE (evals(hdim), evecs(hdim, hdim))
00051   ELSE
00052       ALLOCATE (upevals(hdim), upevecs(hdim, hdim))
00053       ALLOCATE (downevals(hdim), downevecs(hdim,
00054         hdim))
00055   ENDIF
00056
00057   ELSE ! Allocate arrays if we're doing k-space
00058
00059       ! Using ZHEEV..
00060
00061       diag_lzwork = 2*hdim-1
00062       diag_lrwork = 3*hdim-2
00063
00064       ALLOCATE(diag_rwork(diag_lrwork), diag_zwork(
00065         diag_lzwork))
00066
00067       ! Using ZHEEVD
00068
00069       !       ZHEEVD_LWORK = 2*HDIM + HDIM*HDIM
00070       !       ZHEEVD_LRWORK = 1 + 5*HDIM + 2*HDIM*HDIM
00071       !       ZHEEVD_LIWORK = 3 + 5*HDIM
00072
00073       !       ALLOCATE (ZHEEVD_WORK (ZHEEVD_LWORK), ZHEEVD_RWORK (ZHEEVD_LRWORK), &
00074       !       ZHEEVD_IWORK (ZHEEVD_LIWORK))
00075
00076       !       ALLOCATE (KHTMP (HDIM, HDIM))
00077
00078       IF (spinon .EQ. 0) THEN
00079           ALLOCATE (kevals(hdim, nktot), kevecs(hdim,
00080             hdim, nktot), &
00081             cplist(hdim*nktot))
00082       ENDIF
00083   ENDIF
00084
00085
00086   numlimit = exp(-exptol)
00087
00088   RETURN
00089

```

```
00090 END SUBROUTINE allocatediag
```

8.13 [allocatenebarrays.f90](#) File Reference

Functions/Subroutines

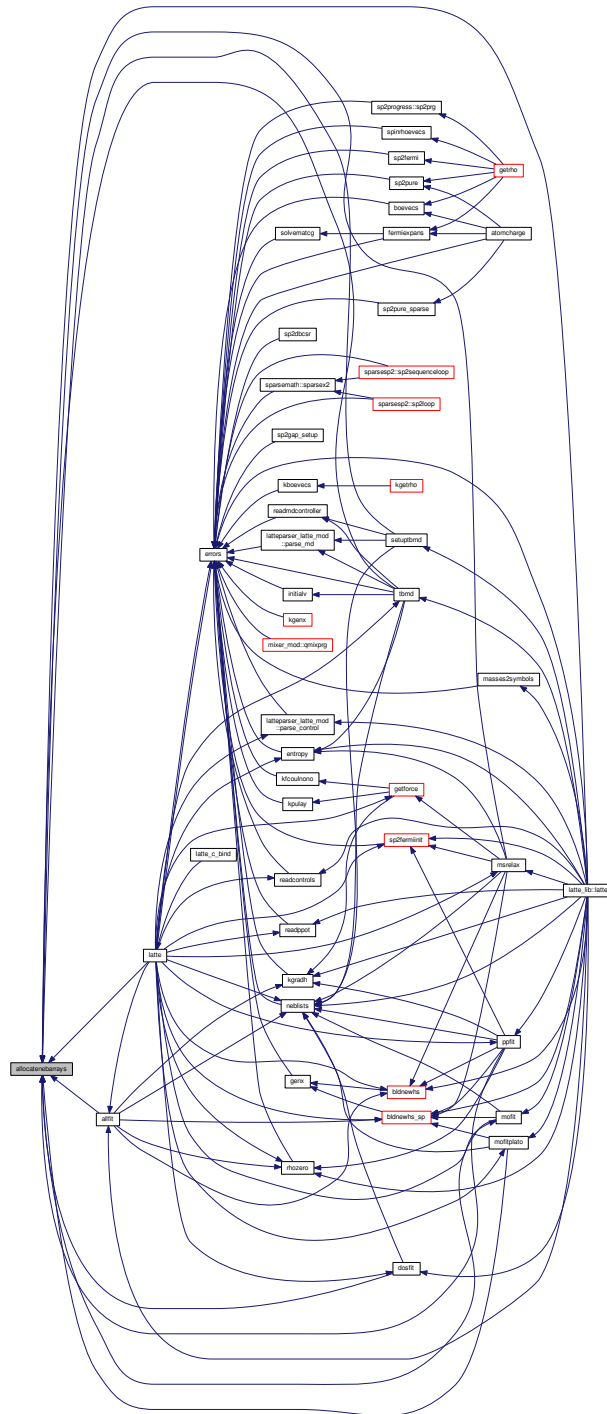
- subroutine [allocatenebarrays](#)

8.13.1 Function/Subroutine Documentation

8.13.1.1 subroutine [allocatenebarrays](#) ()

Definition at line [23](#) of file [allocatenebarrays.f90](#).

Here is the caller graph for this function:



8.14 allocatenebarrays.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatenebarrays
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE neblistarray
00027
00028     IMPLICIT NONE
00029     IF (existerror) RETURN
00030
00031     IF (.NOT. ALLOCATED(totnebtb)) ALLOCATE(totnebtb(nats))
00032     IF (ppoton .GT. 0 .AND. .NOT. ALLOCATED(totnebpp)) ALLOCATE(
totnebpp(nats))
00033     IF (electro .EQ. 1 .AND. .NOT. ALLOCATED(totnebcoul)) ALLOCATE(
totnebcoul(nats))
00034
00035
00036     RETURN
00037
00038 END SUBROUTINE allocatenebarrays

```

8.15 allocatenono.f90 File Reference

Functions/Subroutines

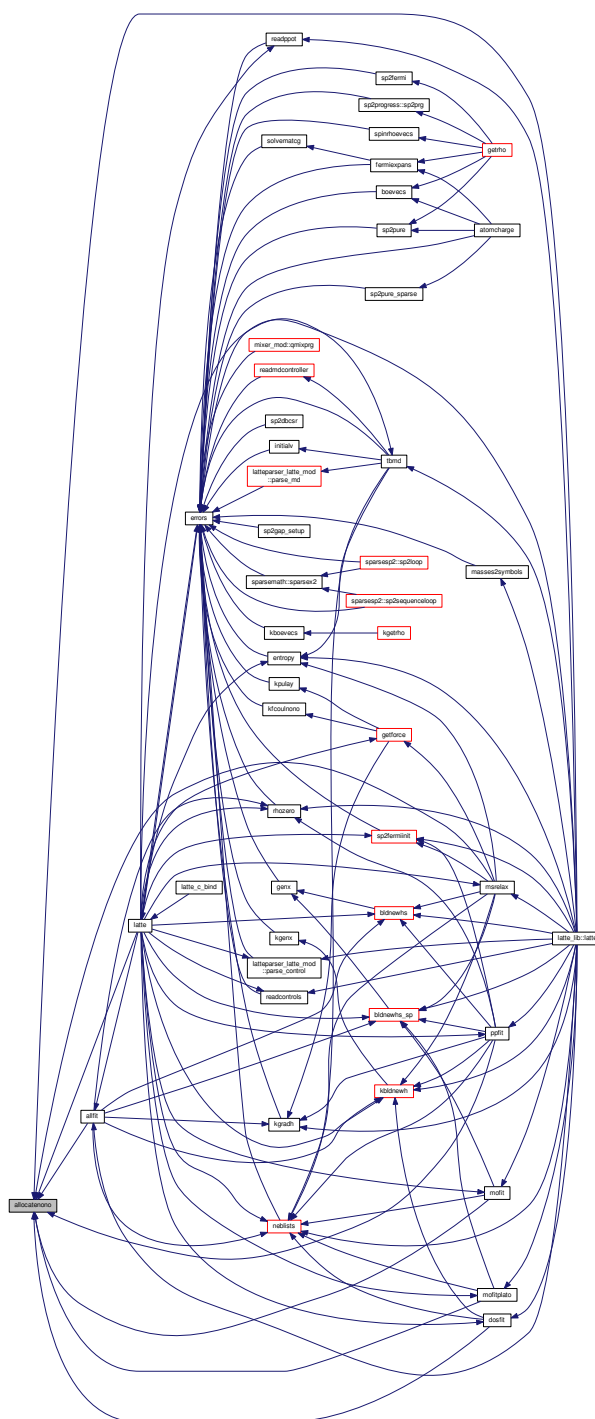
- subroutine [allocatenono](#)

8.15.1 Function/Subroutine Documentation

8.15.1.1 subroutine [allocatenono](#) ()

Definition at line 23 of file [allocatenono.f90](#).

Here is the caller graph for this function:



8.16 allocatenono.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatenono
00023
00024     USE constants_mod
00025     USE nonoarray
00026     USE kspacearray
00027
00028     IMPLICIT NONE
00029     IF (existerror) RETURN
00030
00031     IF (kon .EQ. 0) THEN
00032
00033 #ifdef XSYEV
00034
00035         nono_lwork = 3*hdim - 1
00036         ALLOCATE(nono_work( nono_lwork ))
00037
00038 #elif defined(XSYEVD)
00039
00040         nono_lwork = 1 + 6*hdim + 2*hdim*hdim
00041         nono_liwork = 3 + 5*hdim
00042
00043         ALLOCATE(nono_work( nono_lwork ), nono_iwork(
nono_liwork ))
00044
00045 #endif
00046
00047
00048
00049         ALLOCATE(nono_evals(hdim), umat(hdim, hdim))
00050         ALLOCATE(xmat(hdim, hdim), smat(hdim, hdim),
nonotmp(hdim, hdim))
00051
00052         ALLOCATE(x2hrho(hdim, hdim))
00053
00054         ALLOCATE(hjj(hdim))
00055
00056         IF (spinon .EQ. 0) THEN
00057             ALLOCATE(orthoh(hdim, hdim))
00058         ELSE
00059             ALLOCATE(orthohup(hdim, hdim), orthohdown(hdim,
hdim))
00060             ALLOCATE(spintmp(hdim, hdim), sh2(hdim, hdim))
00061         ENDIF
00062
00063     ELSEIF (kon .EQ. 1) THEN ! We're now doing k-space integration
00064
00065         ALLOCATE(sk(hdim, hdim, nktot), kxmat(hdim, hdim,
nktot), korthoh(hdim, hdim, nktot))
00066         ALLOCATE(zhjj(hdim))
00067
00068     ENDIF
00069
00070     RETURN
00071
00072 END SUBROUTINE allocatenono

```

8.17 allocatpure.f90 File Reference

Functions/Subroutines

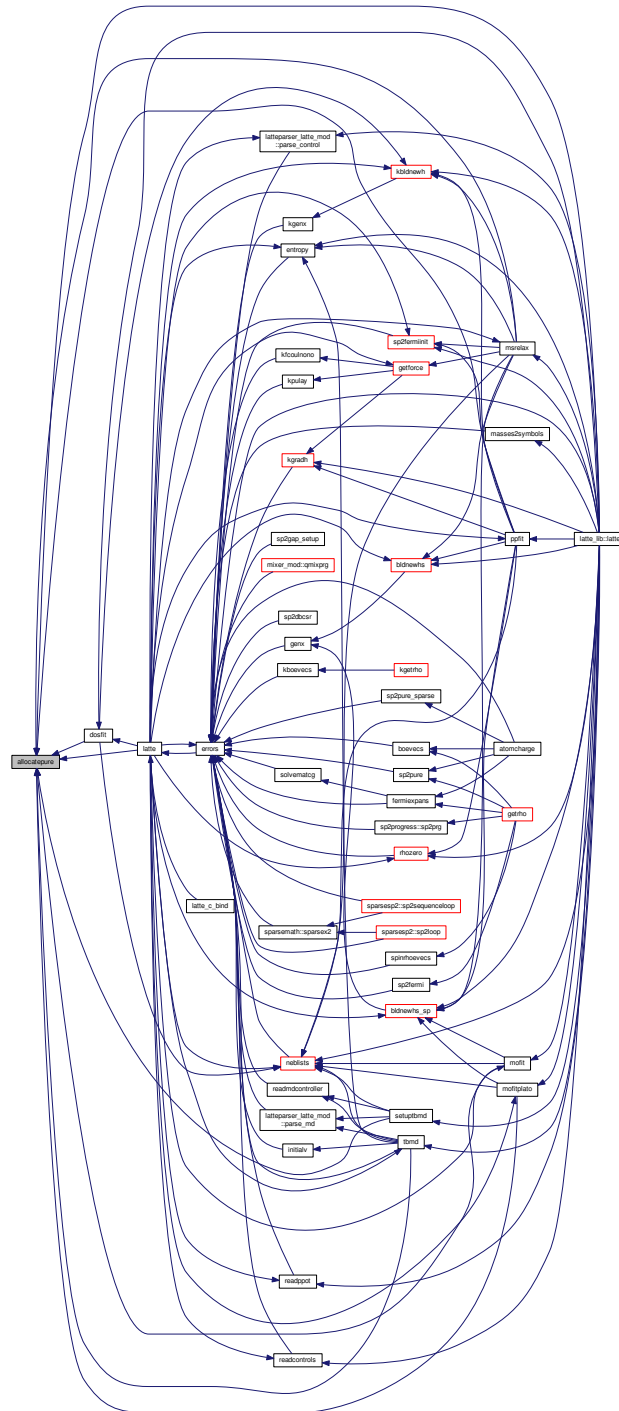
- subroutine [allocatpure](#)

8.17.1 Function/Subroutine Documentation

8.17.1.1 subroutine allocatpure ()

Definition at line 23 of file [allocatpure.f90](#).

Here is the caller graph for this function:



8.18 allocatpure.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatpure
00023
00024 #ifdef DBCSR_ON
00025
00026     USE dbcsr_var_mod
00027
00028 #endif
00029
00030     USE constants_mod
00031     USE purearray
00032     USE sparsearray
00033
00034     IMPLICIT NONE
00035     IF (existerror) RETURN
00036
00037     IF (control .EQ. 5) THEN
00038         ALLOCATE(signlist(norecs))
00039     ENDIF
00040
00041     IF (sparseon .EQ. 0) THEN
00042
00043         IF (spinon .EQ. 0) THEN
00044             ALLOCATE (x2(hdim,hdim) )
00045         ELSE
00046             ALLOCATE(x2up(hdim, hdim), x2down(hdim, hdim))
00047         ENDIF
00048
00049     ELSE
00050
00051         ! ALLOCATE(PP(100))
00052         ! ALLOCATE(VV(100))
00053
00054 #ifdef DBCSR_ON
00055
00056         ALLOCATE(bo_padded(blksz*nblkrows_total, blksz*
nblkcols_total))
00057
00058 #elif defined(DBCSR_OFF)
00059
00060         ALLOCATE(rx(hdim + 1), rxtmp(hdim + 1), work(hdim),
xb(hdim))
00061
00062 #endif
00063
00064     ENDIF
00065
00066     RETURN
00067
00068 END SUBROUTINE allocatpure

```

8.19 allocatesubgraph.f90 File Reference

Functions/Subroutines

- subroutine [allocatesubgraph](#)

8.19.1 Function/Subroutine Documentation

8.19.1.1 subroutine allocatesubgraph ()

Definition at line 23 of file [allocatesubgraph.f90](#).

8.20 allocatesubgraph.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatesubgraph
00023
00024   USE constants_mod
00025   USE xboarray
00026   USE spinarray
00027   USE myprecision
00028   USE subgraph
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I
00033   IF (existerror) RETURN
00034
00035   i = hdim
00036   ALLOCATE(g(i,i))
00037   !ALLOCATE(G(I,I),G0(I,I),G1(I,I),G2(I,I),G3(I,I),G4(I,I),G5(I,I),G6(I,I))
00038   first_step = 1
00039
00040   RETURN
00041
00042 END SUBROUTINE allocatesubgraph

```

8.21 allocatexbo.f90 File Reference

Functions/Subroutines

- subroutine [allocatexbo](#)

8.21.1 Function/Subroutine Documentation

8.21.1.1 subroutine allocatexbo ()

Definition at line 23 of file [allocatexbo.f90](#).

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE allocatexbo
00023
00024   USE constants_mod
00025   USE xboarray
00026   USE spinarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: ARRAYDIM
00032   IF (existerror) RETURN
00033
00034   cnk = zero
00035
00036   !
00037   ! This depends on whether we're propagating the
00038   ! on-site H matrix elements (in the case of LCN calculations)
00039   ! or the atomic partial charges (full SCC-TB)
00040   !
00041
00042   IF (electro .EQ. 0) THEN
00043     arraydim = nats
00044     !     ARRAYDIM = HDIM
00045   ELSE
00046     arraydim = nats
00047   ENDIF
00048
00049   IF (xbodison .EQ. 0) THEN
00050
00051     ALLOCATE(pnk(2,arraydim))
00052
00053     !
00054     ! If we need to propagate the chemical potential
00055     !
00056
00057     IF (control .EQ. 1 .OR. control .EQ. 3 &
00058         .OR. control .EQ. 5) ALLOCATE( chempot_pnk(2) )
00059
00060     !
00061     ! If we need to propagate the spin difference
00062     ! densities too:
00063     !
00064
00065     IF (spinon .EQ. 1) ALLOCATE( spin_pnk(2, deltadim) )
00066
00067   ELSE
00068
00069     ALLOCATE( pnk(xbodisorder + 1, arraydim) )
00070
00071     !
00072     ! If we need to propagate the chemical potential
00073     !
00074
00075     IF (control .EQ. 1 .OR. control .EQ. 3 .OR. &
00076         control .EQ. 5) ALLOCATE( chempot_pnk(xbodisorder + 1) )
00077
00078     !
00079     ! If we need to propagate the spin difference
00080     ! densities too:
00081     !
00082
00083     IF (spinon .EQ. 1) ALLOCATE( spin_pnk(xbodisorder + 1,
00084         deltadim) )
00085
00086     IF (xbodisorder .EQ. 3) THEN
00087
00088       alpha_xbo = 150.0d-3
00089       kappa_xbo = 1.69d0
00090       cnk(1) = -2.0d0
00091       cnk(2) = 3.0d0
00092       cnk(3) = 0.0d0

```

```
00093         cnk(4) = -1.0d0
00094
00095     ELSEIF (xbodisorder .EQ. 4) THEN
00096
00097         alpha_xbo = 57.0d-3
00098         kappa_xbo = 1.75d0
00099         cnk(1) = -3.0d0
00100         cnk(2) = 6.0d0
00101         cnk(3) = -2.0d0
00102         cnk(4) = -2.0d0
00103         cnk(5) = 1.0d0
00104
00105     ELSEIF (xbodisorder .EQ. 5) THEN
00106
00107         alpha_xbo = 18.0d-3
00108         kappa_xbo = 1.82d0
00109         cnk(1) = -6.0d0
00110         cnk(2) = 14.0d0
00111         cnk(3) = -8.0d0
00112         cnk(4) = -3.0d0
00113         cnk(5) = 4.0d0
00114         cnk(6) = -1.0d0
00115
00116     ELSEIF (xbodisorder .EQ. 6) THEN
00117
00118         alpha_xbo = 5.5d-3
00119         kappa_xbo = 1.84d0
00120         cnk(1) = -14.0d0
00121         cnk(2) = 36.0d0
00122         cnk(3) = -27.0d0
00123         cnk(4) = -2.0d0
00124         cnk(5) = 12.0d0
00125         cnk(6) = -6.0d0
00126         cnk(7) = 1.0d0
00127
00128     ELSEIF (xbodisorder .EQ. 7) THEN
00129
00130         alpha_xbo = 1.6d-3
00131         kappa_xbo = 1.86d0
00132         cnk(1) = -36.0d0
00133         cnk(2) = 99.0d0
00134         cnk(3) = -88.0d0
00135         cnk(4) = 11.0d0
00136         cnk(5) = 32.0d0
00137         cnk(6) = -25.0d0
00138         cnk(7) = 8.0d0
00139         cnk(8) = -1.0d0
00140
00141     ELSEIF (xbodisorder .EQ. 8) THEN
00142
00143         alpha_xbo = 0.44d-3
00144         kappa_xbo = 1.88d0
00145         cnk(1) = -99.0d0
00146         cnk(2) = 286.0d0
00147         cnk(3) = -286.0d0
00148         cnk(4) = 78.0d0
00149         cnk(5) = 78.0d0
00150         cnk(6) = -90.0d0
00151         cnk(7) = 42.0d0
00152         cnk(8) = -10.0d0
00153         cnk(9) = 1.0d0
00154
00155     ELSEIF (xbodisorder .EQ. 9) THEN
00156
00157         alpha_xbo = 0.12d-3
00158         kappa_xbo = 1.89d0
00159         cnk(1) = -286.0d0
00160         cnk(2) = 858.0d0
00161         cnk(3) = -936.0d0
00162         cnk(4) = 364.0d0
00163         cnk(5) = 168.0d0
00164         cnk(6) = -300.0d0
00165         cnk(7) = 184.0d0
00166         cnk(8) = -63.0d0
00167         cnk(9) = 12.0d0
00168         cnk(10) = -1.0d0
00169
00170     ENDIF
00171
00172 ENDIF
00173
00174 RETURN
00175
00176 END SUBROUTINE allocatexbo
```

8.23 am.f90 File Reference

Functions/Subroutines

- `real(latteprec)` function `am` (M, ALPHA)

8.23.1 Function/Subroutine Documentation

8.23.1.1 `real(latteprec)` function `am` (integer *M*, `real(latteprec)` *ALPHA*)

Definition at line 23 of file `am.f90`.

8.24 am.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION am(M, ALPHA)
00023
00024 ! Build Am function defined in eqn. (5) of PRB 72 165107
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 REAL(LATTEPREC) :: AM, ALPHA
00031 ! REAL(LATTEPREC), EXTERNAL:: HEAVI
00032 INTEGER :: M
00033
00034 ! Removed the call to the Heaviside step function to save on
00035 ! cos or sin calculation in M.NE. 0
00036
00037 ! MJC 1/10/14
00038
00039 IF (m == 0) THEN
00040   am = one / sqrt2
00041 ELSE IF ( m .GT. 0) THEN
00042   am = REAL((-1)**M, LATTEPREC) * COS(abs(m)*alpha)
00043 ELSE IF ( m .LT. 0) THEN
00044   am = REAL((-1)**M, LATTEPREC) * SIN(abs(m)*alpha)
00045 ENDIF
00046
00047 RETURN
00048
00049 END FUNCTION am

```

8.25 assessocc.f90 File Reference

Functions/Subroutines

- subroutine `assessocc`

8.25.1 Function/Subroutine Documentation

8.25.1.1 subroutine assessocc ()

Definition at line 23 of file [assessocc.f90](#).

8.26 assessocc.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE assessocc
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I, LWORK, INFO
00031   REAL(LATTEPREC), ALLOCATABLE :: WORK(:), EVECS(:, :)
00032   REAL(LATTEPREC), ALLOCATABLE :: HEVALS(:), RHOEVALS(:)
00033   CHARACTER(LEN=1), PARAMETER :: JOBZ = "V", uplo = "U"
00034
00035   lwork = 3*hdim - 1
00036
00037   ALLOCATE(work(lwork), evecs(hdim, hdim), hevals(hdim), rhoevals(
00038     hdim))
00039   !
00040   ! First get the eigenvalues of the Hamiltonian
00041   !
00042   evecs = h
00043
00044   #ifdef DOUBLEPREC
00045     CALL dsyev(jobz, uplo, hdim, evecs, hdim, hevals, work, lwork, info)
00046   #elif defined(SINGLEPREC)
00047     CALL ssyev(jobz, uplo, hdim, evecs, hdim, hevals, work, lwork, info)
00048   #endif
00049
00050   !
00051   ! Now the eigenvalues of the density matrix
00052   !
00053   evecs = half*bo
00054
00055   #ifdef DOUBLEPREC
00056     CALL dsyev(jobz, uplo, hdim, evecs, hdim, rhoevals, work, lwork, info)
00057   #elif defined(SINGLEPREC)
00058     CALL ssyev(jobz, uplo, hdim, evecs, hdim, rhoevals, work, lwork, info)
00059   #endif
00060
00061   !
00062   ! ... and print them out
00063   !
00064   ! Blas lists the eigenvalues of H in ascending order to we'll have

```



```
00070      ! to flip this:
00071      !
00072
00073      OPEN(unit=50, status="UNKNOWN", file="checkoccupancy.dat")
00074
00075      DO i = 1, hdim
00076          WRITE(50,10) hevals(i), one-rhoevals(i)
00077      ENDDO
00078
00079 10 FORMAT(2f18.9)
00080
00081      CLOSE(50)
00082
00083      DEALLOCATE(work, evecs, hevals, rhoevals)
00084
00085      RETURN
00086
00087 END SUBROUTINE assessocc
00088
```

8.27 atomcharge.f90 File Reference

Functions/Subroutines

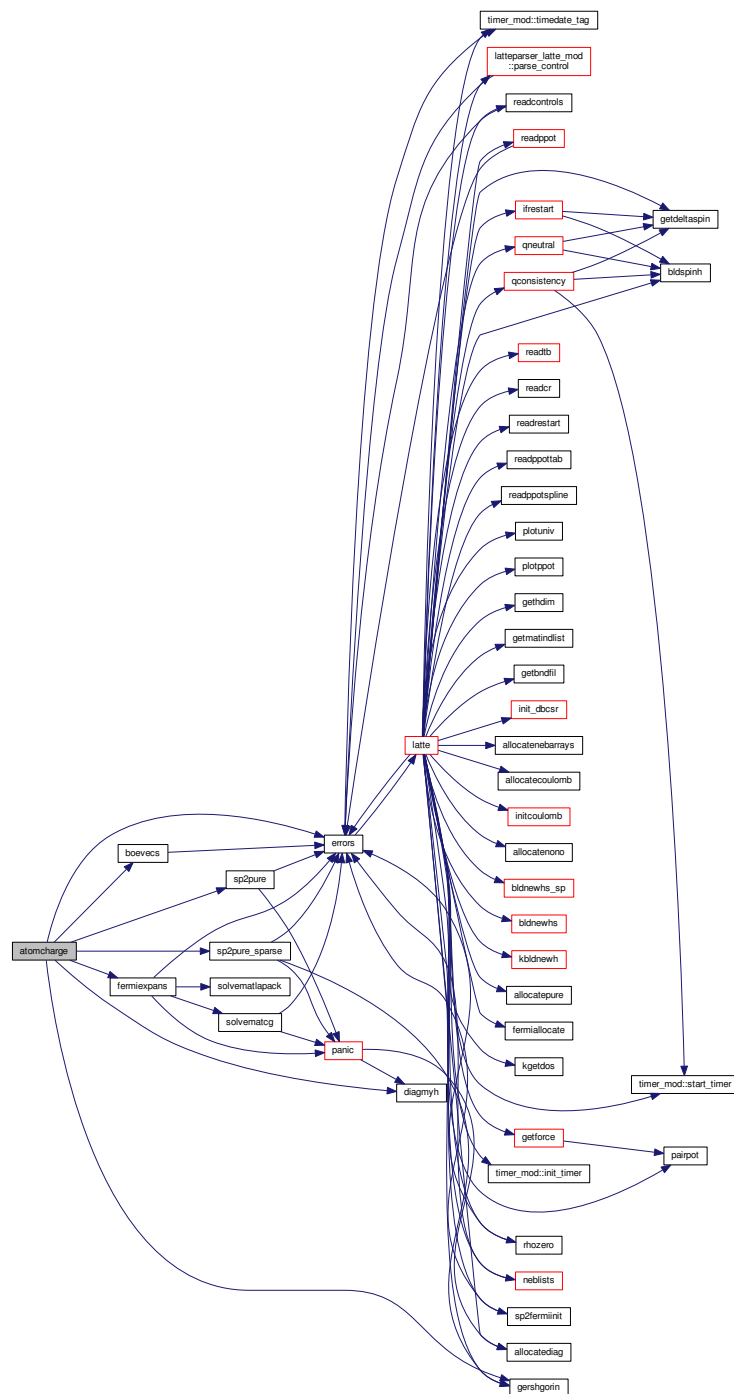
- subroutine [atomcharge](#) (SWITCH)

8.27.1 Function/Subroutine Documentation

8.27.1.1 subroutine atomcharge (integer SWITCH)

Definition at line 23 of file [atomcharge.f90](#).

Here is the call graph for this function:



8.28 atomcharge.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS            !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such        !
00010 ! modified software should be clearly marked, so as not to confuse it         !
00011 ! with the version available from LANL.                                       !
00012 !                                                                              !
00013 ! Additionally, this program is free software; you can redistribute it        !
00014 ! and/or modify it under the terms of the GNU General Public License as       !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE atomcharge(SWITCH)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: SWITCH
00031   INTEGER :: I, J, MYINDEX, PREVMYINDEX, ALLOK, ITER, NEWBUILD, II
00032   REAL(LATTEPREC) :: ATCHG
00033   REAL(LATTEPREC) :: CORRCHG, MAXCHG
00034   REAL(LATTEPREC), PARAMETER :: CPROPMAX = 0.8d0, cpropmin = 0.01d0
00035   REAL(LATTEPREC) :: DELTAHII
00036   REAL(LATTEPREC) :: RIGHTFACT, WRONGFACT
00037   REAL(LATTEPREC), ALLOCATABLE :: PREVDQ(:), CPROP(:)
00038   IF (existerror) RETURN
00039
00040   ALLOCATE(deltaq(nats), prevdq(nats), cprop(nats))
00041
00042   cprop = 0.25d0
00043
00044   rightfact = 1.1d0
00045   wrongfact = 0.45d0
00046
00047   maxchg = zero
00048   myindex = 0
00049   prevmyindex = 0
00050   allok = 0
00051
00052   !
00053   ! If we're applying full local charge neutrality, we need to run the
00054   ! whole caboodle until the DO WHILE loop is satisfied (ie, all atoms
00055   ! possess a number of electrons to within CHTOL
00056   !
00057   ! On the other hand, running one or two iterations of this should be
00058   ! good enough for an MD simulation when using XBO - we just need an
00059   ! infinitesimal improvement to ensure stability
00060   !
00061
00062   scfs_ii = 0
00063
00064   ! LCNON = 1: we're going full self-consistency
00065
00066   IF (lcnon .EQ. 1 .OR. switch .EQ. 0) THEN
00067
00068     DO i = 1, nats
00069
00070       atchg = zero
00071
00072       DO j = 1, noelem
00073
00074         IF (atele(i) .EQ. ele(j)) THEN
00075           corrchg = atocc(j)
00076           IF (basis(j) .EQ. "sp") THEN
00077             myindex = myindex + 4
00078           ELSEIF (basis(j) .EQ. "s") THEN
00079             myindex = myindex + 1
00080           ENDIF
00081         ENDIF
00082       ENDDO
00083
00084       DO j = prevmyindex+1, myindex
00085         atchg = atchg + bo(j,j)
00086       ENDDO
00087
00088       prevmyindex = myindex
00089
00090       deltaq(i) = atchg - corrchg
00091
00092       IF (abs(deltaq(i)) .GT. chtol) THEN
00093

```

```

00094         alloc = alloc + 1
00095     ENDIF
00096
00097 ENDDO
00098
00099 iter = 0
00100 newbuild = 0
00101 !     SCFS_II = 0
00102
00103 DO WHILE (alloc .NE. 0)
00104
00105     iter = iter + 1
00106
00107     scfs_ii = scfs_ii + 1
00108
00109     myindex = 0
00110     prevmyindex = 0
00111
00112     DO i = 1, nats
00113
00114         IF (iter .GT. 1) THEN
00115
00116             IF (abs(deltaq(i)) .LT. abs(prevdq(i))) THEN
00117
00118                 ! Going in the right direction, so go further
00119                 cprop(i) = rightfact*cprop(i)
00120
00121             ELSEIF (abs(deltaq(i)) .GT. abs(prevdq(i))) THEN
00122
00123                 ! Going in the wrong direction, so reverse
00124                 cprop(i) = wrongfact*cprop(i)
00125
00126             ENDIF
00127         ENDIF
00128
00129         cprop(i) = min(cprop(i), cpropmax)
00130         cprop(i) = max(cprop(i), cpropmin)
00131
00132         deltahii = deltaq(i)*cprop(i)
00133
00134         DO j = 1, noelem
00135             IF (atele(i) .EQ. ele(j)) THEN
00136                 IF (basis(j) .EQ. "sp") THEN
00137                     myindex = myindex + 4
00138                 ELSEIF (basis(j) .EQ. "s") THEN
00139                     myindex = myindex + 1
00140                 ENDIF
00141             ENDIF
00142         ENDDO
00143
00144         DO j = prevmyindex + 1, myindex
00145             h(j,j) = h(j,j) + deltahii
00146         ENDDO
00147
00148         prevmyindex = myindex
00149
00150     ENDDO
00151
00152     IF (iter .EQ. 100 ) THEN
00153
00154         DO i = 1, nats
00155             WRITE(6,99) i, deltaq(i), atele(i), cprop(i)
00156         ENDDO
00157 99    FORMAT(i6,1x,f18.9,1x,a2,1x, f12.8)
00158
00159         CALL errors("atomcharge","LCN not converging: STOP!")
00160
00161     ENDIF
00162
00163     IF (control .EQ. 1) THEN
00164
00165         CALL diagmyh()
00166         CALL boevecs()
00167
00168     ELSEIF (control .EQ. 2) THEN
00169
00170         CALL gershgorin
00171
00172         IF (sparseon .EQ. 0) THEN
00173             CALL sp2pure
00174         ELSEIF (sparseon .EQ. 1) THEN
00175
00176             CALL sp2pure_sparse
00177
00178         ENDIF
00179
00180     ELSEIF (control .EQ. 3) THEN

```

```

00181         !           IF (SPARSEON .EQ. 0) THEN
00182         CALL fermiexpans
00183         !           ELSEIF (SPARSEON .EQ. 1) THEN
00184         !           CALL ALLOCATEPURE
00185         !           CALL GERSHGORIN
00186         !           CALL INITSPARSESP2
00187         !           CALL DEALLOCATEPURE
00188         !           CALL FERMIEXPANSSPARSE
00189         !           ENDF
00190     ENDF
00191
00192     alloc = 0
00193
00194     myindex = 0
00195     prevmyindex = 0
00196
00197     prevdq = deltaq
00198
00199     DO i = 1, nats
00200
00201         atchg = zero
00202
00203         DO j = 1, noelem
00204
00205             IF (atele(i) .EQ. ele(j)) THEN
00206                 corrchg = atocc(j)
00207                 IF (basis(j) .EQ. "sp") THEN
00208                     myindex = myindex + 4
00209                 ELSEIF (basis(j) .EQ. "s") THEN
00210                     myindex = myindex + 1
00211                 ENDF
00212             ENDF
00213
00214         ENDDO
00215
00216         DO j = prevmyindex+1, myindex
00217             atchg = atchg + bo(j,j)
00218         ENDDO
00219
00220         prevmyindex = myindex
00221
00222         deltaq(i) = atchg - corrchg
00223
00224         IF (abs(deltaq(i)) .GT. chtol) THEN
00225             alloc = alloc + 1
00226         ENDF
00227     ENDDO
00228
00229
00230     ENDDO
00231
00232     ! LCNON = 0: we're doing just some specified number of
00233     ! iterations (= LCNITER)
00234
00235     ELSEIF (lcnon .EQ. 0 .AND. switch .NE. 0) THEN
00236
00237         DO ii = 1, lcniter
00238
00239             myindex = 0
00240             prevmyindex = 0
00241
00242             scfs_ii = scfs_ii + 1
00243
00244             DO i = 1, nats
00245
00246                 atchg = zero
00247
00248                 DO j = 1, noelem
00249
00250                     IF (atele(i) .EQ. ele(j)) THEN
00251                         corrchg = atocc(j)
00252                         IF (basis(j) .EQ. "sp") THEN
00253                             myindex = myindex + 4
00254                         ELSEIF (basis(j) .EQ. "s") THEN
00255                             myindex = myindex + 1
00256                         ENDF
00257                     ENDF
00258                 ENDDO
00259
00260                 DO j = prevmyindex+1, myindex
00261                     atchg = atchg + bo(j,j)
00262                 ENDDO
00263
00264                 !
00265                 ! Note that this factor of 0.25 is an empirical
00266                 ! value that the user is free to adjust. I should probably
00267

```

```

00268      ! move it to an input file
00269      !
00270
00271      deltahii = quarter*(atchg - corrchg)
00272
00273      DO j = prevmyindex+1, myindex
00274          h(j,j) = h(j,j) + deltahii
00275      ENDDO
00276
00277      prevmyindex = myindex
00278
00279      ENDDO
00280
00281      IF ( ii .LT. lcniter ) THEN
00282
00283          IF (control .EQ. 1) THEN
00284
00285              CALL diagmyh()
00286              CALL boevecs()
00287
00288          ELSEIF (control .EQ. 2) THEN
00289
00290              CALL gershgorin
00291
00292              IF (sparseon .EQ. 0) THEN
00293                  CALL sp2pure
00294              ELSEIF (sparseon .EQ. 1) THEN
00295
00296                  CALL sp2pure_sparse
00297              ENDIF
00298
00299          ELSEIF (control .EQ. 3) THEN
00300              ! IF (SPARSEON .EQ. 0) THEN
00301              CALL fermiexpans
00302              !
00303              ! ELSEIF (SPARSEON .EQ. 1) THEN
00304              !     CALL ALLOCATEPURE
00305              !     CALL GERSHGORIN
00306              !     CALL INITSPARSESP2
00307              !     CALL DEALLOCATEPURE
00308              !     CALL FERMIEXPANSSPARSE
00309              !
00310              ENDIF
00311          ENDIF
00312
00313      ENDDO
00314
00315      ENDIF
00316
00317      DEALLOCATE(deltaq, prevdq, cprop)
00318
00319      RETURN
00320
00321  END SUBROUTINE atomcharge

```

8.29 avepress.f90 File Reference

Functions/Subroutines

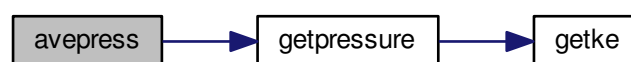
- subroutine [avepress](#)

8.29.1 Function/Subroutine Documentation

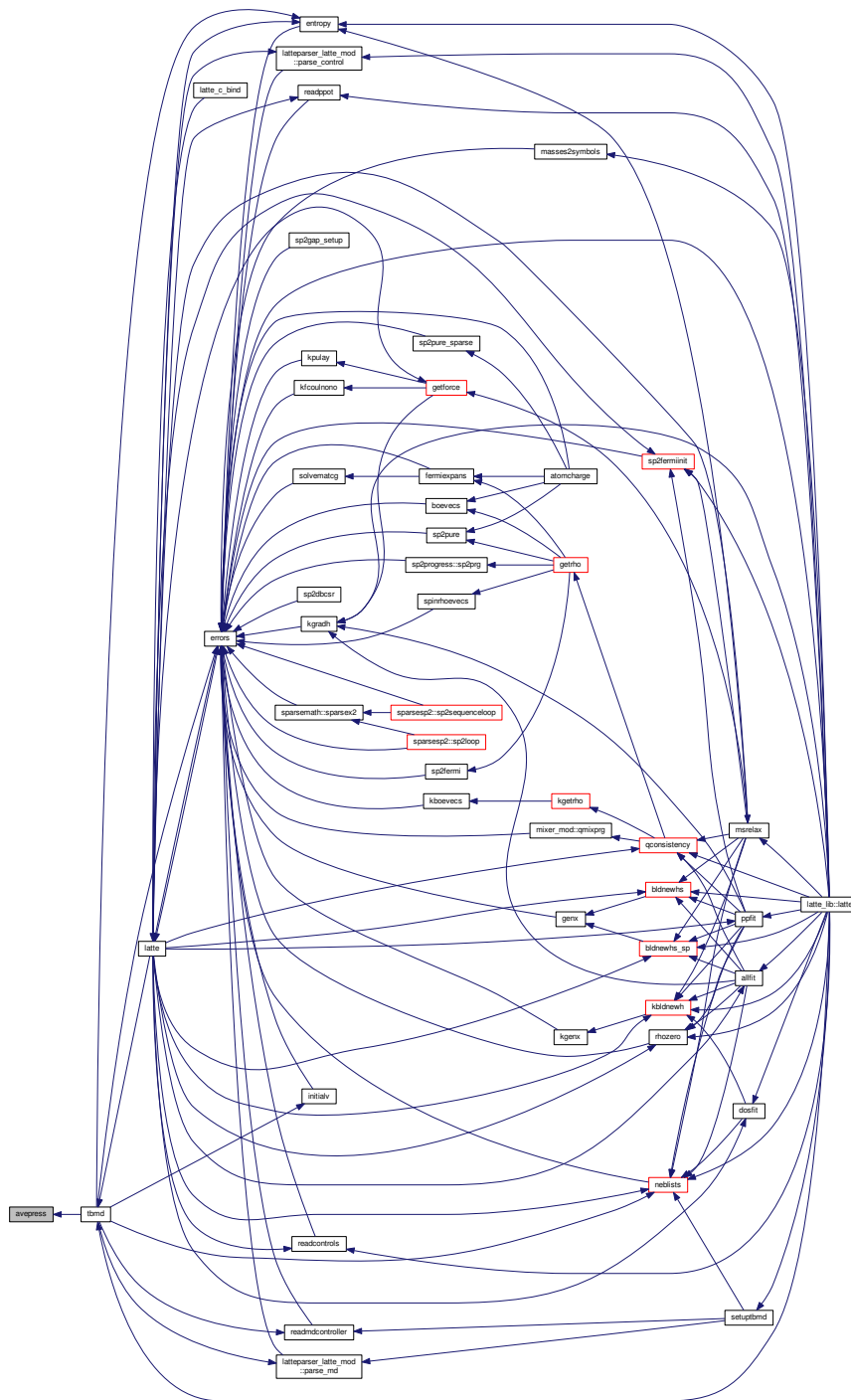
8.29.1.1 subroutine [avepress](#) ()

Definition at line 23 of file [avepress.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.30 avepress.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE avepress
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE virialarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I
00033   IF (existerror) RETURN
00034
00035   CALL getpressure
00036
00037   IF (npttype .EQ. "ISO") THEN
00038
00039     DO i = 1, aveper - 1
00040
00041       phist(i) = phist(i + 1)
00042
00043     ENDDO
00044
00045     phist(aveper) = pressure
00046
00047   ELSE
00048
00049     DO i = 1, aveper - 1
00050
00051       phistx(i) = phistx(i + 1)
00052       phisty(i) = phisty(i + 1)
00053       phistz(i) = phistz(i + 1)
00054
00055     ENDDO
00056
00057     phistx(aveper) = strten(1)
00058     phisty(aveper) = strten(2)
00059     phistz(aveper) = strten(3)
00060
00061   ENDIF
00062
00063   RETURN
00064
00065 END SUBROUTINE avepress

```

8.31 avesforhug.f90 File Reference

Functions/Subroutines

- subroutine [avesforhug](#) (MYPRESSURE, MYENERGY, MYTEMPERATURE, MYVOL)

8.31.1 Function/Subroutine Documentation

8.31.1.1 subroutine [avesforhug](#) (real(latteprec), intent(in) *MYPRESSURE*, real(latteprec), intent(in) *MYENERGY*, real(latteprec), intent(in) *MYTEMPERATURE*, real(latteprec), intent(in) *MYVOL*)

Definition at line 23 of file [avesforhug.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE avesforhug(MYPRESSURE, MYENERGY, MYTEMPERATURE, MYVOL)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I
00032   REAL(LATTEPREC), INTENT(IN) :: MYPRESSURE, MYENERGY, MYTEMPERATURE, MYVOL
00033   IF (existerror) RETURN
00034
00035   ! Record the pressure, volume, and energy
00036
00037   DO i = 1, (aveper/wrtfreq) - 1
00038
00039     phist(i) = phist(i + 1)
00040     vhist(i) = vhist(i + 1)
00041     ehist(i) = ehist(i + 1)
00042     thist(i) = thist(i + 1)
00043
00044   ENDDO
00045
00046   phist(aveper/wrtfreq) = mypressure
00047   vhist(aveper/wrtfreq) = myvol
00048   ehist(aveper/wrtfreq) = myenergy
00049   thist(aveper/wrtfreq) = mytemperature
00050
00051   RETURN
00052
00053 END SUBROUTINE avesforhug

```

8.33 avetemp.f90 File Reference

Functions/Subroutines

- subroutine [avetemp](#)

8.33.1 Function/Subroutine Documentation

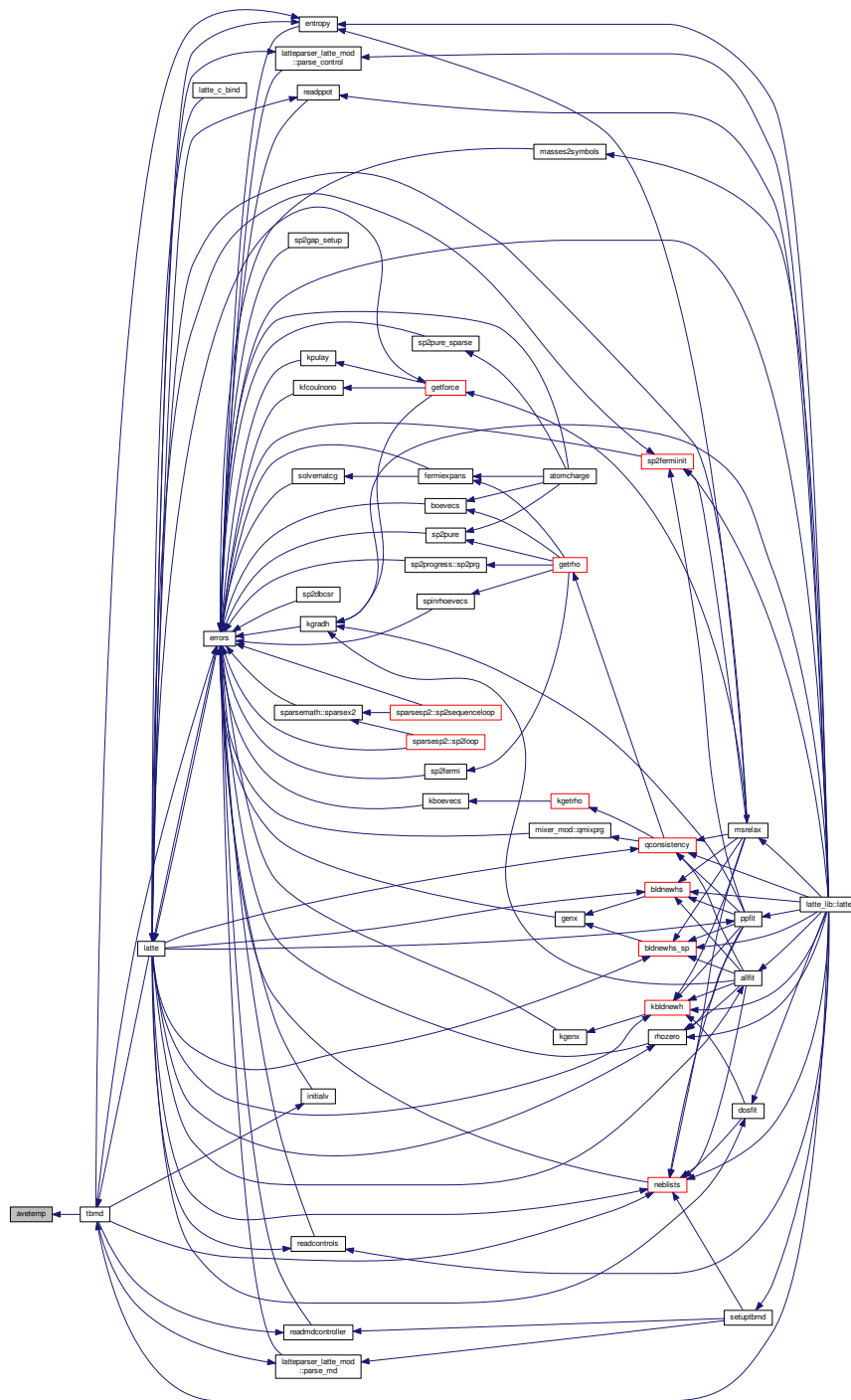
8.33.1.1 subroutine avetemp ()

Definition at line 23 of file [avetemp.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.34 avetemp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE avetemp
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I
00032   IF (existerror) RETURN
00033
00034   CALL getke
00035
00036   DO i = 1, aveper - 1
00037
00038       thist(i) = thist(i + 1)
00039
00040   ENDDO
00041
00042   thist(aveper) = temperature
00043
00044   RETURN
00045
00046 END SUBROUTINE avetemp

```

8.35 bldnewH.f90 File Reference

Functions/Subroutines

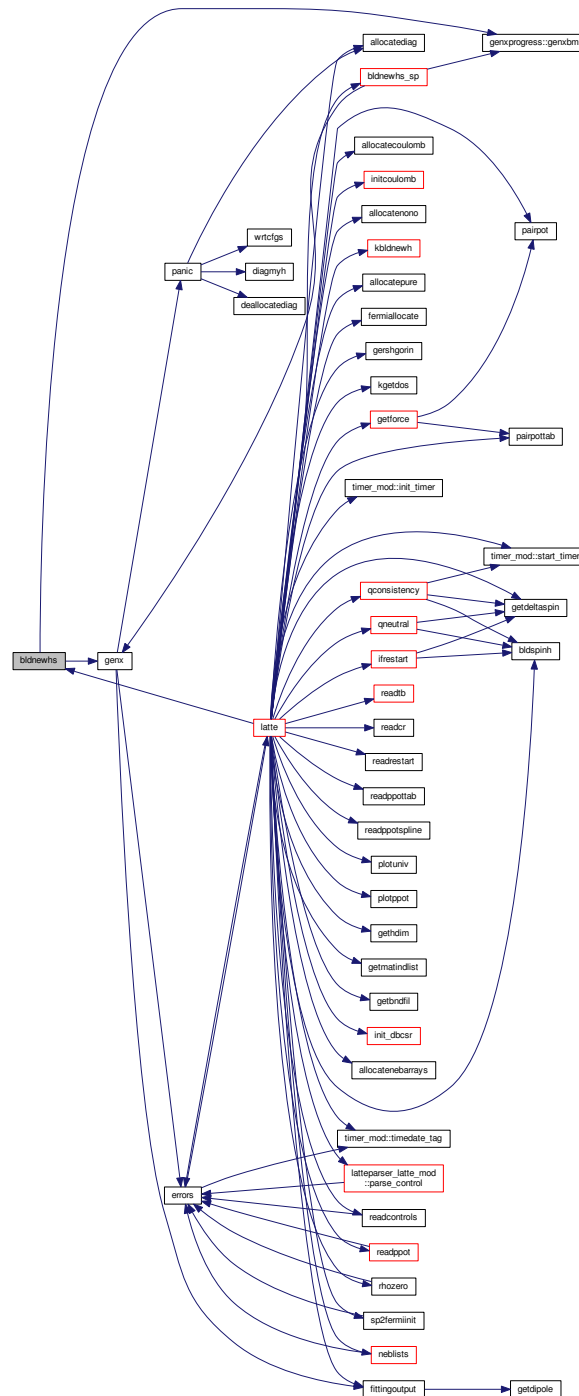
- subroutine [bldnewhs](#)

8.35.1 Function/Subroutine Documentation

8.35.1.1 subroutine bldnewhs ()

Definition at line 23 of file [bldnewH.f90](#).

Here is the call graph for this function:



```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE bldnewhs
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE xboarray
00028   USE nonoarray
00029   USE univarray
00030   USE myprecision
00031
00032   #ifdef PROGRESSON
00033     USE genxprogress
00034   #endif
00035
00036   IMPLICIT NONE
00037
00038   INTEGER :: I, J, NEWJ, K, L, II, JJ, KK, MM, MP, NN, SUBI
00039   INTEGER :: IBRA, IKET, LBRA, LKET, MBRA, MKET
00040   INTEGER :: INDEX, INDI, INDJ
00041   INTEGER :: SWITCH, PREVJ
00042   INTEGER :: PBCI, PBCJ, PBCK
00043   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00044   REAL(LATTEPREC) :: ALPHA, BETA, COSBETA, PHI, TMP, PERM
00045   REAL(LATTEPREC) :: RIJ(3), MAGR2, MAGR, MAGRP, RCUTB
00046   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00047   REAL(LATTEPREC) :: ANGFACTOR, AMMBRA, WIGLBRAMBRA
00048   ! REAL(LATTEPREC), ALLOCATABLE :: SAVEHDIAG(:)
00049   REAL(LATTEPREC), EXTERNAL :: UNIVSCALE, WIGNERD, SLMMP, TLMMP, AM, BM
00050   IF (existerror) RETURN
00051
00052
00053   h = zero
00054
00055   IF (basistype .EQ. "NONORTHO") THEN
00056
00057     smat = zero
00058     DO i = 1, hdim
00059       smat(i,i) = one
00060     ENDDO
00061
00062   ENDIF
00063
00064   index = 0
00065
00066   ! Build diagonal elements
00067   DO i = 1, nats
00068
00069     k = elempointer(i)
00070
00071     SELECT CASE(basis(k))
00072
00073     CASE("s")
00074
00075       index = index + 1
00076       h(index, index) = hes(k)
00077
00078     CASE("p")
00079
00080       DO subi = 1, 3
00081         index = index + 1
00082         h(index,index) = hep(k)
00083       ENDDO
00084
00085     CASE("d")
00086
00087       DO subi = 1, 5
00088         index = index + 1
00089         h(index,index) = hed(k)
00090       ENDDO
00091
00092     CASE("f")

```



```

00094      DO subi = 1, 7
00095          index = index + 1
00096          h(index,index) = hef(k)
00097      ENDDO
00098
00099      CASE("sp")
00100
00101      DO subi = 1, 4
00102
00103          index = index + 1
00104          IF (subi .EQ. 1) THEN
00105              h(index,index) = hes(k)
00106          ELSE
00107              h(index,index) = hep(k)
00108          ENDIF
00109
00110      ENDDO
00111
00112      CASE("sd")
00113
00114      DO subi = 1, 6
00115
00116          index = index + 1
00117          IF (subi .EQ. 1) THEN
00118              h(index,index) = hes(k)
00119          ELSE
00120              h(index,index) = hed(k)
00121          ENDIF
00122
00123      ENDDO
00124
00125      CASE("sf")
00126
00127      DO subi = 1, 8
00128
00129          index = index + 1
00130          IF (subi .EQ. 1) THEN
00131              h(index,index) = hes(k)
00132          ELSE
00133              h(index,index) = hef(k)
00134          ENDIF
00135
00136      ENDDO
00137
00138      CASE("pd")
00139
00140      DO subi = 1, 8
00141
00142          index = index + 1
00143          IF (subi .LE. 3) THEN
00144              h(index,index) = hep(k)
00145          ELSE
00146              h(index,index) = hed(k)
00147          ENDIF
00148
00149      ENDDO
00150
00151      CASE("pf")
00152
00153      DO subi = 1, 10
00154
00155          index = index + 1
00156          IF (subi .LE. 3) THEN
00157              h(index,index) = hep(k)
00158          ELSE
00159              h(index,index) = hef(k)
00160          ENDIF
00161
00162      ENDDO
00163
00164      CASE("df")
00165
00166      DO subi = 1, 12
00167
00168          index = index + 1
00169          IF (subi .LE. 5) THEN
00170              h(index,index) = hed(k)
00171          ELSE
00172              h(index,index) = hef(k)
00173          ENDIF
00174
00175      ENDDO
00176
00177      CASE("spd")
00178
00179      DO subi = 1, 9
00180

```

```

00181         index = index + 1
00182         IF (subi .EQ. 1) THEN
00183             h(index,index) = hes(k)
00184         ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00185             h(index,index) = hep(k)
00186         ELSE
00187             h(index,index) = hed(k)
00188         ENDIF
00189
00190     ENDDO
00191
00192     CASE("spf")
00193
00194     DO subi = 1, 11
00195
00196         index = index + 1
00197         IF (subi .EQ. 1) THEN
00198             h(index,index) = hes(k)
00199         ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00200             h(index,index) = hep(k)
00201         ELSE
00202             h(index,index) = hef(k)
00203         ENDIF
00204
00205     ENDDO
00206
00207     CASE("sdf")
00208
00209     DO subi = 1, 13
00210
00211         index = index + 1
00212         IF (subi .EQ. 1) THEN
00213             h(index,index) = hes(k)
00214         ELSEIF (subi .GT. 1 .AND. subi .LE. 6) THEN
00215             h(index,index) = hed(k)
00216         ELSE
00217             h(index,index) = hef(k)
00218         ENDIF
00219
00220     ENDDO
00221
00222     CASE("pdf")
00223
00224     DO subi = 1, 15
00225
00226         index = index + 1
00227         IF (subi .LE. 3) THEN
00228             h(index,index) = hep(k)
00229         ELSEIF (subi .GT. 3 .AND. subi .LE. 8) THEN
00230             h(index,index) = hed(k)
00231         ELSE
00232             h(index,index) = hef(k)
00233         ENDIF
00234
00235     ENDDO
00236
00237     CASE("spdf")
00238
00239     DO subi = 1, 16
00240
00241         index = index + 1
00242         IF (subi .EQ. 1) THEN
00243             h(index, index) = hes(k)
00244         ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00245             h(index, index) = hep(k)
00246         ELSEIF (subi .GT. 4 .AND. subi .LE. 9) THEN
00247             h(index, index) = hed(k)
00248         ELSE
00249             h(index, index) = hef(k)
00250         ENDIF
00251
00252     ENDDO
00253
00254     END SELECT
00255
00256 ENDDO
00257
00258 !$OMP PARALLEL DO DEFAULT (NONE) &
00259 !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00260 !$OMP SHARED(CR, BOX, H, SMAT, NOINT, ATELE, ELE1, ELE2) &
00261 !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00262 !$OMP PRIVATE(I, J, K, NEWJ, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00263 !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP, PHI, ALPHA, BETA, COSBETA) &
00264 !$OMP PRIVATE(LBRAINC, LBRA, MBRA, L, LKETINC, LKET, MKET) &
00265 !$OMP PRIVATE(RCUTTB, IBRA, IKET, AMMBRA, WIGLBRAMBRA, ANGFACOR, MP)
00266
00267

```

```

00268      ! open loop over atoms I in system
00269      DO i = 1, nats
00270
00271          ! Build the lists of orbitals on each atom
00272
00273          SELECT CASE(basis(elempointer(i)))
00274
00275              CASE("s")
00276                  basisi(1) = 0
00277                  basisi(2) = -1
00278              CASE("p")
00279                  basisi(1) = 1
00280                  basisi(2) = -1
00281              CASE("d")
00282                  basisi(1) = 2
00283                  basisi(2) = -1
00284              CASE("f")
00285                  basisi(1) = 3
00286                  basisi(2) = -1
00287              CASE("sp")
00288                  basisi(1) = 0
00289                  basisi(2) = 1
00290                  basisi(3) = -1
00291              CASE("sd")
00292                  basisi(1) = 0
00293                  basisi(2) = 2
00294                  basisi(3) = -1
00295              CASE("sf")
00296                  basisi(1) = 0
00297                  basisi(2) = 3
00298                  basisi(3) = -1
00299              CASE("pd")
00300                  basisi(1) = 1
00301                  basisi(2) = 2
00302                  basisi(3) = -1
00303              CASE("pf")
00304                  basisi(1) = 1
00305                  basisi(2) = 3
00306                  basisi(3) = -1
00307              CASE("df")
00308                  basisi(1) = 2
00309                  basisi(2) = 3
00310                  basisi(3) = -1
00311              CASE("spd")
00312                  basisi(1) = 0
00313                  basisi(2) = 1
00314                  basisi(3) = 2
00315                  basisi(4) = -1
00316              CASE("spf")
00317                  basisi(1) = 0
00318                  basisi(2) = 1
00319                  basisi(3) = 3
00320                  basisi(4) = -1
00321              CASE("sdf")
00322                  basisi(1) = 0
00323                  basisi(2) = 2
00324                  basisi(3) = 3
00325                  basisi(4) = -1
00326              CASE("pdf")
00327                  basisi(1) = 1
00328                  basisi(2) = 2
00329                  basisi(3) = 3
00330                  basisi(4) = -1
00331              CASE("spdf")
00332                  basisi(1) = 0
00333                  basisi(2) = 1
00334                  basisi(3) = 2
00335                  basisi(4) = 3
00336                  basisi(5) = -1
00337          END SELECT
00338
00339          indi = matindlist(i)
00340
00341          ! open loop over neighbors J of atom I
00342          DO newj = 1, totnebtb(i)
00343
00344              j = nebtb(1, newj, i)
00345
00346              IF ( j .GE. i ) THEN
00347
00348                  pbci = nebtb(2, newj, i)
00349                  pbcj = nebtb(3, newj, i)
00350                  pbck = nebtb(4, newj, i)
00351
00352                  rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00353                      REAL(pbck)*BOX(3,1) - CR(1,i)
00354

```

```

00355      rij(2) = cr(2,j) + REAL(pbc1)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00356      REAL(pbc2)*BOX(3,2) - CR(2,i)
00357
00358      rij(3) = cr(3,j) + REAL(pbc1)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00359      REAL(pbc2)*BOX(3,3) - CR(3,i)
00360
00361      magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00362
00363      rcuttb = zero
00364      DO k = 1, noint
00365
00366          IF ( (atele(i) .EQ. ele1(k) .AND. &
00367              atele(j) .EQ. ele2(k)) .OR. &
00368              (atele(j) .EQ. ele1(k) .AND. &
00369              atele(i) .EQ. ele2(k) ) ) THEN
00370
00371              IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00372
00373              IF (basistype .EQ. "NONORTHO") THEN
00374                  IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00375              ENDIF
00376
00377          ENDIF
00378
00379      ENDDO
00380
00381      IF (magr2 .LT. rcuttb*rcuttb) THEN
00382
00383          magr = sqrt(magr2)
00384
00385          SELECT CASE(basis(elempointer(j)))
00386          CASE("s")
00387              basisj(1) = 0
00388              basisj(2) = -1
00389          CASE("p")
00390              basisj(1) = 1
00391              basisj(2) = -1
00392          CASE("d")
00393              basisj(1) = 2
00394              basisj(2) = -1
00395          CASE("f")
00396              basisj(1) = 3
00397              basisj(2) = -1
00398          CASE("sp")
00399              basisj(1) = 0
00400              basisj(2) = 1
00401              basisj(3) = -1
00402          CASE("sd")
00403              basisj(1) = 0
00404              basisj(2) = 2
00405              basisj(3) = -1
00406          CASE("sf")
00407              basisj(1) = 0
00408              basisj(2) = 3
00409              basisj(3) = -1
00410          CASE("pd")
00411              basisj(1) = 1
00412              basisj(2) = 2
00413              basisj(3) = -1
00414          CASE("pf")
00415              basisj(1) = 1
00416              basisj(2) = 3
00417              basisj(3) = -1
00418          CASE("df")
00419              basisj(1) = 2
00420              basisj(2) = 3
00421              basisj(3) = -1
00422          CASE("spd")
00423              basisj(1) = 0
00424              basisj(2) = 1
00425              basisj(3) = 2
00426              basisj(4) = -1
00427          CASE("spf")
00428              basisj(1) = 0
00429              basisj(2) = 1
00430              basisj(3) = 3
00431              basisj(4) = -1
00432          CASE("sdf")
00433              basisj(1) = 0
00434              basisj(2) = 2
00435              basisj(3) = 3
00436              basisj(4) = -1
00437          CASE("pdf")
00438              basisj(1) = 1
00439              basisj(2) = 2
00440              basisj(3) = 3
00441              basisj(4) = -1

```

```

00442      CASE("spdf")
00443          basisj(1) = 0
00444          basisj(2) = 1
00445          basisj(3) = 2
00446          basisj(4) = 3
00447          basisj(5) = -1
00448      END SELECT
00449
00450      indj = matindlist(j)
00451
00452      magrp = sqrt(rij(1) * rij(1) + rij(2) * rij(2))
00453
00454      ! transform to system in which z-axis is aligned with RIJ,
00455      IF (abs(rij(1)) .GT. 1.0e-12) THEN
00456
00457          IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00458              phi = zero
00459          ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00460              phi = two * pi
00461          ELSE
00462              phi = pi
00463          ENDIF
00464          alpha = atan(rij(2) / rij(1)) + phi
00465
00466      ELSEIF (abs(rij(2)) .GT. 1.0e-12) THEN
00467
00468          IF (rij(2) .GT. 1.0e-12) THEN
00469              alpha = pi / two
00470          ELSE
00471              alpha = three * pi / two
00472          ENDIF
00473
00474      ELSE
00475          ! pathological case: beta=0 and alpha undefined, but
00476          ! this doesn't matter for matrix elements
00477
00478          alpha = zero
00479
00480      ENDIF
00481
00482      cosbeta = rij(3)/magr
00483      beta = acos(rij(3) / magr)
00484
00485      ! Build matrix elements using eqns (1)-(9) in PRB 72 165107
00486
00487      ! The loops over LBRA and LKET need to take into account
00488      ! the orbitals assigned to each atom, e.g., sd rather than
00489      ! spd...
00490
00491      lbra = indi + 1
00492
00493      lbrainc = 1
00494      DO WHILE (basisi(lbrainc) .NE. -1)
00495
00496          lbra = basisi(lbrainc)
00497          lbrainc = lbrainc + 1
00498
00499          DO mbra = -lbra, lbra
00500
00501              ! We can calculate these two outside the
00502              ! MKET loop...
00503
00504              ammbra = am(mbra, alpha)
00505              wiglbrambra = wignerd(lbra, abs(mbra), 0, cosbeta)
00506
00507              iket = indj + 1
00508
00509              lketinc = 1
00510              DO WHILE (basisj(lketinc) .NE. -1)
00511
00512                  lket = basisj(lketinc)
00513                  lketinc = lketinc + 1
00514
00515                  DO mket = -lket, lket
00516
00517                      ! This is the sigma bonds (mp = 0)
00518
00519                      ! Hamiltonian build
00520
00521                      ! Pre-compute the angular part so we can use it
00522                      ! again later if we're building the S matrix too
00523
00524                      angfactor = two * ammbra * &
00525                          am(mket, alpha) * &
00526                          wiglbrambra * &
00527                          wignerd(lket, abs(mket), 0, cosbeta)
00528

```

```

00529             h(ibra, iket) = h(ibra, iket) + angfactor * &
00530             !                                     H(IKET, IBRA) = H(IKET, IBRA) + ANGFACOR * &
00531             univscale(i, j, lbra, lket, 0, magr, "H")
00532
00533             ! Overlap matrix build
00534
00535             IF (basistype .EQ. "NONORTHO") THEN
00536
00537                 smat(ibra, iket) = smat(ibra, iket) + angfactor * &
00538                 univscale(i, j, lbra, lket, 0, magr, "S")
00539
00540             ENDIF
00541
00542             ! everything else
00543
00544             DO mp = 1, min(lbra, lket)
00545
00546                 angfactor = slmmp(lbra, mbra, mp, alpha, cosbeta) * &
00547                 slmmp(lket, mket, mp, alpha, cosbeta) + &
00548                 tlmmp(lbra, mbra, mp, alpha, cosbeta) * &
00549                 tlmmp(lket, mket, mp, alpha, cosbeta)
00550
00551                 h(ibra, iket) = h(ibra, iket) + angfactor * &
00552                 !H(IKET, IBRA) = H(IKET, IBRA) + ANGFACOR * &
00553                 univscale(i, j, lbra, lket, mp, magr, "H")
00554
00555                 IF (basistype .EQ. "NONORTHO") THEN
00556
00557                     smat(ibra, iket) = smat(ibra, iket) + &
00558                     angfactor * &
00559                     univscale(i, j, lbra, lket, mp, magr, "S")
00560
00561                 ENDIF
00562
00563             ENDDO
00564
00565             h(iket, ibra) = h(ibra, iket)
00566             !H(IBRA, IKET) = H(IKET, IBRA)
00567             IF (basistype .EQ. "NONORTHO") &
00568                 smat(iket, ibra) = smat(ibra, iket)
00569
00570             iket = iket + 1
00571
00572         ENDDO
00573
00574     ENDDO
00575
00576     ibra = ibra + 1
00577
00578 ENDDO
00579 ENDDO
00580 ENDDIF
00581 ENDDIF
00582 ENDDO
00583
00584 !     INDI = INDI + NORBI
00585
00586 ENDDO
00587
00588 !$OMP END PARALLEL DO
00589
00590 DO i = 1, hdim
00591     hdiag(i) = h(i,i)
00592 ENDDO
00593
00594
00595 IF (basistype .EQ. "NONORTHO") THEN
00596
00597     h0 = h
00598
00599 #ifdef PROGRESSON
00600     IF(latteinexists) THEN
00601         CALL genxbml
00602     ELSE
00603         CALL genx
00604     ENDIF
00605 #else
00606     CALL genx
00607 #endif
00608
00609
00610 IF (debugon .EQ. 1) THEN
00611
00612     OPEN(unit=30, status="UNKNOWN", file="myS.dat")
00613     OPEN(unit=31, status="UNKNOWN", file="myH0.dat")
00614
00615     print*, "Caution - the Slater-Koster H and overlap matrices are being written to file"

```

```
00616
00617      DO i = 1, hdim
00618          WRITE(30,10) (smat(i,j), j = 1, hdim)
00619          WRITE(31,10) (h0(i,j), j = 1, hdim)
00620      ENDDO
00621
00622      CLOSE(30)
00623      CLOSE(31)
00624
00625 10      FORMAT(100f12.6)
00626
00627      ENDIF
00628
00629  ENDIF
00630
00631  RETURN
00632
00633 END SUBROUTINE bldnewhs
```

8.37 bldnewHS_sp.f90 File Reference

Functions/Subroutines

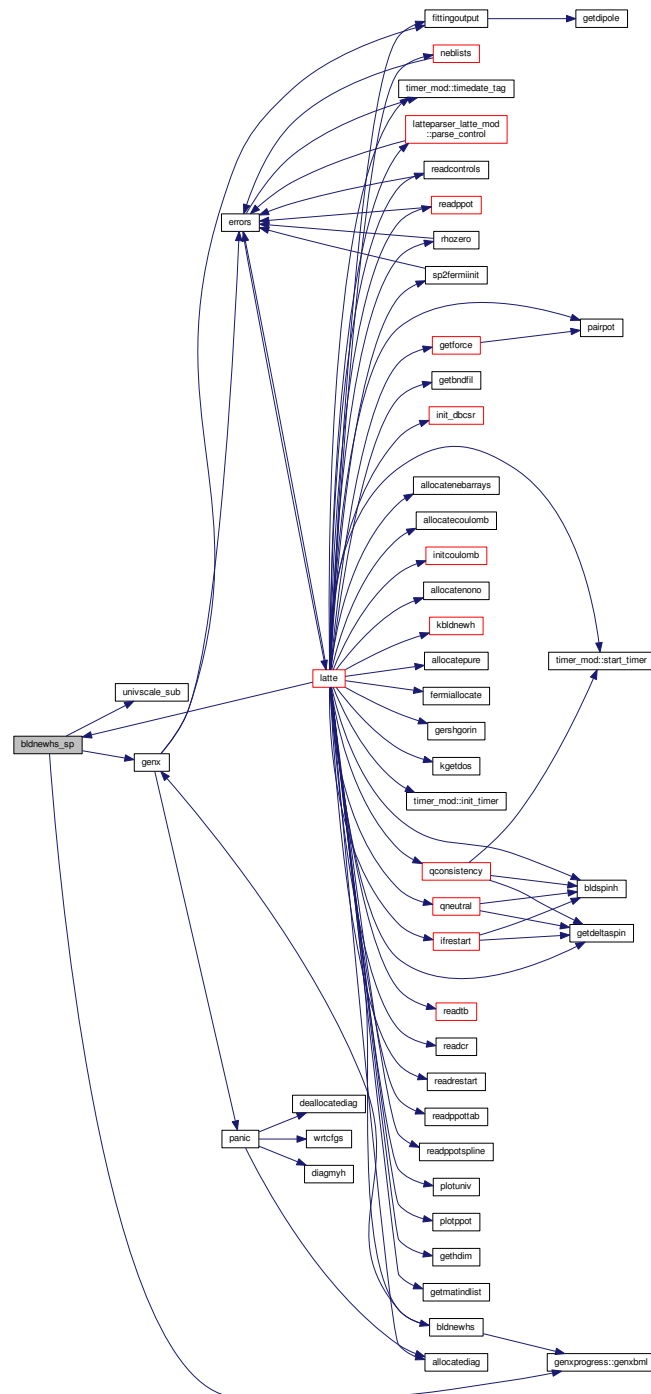
- subroutine [bldnewhs_sp](#)

8.37.1 Function/Subroutine Documentation

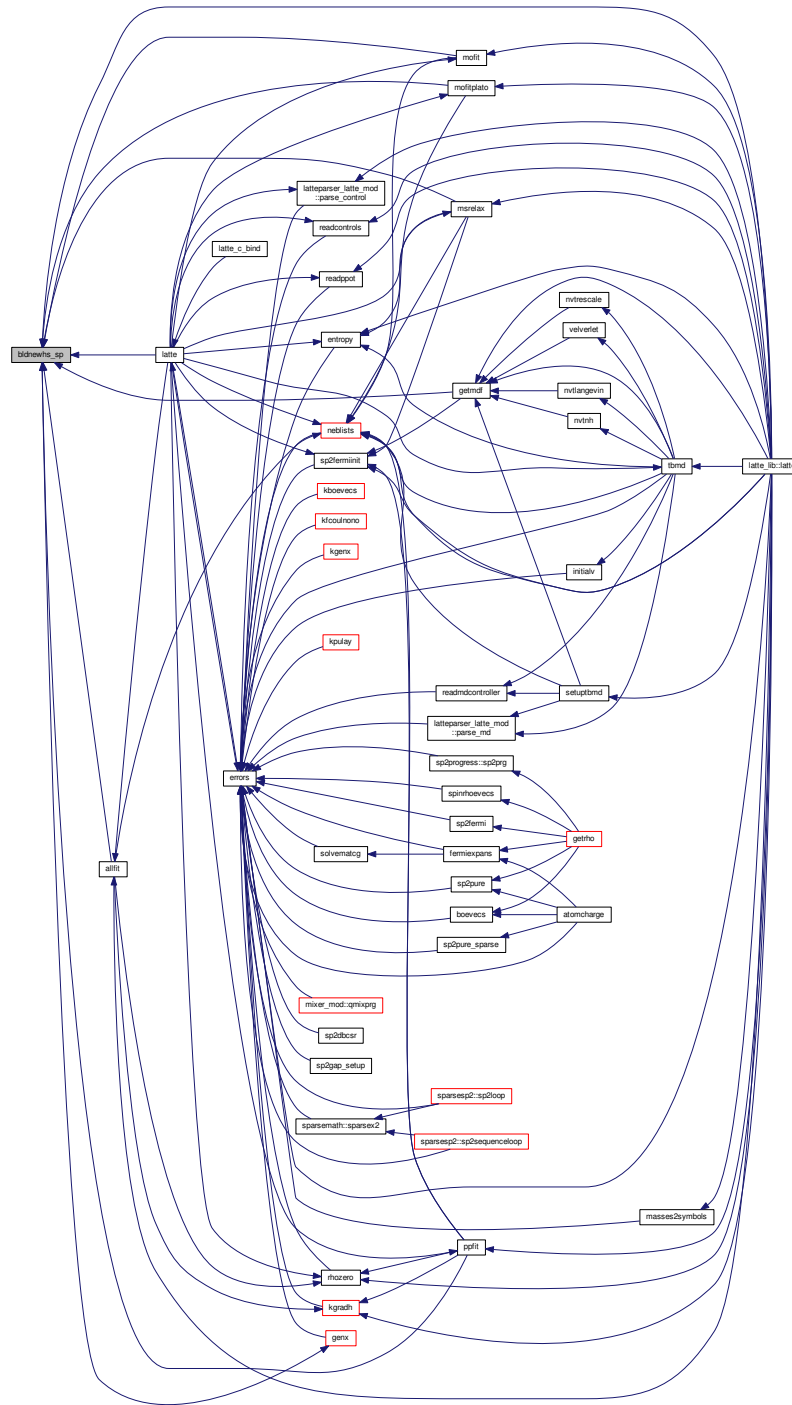
8.37.1.1 subroutine [bldnewhs_sp](#) ()

Definition at line 23 of file [bldnewHS_sp.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.38 bldnewHS_sp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE bldnewhs_sp
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE xboarray
00028   USE coulombarray
00029   USE nonoarray
00030   USE univarray
00031   USE myprecision
00032
00033   #ifdef PROGRESSON
00034     USE genxprogress
00035   #endif
00036
00037   IMPLICIT NONE
00038
00039   INTEGER :: I, J, NEWJ, K, KK, SUBI, NNZ
00040   INTEGER :: MYINDEX, INDI, INDJ
00041   INTEGER :: PBCI, PBCJ, PBCK
00042   REAL(LATTEPREC) :: HPPS, HPPP, PPSMPPP
00043   REAL(LATTEPREC) :: L, M, N, RIJ(3), MAGR, MAGR2, RCUTTB
00044
00045   !
00046   ! The sp H-matrix elements so we don't confuse ourselves...
00047   !
00048
00049   REAL(LATTEPREC) :: SISJ = zero
00050   REAL(LATTEPREC) :: SIPXJ = zero, sipyj = zero, sipzj = zero
00051   REAL(LATTEPREC) :: PXISJ = zero, pyisj = zero, pzisj = zero
00052   REAL(LATTEPREC) :: PXPX = zero, ppxy = zero, pxpz = zero
00053   REAL(LATTEPREC) :: PYPX = zero, pypy = zero, pypz = zero
00054   REAL(LATTEPREC) :: PZPX = zero, pzpy = zero, pzpz = zero
00055
00056   REAL(LATTEPREC) :: PISJ, OLPXISJ, OLPYISJ, OLPZISJ
00057   REAL(LATTEPREC) :: OLSIPXJ, OLSIPYJ, OLSIPZJ, OLSISJ, SIPJ
00058   REAL(LATTEPREC) :: SPPP, SPPS
00059
00060   !
00061
00062   CHARACTER(LEN=2) :: BASISI, BASISJ
00063   IF (existerror) RETURN
00064
00065   h = zero
00066
00067   myindex = 0
00068
00069   DO i = 1, nats
00070
00071     IF (basis(elempointer(i)) .EQ. "sp") THEN
00072
00073       myindex = myindex + 1
00074       h(myindex, myindex) = hes(elempointer(i))
00075       myindex = myindex + 1
00076       h(myindex, myindex) = hep(elempointer(i))
00077       myindex = myindex + 1
00078       h(myindex, myindex) = hep(elempointer(i))
00079       myindex = myindex + 1
00080       h(myindex, myindex) = hep(elempointer(i))
00081
00082     ELSEIF (basis(elempointer(i)) .EQ. "s") THEN
00083
00084       myindex = myindex + 1
00085       h(myindex, myindex) = hes(elempointer(i))
00086
00087     ENDIF
00088
00089   ENDDO
00090
00091   IF (basistype .EQ. "NONORTHO") THEN
00092
00093

```

```

00094      smat = zero
00095
00096      DO i = 1, hdim
00097          smat(i,i) = one
00098      ENDDO
00099
00100  ENDIF
00101
00102  !
00103  ! In the following, just to be clear,
00104  !
00105  ! SSS = S S SIGMA
00106  ! SPS = S P SIGMA
00107  ! PSS = P S SIGMA
00108  ! PPS = P P SIGMA
00109  ! PPP = P P PI,
00110  !
00111  ! are the fundamental Slater-Koster bond integrals
00112  !
00113
00114  !$OMP PARALLEL DO DEFAULT (NONE) &
00115  !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00116  !$OMP SHARED(CR, BOX, H, SMAT, NOINT, ATELE, ELE1, ELE2) &
00117  !$OMP SHARED(BOND, OVERL, MATINDLIST, BTYPE, BASISTYPE) &
00118  !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00119  !$OMP PRIVATE(RIJ, MAGR, MAGR2, SISJ, SIPXJ, SIPYJ, SIPZJ, PXISJ, PYISJ, PZISJ) &
00120  !$OMP PRIVATE(PXPX, XPY, XPZ, PYPX, PYPY, PYPZ, PZPX, PZPY, PZPZ) &
00121  !$OMP PRIVATE(PISJ, OLPXISJ, OLPYISJ, OLPZISJ, OLSIPXJ, OLSIPYJ, OLSIPZJ, OLSISJ) &
00122  !$OMP PRIVATE(SIPJ, SPPP, SPPS, L, M, N, HPPS, HPPP, PPSMPPP)
00123
00124
00125  DO i = 1, nats
00126
00127      basisi = basis(elempointer(i))
00128      indi = matindlist(i)
00129
00130      DO newj = 1, totnebtb(i)
00131
00132          !
00133          ! Getting neighbors from the TB neighborlist
00134          !
00135
00136          j = nebtb(1, newj, i)
00137
00138          IF (j .GE. i) THEN !bug fix by MJC
00139
00140              pbci = nebtb(2, newj, i)
00141              pbcj = nebtb(3, newj, i)
00142              pbck = nebtb(4, newj, i)
00143
00144              indj = matindlist(j)
00145
00146              !
00147              ! Which orbitals does J have? (s, sp, etc.)?
00148              !
00149
00150              basisj = basis(elempointer(j))
00151
00152              rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00153                  REAL(pbck)*BOX(3,1) - CR(1,i)
00154
00155              rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00156                  REAL(pbck)*BOX(3,2) - CR(2,i)
00157
00158              rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00159                  REAL(pbck)*BOX(3,3) - CR(3,i)
00160
00161              magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00162
00163              magr = sqrt(magr2)
00164
00165              !
00166              ! Direction cosines
00167              !
00168
00169              l = rij(1)/magr
00170              m = rij(2)/magr
00171              n = rij(3)/magr
00172
00173              !
00174              ! 5/3/10: Major bug fix by Ed Sanville for when
00175              ! the periodic cell measures less than the cut-off
00176              ! for the bond integrals.
00177              !
00178
00179              IF (basisi .EQ. "s") THEN
00180

```

```

00181      IF (basisj .EQ. "s") THEN
00182
00183          ! SISJ
00184
00185          DO k = 1, noint
00186              IF ((atele(i) .EQ. ele1(k) .AND. &
00187                  atele(j) .EQ. ele2(k)) .OR. &
00188                  (atele(i) .EQ. ele2(k) .AND. &
00189                  atele(j) .EQ. ele1(k))) THEN
00190
00191                  IF (btype(k) .EQ. "sss") THEN
00192
00193                      CALL univscale_sub(magr, bond(:,k), sisj)
00194
00195                      IF (basistype .EQ. "NONORTHO") &
00196                          CALL univscale_sub(magr, overl(:,k), olsisj)
00197
00198                      ENDIF
00199
00200                  ENDIF
00201              ENDDO
00202
00203              h(indi+1, indj+1) = h(indi+1, indj+1) + sisj
00204
00205              IF (basistype .EQ. "NONORTHO") &
00206                  smat(indi+1, indj+1) = smat(indi+1, indj+1) + olsisj
00207
00208          ELSEIF (basisj .EQ. "sp") THEN
00209
00210              ! SISJ, SIPXJ, SIPYJ, SIPZJ
00211
00212              DO k = 1, noint
00213
00214                  IF ((atele(i) .EQ. ele1(k) .AND. &
00215                      atele(j) .EQ. ele2(k)) .OR. &
00216                      (atele(i) .EQ. ele2(k) .AND. &
00217                      atele(j) .EQ. ele1(k))) THEN
00218
00219                      IF (btype(k) .EQ. "sss") THEN
00220
00221                          CALL univscale_sub(magr, bond(:,k), sisj)
00222
00223                          IF (basistype .EQ. "NONORTHO") &
00224                              CALL univscale_sub(magr, overl(:,k), olsisj)
00225
00226                      ELSEIF (btype(k) .EQ. "sps") THEN
00227
00228                          CALL univscale_sub(magr, bond(:,k), sipj)
00229
00230                          sipxj = 1 * sipj
00231                          sipyj = m * sipj
00232                          sipzj = n * sipj
00233
00234                          IF (basistype .EQ. "NONORTHO") THEN
00235
00236                              CALL univscale_sub(magr, overl(:,k), sipj)
00237
00238                              olsipxj = 1 * sipj
00239                              olsipyj = m * sipj
00240                              olsipzj = n * sipj
00241
00242                          ENDIF
00243
00244                      ENDIF
00245                  ENDDO
00246
00247              h(indi+1, indj+1) = h(indi+1, indj+1) + sisj
00248              h(indi+1, indj+2) = h(indi+1, indj+2) + sipxj
00249              h(indi+1, indj+3) = h(indi+1, indj+3) + sipyj
00250              h(indi+1, indj+4) = h(indi+1, indj+4) + sipzj
00251
00252              IF (basistype .EQ. "NONORTHO") THEN
00253
00254                  smat(indi+1, indj+1) = smat(indi+1, indj+1) + olsisj
00255                  smat(indi+1, indj+2) = smat(indi+1, indj+2) + olsipxj
00256                  smat(indi+1, indj+3) = smat(indi+1, indj+3) + olsipyj
00257                  smat(indi+1, indj+4) = smat(indi+1, indj+4) + olsipzj
00258
00259              ENDIF
00260
00261          ENDIF
00262
00263      ENDIF
00264
00265      ELSEIF (basisi .EQ. "sp") THEN
00266
00267          IF (basisj .EQ. "s") THEN

```

```

00268      ! SISJ, PXISJ, PYISJ, PZISJ
00269
00270      DO k = 1, noint
00271
00272      IF ((atele(i) .EQ. ele1(k) .AND. &
00273          atele(j) .EQ. ele2(k)) .OR. &
00274          (atele(i) .EQ. ele2(k) .AND. &
00275          atele(j) .EQ. ele1(k))) THEN
00276
00277      IF (btype(k) .EQ. "sss") THEN
00278
00279          CALL univscale_sub(magr, bond(:,k), sisj)
00280
00281      IF (basistype .EQ. "NONORTHO") &
00282          CALL univscale_sub(magr, overl(:,k), olsisj)
00283
00284      ELSEIF (btype(k) .EQ. "sps") THEN
00285
00286          CALL univscale_sub(magr, bond(:,k), pisj)
00287
00288          pxisj = -l * pisj
00289          pyisj = -m * pisj
00290          pzisj = -n * pisj
00291
00292      IF (basistype .EQ. "NONORTHO") THEN
00293
00294          CALL univscale_sub(magr, overl(:,k), pisj)
00295
00296          olpxisj = -l * pisj
00297          olpyisj = -m * pisj
00298          olpzisj = -n * pisj
00299
00300      ENDIF
00301
00302      ENDIF
00303      ENDDO
00304
00305      h(indi+1, indj+1) = h(indi+1, indj+1) + sisj
00306      h(indi+2, indj+1) = h(indi+2, indj+1) + pxisj
00307      h(indi+3, indj+1) = h(indi+3, indj+1) + pyisj
00308      h(indi+4, indj+1) = h(indi+4, indj+1) + pzisj
00309
00310      IF (basistype .EQ. "NONORTHO") THEN
00311
00312          smat(indi+1, indj+1) = smat(indi+1, indj+1) + olsisj
00313          smat(indi+2, indj+1) = smat(indi+2, indj+1) + olpxisj
00314          smat(indi+3, indj+1) = smat(indi+3, indj+1) + olpyisj
00315          smat(indi+4, indj+1) = smat(indi+4, indj+1) + olpzisj
00316
00317      ENDIF
00318
00319      ELSEIF (basisj .EQ. "sp") THEN
00320
00321      ! element I = element J is a bit simpler
00322
00323      IF (atele(i) .EQ. atele(j)) THEN
00324
00325      DO k = 1, noint
00326
00327      IF (atele(i) .EQ. ele1(k) .AND. &
00328          atele(j) .EQ. ele2(k)) THEN
00329
00330      IF (btype(k) .EQ. "sss") THEN
00331
00332          CALL univscale_sub(magr, bond(:,k), sisj)
00333
00334      IF (basistype .EQ. "NONORTHO") &
00335          CALL univscale_sub(magr, overl(:,k), olsisj)
00336
00337      ELSEIF (btype(k) .EQ. "sps") THEN
00338
00339          CALL univscale_sub(magr, bond(:,k), sipj)
00340
00341          sipxj = l * sipj
00342          sipyj = m * sipj
00343          sipzj = n * sipj
00344
00345          pxisj = -sipxj
00346          pyisj = -sipyj
00347          pzisj = -sipzj
00348
00349      IF (basistype .EQ. "NONORTHO") THEN
00350
00351          CALL univscale_sub(magr, overl(:,k), sipj)
00352
00353          olsipxj = l * sipj
00354

```

```

00355             olsipyj = m * sipj
00356             olsipzj = n * sipj
00357
00358             olpxisj = -olsipxj
00359             olpyisj = -olsipyj
00360             olpzisj = -olsipzj
00361
00362         ENDIF
00363
00364     ELSEIF (btype(k) .EQ. "pps") THEN
00365
00366         CALL univscale_sub(magr, bond(:,k), hpps)
00367
00368         IF (basistype .EQ. "NONORTHO") &
00369             CALL univscale_sub(magr, overl(:,k), spps)
00370
00371     ELSEIF (btype(k) .EQ. "ppp") THEN
00372
00373         CALL univscale_sub(magr, bond(:,k), hppp)
00374
00375         IF (basistype .EQ. "NONORTHO") &
00376             CALL univscale_sub(magr, overl(:,k), sppp)
00377
00378     ENDIF
00379 ENDIF
00380 ENDDO
00381
00382 ELSEIF (atele(i) .NE. atele(j)) THEN
00383
00384     DO k = 1, noint
00385
00386         IF (atele(i) .EQ. ele1(k) .AND. &
00387             atele(j) .EQ. ele2(k)) THEN
00388
00389             IF (btype(k) .EQ. "sss") THEN
00390
00391                 CALL univscale_sub(magr, bond(:,k), sisj)
00392
00393                 IF (basistype .EQ. "NONORTHO") &
00394                     CALL univscale_sub(magr, overl(:,k), olsisj)
00395
00396             ELSEIF (btype(k) .EQ. "sps") THEN
00397
00398                 CALL univscale_sub(magr, bond(:,k), sipj)
00399
00400                 sipxj = 1 * sipj
00401                 sipyj = m * sipj
00402                 sipzj = n * sipj
00403
00404                 IF (basistype .EQ. "NONORTHO") THEN
00405
00406                     CALL univscale_sub(magr, overl(:,k), sipj)
00407
00408                     olsipxj = 1 * sipj
00409                     olsipyj = m * sipj
00410                     olsipzj = n * sipj
00411
00412                 ENDIF
00413
00414             ELSEIF (btype(k) .EQ. "pps") THEN
00415
00416                 CALL univscale_sub(magr, bond(:,k), hpps)
00417
00418                 IF (basistype .EQ. "NONORTHO") &
00419                     CALL univscale_sub(magr, overl(:,k), spps)
00420
00421             ELSEIF (btype(k) .EQ. "ppp") THEN
00422
00423                 CALL univscale_sub(magr, bond(:,k), hppp)
00424
00425                 IF (basistype .EQ. "NONORTHO") &
00426                     CALL univscale_sub(magr, overl(:,k), sppp)
00427
00428             ENDIF
00429
00430         ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00431             atele(j) .EQ. ele1(k)) THEN
00432
00433             IF (btype(k) .EQ. "sss") THEN
00434
00435                 CALL univscale_sub(magr, bond(:,k), sisj)
00436
00437                 IF (basistype .EQ. "NONORTHO") &
00438                     CALL univscale_sub(magr, overl(:,k), olsisj)
00439
00440             ELSEIF (btype(k) .EQ. "sps") THEN

```

```

00442          CALL univscale_sub(magr, bond(:,k), sipj)
00443
00444          pxisj = -l * sipj
00445          pyisj = -m * sipj
00446          pzisj = -n * sipj
00447
00448          IF (basistype .EQ. "NONORTHO") THEN
00449
00450              CALL univscale_sub(magr, overl(:,k), sipj)
00451
00452              olpxisj = -l * sipj
00453              olpyisj = -m * sipj
00454              olpzisj = -n * sipj
00455
00456          ENDIF
00457
00458          ELSEIF (btype(k) .EQ. "pps") THEN
00459
00460              CALL univscale_sub(magr, bond(:,k), hpps)
00461
00462              IF (basistype .EQ. "NONORTHO") &
00463                  CALL univscale_sub(magr, overl(:,k), spps)
00464
00465          ELSEIF (btype(k) .EQ. "ppp") THEN
00466
00467              CALL univscale_sub(magr, bond(:,k), hppp)
00468
00469              IF (basistype .EQ. "NONORTHO") &
00470                  CALL univscale_sub(magr, overl(:,k), sppp)
00471
00472          ENDIF
00473
00474          ENDDO
00475
00476      ENDIF
00477
00478      ppsmppp = hpps - hppp
00479
00480      pxpx = hppp + l*l*ppsmppp
00481      pxpy = l*m*ppsmppp
00482      pxpz = l*n*ppsmppp
00483      pypx = m*l*ppsmppp
00484      pypy = hppp + m*m*ppsmppp
00485      pypz = m*n*ppsmppp
00486      pzpx = n*l*ppsmppp
00487      pzpy = n*m*ppsmppp
00488      pzpz = hppp + n*n*ppsmppp
00489
00490      h(indi+1,indj+1) = h(indi+1,indj+1) + sisj
00491      h(indi+1,indj+2) = h(indi+1,indj+2) + sipxj
00492      h(indi+1,indj+3) = h(indi+1,indj+3) + sipyj
00493      h(indi+1,indj+4) = h(indi+1,indj+4) + sipzj
00494
00495      h(indi+2,indj+1) = h(indi+2,indj+1) + pxisj
00496      h(indi+2,indj+2) = h(indi+2,indj+2) + pxpx
00497      h(indi+2,indj+3) = h(indi+2,indj+3) + pxpy
00498      h(indi+2,indj+4) = h(indi+2,indj+4) + pxpz
00499      h(indi+3,indj+1) = h(indi+3,indj+1) + pyisj
00500      h(indi+3,indj+2) = h(indi+3,indj+2) + pypx
00501      h(indi+3,indj+3) = h(indi+3,indj+3) + pypy
00502      h(indi+3,indj+4) = h(indi+3,indj+4) + pypz
00503      h(indi+4,indj+1) = h(indi+4,indj+1) + pzisj
00504      h(indi+4,indj+2) = h(indi+4,indj+2) + pzpx
00505      h(indi+4,indj+3) = h(indi+4,indj+3) + pzpy
00506      h(indi+4,indj+4) = h(indi+4,indj+4) + pzpz
00507
00508      IF (basistype .EQ. "NONORTHO") THEN
00509
00510          ppsmppp = spps - sppp
00511
00512          pxpx = sppp + l*l*ppsmppp
00513          pxpy = l*m*ppsmppp
00514          pxpz = l*n*ppsmppp
00515          pypx = m*l*ppsmppp
00516          pypy = sppp + m*m*ppsmppp
00517          pypz = m*n*ppsmppp
00518          pzpx = n*l*ppsmppp
00519          pzpy = n*m*ppsmppp
00520          pzpz = sppp + n*n*ppsmppp
00521
00522          smat(indi+1,indj+1) = smat(indi+1,indj+1) + olsisj
00523          smat(indi+1,indj+2) = smat(indi+1,indj+2) + olsipxj
00524          smat(indi+1,indj+3) = smat(indi+1,indj+3) + olsipyj
00525          smat(indi+1,indj+4) = smat(indi+1,indj+4) + olsipzj
00526
00527          smat(indi+2,indj+1) = smat(indi+2,indj+1) + olpxisj

```

```

00529          smat(indi+2,indj+2) = smat(indi+2,indj+2) + ppxx
00530          smat(indi+2,indj+3) = smat(indi+2,indj+3) + ppxy
00531          smat(indi+2,indj+4) = smat(indi+2,indj+4) + ppxz
00532          smat(indi+3,indj+1) = smat(indi+3,indj+1) + olpyisj
00533          smat(indi+3,indj+2) = smat(indi+3,indj+2) + pypx
00534          smat(indi+3,indj+3) = smat(indi+3,indj+3) + pypy
00535          smat(indi+3,indj+4) = smat(indi+3,indj+4) + pypz
00536          smat(indi+4,indj+1) = smat(indi+4,indj+1) + olpzisj
00537          smat(indi+4,indj+2) = smat(indi+4,indj+2) + pzpx
00538          smat(indi+4,indj+3) = smat(indi+4,indj+3) + pzpy
00539          smat(indi+4,indj+4) = smat(indi+4,indj+4) + pzpz
00540
00541          ENDIF
00542
00543
00544
00545          ENDIF
00546      ENDIF
00547  ENDIF
00548  ENDDO
00549
00550      !      IF (BASISI .EQ. "sp") INDI = INDI + 4
00551      !      IF (BASISI .EQ. "s") INDI = INDI + 1
00552
00553  ENDDO
00554
00555  !$OMP END PARALLEL DO
00556
00557  DO i = 1, hdim
00558      hdiag(i) = h(i,i)
00559  ENDDO
00560
00561  !
00562  ! ... and fill in the other half of the matrix
00563  !
00564
00565  IF (basistype .EQ. "ORTHO") THEN
00566
00567      DO i = 1, hdim
00568          DO j = i, hdim
00569              h(j,i) = h(i,j)
00570          ENDDO
00571      ENDDO
00572
00573      !      NNZ = 0
00574      !      DO I = 1, HDIM
00575      !          DO J = 1, HDIM
00576      !              IF (ABS(H(J,I)) .GT. 1.0D-12) NNZ = NNZ + 1
00577      !          ENDDO
00578      !      ENDDO
00579
00580
00581      !      OPEN(UNIT=50, STATUS="UNKNOWN", FILE="myham.mtx")
00582
00583      !      WRITE(50, '("%MatrixMarket matrix coordinate real general")')
00584      !      WRITE(50,*) HDIM, HDIM, NNZ
00585      !      DO I = 1, HDIM
00586      !          DO J = 1, HDIM
00587      !              IF (ABS(H(I,J)) .GT. 1.0D-12) WRITE(50,50) I, J, H(I,J)
00588      !          ENDDO
00589      !      ENDDO
00590
00591      !50      FORMAT(2I8, G24.12)
00592      !      STOP
00593
00594  ELSE
00595
00596      DO i = 1, hdim
00597          DO j = i, hdim
00598              h(j,i) = h(i,j)
00599              smat(j,i) = smat(i,j)
00600          ENDDO
00601      ENDDO
00602
00603      h0 = h
00604
00605  #ifdef PROGRESSON
00606      IF(latteinexists) THEN
00607          CALL genxbml
00608      ELSE
00609          CALL genx
00610      ENDIF
00611  #else
00612      CALL genx
00613  #endif
00614
00615      IF (debugon .EQ. 1) THEN

```



```
00616
00617     OPEN(unit=30, status="UNKNOWN", file="myS.dat")
00618     OPEN(unit=31, status="UNKNOWN", file="myH0.dat")
00619
00620     print*, "Caution - the Slater-Koster H and overlap matrices are being written to file"
00621
00622     DO i = 1, hdim
00623         WRITE(30,10) (smat(i,j), j = 1, hdim)
00624         WRITE(31,10) (h0(i,j), j = 1, hdim)
00625     ENDDO
00626
00627     CLOSE(30)
00628     CLOSE(31)
00629
00630     ENDIF
00631
00632     ENDIF
00633
00634     ! IF (BASISTYPE .EQ. "NONORTHO") CALL GENX
00635
00636 10 FORMAT(100f12.6)
00637
00638     RETURN
00639
00640 END SUBROUTINE bldnewhs_sp
```

8.39 bldspinH.f90 File Reference

Functions/Subroutines

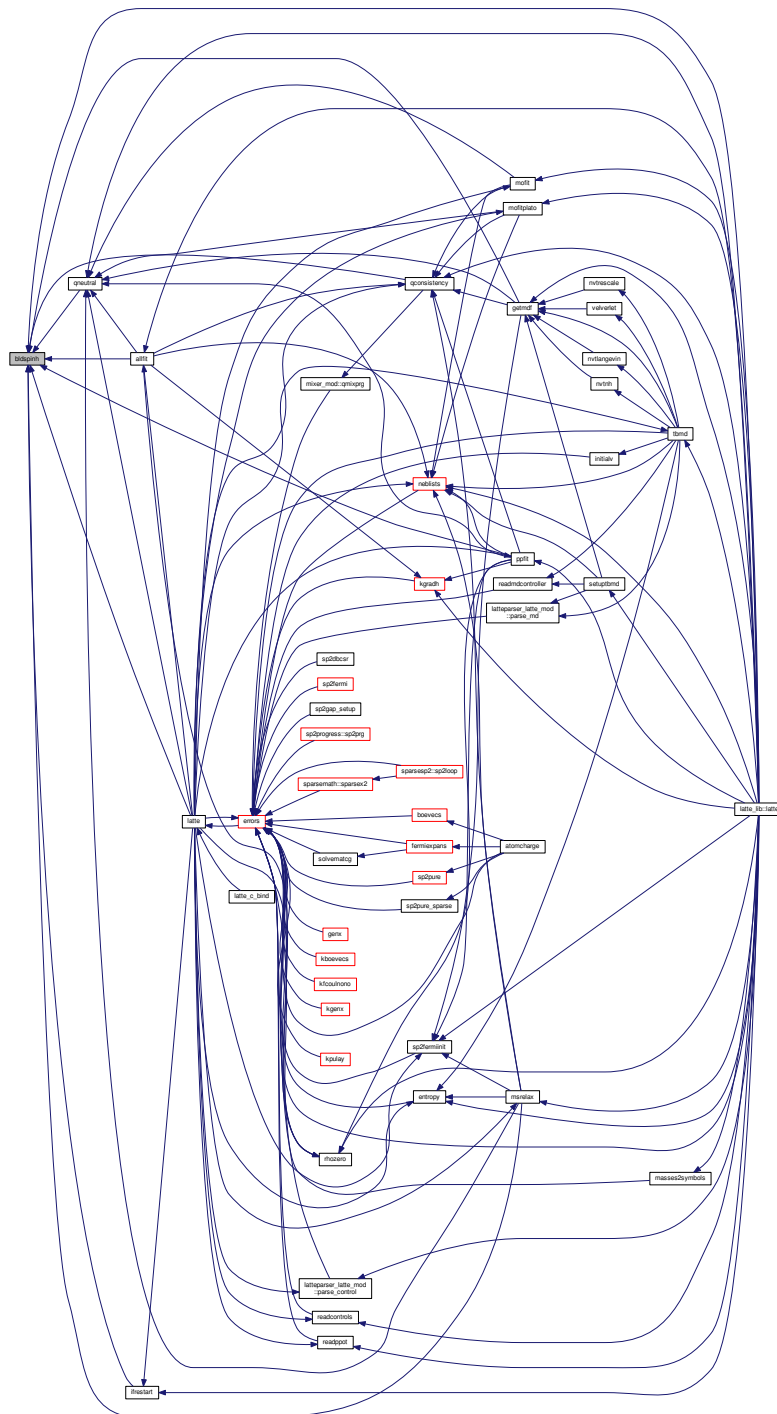
- subroutine [bldspin](#)

8.39.1 Function/Subroutine Documentation

8.39.1.1 subroutine [bldspin](#) ()

Definition at line 23 of file [bldspinH.f90](#).

Here is the caller graph for this function:



8.40 bldspinH.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE bldspinh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   !
00033   ! Here we build the spin-dependent Hamiltonian
00034   !
00035   ! Only the onsite blocks are non-zero, and there are only
00036   ! three unique elements in the case of sp-valent atoms, so we'll
00037   ! store this H matrix as a vector - suspended, to be implemented later
00038   !
00039
00040   INTEGER :: I, J, K
00041   INTEGER :: INDEX, HINDEX
00042   REAL(LATTEPREC) :: DSPINS, DSPINP
00043   REAL(LATTEPREC) :: HSS, HSP, HPP, HDD, HFF
00044   IF (existerror) RETURN
00045
00046   index = 0
00047   hindex = 0
00048
00049   ! We start with the spin-indepentent H matrix and
00050   ! either add or subtract the spin dependent one for
00051   ! the up-side and down-spin channels, respectively.
00052
00053
00054   DO i = 1, nats
00055
00056     SELECT CASE(basis(elempointer(i)))
00057
00058     CASE("s")
00059
00060       hss = wss(elempointer(i))*deltaspin(index + 1)
00061       index = index + 1
00062
00063       h2vect(hindex + 1) = hss
00064       hindex = hindex + 1
00065
00066     CASE("p")
00067
00068       hpp = wpp(elempointer(i))*deltaspin(index + 1)
00069       index = index + 1
00070
00071       h2vect(hindex + 1) = hpp
00072       h2vect(hindex + 2) = hpp
00073       h2vect(hindex + 3) = hpp
00074       hindex = hindex + 3
00075
00076     CASE("d")
00077
00078       hdd = wdd(elempointer(i))*deltaspin(index + 1)
00079       index = index + 1
00080
00081       h2vect(hindex + 1) = hdd
00082       h2vect(hindex + 2) = hdd
00083       h2vect(hindex + 3) = hdd
00084       h2vect(hindex + 4) = hdd
00085       h2vect(hindex + 5) = hdd
00086       hindex = hindex + 5
00087
00088     CASE("f")
00089
00090       hff = wff(elempointer(i))*deltaspin(index + 1)
00091       index = index + 1
00092
00093       h2vect(hindex + 1) = hff

```

```

00094         h2vect(hindex + 2) = hff
00095         h2vect(hindex + 3) = hff
00096         h2vect(hindex + 4) = hff
00097         h2vect(hindex + 5) = hff
00098         h2vect(hindex + 6) = hff
00099         h2vect(hindex + 7) = hff
00100         hindex = hindex + 7
00101
00102     CASE("sp")
00103
00104         hss = wss(elempointer(i))*deltaspin(index + 1)
00105         hpp = wpp(elempointer(i))*deltaspin(index + 2)
00106         index = index + 2
00107
00108         h2vect(hindex + 1) = hss
00109         h2vect(hindex + 2) = hpp
00110         h2vect(hindex + 3) = hpp
00111         h2vect(hindex + 4) = hpp
00112         hindex = hindex + 4
00113
00114     CASE("sd")
00115
00116         hss = wss(elempointer(i))*deltaspin(index + 1)
00117         hdd = wdd(elempointer(i))*deltaspin(index + 2)
00118         index = index + 2
00119
00120         h2vect(hindex + 1) = hss
00121         h2vect(hindex + 2) = hdd
00122         h2vect(hindex + 3) = hdd
00123         h2vect(hindex + 4) = hdd
00124         h2vect(hindex + 5) = hdd
00125         h2vect(hindex + 6) = hdd
00126         hindex = hindex + 6
00127
00128     CASE("sf")
00129
00130         hss = wss(elempointer(i))*deltaspin(index + 1)
00131         hff = wff(elempointer(i))*deltaspin(index + 2)
00132         index = index + 2
00133
00134         h2vect(hindex + 1) = hss
00135         h2vect(hindex + 2) = hff
00136         h2vect(hindex + 3) = hff
00137         h2vect(hindex + 4) = hff
00138         h2vect(hindex + 5) = hff
00139         h2vect(hindex + 6) = hff
00140         h2vect(hindex + 7) = hff
00141         h2vect(hindex + 8) = hff
00142         hindex = hindex + 8
00143
00144     CASE("pd")
00145
00146         hpp = wpp(elempointer(i))*deltaspin(index + 1)
00147         hdd = wdd(elempointer(i))*deltaspin(index + 2)
00148         index = index + 2
00149
00150         h2vect(hindex + 1) = hpp
00151         h2vect(hindex + 2) = hpp
00152         h2vect(hindex + 3) = hpp
00153         h2vect(hindex + 4) = hdd
00154         h2vect(hindex + 5) = hdd
00155         h2vect(hindex + 6) = hdd
00156         h2vect(hindex + 7) = hdd
00157         h2vect(hindex + 8) = hdd
00158         hindex = hindex + 8
00159
00160     CASE("pf")
00161
00162         hpp = wpp(elempointer(i))*deltaspin(index + 1)
00163         hff = wff(elempointer(i))*deltaspin(index + 2)
00164         index = index + 2
00165
00166         h2vect(hindex + 1) = hpp
00167         h2vect(hindex + 2) = hpp
00168         h2vect(hindex + 3) = hpp
00169         h2vect(hindex + 4) = hff
00170         h2vect(hindex + 5) = hff
00171         h2vect(hindex + 6) = hff
00172         h2vect(hindex + 7) = hff
00173         h2vect(hindex + 8) = hff
00174         h2vect(hindex + 9) = hff
00175         h2vect(hindex + 10) = hff
00176         hindex = hindex + 10
00177
00178
00179     CASE("df")
00180

```

```

00181      hdd = wdd(elempointer(i))*deltaspin(index + 1)
00182      hff = wff(elempointer(i))*deltaspin(index + 2)
00183      index = index + 2
00184
00185      h2vect(hindex + 1) = hdd
00186      h2vect(hindex + 2) = hdd
00187      h2vect(hindex + 3) = hdd
00188      h2vect(hindex + 4) = hdd
00189      h2vect(hindex + 5) = hdd
00190      h2vect(hindex + 6) = hff
00191      h2vect(hindex + 7) = hff
00192      h2vect(hindex + 8) = hff
00193      h2vect(hindex + 9) = hff
00194      h2vect(hindex + 10) = hff
00195      h2vect(hindex + 11) = hff
00196      h2vect(hindex + 12) = hff
00197      hindex = hindex + 12
00198
00199      CASE("spd")
00200
00201      hss = wss(elempointer(i))*deltaspin(index + 1)
00202      hpp = wpp(elempointer(i))*deltaspin(index + 2)
00203      hdd = wdd(elempointer(i))*deltaspin(index + 3)
00204      index = index + 3
00205
00206      h2vect(hindex + 1) = hss
00207      h2vect(hindex + 2) = hpp
00208      h2vect(hindex + 3) = hpp
00209      h2vect(hindex + 4) = hpp
00210      h2vect(hindex + 5) = hdd
00211      h2vect(hindex + 6) = hdd
00212      h2vect(hindex + 7) = hdd
00213      h2vect(hindex + 8) = hdd
00214      h2vect(hindex + 9) = hdd
00215      hindex = hindex + 9
00216
00217      CASE("spf")
00218
00219      hss = wss(elempointer(i))*deltaspin(index + 1)
00220      hpp = wpp(elempointer(i))*deltaspin(index + 2)
00221      hff = wff(elempointer(i))*deltaspin(index + 3)
00222      index = index + 3
00223
00224      h2vect(hindex + 1) = hss
00225      h2vect(hindex + 2) = hpp
00226      h2vect(hindex + 3) = hpp
00227      h2vect(hindex + 4) = hpp
00228      h2vect(hindex + 5) = hff
00229      h2vect(hindex + 6) = hff
00230      h2vect(hindex + 7) = hff
00231      h2vect(hindex + 8) = hff
00232      h2vect(hindex + 9) = hff
00233      h2vect(hindex + 10) = hff
00234      h2vect(hindex + 11) = hff
00235      hindex = hindex + 11
00236
00237      CASE("sdf")
00238
00239      hss = wss(elempointer(i))*deltaspin(index + 1)
00240      hdd = wdd(elempointer(i))*deltaspin(index + 2)
00241      hff = wff(elempointer(i))*deltaspin(index + 3)
00242      index = index + 3
00243
00244      h2vect(hindex + 1) = hss
00245      h2vect(hindex + 2) = hdd
00246      h2vect(hindex + 3) = hdd
00247      h2vect(hindex + 4) = hdd
00248      h2vect(hindex + 5) = hdd
00249      h2vect(hindex + 6) = hdd
00250      h2vect(hindex + 7) = hff
00251      h2vect(hindex + 8) = hff
00252      h2vect(hindex + 9) = hff
00253      h2vect(hindex + 10) = hff
00254      h2vect(hindex + 11) = hff
00255      h2vect(hindex + 12) = hff
00256      h2vect(hindex + 13) = hff
00257      hindex = hindex + 13
00258
00259      CASE("pdf")
00260
00261      hpp = wpp(elempointer(i))*deltaspin(index + 1)
00262      hdd = wdd(elempointer(i))*deltaspin(index + 2)
00263      hff = wff(elempointer(i))*deltaspin(index + 3)
00264      index = index + 3
00265
00266      h2vect(hindex + 1) = hpp
00267      h2vect(hindex + 2) = hpp

```

```

00268         h2vect(hindex + 3) = hpp
00269         h2vect(hindex + 4) = hdd
00270         h2vect(hindex + 5) = hdd
00271         h2vect(hindex + 6) = hdd
00272         h2vect(hindex + 7) = hdd
00273         h2vect(hindex + 8) = hdd
00274         h2vect(hindex + 9) = hff
00275         h2vect(hindex + 10) = hff
00276         h2vect(hindex + 11) = hff
00277         h2vect(hindex + 12) = hff
00278         h2vect(hindex + 13) = hff
00279         h2vect(hindex + 14) = hff
00280         h2vect(hindex + 15) = hff
00281         hindex = hindex + 15
00282
00283         CASE("spdf")
00284
00285         hss = wss(elempointer(i))*deltaspin(index + 1)
00286         hpp = wpp(elempointer(i))*deltaspin(index + 2)
00287         hdd = wdd(elempointer(i))*deltaspin(index + 3)
00288         hff = wff(elempointer(i))*deltaspin(index + 4)
00289         index = index + 4
00290
00291         h2vect(hindex + 1) = hss
00292         h2vect(hindex + 2) = hpp
00293         h2vect(hindex + 3) = hpp
00294         h2vect(hindex + 4) = hpp
00295         h2vect(hindex + 5) = hdd
00296         h2vect(hindex + 6) = hdd
00297         h2vect(hindex + 7) = hdd
00298         h2vect(hindex + 8) = hdd
00299         h2vect(hindex + 9) = hdd
00300         h2vect(hindex + 10) = hff
00301         h2vect(hindex + 11) = hff
00302         h2vect(hindex + 12) = hff
00303         h2vect(hindex + 13) = hff
00304         h2vect(hindex + 14) = hff
00305         h2vect(hindex + 15) = hff
00306         h2vect(hindex + 16) = hff
00307         hindex = hindex + 16
00308
00309         END SELECT
00310
00311     ENDDO
00312
00313     IF (basistype .EQ. "ORTHO") THEN
00314
00315         hup = h
00316         hdown = h
00317
00318         DO i = 1, hdim
00319             hup(i,i) = hup(i,i) + h2vect(i)
00320             hdown(i,i) = hdown(i,i) - h2vect(i)
00321         ENDDO
00322
00323     ELSEIF (basistype .EQ. "NONORTHO") THEN
00324
00325         ! Now we have H_2, we can form SH_2, and then symmetrize it
00326
00327         DO i = 1, hdim
00328             DO j = 1, hdim
00329                 sh2(j,i) = smat(j,i)*h2vect(i)
00330             ENDDO
00331         ENDDO
00332
00333         sh2 = (sh2 + transpose(sh2))/two
00334
00335         ! H_up = H + SH2 ; H_down = H - SH2
00336
00337         hup = h + sh2
00338         hdown = h - sh2
00339
00340     ENDIF
00341
00342     RETURN
00343
00344 END SUBROUTINE bldspinh
00345
00346
00347
00348
00349
00350
00351

```

8.41 bm.f90 File Reference

Functions/Subroutines

- real(latteprec) function [bm](#) (M, ALPHA)

8.41.1 Function/Subroutine Documentation

8.41.1.1 real(latteprec) function bm (integer M, real(latteprec) ALPHA)

Definition at line 23 of file [bm.f90](#).

8.42 bm.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION bm(M, ALPHA)
00023
00024 ! Build Bm function defined in eqn. (6) of PRB 72 165107
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 REAL(LATTEPREC) :: BM, ALPHA
00031 ! REAL(LATTEPREC), EXTERNAL :: HEAVI
00032 INTEGER :: M
00033
00034 ! Removed the Heaviside step function: MJC 1/10/13
00035
00036 IF (m == 0) THEN
00037   bm = zero
00038 ELSE IF (m .GT. 0) THEN
00039   bm = -REAL((-1)**m, latteprec)*SIN(abs(m) * alpha)
00040 ELSE IF (m .LT. 0) THEN
00041   bm = REAL((-1)**M, LATTEPREC) * COS(abs(m) * alpha)
00042 ENDIF
00043
00044 RETURN
00045
00046 END FUNCTION bm

```

8.43 bodirect.f90 File Reference

Functions/Subroutines

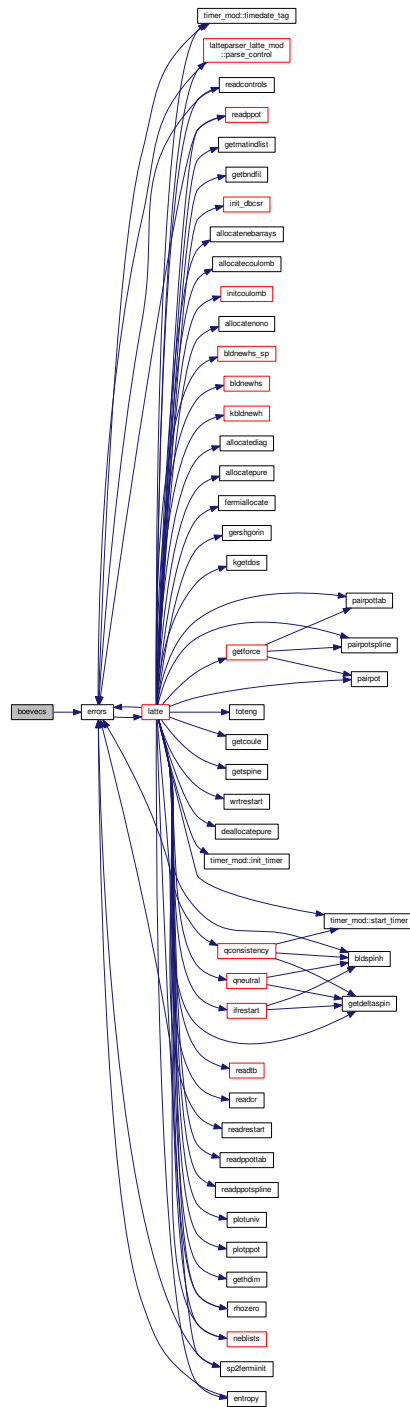
- subroutine [boevecs](#)

8.43.1 Function/Subroutine Documentation

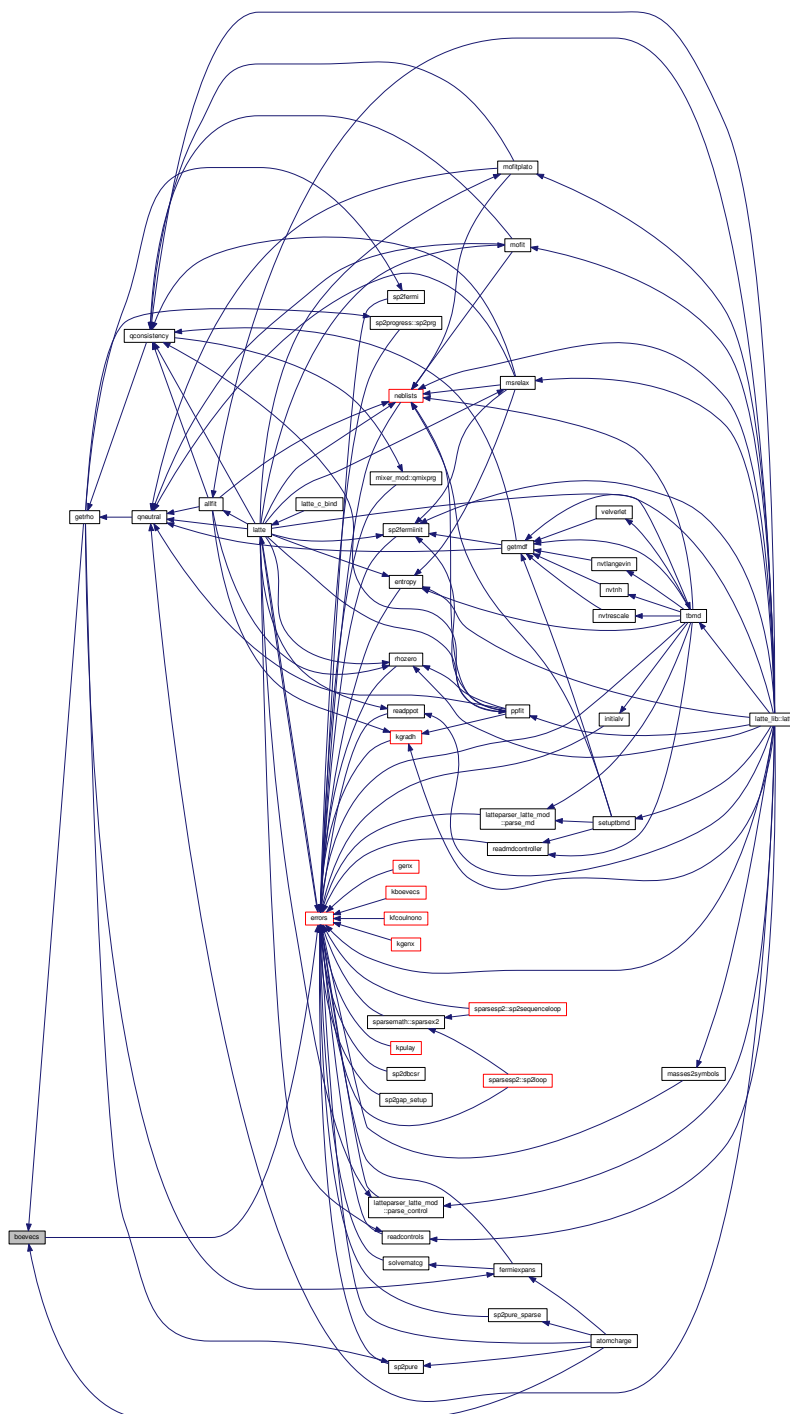
8.43.1.1 subroutine boevecs ()

Definition at line 23 of file [bodirect.f90](#).

Here is the call graph for this function:



8.44 bodirect.f90



```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE boevecs
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE myprecision
00028   USE mdarray
00029
00030   #ifdef PROGRESSON
00031     USE prg_densitymatrix_mod
00032   #endif
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K
00037   INTEGER :: ITER, BREAKLOOP, LOOPTARGET
00038   REAL(LATTEPREC) :: OCCTARGET, OCC, FDIRAC, DFDIRAC
00039   REAL(LATTEPREC) :: OCCERROR, SHIFTCP, FDIRACARG, EXPARG
00040   REAL(LATTEPREC), PARAMETER :: MAXSHIFT = one
00041   REAL(LATTEPREC) :: EBAND, QMIXORIG
00042   REAL(LATTEPREC) :: S, OCCLOGOCC_ELECTRONS, OCCLOGOCC_HOLES
00043
00044   IF (existerror) RETURN
00045
00046   bo = zero
00047
00048   occtarget = bndfil*REAL(hdim)
00049
00050   ! PRINT*, TOTNE, OCCTARGET
00051
00052   iter = 0
00053
00054   breakloop = 0
00055
00056   occerror = 1000000000.0
00057
00058   !
00059   ! The do-while loop uses a Newton-Raphson optimization of the chemical
00060   ! potential to obtain the correct occupation
00061   !
00062
00063   IF (kbt .GT. 0.000001) THEN ! This bit is for a finite electronic temperature
00064
00065     IF(verbose >= 2)WRITE(*,*)"Total charge =",sum(deltaq)
00066
00067   #ifdef PROGRESSON
00068     CALL prg_get_flevel(evals,kbt,bndfil,breaktol,chempot)
00069
00070   #else
00071
00072     DO WHILE (abs(occerror) .GT. breaktol .AND. iter .LT. 100)
00073
00074       iter = iter + 1
00075       occ = zero
00076       dffdirac = zero
00077
00078       DO i = 1, hdim
00079
00080         fdiracarg = (evals(i) - chempot)/kbt
00081
00082         fdiracarg = max(fdiracarg, -exptol)
00083         fdiracarg = min(fdiracarg, exptol)
00084
00085         exparg = exp(fdiracarg)
00086         fdirac = one/(one + exparg)
00087         occ = occ + fdirac
00088         dffdirac = dffdirac + exparg*fdirac*fdirac
00089
00090       ENDDO
00091
00092       dffdirac = dffdirac/kbt

```

```

00094
00095     occerror = occtarget - occ
00096
00097     IF (abs(dfdirac) .LT. numlimit) dfdirac = sign(numlimit, dfdirac)
00098
00099     shiftcp = occerror/dfdirac
00100
00101     IF (abs(shiftcp) .GT. maxshift) shiftcp = sign(maxshift, shiftcp)
00102
00103     chempot = chempot + shiftcp
00104
00105     !          PRINT*, CHEMPOT, OCCERROR
00106
00107     IF(verbose >= 2)WRITE(*,*)"Occupation error = ",occerror," Chemical potential = ",
chempot
00108
00109     ENDDO
00110
00111     IF (iter .EQ. 100) THEN
00112         CALL errors("bodirect","Newton-Raphson scheme to find the Chemical potential does not
converge")
00113         IF (existerror) RETURN
00114     ENDIF
00115
00116     ! Now we have the chemical potential we can build the density matrix
00117 #endif
00118
00119     s = zero
00120
00121     IF (mdon .EQ. 0 .OR. &
00122         (mdon .EQ. 1 .AND. mod(entropyiter, wrtfreq) .EQ. 0 )) THEN
00123
00124         DO i = 1, hdim
00125
00126             fdiracarg = (evals(i) - chempot)/kbt
00127
00128             fdiracarg = max(fdiracarg, -exptol)
00129             fdiracarg = min(fdiracarg, exptol)
00130
00131             fdirac = one/(one + exp(fdiracarg))
00132
00133             occlogocc_electrons = fdirac * log(fdirac)
00134             occlogocc_holes = (one - fdirac) * log(one - fdirac)
00135
00136             s = s + two*(occlogocc_electrons + occlogocc_holes)
00137
00138         ENDDO
00139
00140         ! Compute the gap only when we have to...
00141
00142         ! If we have an even number of electrons
00143
00144         IF (mod(int(totne),2) .EQ. 0) THEN
00145             egap = evals(int(occtarget) + 1) - evals(int(occtarget))
00146         ELSE
00147             egap = zero
00148         ENDIF
00149
00150     ENDIF
00151
00152     ente = -kbt*s
00153
00154     DO i = 1, hdim
00155
00156         fdiracarg = (evals(i) - chempot)/kbt
00157
00158         fdiracarg = max(fdiracarg, -exptol)
00159         fdiracarg = min(fdiracarg, exptol)
00160
00161         fdirac = one/(one + exp(fdiracarg))
00162
00163 #ifdef DOUBLEPREC
00164         CALL dger(hdim, hdim, fdirac, evecs(:,i), 1, evecs(:,i), 1,
bo, hdim)
00165 #elif defined(SINGLEPREC)
00166         CALL sger(hdim, hdim, fdirac, evecs(:,i), 1, evecs(:,i), 1,
bo, hdim)
00167 #endif
00168
00169     ENDDO
00170
00171     ELSE ! This bit is for zero electronic temperature
00172
00173     IF (mod(int(totne),2) .NE. 0) THEN
00174         CALL errors("bodirect","Odd number of electrons - run a spin-polarized calculation &
& or use a finite electron temperature")
00175     ENDIF
00176

```

```

00177
00178      !
00179      ! This definition of the chemical potential is a little arbitrary
00180      !
00181
00182      looptarget = nint(occtarget)
00183
00184      chempot = half*(evals(looptarget) + evals(looptarget + 1))
00185      egap = evals(looptarget + 1) - evals(looptarget)
00186
00187      DO i = 1, looptarget
00188
00189      #ifdef DOUBLEPREC
00190          CALL dger(hdim, hdim, one, evecs(:,i), 1, evecs(:,i), 1,
00191                  bo, hdim)
00192      #elif defined(SINGLEPREC)
00193          CALL sger(hdim, hdim, one, evecs(:,i), 1, evecs(:,i), 1,
00194                  bo, hdim)
00195      #endif
00196      ENDDO
00197
00198      IF (mdon .EQ. 1 .AND. mdadapt .EQ. 1) THEN
00199
00200          fullqconv = 0
00201
00202          IF (egap .LT. 1.0d0) THEN
00203              fullqconv = 1
00204              mdmix = 0.1
00205          ELSE
00206              qiter = 1
00207              mdmix = 0.25
00208          ENDIF
00209
00210      ENDIF
00211
00212      bo = two * bo
00213
00214      RETURN
00215
00216 END SUBROUTINE boevecs

```

8.45 bodirectprogress.f90 File Reference

Functions/Subroutines

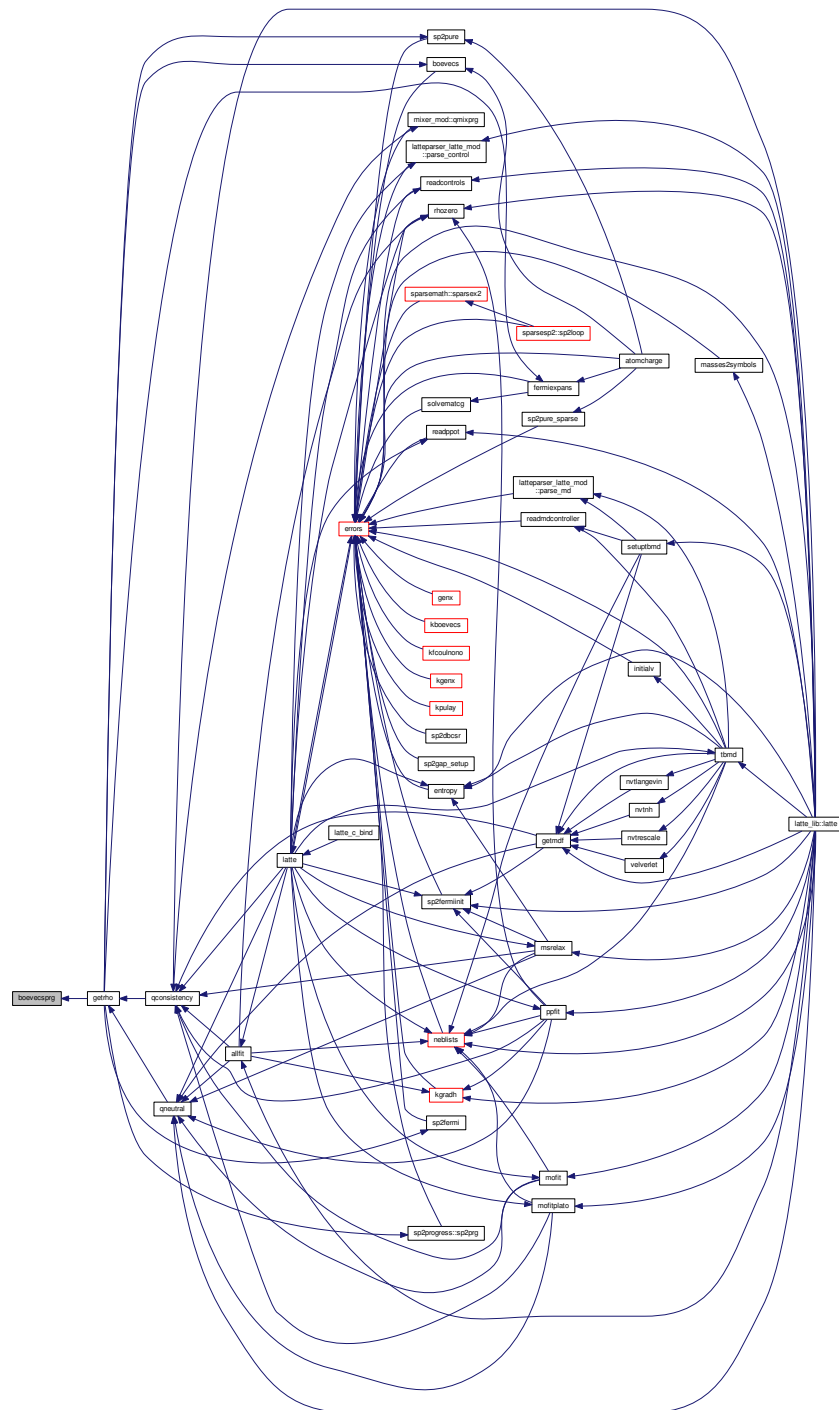
- subroutine [boevecsprg](#)

8.45.1 Function/Subroutine Documentation

8.45.1.1 subroutine boevecsprg ()

Definition at line 23 of file [bodirectprogress.f90](#).

Here is the caller graph for this function:



8.46 bodirectprogress.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE boevecsprg
00023
00024 #ifdef PROGRESSON
00025
00026     USE constants_mod
00027     USE setuparray
00028     USE diagarray
00029     USE myprecision
00030     USE mdarray
00031     USE nonoarray
00032     USE sparsearray
00033
00034     USE bml
00035     USE prg_densitymatrix_mod
00036
00037     IMPLICIT NONE
00038
00039     INTEGER :: I, J, K
00040     INTEGER :: ITER, BREAKLOOP, LOOPTARGET
00041     REAL(LATTEPREC) :: OCCTARGET, OCC, FDIRAC, FDIRAC
00042     REAL(LATTEPREC) :: OCCERROR, SHIFTCP, FDIRACARG, EXPARG
00043     REAL(LATTEPREC), PARAMETER :: MAXSHIFT = one
00044     REAL(LATTEPREC) :: EBAND, QMIXORIG
00045     REAL(LATTEPREC) :: S, OCCLOGOCC_ELECTRONS, OCCLOGOCC_HOLES
00046     TYPE(bml_matrix_t) :: ORTHOH_BML, BO_BML
00047
00048     IF (verbose .GE. 1) WRITE(*,*) "In BOEVECSPRG ..."
00049
00050     bo = zero
00051
00052     ! OCCTARGET = BNDFIL*REAL(HDIM)
00053
00054     !! Convert Hamiltonian to bml format
00055     !! H should be in orthogonal form, ORTHOH
00056     CALL bml_zero_matrix(bml_matrix_dense, bml_element_real, &
00057         latteprec, hdim, hdim, orthoh_bml)
00058     CALL bml_zero_matrix(bml_matrix_dense, bml_element_real, &
00059         latteprec, hdim, hdim, bo_bml)
00060     CALL bml_import_from_dense(bml_matrix_dense, &
00061         orthoh, orthoh_bml, zero, hdim)
00062
00063     IF (kbt .GT. 0.000001) THEN ! This bit is for a finite electronic temperature
00064         CALL prg_build_density_t(orthoh_bml,bo_bml,numthresh,bndfil,
00065             kbt,chempot)
00066     ELSE ! This bit is for zero electronic temperature
00067         CALL prg_build_density_t0(orthoh_bml,bo_bml,numthresh,bndfil)
00068     ENDIF
00069
00070     CALL bml_export_to_dense(bo_bml, bo)
00071
00072     CALL bml_deallocate(bo_bml)
00073     CALL bml_deallocate(orthoh_bml)
00074
00075     IF (mdon .EQ. 1 .AND. mdadapt .EQ. 1) THEN
00076         fullqconv = 0
00077         IF (egap .LT. 1.0d0) THEN
00078             fullqconv = 1
00079             mdmix = 0.1
00080         ELSE
00081             qiter = 1
00082             mdmix = 0.25
00083         ENDIF
00084     ENDIF
00085
00086 ENDIF

```

```

00093
00094      ! BO = TWO * BO
00095
00096      RETURN
00097
00098 #endif
00099
00100 END SUBROUTINE boevecsprg

```

8.47 caselist.f90 File Reference

8.48 caselist.f90

```

00001      CASE ("s")
00002
00003
00004
00005      CASE ("p")
00006
00007
00008      CASE ("d")
00009
00010
00011
00012      CASE ("f")
00013
00014
00015
00016      CASE ("sp")
00017
00018
00019      CASE ("sd")
00020
00021
00022
00023      CASE ("sf")
00024
00025
00026
00027
00028      CASE ("pd")
00029
00030
00031
00032      CASE ("pf")
00033
00034
00035      CASE ("df")
00036
00037
00038      CASE ("spd")
00039
00040
00041
00042      CASE ("spf")
00043
00044
00045
00046      CASE ("sdf")
00047
00048
00049
00050      CASE ("pdf")
00051
00052
00053
00054      CASE ("spdf")
00055
00056
00057
00058      END SELECT

```

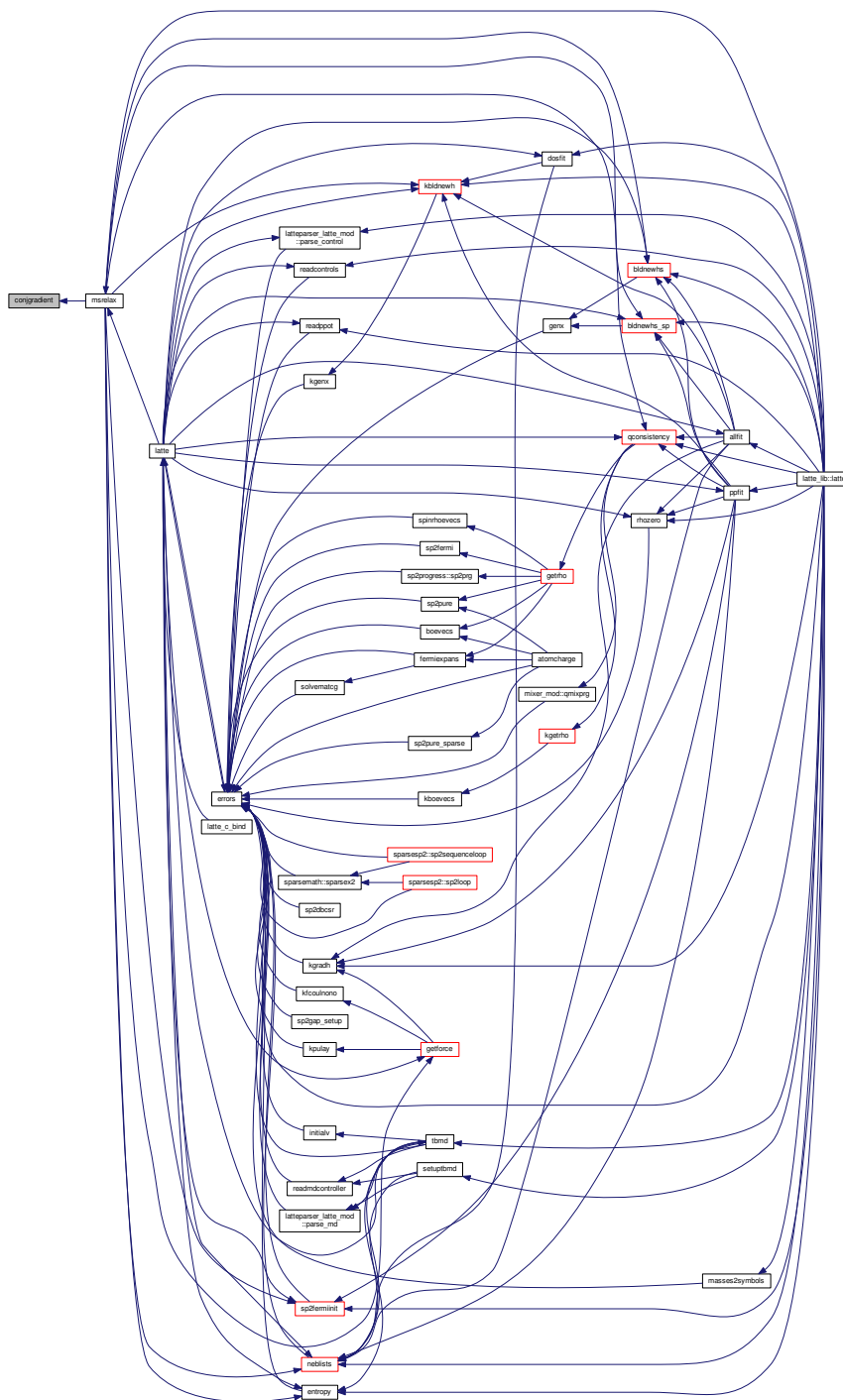
8.49 conjgradient.f90 File Reference

Functions/Subroutines

- subroutine [conjgradient](#) (ITER, DELTAENERGY)

8.49.1.1 subroutine conjgradient (integer *ITER*, real(latteprec) *DELTAENERGY*)

Here is the caller graph for this function:



8.50 conjgradient.f90

```

00001 SUBROUTINE conjgradient (ITER, DELTAENERGY)
00002
00003   USE constants_mod
00004   USE setuparray
00005   USE relaxcommon
00006   USE myprecision
00007
00008   IMPLICIT NONE
00009
00010   INTEGER :: I, J, ITER
00011   REAL(LATTEPREC) :: DELTAENERGY
00012   REAL(LATTEPREC) :: MAXSHIFT = 0.1d0, kappa = 0.01d0
00013   REAL(LATTEPREC) :: BETA, BETA1, BETA2, MYSHIFT
00014   IF (existerror) RETURN
00015
00016   ! On the first step, or every so often because
00017   ! of a loss of conjugacy we do a molecular statics step
00018
00019   IF ( iter .EQ. 1 .OR. mod(iter, 5) .EQ. 0 ) THEN
00020
00021       DO i = 1, nats
00022           DO j = 1, 3
00023
00024               !
00025               ! Limit maximum displacement in case the initial
00026               ! geometry is not good
00027               !
00028
00029               myshift = kappa*ftot(j,i)
00030
00031               IF (abs( myshift ) .GT. maxshift) THEN
00032                   myshift = sign(maxshift, myshift)
00033               ENDIF
00034
00035               cr(j,i) = cr(j,i) + myshift
00036
00037           ENDDO
00038       ENDDO
00039
00040       oldf = ftot
00041       oldd = ftot
00042
00043   ELSE
00044
00045       beta1 = zero
00046       beta2 = zero
00047       DO i = 1, nats
00048
00049           !
00050           ! Use the Polak-Ribero Beta
00051           !
00052
00053           DO j = 1, 3
00054
00055               beta1 = beta1 + ftot(j,i) * (ftot(j,i) - oldf(j,i))
00056               beta2 = beta2 + oldf(j,i) * oldf(j,i)
00057
00058           ENDDO
00059       ENDDO
00060
00061       IF ( abs(beta2) .LT. 1.0d-6 ) THEN
00062
00063           beta = zero
00064
00065       ELSE
00066
00067           beta = beta1/beta2
00068
00069       ENDIF
00070
00071       DO i = 1, nats
00072
00073           DO j = 1, 3
00074
00075               dl(j,i) = ftot(j,i) + beta*oldd(j,i)
00076
00077               myshift = kappa*dl(j,i)
00078
00079               IF ( abs( myshift ) .GT. maxshift ) THEN
00080                   myshift = sign( maxshift, myshift )
00081               ENDIF
00082
00083               cr(j,i) = cr(j,i) + myshift
00084

```

```

00085
00086         ENDDO
00087     ENDDO
00088
00089     oldf = ftot
00090     oldd = d1
00091
00092     ENDIF
00093
00094     RETURN
00095
00096 END SUBROUTINE conjgradient

```

8.51 constants_mod.f90 File Reference

Modules

- module [constants_mod](#)

Variables

- integer [constants_mod::allfiton](#)
- character(len=20) [constants_mod::basistype](#)
- integer [constants_mod::binfitstep](#)
- integer [constants_mod::bint2fit](#)
- integer [constants_mod::blkksz](#)
- real(latteprec) [constants_mod::bndfil](#)
- real(latteprec), dimension(3, 3) [constants_mod::box](#)
- real(latteprec), dimension(3, 3) [constants_mod::box_old](#)
- real(latteprec), dimension(3) [constants_mod::boxdims](#)
- real(latteprec) [constants_mod::breaktol](#)
- integer [constants_mod::cgorlib](#)
- integer [constants_mod::charge](#)
- real(latteprec) [constants_mod::chempot](#)
- real(latteprec) [constants_mod::chtol](#)
- integer [constants_mod::compforce](#)
- integer [constants_mod::control](#)
- character(len=100) [constants_mod::coordsfile](#) = `"/bl/inputblock.dat"`
- real(latteprec) [constants_mod::cove](#)
- integer [constants_mod::debugon](#)
- integer [constants_mod::dosfiton](#)
- real(latteprec) [constants_mod::ecoul](#)
- real(latteprec) [constants_mod::egap](#)
- real(latteprec) [constants_mod::ehomo](#)
- real(latteprec) [constants_mod::elec_etol](#)
- real(latteprec) [constants_mod::elec_qtol](#)
- integer [constants_mod::elecmeth](#)
- integer [constants_mod::electro](#)
- real(latteprec) [constants_mod::elumo](#)
- integer [constants_mod::emeth](#)
- real(latteprec) [constants_mod::ente](#)
- integer [constants_mod::entropykind](#)
- real(latteprec) [constants_mod::eps](#)
- real(latteprec) [constants_mod::erep](#)
- real(latteprec) [constants_mod::espin](#)

- real(latteprec) [constants_mod::espin_zero](#)
- logical(1) [constants_mod::existerror](#)
- real(latteprec), parameter [constants_mod::f2v](#) = 9.6484504393669415d-003
- integer [constants_mod::fiton](#)
- integer [constants_mod::freeze](#)
- integer [constants_mod::fullqconv](#)
- integer [constants_mod::hdim](#)
- integer [constants_mod::int2fit](#)
- character(len=20) [constants_mod::job](#)
- real(latteprec) [constants_mod::kbt](#)
- real(latteprec), parameter [constants_mod::ke2t](#) = 1.0/0.000086173435
- real(latteprec) [constants_mod::kee](#)
- integer [constants_mod::kon](#)
- logical [constants_mod::latteinexists](#)
- integer [constants_mod::lcniter](#)
- integer [constants_mod::lcnon](#)
- integer [constants_mod::libcalls](#) = 0
- logical [constants_mod::libinit](#) = .FALSE.
- logical [constants_mod::librun](#) = .FALSE.
- real(latteprec) [constants_mod::massden](#)
- real(latteprec) [constants_mod::maxeval](#)
- real(latteprec) [constants_mod::maxminusmin](#)
- integer [constants_mod::maxscf](#)
- real(latteprec) [constants_mod::mcbeta](#)
- real(latteprec) [constants_mod::mcsigma](#)
- integer [constants_mod::mdadapt](#)
- real(latteprec) [constants_mod::mdmix](#)
- integer [constants_mod::mdon](#)
- real(latteprec) [constants_mod::mineval](#)
- integer [constants_mod::minsp2iter](#)
- integer [constants_mod::mixer](#)
- real(latteprec), parameter [constants_mod::mvv2ke](#) = 166.0538782/1.602176487
- real(latteprec), parameter [constants_mod::mvv2t](#) = 1660538.782/1.3806504
- integer [constants_mod::nats](#)
- integer [constants_mod::nfitstep](#)
- integer [constants_mod::ngpu](#)
- integer [constants_mod::nmat](#)
- integer [constants_mod::noelem](#)
- integer [constants_mod::noint](#)
- integer [constants_mod::norecs](#)
- integer [constants_mod::numscf](#)
- real(latteprec) [constants_mod::occerrlimit](#)
- integer [constants_mod::occsteps](#)
- integer [constants_mod::ordernmol](#)
- character(len=100) [constants_mod::parampath](#) = ".TBparam"
- integer [constants_mod::parrep](#)
- integer [constants_mod::pbcon](#)
- real(latteprec), parameter [constants_mod::pi](#) = TWO*ACOS(ZERO)
- integer [constants_mod::pp2fit](#)
- real(latteprec) [constants_mod::ppbeta](#)
- integer [constants_mod::ppfiton](#)
- integer [constants_mod::ppnfitstep](#)
- integer [constants_mod::ppngeom](#)
- integer [constants_mod::ppnmol](#)
- integer [constants_mod::ppoton](#)

- real(latteprec) `constants_mod::ppsigma`
- integer `constants_mod::qfit`
- integer `constants_mod::qiter`
- real(latteprec) `constants_mod::qmix`
- integer `constants_mod::relaxme`
- integer `constants_mod::restart`
- integer `constants_mod::restartlib`
- integer `constants_mod::rslevel`
- integer `constants_mod::scfs`
- integer `constants_mod::scfs_ii`
- integer `constants_mod::scfstep` = 0
- character(len=3) `constants_mod::sp2conv`
- integer `constants_mod::sparseon`
- real(latteprec) `constants_mod::spnmix`
- integer `constants_mod::spinon`
- real(latteprec) `constants_mod::spintol`
- integer `constants_mod::sponly`
- real(latteprec) `constants_mod::summass`
- real(latteprec), parameter `constants_mod::togpa` = 160.2176487
- real(latteprec) `constants_mod::tote`
- real(latteprec) `constants_mod::totne`
- real(latteprec) `constants_mod::tracelimit`
- real(latteprec) `constants_mod::trrhoh`
- real(latteprec) `constants_mod::tscale`
- integer `constants_mod::vardt`
- integer `constants_mod::vdwon`
- integer `constants_mod::verbose` = 1
- integer `constants_mod::xbodison`
- integer `constants_mod::xbodisorder`
- integer `constants_mod::xboon`

8.52 constants_mod.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE constants_mod
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER :: nats, nmat, noint, noelem, hdim
00030   INTEGER :: control, relaxme, pbcon, restart, mdon
00031   INTEGER :: xboon, xbodison, xbodisorder

```

```

00032 INTEGER :: scfs, scfs_ii
00033 INTEGER :: lcnon, lcniter
00034 INTEGER :: electro, elecmeth
00035 INTEGER :: ppoton, vdwon
00036 INTEGER :: fullqconv, qiter
00037 INTEGER :: emeth, cgorlib
00038 INTEGER :: spinon, sparseon
00039 INTEGER :: ordernmol
00040 INTEGER :: maxscf, minsp2iter
00041 INTEGER :: vardt
00042 INTEGER :: norecs, entropykind
00043 INTEGER :: debugon, fiton
00044 INTEGER :: blkksz ! Block size when using DBCSR
00045 INTEGER :: ngpu ! Number of GPUs (blimey!)
00046 INTEGER :: numscf
00047 INTEGER :: charge
00048 INTEGER :: kon ! K-SPACE FLAG
00049 INTEGER :: compforce
00050 INTEGER :: sponly ! If we only have sp-bonded elements: faster gradH
00051 INTEGER :: dosfiton, int2fit, nfitstep, qfit ! For the simulated annealing
00052 INTEGER :: pp2fit, bint2fit, ppnfitstep, ppmol,
ppngeom, binfitstep
00053 INTEGER :: pffiton, allfiton
00054 INTEGER :: mdadapt
00055 INTEGER :: parrep
00056 INTEGER :: verbose = 1
00057 INTEGER :: mixer
00058 INTEGER :: rslevel
00059 INTEGER :: restartlib
00060 INTEGER :: freeze
00061 REAL(LATTEPREC) :: box(3,3), box_old(3,3), boxdims(3)
00062 REAL(LATTEPREC) :: bndfil, totne
00063 REAL(LATTEPREC) :: cove, tote, ente, kee, ecoul, erep,
trrhoh
00064 REAL(LATTEPREC) :: espin, espin_zero
00065 REAL(LATTEPREC) :: mineval, maxeval, maxminusmin
00066 REAL(LATTEPREC) :: chempot, kbt
00067 REAL(LATTEPREC) :: egap, ehomo, elumo
00068 REAL(LATTEPREC) :: chtol, spintol
00069 REAL(LATTEPREC) :: elec_etol, elec_qtol
00070 REAL(LATTEPREC) :: qmix, spinmix, mdmix
00071 REAL(LATTEPREC) :: breaktol
00072 REAL(LATTEPREC) :: summass, massden
00073 REAL(LATTEPREC) :: mcbeta, mcsigma ! Temperature in simulated annealing
00074 REAL(LATTEPREC) :: ppbeta, ppsigma
00075 CHARACTER(LEN = 3) :: sp2conv
00076 CHARACTER(LEN = 20) :: basistype
00077
00078 CHARACTER(LEN = 100) :: parampath = "./TBparam"
00079 CHARACTER(LEN = 100) :: coordsfile = "./bl/inputblock.dat"
00080
00081 ! For the latte lib
00082 CHARACTER(LEN = 20) :: job
00083 LOGICAL :: libinit = .false.
00084 INTEGER :: libcalls = 0
00085 LOGICAL(1) :: existerror
00086 LOGICAL :: librun = .false.
00087
00088 !For the new input file parser
00089 LOGICAL :: latteinexists
00090
00091 !! For truncated SP2 and entropy calculation
00092 INTEGER :: scfstep = 0
00093 INTEGER :: occsteps
00094 REAL(LATTEPREC) :: tscale
00095 REAL(LATTEPREC) :: occerlimit, tracelimit, eps
00096
00097 ! Some often-used constants
00098
00099 REAL(LATTEPREC), PARAMETER :: mvv2ke = 166.0538782/1.602176487
00100 REAL(LATTEPREC), PARAMETER :: ke2t = 1.0/0.000086173435
00101 REAL(LATTEPREC), PARAMETER :: f2v = 9.6484504393669415d-003
00102 REAL(LATTEPREC), PARAMETER :: togpa = 160.2176487
00103 REAL(LATTEPREC), PARAMETER :: mvv2t = 1660538.782/1.3806504
00104 ! REAL(LATTEPREC), PARAMETER :: PI = 3.14159265358979323846264D0
00105 REAL(LATTEPREC), PARAMETER :: pi = two*acos(zero)
00106
00107 END MODULE constraints_mod

```

8.53 constraints_mod.f90 File Reference

Data Types

- type `constraints_mod::constrains_type`

General input variables for constrains.

Modules

- module `constraints_mod`

! To add constrains to the system. This module can be used to add constrains to the system. For example, fixing the position of certain atoms.

Functions/Subroutines

- subroutine, public `constraints_mod::freeze_atoms` (FTOT, VEL)

To freeze a group of atoms. The atoms indices to be frozen are read from a file. The file is formatted as follows:

8.54 constraints_mod.f90

```

00001
00002 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00003 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00004 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00005 ! National Laboratory (LANL), which is operated by Los Alamos National !
00006 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00007 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00008 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00009 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00010 ! SOFTWARE. If software is modified to produce derivative works, such !
00011 ! modified software should be clearly marked, so as not to confuse it !
00012 ! with the version available from LANL. !
00013 ! !
00014 ! Additionally, this program is free software; you can redistribute it !
00015 ! and/or modify it under the terms of the GNU General Public License as !
00016 ! published by the Free Software Foundation; version 2.0 of the License. !
00017 ! Accordingly, this program is distributed in the hope that it will be !
00018 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00019 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00020 ! Public License for more details. !
00021 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00022
00023
00029 MODULE constraints_mod
00030
00031     USE constants_mod
00032
00033     USE kernelparser_mod
00034
00035     #ifdef PROGRESSON
00036     USE bml
00037     #endif
00038
00039     IMPLICIT NONE
00040
00041     PRIVATE
00042
00043     INTEGER, PARAMETER :: dp = latteprec
00044
00045     PUBLIC :: freeze_atoms
00046
00050     TYPE, PUBLIC :: constrains_type
00051
00053         INTEGER :: verbose
00054
00056         CHARACTER(20) :: method
00057
00058     END TYPE constrains_type
00059
00060 CONTAINS
00061
00075 SUBROUTINE freeze_atoms(FTOT,VEL)

```

```

00076
00077     INTEGER, SAVE :: NFREEZE
00078     INTEGER, SAVE, ALLOCATABLE :: FREEZEID(:)
00079     INTEGER :: I
00080     REAL(DP), OPTIONAL, INTENT(INOUT) :: VEL(:, :)
00081     REAL(DP), INTENT(INOUT) :: FTOT(:, :)
00082
00083     IF (.NOT. ALLOCATED(freezeid)) THEN
00084         OPEN(444, file="freeze.in")
00085         READ(444, *) nfreeze
00086         ALLOCATE(freezeid(nfreeze))
00087         DO i = 1, nfreeze
00088             READ(444, *) freezeid(i)
00089         ENDDO
00090         CLOSE(444)
00091     ENDIF
00092
00093     IF (PRESENT(vel)) THEN
00094         DO i = 1, nfreeze
00095             vel(:, freezeid(i)) = 0.0d0
00096             ftot(:, freezeid(i)) = 0.0d0
00097         ENDDO
00098     ELSE
00099         DO i = 1, nfreeze
00100             ftot(:, freezeid(i)) = 0.0d0
00101         ENDDO
00102     ENDIF
00103
00104     END SUBROUTINE freeze_atoms
00105
00106 END MODULE constraints_mod

```

8.55 coulomb_ewald.f90 File Reference

Functions/Subroutines

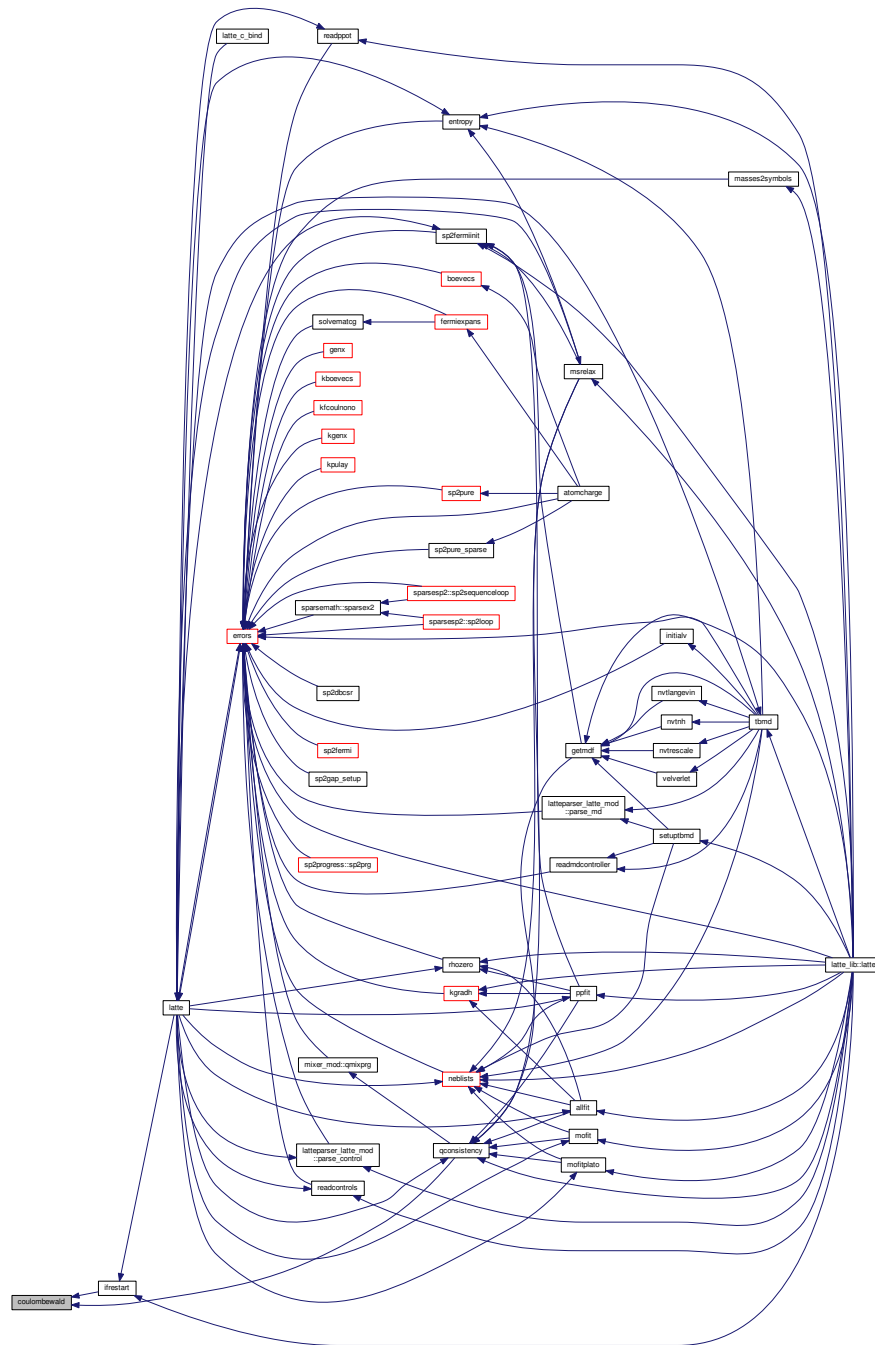
- subroutine [coulombewald](#)

8.55.1 Function/Subroutine Documentation

8.55.1.1 subroutine [coulombewald](#) ()

Definition at line 23 of file [coulomb_ewald.f90](#).

Here is the caller graph for this function:



8.56 coulomb_ewald.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !

```



```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE coulombewald
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE virialarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, L, M, N, IK
00033   INTEGER :: LMIN, MMIN, NMIN
00034   REAL(LATTEPREC) :: PREFACTOR, KEPREF, K2, K(3), DOT
00035   REAL(LATTEPREC) :: COSSUM, SINUM, COSSUM2, SINUM2
00036   REAL(LATTEPREC) :: FLL, FLM, FLN
00037   REAL(LATTEPREC) :: FORCE
00038   REAL(LATTEPREC) :: PREVIR, CORRFACT
00039   REAL(LATTEPREC) :: L11, L12, L13, M21, M22, M23
00040
00041   IF (existerror) RETURN
00042
00043   ! $OMP PARALLEL DO DEFAULT(NONE) &
00044   ! $OMP SHARED(NATS,DELTAQ,K1_LIST,K2_LIST,K3_LIST,KSQ_LIST) &
00045   ! $OMP SHARED(KCUTOFF2,EIGHTPI,FOURCALPHA2,COULVOL,KECONST,CR) &
00046   ! $OMP PRIVATE(I,IK,DOT,FORCE,SINLIST,COSLIST,COSSUM,SINUM) &
00047   ! $OMP PRIVATE(K2,PREFACTOR,PREVIR,COSSUM2,SINUM2,KEPREF) &
00048   ! $OMP REDUCTION(+:FCOUL,VIRCOUL,COULOMBV)
00049   DO ik = 1, nk
00050
00051     k2 = ksq_list(ik)
00052
00053     IF (k2 .LE. kcutoff2) THEN
00054
00055       prefactor = eightpi*exp( -k2/(fourcalpha2) ) / &
00056         (coulvol*k2)
00057
00058       previr = (two/k2) + (two/(fourcalpha2))
00059
00060       cossum = zero
00061       sinsum = zero
00062
00063       ! Doing the sin and cos sums
00064
00065       DO i = 1, nats
00066
00067         dot = k1_list(ik)*cr(1,i) + k2_list(ik)*cr(2,i) +
00068           k3_list(ik)*cr(3,i)
00069
00070         ! We re-use these in the next loop...
00071
00072         sinlist(i) = sin(dot)
00073         coslist(i) = cos(dot)
00074
00075         cossum = cossum + deltaq(i)*coslist(i)
00076         sinsum = sinsum + deltaq(i)*sinlist(i)
00077
00078       ENDDO
00079
00080       cossum2 = cossum*cossum
00081       sinsum2 = sinsum*sinsum
00082
00083       ! Add up energy and force contributions
00084
00085       kepref = keconst*prefactor
00086
00087       DO i = 1, nats
00088
00089         coulombv(i) = coulombv(i) + &
00090           kepref*(coslist(i)*cossum + sinlist(i)*sinsum)
00091
00092         force = kepref * deltaq(i) * &
00093           (sinlist(i)*cossum - coslist(i)*sinsum)
00094
00095         fcoul(1,i) = fcoul(1,i) + force*k1_list(ik)
00096         fcoul(2,i) = fcoul(2,i) + force*k2_list(ik)

```

```

00096         fcoul(3,i) = fcoul(3,i) + force*k3_list(ik)
00097
00098     ENDDO
00099
00100     kepref = -kepref * (cossum2 + sinsum2)/two
00101
00102     vircou(1) = vircou(1) + kepref * &
00103         (one - previr*k1_list(ik)*k1_list(ik))
00104     vircou(2) = vircou(2) + kepref * &
00105         (one - previr*k2_list(ik)*k2_list(ik))
00106     vircou(3) = vircou(3) + kepref * &
00107         (one - previr*k3_list(ik)*k3_list(ik))
00108     vircou(4) = vircou(4) - kepref*previr* &
00109         k1_list(ik)*k2_list(ik)
00110     vircou(5) = vircou(5) - kepref*previr* &
00111         k2_list(ik)*k3_list(ik)
00112     vircou(6) = vircou(6) - kepref*previr* &
00113         k3_list(ik)*k1_list(ik)
00114
00115     ENDIF
00116
00117 END DO
00118 ! $OMP END PARALLEL DO
00119
00120 ! Point self energy
00121
00122 corrfact = two*keconst*calpha/sqrtpi
00123
00124 coulombv = coulombv - corrfact*deltaq
00125
00126 RETURN
00127
00128 END SUBROUTINE coulombewald

```

8.57 coulomb_oldskool.f90 File Reference

Functions/Subroutines

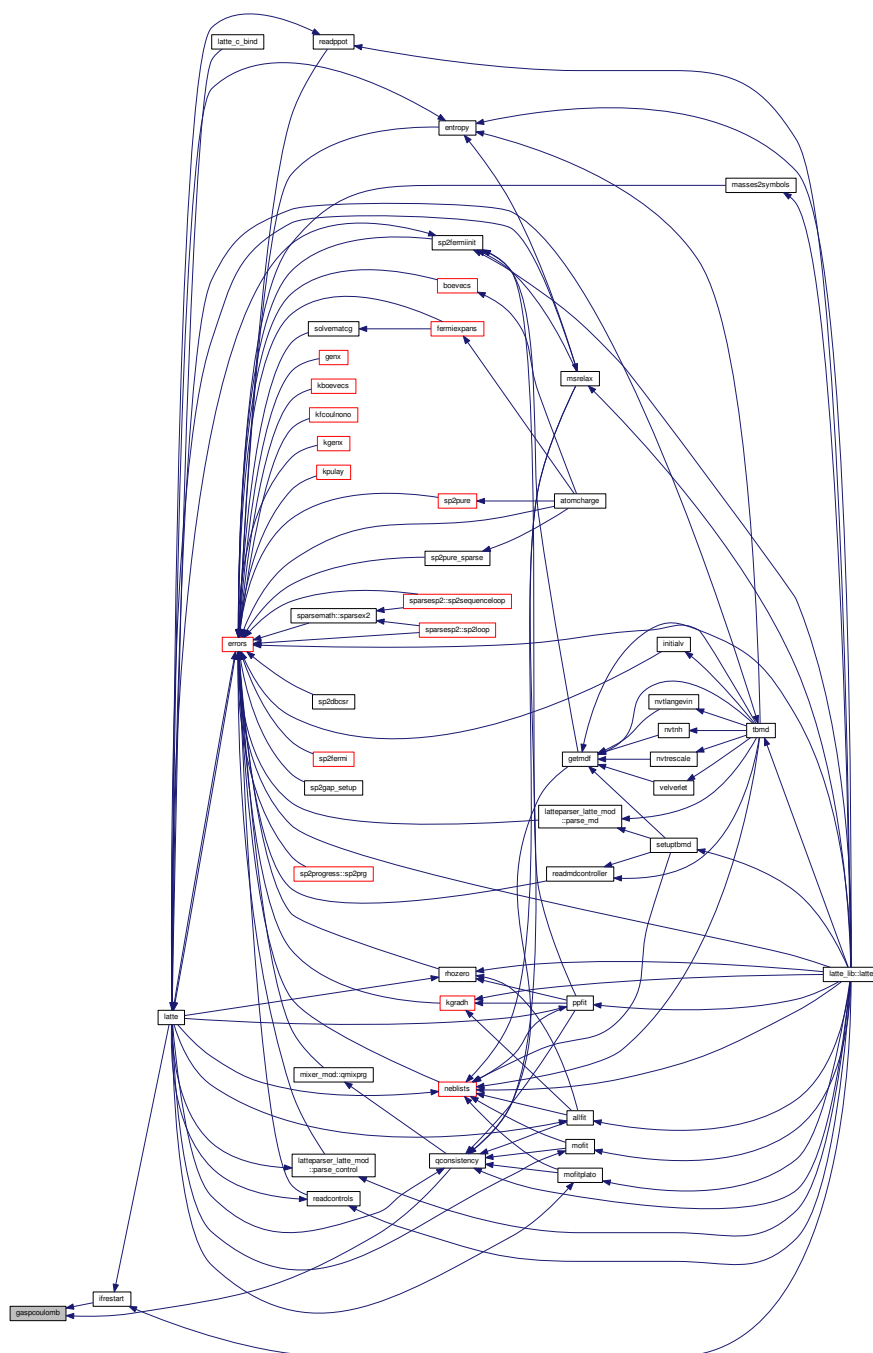
- subroutine [gaspoulomb](#)

8.57.1 Function/Subroutine Documentation

8.57.1.1 subroutine [gaspoulomb](#) ()

Definition at line 23 of file [coulomb_oldskool.f90](#).

Here is the caller graph for this function:



8.58 coulomb_oldskool.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National     !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,       !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE. If software is modified to produce derivative works, such
```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                   !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gaspcoulomb
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE neblistarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, NEWJ
00034   INTEGER :: PBCI, PBCJ, PBCK
00035   REAL(LATTEPREC) :: RIJ(3), MAGR, MAGR2, DC(3)
00036   REAL(LATTEPREC) :: FORCE, TI, TJ
00037   REAL(LATTEPREC) :: TI2, TI3, TI4, TI6, TJ2, TJ3, TJ4, TJ6
00038   REAL(LATTEPREC) :: EXPTI, EXPTJ
00039   REAL(LATTEPREC) :: TI2MTJ2, TJ2MTI2
00040   REAL(LATTEPREC) :: SA, SB, SC, SD, SE, SF
00041   REAL(LATTEPREC) :: SSA, SSB, SSC, SSD, SSE
00042   REAL(LATTEPREC) :: TAILR, DTAILR, MYR, MYR2, MYR3, MYR4, MYR5
00043
00044   IF (existerror) RETURN
00045
00046   keconst = 14.3996437701414*relperm
00047   tfact = 16.0/(5.0 * keconst)
00048
00049   fcoul = zero
00050   coulombv = zero
00051
00052   vircoul = zero
00053
00054   DO i = 1, nats
00055
00056     ti = tfact*hubbardu(elempointer(i))
00057
00058     ti2 = ti*ti
00059     ti3 = ti2*ti
00060     ti4 = ti2*ti2
00061     ti6 = ti4*ti2
00062
00063     ssa = ti
00064     ssb = ti3/fortyeight
00065     ssc = three*ti2/sixteen
00066     ssd = eleven*ti/sixteen
00067     sse = one
00068
00069     DO newj = 1, totnebcoul(i)
00070
00071       j = nebcoul(1, newj, i)
00072       pbcj = nebcoul(2, newj, i)
00073       pbcj = nebcoul(3, newj, i)
00074       pbck = nebcoul(4, newj, i)
00075
00076       rij(1) = cr(1,j) + REAL(pbcj)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00077         REAL(pbck)*BOX(3,1) - CR(1,i)
00078
00079       rij(2) = cr(2,j) + REAL(pbcj)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00080         REAL(pbck)*BOX(3,2) - CR(2,i)
00081
00082       rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00083         REAL(pbck)*BOX(3,3) - CR(3,i)
00084
00085
00086       !      RIJ(1) = CR(1,J) + PBCI*BOXDIMS(1) - CR(1,I)
00087       !      RIJ(2) = CR(2,J) + PBCJ*BOXDIMS(2) - CR(2,I)
00088       !      RIJ(3) = CR(3,J) + PBCK*BOXDIMS(3) - CR(3,I)
00089
00090       magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00091
00092       IF (magr2 .LE. coulcut2) THEN
00093
00094         magr = sqrt(magr2)
00095
00096         !

```

```

00097      ! Direction cosines
00098      !
00099
00100      dc = rij/magr
00101
00102      IF (magr .LT. coulrl) THEN
00103
00104          coulombv(i) = coulombv(i) + deltaq(j)/magr
00105
00106          force = -keconst*deltaq(i)*deltaq(j)/magr2
00107
00108          expti = exp( -ti*magr )
00109
00110          IF (elempointer(i) .EQ. elempointer(j)) THEN
00111
00112              coulombv(i) = coulombv(i) - deltaq(j)*expti* &
00113                  (ssb*magr2 + ssc*magr + ssd + sse/magr)
00114
00115              force = force + (keconst*deltaq(i)*deltaq(j)*expti) * &
00116                  ((sse/magr2 - two*ssb*magr - ssc) + &
00117                  ssa*(ssb*magr2 + ssc*magr + ssd + sse/magr))
00118
00119          ELSE
00120
00121              tj = tfact*hubbardu(elempointer(j))
00122
00123              tj2 = tj*tj
00124              tj3 = tj2*tj
00125              tj4 = tj2*tj2
00126              tj6 = tj4*tj2
00127
00128              exptj = exp( -tj*magr )
00129
00130              ti2mtj2 = ti2 - tj2
00131              tj2mti2 = -ti2mtj2
00132
00133              sa = ti
00134              sb = tj4*ti/(two * ti2mtj2 * ti2mtj2)
00135              sc = (tj6 - three*tj4*ti2)/(ti2mtj2 * ti2mtj2 * ti2mtj2)
00136
00137              sd = tj
00138              se = ti4*tj/(two * tj2mti2 * tj2mti2)
00139              sf = (ti6 - three*ti4*tj2)/(tj2mti2*tj2mti2*tj2mti2)
00140
00141              coulombv(i) = coulombv(i) - (deltaq(j) * &
00142                  (expti*(sb - (sc/magr)) + exptj*(se - (sf/magr))))
00143
00144              force = force + keconst*deltaq(i)*deltaq(j) * &
00145                  ((expti * (sa*(sb - (sc/magr)) - (sc/magr2))) + &
00146                  (exptj * (sd*(se - (sf/magr)) - (sf/magr2))))
00147
00148          ENDIF
00149
00150      ELSEIF (magr .GE. coulrl) THEN
00151
00152          myr = magr - coulrl
00153          myr2 = myr*myr
00154          myr3 = myr2*myr
00155          myr4 = myr3*myr
00156          myr5 = myr4*myr
00157
00158          tailr = coulrb(1) + coulrb(2)*myr + coulrb(3)*myr2 + &
00159              coulrb(4)*myr3 + coulrb(5)*myr4 + coulrb(6)*myr5
00160
00161          dtailr = coulrb(2) + two*coulrb(3)*myr + &
00162              three*coulrb(4)*myr2 + four*coulrb(5)*myr3 + &
00163              five*coulrb(6)*myr4
00164
00165          coulombv(i) = coulombv(i) + deltaq(j)*tailr
00166
00167          force = -keconst*deltaq(i)*deltaq(j)*dtailr
00168
00169      ENDIF
00170
00171      fcoul(1,i) = fcoul(1,i) + dc(1)*force
00172      fcoul(2,i) = fcoul(2,i) + dc(2)*force
00173      fcoul(3,i) = fcoul(3,i) + dc(3)*force
00174
00175      virrcoul(1) = virrcoul(1) + rij(1)*dc(1)*force
00176      virrcoul(2) = virrcoul(2) + rij(2)*dc(2)*force
00177      virrcoul(3) = virrcoul(3) + rij(3)*dc(3)*force
00178      virrcoul(4) = virrcoul(4) + rij(1)*dc(2)*force
00179      virrcoul(5) = virrcoul(5) + rij(2)*dc(3)*force
00180      virrcoul(6) = virrcoul(6) + rij(3)*dc(1)*force
00181
00182  ENDIF
00183

```

```
00184      ENDDO
00185
00186      ENDDO
00187
00188      coulombv = keconst*coulombv
00189      vir coul = half * vir coul
00190
00191      RETURN
00192
00193 END SUBROUTINE gaspcoulomb
```

8.59 coulomb_rspace.f90 File Reference

Functions/Subroutines

- subroutine [coulombrspace](#)

8.59.1 Function/Subroutine Documentation

8.59.1.1 subroutine [coulombrspace](#) ()

Definition at line 23 of file [coulomb_rspace.f90](#).

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such

```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE coulombspace
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE neblistarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, NEWJ
00034   INTEGER :: PBCI, PBCJ, PBCK
00035   REAL(LATTEPREC) :: RIJ(3), MAGR, MAGR2, DC(3)
00036   REAL(LATTEPREC) :: CA, TI, TJ, FORCE
00037   REAL(LATTEPREC) :: TI2, TI3, TI4, TI6, TJ2, TJ3, TJ4, TJ6
00038   REAL(LATTEPREC) :: EXPTI, EXPTJ
00039   REAL(LATTEPREC) :: TI2MTJ2, TJ2MTI2
00040   REAL(LATTEPREC) :: SA, SB, SC, SD, SE, SF
00041   REAL(LATTEPREC) :: SSA, SSB, SSC, SSD, SSE
00042   REAL(LATTEPREC), PARAMETER :: C1 = -1.26551223, c2 = 1.00002368
00043   REAL(LATTEPREC), PARAMETER :: C3 = 0.37409196, c4 = 0.09678418
00044   REAL(LATTEPREC), PARAMETER :: C5 = -0.18628806, c6 = 0.27886807
00045   REAL(LATTEPREC), PARAMETER :: C7 = -1.13520398, c8 = 1.48851587
00046   REAL(LATTEPREC), PARAMETER :: C9 = -0.82215223, c10 = 0.17087277
00047   REAL(LATTEPREC) :: Z, T, NUMREP_ERFC
00048   IF (existerror) RETURN
00049   ! REAL(LATTEPREC), EXTERNAL :: NUMREP_ERFC
00050
00051   fcoul = zero
00052   coulombv = zero
00053
00054   vircoul = zero
00055
00056   !$OMP PARALLEL DO DEFAULT(NONE) &
00057   !$OMP SHARED(NATS, BOX, CR, COULCUT2, TOTNEBCOUL, NEBCOUL, HUBBARDU, ELEMPONTER, DELTAQ, COULOMBV) &
00058   !$OMP SHARED(CALPHA, CALPHA2, SQRTPI, FCoul, KECONST, TFACT) &
00059   !$OMP PRIVATE(I, J, NEWJ, PBCI, PBCJ, PBCK, RIJ, MAGR2, MAGR, TI, TI2, TI3, TI4, TI6, SSA) &
00060   !$OMP PRIVATE(SSB, SSC, SSD, SSE, SA, SB, SC, SD, SE, SF) &
00061   !$OMP PRIVATE(TI2MTJ2, TJ2MTI2, EXPTI, EXPTJ, TJ, TJ2, TJ3, TJ4, TJ6, FORCE, CA, DC, Z, T, NUMREP_ERFC) &
00062   !$OMP REDUCTION(+:VIRCOUL)
00063   DO i = 1, nats
00064
00065       ti = tfact*hubbardu(elempointer(i))
00066
00067       ti2 = ti*ti
00068       ti3 = ti2*ti
00069       ti4 = ti2*ti2
00070       ti6 = ti4*ti2
00071
00072       ssa = ti
00073       ssb = ti3/fortyeight
00074       ssc = three*ti2/sixteen
00075       ssd = eleven*ti/sixteen
00076       sse = one
00077
00078       DO newj = 1, totnebcoul(i)
00079
00080           j = nebcoul(1, newj, i)
00081           pbcI = nebcoul(2, newj, i)
00082           pbcj = nebcoul(3, newj, i)
00083           pbck = nebcoul(4, newj, i)
00084
00085           rij(1) = cr(1,j) + REAL(pbcI)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00086             REAL(pbck)*BOX(3,1) - CR(1,i)
00087
00088           rij(2) = cr(2,j) + REAL(pbcI)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00089             REAL(pbck)*BOX(3,2) - CR(2,i)
00090
00091           rij(3) = cr(3,j) + REAL(pbcI)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00092             REAL(pbck)*BOX(3,3) - CR(3,i)
00093
00094           magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00095
00096           IF (magr2 .LE. coulcut2) THEN

```



```

00097
00098      tj = tfact*hubbardu(elempointer(j))
00099
00100      magr = sqrt(magr2)
00101
00102      !
00103      ! Direction cosines
00104      !
00105
00106      dc = rij/magr
00107
00108      ! Using Numerical Recipes ERFc
00109
00110      z = abs(calpha*magr)
00111
00112      t = one/(one + half*z)
00113
00114      numrep_erfc = t * exp(-z*z + c1 + t*(c2 + t*(c3 + t*(c4 + t*(c5 + &
00115      t*(c6 + t*(c7 + t*(c8 + t*(c9 + t*c10))))))))))
00116
00117      IF ( calpha*magr .LT. zero ) numrep_erfc = two - numrep_erfc
00118
00119      ca = numrep_erfc/magr
00120
00121      coulombv(i) = coulombv(i) + deltaq(j)*ca
00122
00123      ca = ca + two*calpha*exp( -calpha2*magr2 )/sqrtpi
00124
00125      force = -keconst*deltaq(i)*deltaq(j)*ca/magr
00126
00127      expti = exp( -ti*magr )
00128
00129      IF (elempointer(i) .EQ. elempointer(j)) THEN
00130
00131          coulombv(i) = coulombv(i) - deltaq(j)*expti * &
00132          (ssb*magr2 + ssc*magr + ssd + sse/magr)
00133
00134          force = force + (keconst*deltaq(i)*deltaq(j)*expti) * &
00135          ((sse/magr2 - two*ssb*magr - ssc) + &
00136          ssa*(ssb*magr2 + ssc*magr + ssd + sse/magr))
00137
00138      ELSE
00139
00140          tj2 = tj*tj
00141          tj3 = tj2*tj
00142          tj4 = tj2*tj2
00143          tj6 = tj4*tj2
00144
00145          exptj = exp( -tj*magr )
00146
00147          ti2mtj2 = ti2 - tj2
00148          tj2mti2 = -ti2mtj2
00149
00150          sa = ti
00151          sb = tj4*ti/(two * ti2mtj2 * ti2mtj2)
00152          sc = (tj6 - three*tj4*ti2)/(ti2mtj2 * ti2mtj2 * ti2mtj2)
00153
00154          sd = tj
00155          se = ti4*tj/(two * tj2mti2 * tj2mti2)
00156          sf = (ti6 - three*ti4*tj2)/(tj2mti2 * tj2mti2 * tj2mti2)
00157
00158          coulombv(i) = coulombv(i) - (deltaq(j) * &
00159          (expti*(sb - (sc/magr)) + exptj*(se - (sf/magr))))
00160
00161          force = force + keconst*deltaq(i)*deltaq(j) * &
00162          ((expti * (sa*(sb - (sc/magr)) - (sc/magr2))) + &
00163          (exptj * (sd*(se - (sf/magr)) - (sf/magr2))))
00164
00165      ENDIF
00166
00167      fcoul(1,i) = fcoul(1,i) + dc(1)*force
00168      fcoul(2,i) = fcoul(2,i) + dc(2)*force
00169      fcoul(3,i) = fcoul(3,i) + dc(3)*force
00170
00171      virrcoul(1) = virrcoul(1) + rij(1)*dc(1)*force
00172      virrcoul(2) = virrcoul(2) + rij(2)*dc(2)*force
00173      virrcoul(3) = virrcoul(3) + rij(3)*dc(3)*force
00174      virrcoul(4) = virrcoul(4) + rij(1)*dc(2)*force
00175      virrcoul(5) = virrcoul(5) + rij(2)*dc(3)*force
00176      virrcoul(6) = virrcoul(6) + rij(3)*dc(1)*force
00177
00178      ENDIF
00179
00180      ENDDO
00181
00182      ENDDO
00183      !$OMP END PARALLEL DO

```

```

00184
00185
00186   coulombv = keconst * coulombv
00187   vir coul = half * vir coul
00188
00189   RETURN
00190
00191 END SUBROUTINE coulombspace
00192
00193
00194 ! Taken from: Numerical Recipes in Fortran 77. The Art of Scientific
00195 ! Computing (ISBN 0-521-43064-X).
00196
00197 !FUNCTION NUMREP_ERFC( X )
00198
00199 !  USE MYPRECISION
00200
00201 !  IMPLICIT NONE
00202
00203 !  REAL(LATTEPREC) :: NUMREP_ERFC, X
00204 !  REAL(LATTEPREC) :: T, Z
00205 !  REAL(LATTEPREC), PARAMETER :: C1 = -1.26551223, C2 = 1.00002368
00206 !  REAL(LATTEPREC), PARAMETER :: C3 = 0.37409196, C4 = 0.09678418
00207 !  REAL(LATTEPREC), PARAMETER :: C5 = -0.18628806, C6 = 0.27886807
00208 !  REAL(LATTEPREC), PARAMETER :: C7 = -1.13520398, C8 = 1.48851587
00209 !  REAL(LATTEPREC), PARAMETER :: C9 = -0.82215223, C10 = 0.17087277
00210
00211 !  Z = ABS(X)
00212
00213 !  T = ONE/(ONE + HALF*Z)
00214
00215 !  NUMREP_ERFC = T * EXP(-Z*Z + C1 + T*(C2 + T*(C3 + T*(C4 + T*(C5 + &
00216 !      T*(C6 + T*(C7 + T*(C8 + T*(C9 + T*(C10))))))))
00217
00218 !  IF ( X .LT. ZERO ) NUMREP_ERFC = TWO - NUMREP_ERFC
00219
00220 !  RETURN
00221
00222 !END FUNCTION NUMREP_ERFC
00223
00224

```

8.61 coulombarray.f90 File Reference

Modules

- module [coulombarray](#)

Variables

- real(latteprec) [coulombarray::calpha](#)
- real(latteprec) [coulombarray::calpha2](#)
- real(latteprec), dimension(:), allocatable [coulombarray::coslist](#)
- real(latteprec) [coulombarray::coulacc](#)
- real(latteprec), dimension(6) [coulombarray::coulb](#)
- real(latteprec) [coulombarray::coulcut](#)
- real(latteprec) [coulombarray::coulcut2](#)
- real(latteprec) [coulombarray::coulr1](#)
- real(latteprec) [coulombarray::coulvol](#)
- real(latteprec) [coulombarray::eightpi](#)
- real(latteprec) [coulombarray::fourcalpha2](#)
- real(latteprec), dimension(:), allocatable [coulombarray::k1_list](#)
- real(latteprec), dimension(:), allocatable [coulombarray::k2_list](#)
- real(latteprec), dimension(:), allocatable [coulombarray::k3_list](#)
- real(latteprec) [coulombarray::kcutoff](#)
- real(latteprec) [coulombarray::kcutoff2](#)

- real(latteprec) `coulombarray::keconst`
- real(latteprec), dimension(:), allocatable `coulombarray::ksq_list`
- real(latteprec), dimension(3, 3) `coulombarray::latticevecs`
- integer `coulombarray::lmax`
- integer `coulombarray::mmax`
- integer `coulombarray::nk`
- integer `coulombarray::nmax`
- real(latteprec), dimension(:), allocatable `coulombarray::olddeltaqs`
- real(latteprec) `coulombarray::pi2`
- real(latteprec), dimension(3, 3) `coulombarray::recipvecs`
- real(latteprec) `coulombarray::relperm`
- real(latteprec), dimension(:), allocatable `coulombarray::sinlist`
- real(latteprec) `coulombarray::sqrtpi`
- real(latteprec) `coulombarray::tfact`
- real(latteprec) `coulombarray::twopi`

8.62 coulombarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE coulombarray
00023
00024   USE constants_mod
00025   USE myprecision
00026
00027   IMPLICIT NONE
00028   SAVE
00029
00030   INTEGER :: lmax, mmax, nmax, nk
00031   REAL(LATTEPREC) :: coulcut ! Input - cutoff for the real space part
00032   REAL(LATTEPREC) :: coulacc ! Input - accuracy for the k-space part
00033   REAL(LATTEPREC) :: coulrl ! Start of the cut off tail for real space
00034   REAL(LATTEPREC) :: latticevecs(3,3), recipvecs(3,3)
00035   REAL(LATTEPREC), ALLOCATABLE :: k1_list(:), k2_list(:), k3_list(:) !List of
reciprocal points
00036   REAL(LATTEPREC), ALLOCATABLE :: ksq_list(:) !List of the square of the reciprocal points
00037   REAL(LATTEPREC) :: coulcut2, kcutoff, kcutoff2
00038   REAL(LATTEPREC) :: calpha, calpha2, coulvol, fourcalpha2
00039   REAL(LATTEPREC) :: coul(6)
00040   REAL(LATTEPREC) :: twopi, pi2, sqrtpi, eightpi
00041   REAL(LATTEPREC) :: relperm ! Dielectric constant
00042   REAL(LATTEPREC) :: keconst
00043   REAL(LATTEPREC) :: tfact
00044   REAL(LATTEPREC), ALLOCATABLE :: olddeltaqs(:)
00045   REAL(LATTEPREC), ALLOCATABLE :: sinlist(:), coslist(:)
00046
00047
00048 END MODULE coulombarray

```

8.63 coultailcoef.f90 File Reference

Functions/Subroutines

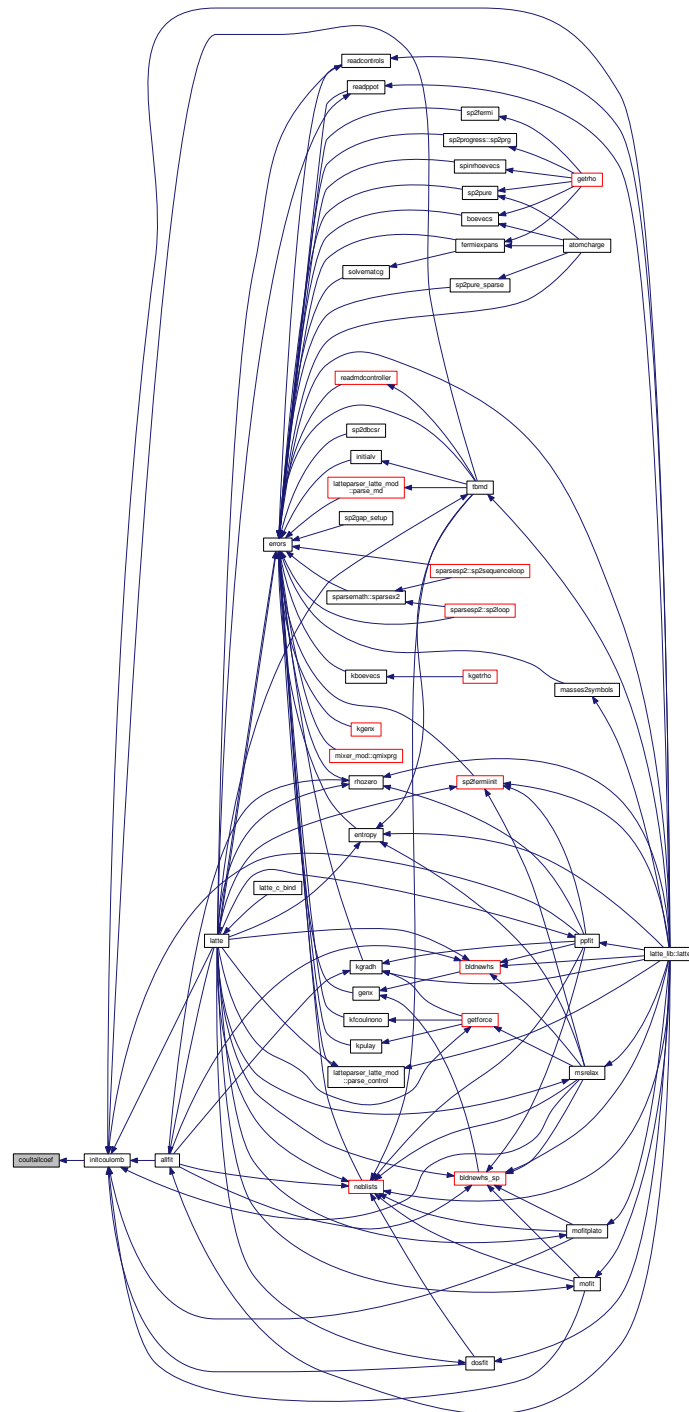
- subroutine [coultailcoef](#)

8.63.1 Function/Subroutine Documentation

8.63.1.1 subroutine [coultailcoef](#) ()

Definition at line [23](#) of file [coultailcoef.f90](#).

Here is the caller graph for this function:



8.64 coultailcoef.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE coultailcoef
00023
00024   USE constants_mod
00025   USE coulombarray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   REAL(LATTEPREC) :: DELTA, DELTA2, DELTA3, DELTA4
00031   REAL(LATTEPREC) :: R1, R12, R13
00032   IF (existerror) RETURN
00033
00034   !
00035   ! The cut-off function looks like this:
00036   !
00037   !  $t(R) = B1 + B2*(R - R1) + B3*(R - R1)^2 + B4*(R - R1)^3 +$ 
00038   !  $B5*(R - R1)^4 + B6*(R - R1)^5$ 
00039   !
00040
00041   r1 = coulrl
00042   r12 = r1*coulrl
00043   r13 = r12*coulrl
00044
00045   coul(1) = one/r1
00046
00047   coul(2) = minusone/r12
00048
00049   coul(3) = one/r13
00050
00051   delta = coulcut - coulrl
00052   delta2 = delta*delta
00053   delta3 = delta2*delta
00054   delta4 = delta3*delta
00055
00056   coul(4) = (minusone/delta3)*(three*coul(3)*delta2 + &
00057             six*coul(2)*delta + ten*coul(1))
00058
00059   coul(5) = (one/delta4)*(three*coul(3)*delta2 + &
00060             eight*coul(2)*delta + fifteen*coul(1))
00061
00062   coul(6) = (minusone/(ten*delta3))*(six*coul(5)*delta2 + &
00063             three*coul(4)*delta + coul(3))
00064
00065   RETURN
00066
00067 END SUBROUTINE coultailcoef
00068

```

8.65 dbcsr_var_mod.f90 File Reference

Modules

- module [dbcsr_var_mod](#)

Functions/Subroutines

- subroutine [dbcsr_var_mod::myset_dist](#) (dist_array, dist_size, nbins)

Variables

- integer, dimension(:), pointer `dbcsr_var_mod::cbs`
- real(latteprec) `dbcsr_var_mod::chksum`
- real(latteprec) `dbcsr_var_mod::chksum2`
- type(array_i1d_obj) `dbcsr_var_mod::col_blk_sizes`
- type(array_i1d_obj) `dbcsr_var_mod::col_dist_a`
- real(latteprec), dimension(:), allocatable `dbcsr_var_mod::diag`
- type(dbcsr_distribution_obj) `dbcsr_var_mod::dist_a`
- type(dbcsr_distribution_obj) `dbcsr_var_mod::dist_b`
- type(dbcsr_distribution_obj) `dbcsr_var_mod::dist_c`
- type(dbcsr_error_type) `dbcsr_var_mod::error`
- logical `dbcsr_var_mod::found`
- integer, dimension(:), allocatable `dbcsr_var_mod::grid_dist`
- integer `dbcsr_var_mod::group`
- integer `dbcsr_var_mod::ierr`
- type(dbcsr_iterator) `dbcsr_var_mod::iter`
- type(dbcsr_obj) `dbcsr_var_mod::matrix_a`
- type(dbcsr_obj) `dbcsr_var_mod::matrix_b`
- integer `dbcsr_var_mod::mp_comm`
- type(dbcsr_mp_obj) `dbcsr_var_mod::mp_env`
- integer `dbcsr_var_mod::mp_group`
- real(latteprec), dimension(2:2) `dbcsr_var_mod::my_block`
- integer `dbcsr_var_mod::mynode`
- integer, dimension(2) `dbcsr_var_mod::myploc`
- integer `dbcsr_var_mod::n`
- integer `dbcsr_var_mod::nblkcols_total`
- integer `dbcsr_var_mod::nblkrows_total`
- integer, dimension(2) `dbcsr_var_mod::npdims`
- integer `dbcsr_var_mod::numnodes`
- integer `dbcsr_var_mod::pcol`
- integer, dimension(:, :), allocatable `dbcsr_var_mod::pgrid`
- integer `dbcsr_var_mod::proc_holds_blk`
- integer `dbcsr_var_mod::prow`
- integer, dimension(:), pointer `dbcsr_var_mod::rbs`
- type(array_i1d_obj) `dbcsr_var_mod::row_blk_sizes`
- type(array_i1d_obj) `dbcsr_var_mod::row_dist_a`
- integer `dbcsr_var_mod::temp`
- logical `dbcsr_var_mod::tr`

8.66 dbcsr_var_mod.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !

```

```

00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022
00023
00024 MODULE dbcsr_var_mod
00025
00026 USE dbcsr_types
00027 USE dbcsr_methods
00028 USE dbcsr_error_handling
00029 USE array_types, ONLY: array_data,&
00030     array_ild_obj,&
00031     array_new,&
00032     array_nullify,&
00033     array_release,&
00034     array_size
00035 USE dbcsr_io
00036 USE dbcsr_operations
00037 USE dbcsr_ptr_util
00038 USE dbcsr_transformations
00039 USE dbcsr_util
00040 USE dbcsr_work_operations
00041 USE dbcsr_message_passing
00042
00043 USE dbcsr_block_access
00044 USE dbcsr_iterator_operations, ONLY: dbcsr_iterator_blocks_left,&
00045     dbcsr_iterator_next_block,&
00046     dbcsr_iterator_start,&
00047     dbcsr_iterator_stop
00048
00049 USE dbcsr_dist_operations, ONLY: create_bl_distribution,&
00050     dbcsr_get_stored_coordinates
00051 IMPLICIT NONE
00052 SAVE
00053
00054 !*****
00055 !sets up dbcsr/mpi variables
00056
00057 TYPE(dbcsr_obj) :: matrix_a, matrix_b
00058 !labels as standard error, necessary dbcsr declaration
00059 TYPE(dbcsr_error_type) :: error
00060
00061 !setting up variables for matrix and mpi
00062 INTEGER, DIMENSION(:), POINTER :: rbs, cbs
00063 TYPE(array_ild_obj) :: row_blk_sizes,
col_blk_sizes, row_dist_a, col_dist_a
00064 INTEGER :: npdims(2), ierr
00065 INTEGER, ALLOCATABLE, DIMENSION(:,:) :: pgrid
00066 INTEGER prow, pcol, mp_comm, mp_group, nblkcols_total,
nblkrows_total, proc_holds_blk
00067 TYPE(dbcsr_mp_obj) :: mp_env
00068 INTEGER :: group, mynode, numnodes,
myproc(2), n
00069 TYPE(dbcsr_distribution_obj) :: dist_a, dist_b, dist_c
00070 REAL(LATTEPREC), DIMENSION(:), ALLOCATABLE :: diag
00071 REAL(LATTEPREC) :: my_block(2:2)
00072 LOGICAL :: tr, found
00073 INTEGER, ALLOCATABLE, DIMENSION(:) :: grid_dist
00074 TYPE(dbcsr_iterator) :: iter
00075 REAL(LATTEPREC) :: chksum, chksum2
00076 INTEGER :: temp
00077
00078 !*****
00079 !sets distribution to processors
00080
00081 CONTAINS
00082
00083 SUBROUTINE myset_dist (dist_array, dist_size, nbins)
00084 TYPE(array_ild_obj), INTENT(OUT) :: dist_array
00085 INTEGER, INTENT(IN) :: dist_size, nbins
00086
00087 INTEGER :: i
00088 INTEGER, ALLOCATABLE, DIMENSION(:) :: grid_dist
00089
00090 ALLOCATE (grid_dist(dist_size))
00091 CALL array_nullify (dist_array)
00092
00093 FORALL (i = 1 : dist_size)
00094     grid_dist(i) = modulo(nbins-i, nbins)
00095 END FORALL
00096
00097 CALL array_new (dist_array, grid_dist, lb=1)
00098 DEALLOCATE (grid_dist)
00099
00100 END SUBROUTINE myset_dist
00101

```



```
00102
00103
00104  !*****
00105
00106  END MODULE dbcsr_var_mod
00107
```

8.67 deallocatell.f90 File Reference

Functions/Subroutines

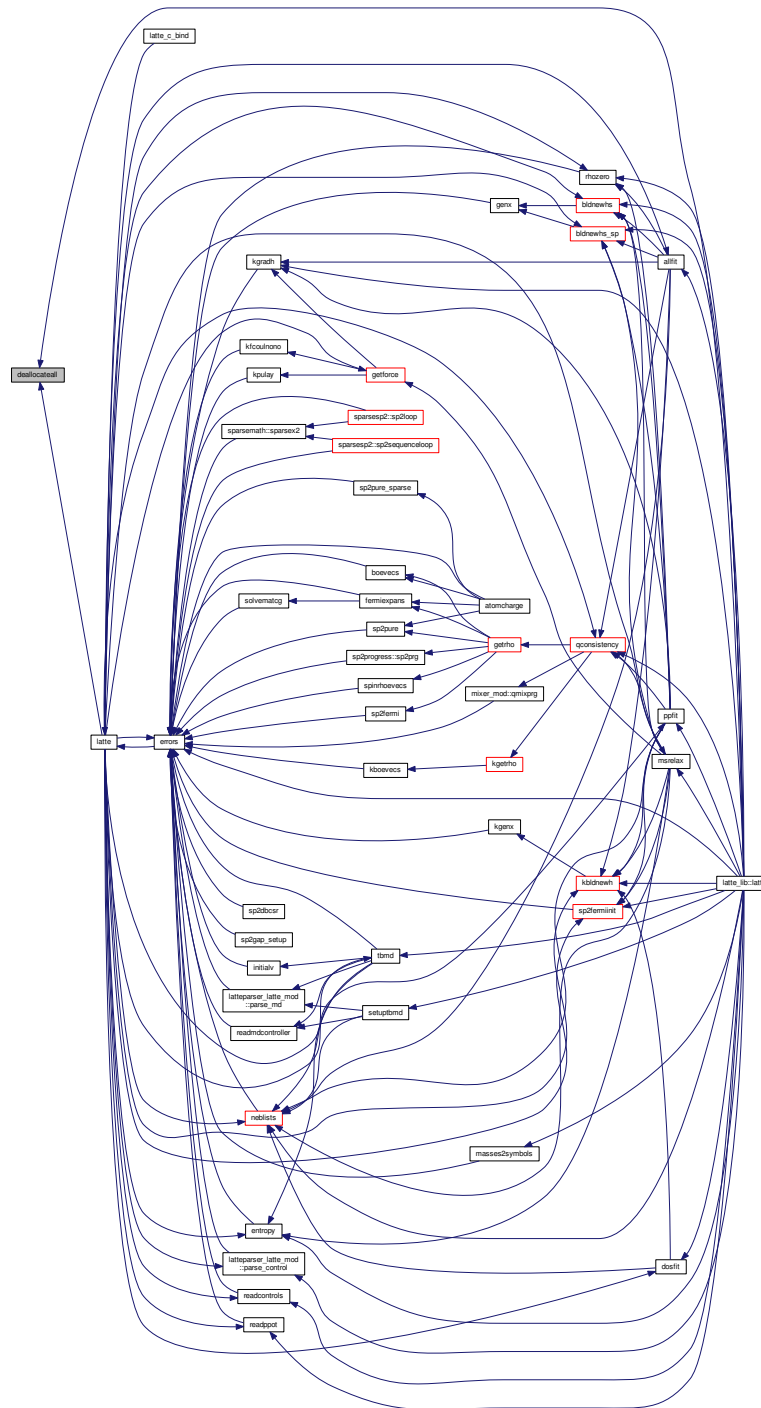
- subroutine [deallocatell](#)

8.67.1 Function/Subroutine Documentation

8.67.1.1 subroutine deallocatell ()

Definition at line 23 of file [deallocatell.f90](#).

Here is the caller graph for this function:



8.68 deallocateall.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS            !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such        !
00010 ! modified software should be clearly marked, so as not to confuse it         !
00011 ! with the version available from LANL.                                       !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it        !
00014 ! and/or modify it under the terms of the GNU General Public License as       !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!! !
00021
00022 SUBROUTINE deallocateall
00023
00024   USE setuparray
00025   USE univarray
00026   USE ppotarray
00027   USE neblastarray
00028   USE mdarray
00029   USE kspacearray
00030   USE constants_mod
00031   USE spinarray
00032   USE restartarray
00033   USE sparsearray
00034   USE xboarray
00035   USE coulombarray
00036   USE diagarray
00037   USE mdarray
00038   USE nonoarray
00039   USE purearray
00040   USE virialarray
00041
00042
00043   IMPLICIT NONE
00044   IF (existerror) RETURN
00045
00046   IF (ALLOCATED( atele )) DEALLOCATE( atele )
00047   IF (ALLOCATED( basis )) DEALLOCATE( basis )
00048   IF (ALLOCATED( btype )) DEALLOCATE( btype )
00049   IF (ALLOCATED( ele )) DEALLOCATE( ele )
00050   IF (ALLOCATED( ele1 )) DEALLOCATE( ele1 )
00051   IF (ALLOCATED( ele2 )) DEALLOCATE( ele2 )
00052   IF (ALLOCATED( ppele1 )) DEALLOCATE( ppele1 )
00053   IF (ALLOCATED( ppele2 )) DEALLOCATE( ppele2 )
00054   IF (ALLOCATED( diag_zwork )) DEALLOCATE( diag_zwork )
00055   IF (ALLOCATED( hk )) DEALLOCATE( hk )
00056   IF (ALLOCATED( hk0 )) DEALLOCATE( hk0 )
00057   IF (ALLOCATED( hkdiag )) DEALLOCATE( hkdiag )
00058   IF (ALLOCATED( kbo )) DEALLOCATE( kbo )
00059   IF (ALLOCATED( kevecs )) DEALLOCATE( kevecs )
00060   IF (ALLOCATED( kf )) DEALLOCATE( kf )
00061   IF (ALLOCATED( khtmp )) DEALLOCATE( khtmp )
00062   IF (ALLOCATED( korthoh )) DEALLOCATE( korthoh )
00063   IF (ALLOCATED( kxmat )) DEALLOCATE( kxmat )
00064   IF (ALLOCATED( sk )) DEALLOCATE( sk )
00065   IF (ALLOCATED( zbo )) DEALLOCATE( zbo )
00066   IF (ALLOCATED( zheevd_work )) DEALLOCATE( zheevd_work )
00067   IF (ALLOCATED( zhjj )) DEALLOCATE( zhjj )
00068   IF (ALLOCATED( diag_iwork )) DEALLOCATE( diag_iwork )
00069   IF (ALLOCATED( elempointer )) DEALLOCATE( elempointer )
00070   IF (ALLOCATED( ifail )) DEALLOCATE( ifail )
00071   IF (ALLOCATED( matindlist )) DEALLOCATE( matindlist )
00072   IF (ALLOCATED( molid )) DEALLOCATE( molid )
00073   IF (ALLOCATED( nebcoul )) DEALLOCATE( nebcoul )
00074   IF (ALLOCATED( nebpp )) DEALLOCATE( nebpp )
00075   IF (ALLOCATED( nebtb )) DEALLOCATE( nebtb )
00076   IF (ALLOCATED( nono_iwork )) DEALLOCATE( nono_iwork )
00077   IF (ALLOCATED( pptablenth )) DEALLOCATE( pptablenth )
00078   IF (ALLOCATED( rx )) DEALLOCATE( rx )
00079   IF (ALLOCATED( rxtmp )) DEALLOCATE( rxtmp )
00080   IF (ALLOCATED( signlist )) DEALLOCATE( signlist )
00081   IF (ALLOCATED( spinindlist )) DEALLOCATE( spinindlist )
00082   IF (ALLOCATED( totnebcoul )) DEALLOCATE( totnebcoul )
00083   IF (ALLOCATED( totnebpp )) DEALLOCATE( totnebpp )
00084   IF (ALLOCATED( totnebtb )) DEALLOCATE( totnebtb )
00085   IF (ALLOCATED( xb )) DEALLOCATE( xb )
00086   IF (ALLOCATED( zheevd_iwork )) DEALLOCATE( zheevd_iwork )
00087   IF (ALLOCATED( atocc )) DEALLOCATE( atocc )
00088   IF (ALLOCATED( bo )) DEALLOCATE( bo )
00089   IF (ALLOCATED( bond )) DEALLOCATE( bond )
00090   IF (ALLOCATED( bozero )) DEALLOCATE( bozero )
00091   IF (ALLOCATED( bo_padded )) DEALLOCATE( bo_padded )
00092   IF (ALLOCATED( chempot_pnk )) DEALLOCATE( chempot_pnk )
00093   IF (ALLOCATED( coslist )) DEALLOCATE( coslist )

```

```

00094 IF ( ALLOCATED( coulombv )) DEALLOCATE( coulombv )
00095 IF ( ALLOCATED( cplist )) DEALLOCATE( cplist )
00096 IF ( ALLOCATED( cr )) DEALLOCATE( cr )
00097 IF ( ALLOCATED( deltaq )) DEALLOCATE( deltaq )
00098 IF ( ALLOCATED( deltaspin )) DEALLOCATE( deltaspin )
00099 IF ( ALLOCATED( diag_rwork )) DEALLOCATE( diag_rwork )
00100 IF ( ALLOCATED( diag_work )) DEALLOCATE( diag_work )
00101 IF ( ALLOCATED( downevals )) DEALLOCATE( downevals )
00102 IF ( ALLOCATED( downvecs )) DEALLOCATE( downvecs )
00103 IF ( ALLOCATED( ehist )) DEALLOCATE( ehist )
00104 IF ( ALLOCATED( evals )) DEALLOCATE( evals )
00105 IF ( ALLOCATED( evecs )) DEALLOCATE( evecs )
00106 IF ( ALLOCATED( f )) DEALLOCATE( f )
00107 IF ( ALLOCATED( fcoul )) DEALLOCATE( fcoul )
00108 IF ( ALLOCATED( fpp )) DEALLOCATE( fpp )
00109 IF ( ALLOCATED( fpul )) DEALLOCATE( fpul )
00110 IF ( ALLOCATED( franprev )) DEALLOCATE( franprev )
00111 IF ( ALLOCATED( fscoul )) DEALLOCATE( fscoul )
00112 IF ( ALLOCATED( fsspin )) DEALLOCATE( fsspin )
00113 IF ( ALLOCATED( ftot )) DEALLOCATE( ftot )
00114 IF ( ALLOCATED( h )) DEALLOCATE( h )
00115 IF ( ALLOCATED( h0 )) DEALLOCATE( h0 )
00116 IF ( ALLOCATED( h2vect )) DEALLOCATE( h2vect )
00117 IF ( ALLOCATED( hdiag )) DEALLOCATE( hdiag )
00118 IF ( ALLOCATED( hdown )) DEALLOCATE( hdown )
00119 IF ( ALLOCATED( hed )) DEALLOCATE( hed )
00120 IF ( ALLOCATED( hef )) DEALLOCATE( hef )
00121 IF ( ALLOCATED( hep )) DEALLOCATE( hep )
00122 IF ( ALLOCATED( hes )) DEALLOCATE( hes )
00123 IF ( ALLOCATED( hjj )) DEALLOCATE( hjj )
00124 IF ( ALLOCATED( hr0 )) DEALLOCATE( hr0 )
00125 IF ( ALLOCATED( hubbardu )) DEALLOCATE( hubbardu )
00126 IF ( ALLOCATED( hup )) DEALLOCATE( hup )
00127 IF ( ALLOCATED( kevals )) DEALLOCATE( kevals )
00128 IF ( ALLOCATED( lcnshift )) DEALLOCATE( lcnshift )
00129 IF ( ALLOCATED( mass )) DEALLOCATE( mass )
00130 IF ( ALLOCATED( mycharge )) DEALLOCATE( mycharge )
00131 IF ( ALLOCATED( nonotmp )) DEALLOCATE( nonotmp )
00132 IF ( ALLOCATED( nono_evals )) DEALLOCATE( nono_evals )
00133 IF ( ALLOCATED( nono_work )) DEALLOCATE( nono_work )
00134 IF ( ALLOCATED( olddeltaqs )) DEALLOCATE( olddeltaqs )
00135 IF ( ALLOCATED( olddeltaspin )) DEALLOCATE( olddeltaspin )
00136 IF ( ALLOCATED( orthoh )) DEALLOCATE( orthoh )
00137 IF ( ALLOCATED( orthohdown )) DEALLOCATE( orthohdown )
00138 IF ( ALLOCATED( orthohup )) DEALLOCATE( orthohup )
00139 IF ( ALLOCATED( orthorho )) DEALLOCATE( orthorho )
00140 IF ( ALLOCATED( overl )) DEALLOCATE( overl )
00141 IF ( ALLOCATED( pair )) DEALLOCATE( pair )
00142 IF ( ALLOCATED( phist )) DEALLOCATE( phist )
00143 IF ( ALLOCATED( phistx )) DEALLOCATE( phistx )
00144 IF ( ALLOCATED( phisty )) DEALLOCATE( phisty )
00145 IF ( ALLOCATED( phistz )) DEALLOCATE( phistz )
00146 IF ( ALLOCATED( pnk )) DEALLOCATE( pnk )
00147 IF ( ALLOCATED( potcoef )) DEALLOCATE( potcoef )
00148 IF ( ALLOCATED( ppr )) DEALLOCATE( ppr )
00149 IF ( ALLOCATED( ppspl )) DEALLOCATE( ppspl )
00150 IF ( ALLOCATED( ppval )) DEALLOCATE( ppval )
00151 IF ( ALLOCATED( qlist )) DEALLOCATE( qlist )
00152 IF ( ALLOCATED( respchi )) DEALLOCATE( respchi )
00153 IF ( ALLOCATED( rhodown )) DEALLOCATE( rhodown )
00154 IF ( ALLOCATED( rhodownzero )) DEALLOCATE( rhodownzero )
00155 IF ( ALLOCATED( rhoup )) DEALLOCATE( rhoup )
00156 IF ( ALLOCATED( rhoupzero )) DEALLOCATE( rhoupzero )
00157 IF ( ALLOCATED( sh2 )) DEALLOCATE( sh2 )
00158 IF ( ALLOCATED( sinlist )) DEALLOCATE( sinlist )
00159 IF ( ALLOCATED( smat )) DEALLOCATE( smat )
00160 IF ( ALLOCATED( spinlist )) DEALLOCATE( spinlist )
00161 IF ( ALLOCATED( spintmp )) DEALLOCATE( spintmp )
00162 IF ( ALLOCATED( spin_pnk )) DEALLOCATE( spin_pnk )
00163 IF ( ALLOCATED( thist )) DEALLOCATE( thist )
00164 IF ( ALLOCATED( tmpbodiag )) DEALLOCATE( tmpbodiag )
00165 IF ( ALLOCATED( tmprhodown )) DEALLOCATE( tmprhodown )
00166 IF ( ALLOCATED( tmprhoup )) DEALLOCATE( tmprhoup )
00167 IF ( ALLOCATED( twoxx2 )) DEALLOCATE( twoxx2 )
00168 IF ( ALLOCATED( umat )) DEALLOCATE( umat )
00169 IF ( ALLOCATED( upevals )) DEALLOCATE( upevals )
00170 IF ( ALLOCATED( upevecs )) DEALLOCATE( upevecs )
00171 IF ( ALLOCATED( v )) DEALLOCATE( v )
00172 IF ( ALLOCATED( vhist )) DEALLOCATE( vhist )
00173 IF ( ALLOCATED( wdd )) DEALLOCATE( wdd )
00174 IF ( ALLOCATED( wff )) DEALLOCATE( wff )
00175 IF ( ALLOCATED( work )) DEALLOCATE( work )
00176 IF ( ALLOCATED( wpp )) DEALLOCATE( wpp )
00177 IF ( ALLOCATED( wss )) DEALLOCATE( wss )
00178 IF ( ALLOCATED( x2 )) DEALLOCATE( x2 )
00179 IF ( ALLOCATED( x2down )) DEALLOCATE( x2down )
00180 IF ( ALLOCATED( x2hrho )) DEALLOCATE( x2hrho )

```

```
00181  IF ( ALLOCATED ( x2up ) ) DEALLOCATE ( x2up )
00182  IF ( ALLOCATED ( xmat ) ) DEALLOCATE ( xmat )
00183  IF ( ALLOCATED ( zheevd_rwork ) ) DEALLOCATE ( zheevd_rwork )
00184
00185  RETURN
00186
00187 END SUBROUTINE deallocateall
```

8.69 deallocatecoulomb.f90 File Reference

Functions/Subroutines

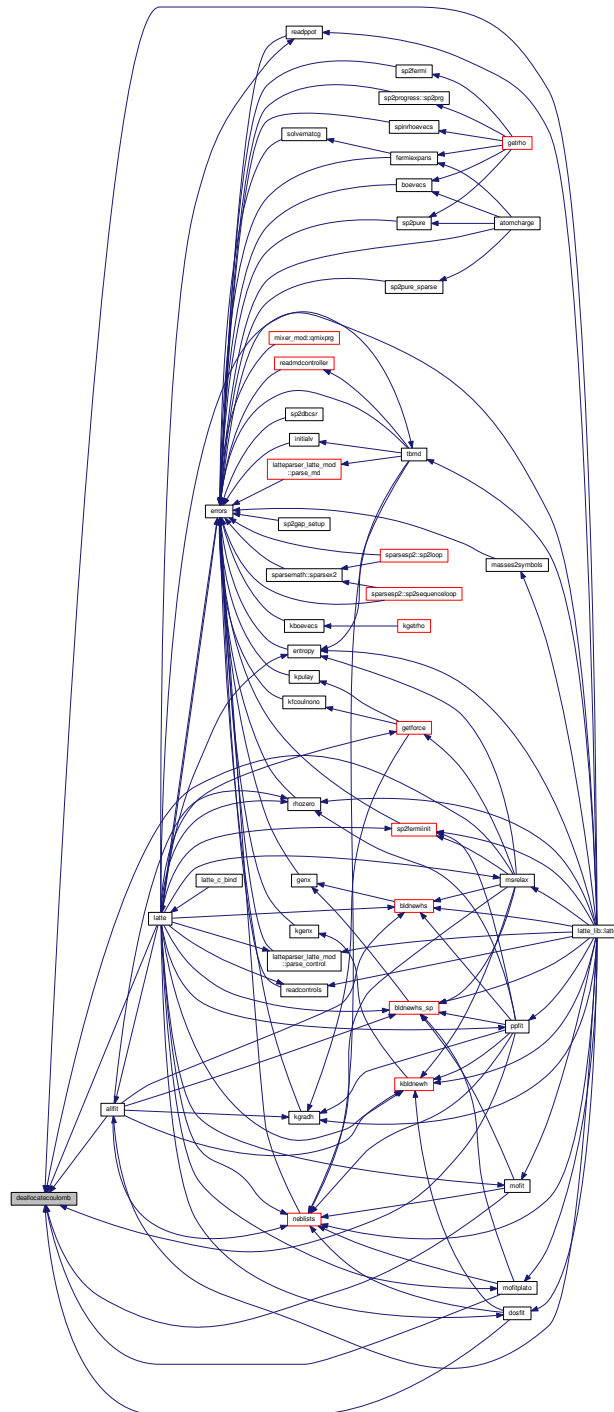
- subroutine [deallocatecoulomb](#)

8.69.1 Function/Subroutine Documentation

8.69.1.1 subroutine deallocatecoulomb ()

Definition at line 23 of file [deallocatecoulomb.f90](#).

Here is the caller graph for this function:



8.70 deallocatecoulomb.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatecoulomb
00023
00024   USE setuparray
00025   USE coulombarray
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   DEALLOCATE(olddeltag, coulombv, fcoul)
00031   DEALLOCATE(sinlist, coslist)
00032
00033 END SUBROUTINE deallocatecoulomb

```

8.71 deallocatediag.f90 File Reference

Functions/Subroutines

- subroutine [deallocatediag](#)

8.71.1 Function/Subroutine Documentation

8.71.1.1 subroutine deallocatediag ()

Definition at line 23 of file [deallocatediag.f90](#).


```

00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS      !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such    !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatEDIAG
00023
00024     USE constants_mod
00025     USE diagarray
00026
00027     IMPLICIT NONE
00028     IF (existerror) RETURN
00029
00030     #ifdef XSYEV
00031         DEALLOCATE(diag_work)
00032     #elif defined(XSYEVD)
00033         DEALLOCATE(diag_work, diag_iwork)
00034     #endif
00035
00036     IF (spinon .EQ. 0) THEN
00037         DEALLOCATE (evals, evecs)
00038     ELSE
00039         DEALLOCATE (upevals, upevecs, downevals, downevecs)
00040     ENDIF
00041
00042     RETURN
00043
00044 END SUBROUTINE deallocatEDIAG

```

8.73 deallocatenebararrays.f90 File Reference

Functions/Subroutines

- subroutine [deallocatenebararrays](#)

8.73.1 Function/Subroutine Documentation

8.73.1.1 subroutine deallocatenebararrays ()

Definition at line 23 of file [deallocatenebararrays.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatenearrays
00023
00024   USE constants_mod
00025   USE neblistarray
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   DEALLOCATE(totnebtb)
00031   IF (ppoton .EQ. 1) DEALLOCATE(totnebpp)
00032   IF (electro .EQ. 1) DEALLOCATE(totnebcoul)
00033
00034   RETURN
00035
00036 END SUBROUTINE deallocatenearrays

```

8.75 deallocatenono.f90 File Reference

Functions/Subroutines

- subroutine [deallocatenono](#)

8.75.1 Function/Subroutine Documentation

8.75.1.1 subroutine deallocatenono ()

Definition at line 23 of file [deallocatenono.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatenono
00023
00024   USE constants_mod
00025   USE nonoarray
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   IF (kon .EQ. 0) THEN
00031
00032   #ifdef XSYEV
00033       DEALLOCATE(nono_work)
00034   #elif defined(XSYEVD)
00035       DEALLOCATE(nono_work, nono_iwork)
00036   #endif
00037
00038       DEALLOCATE(nono_evals, xmat, smat, nonotmp, umat)
00039       DEALLOCATE(x2hrho, hjj)
00040
00041       IF (spinon .EQ. 0) THEN
00042           DEALLOCATE(orthoh)
00043       ELSE
00044           DEALLOCATE(orthohup, orthohdown)
00045           DEALLOCATE(spintmp)
00046       ENDIF
00047
00048   ENDIF
00049
00050   RETURN
00051
00052 END SUBROUTINE deallocatenono

```

8.77 deallocatepure.f90 File Reference

Functions/Subroutines

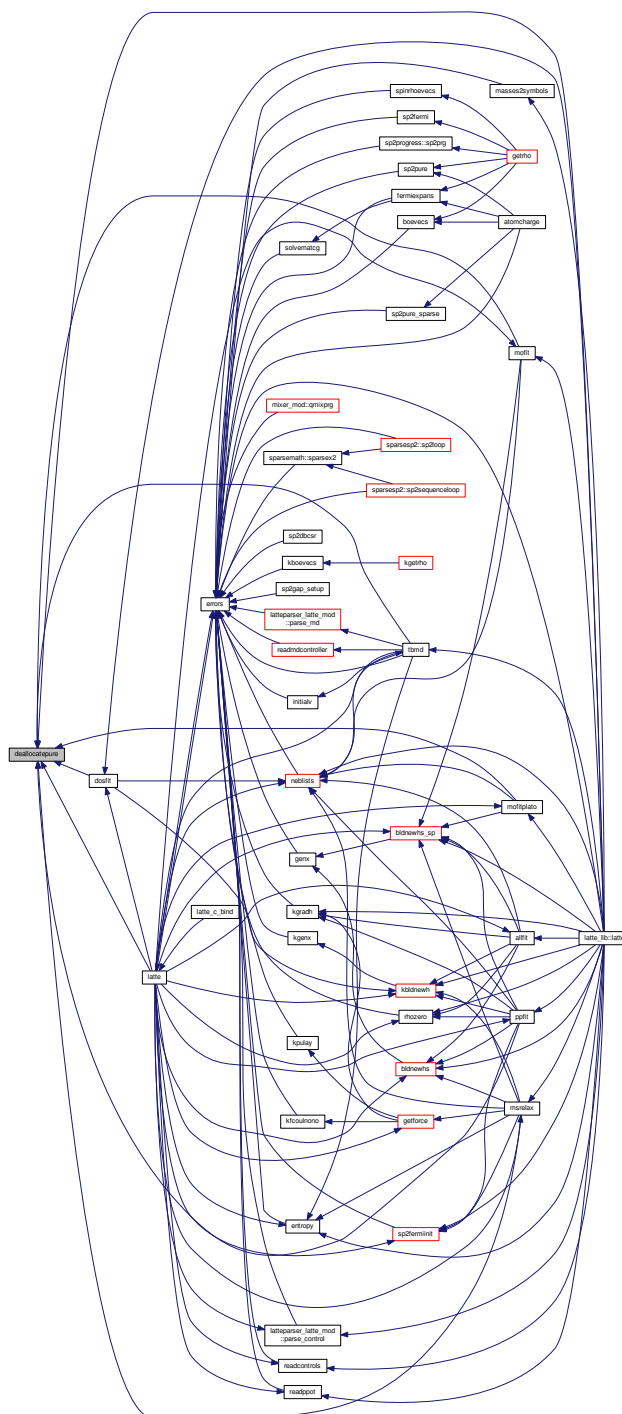
- subroutine [deallocatepure](#)

8.77.1 Function/Subroutine Documentation

8.77.1.1 subroutine deallocatepure ()

Definition at line 23 of file [deallocatepure.f90](#).

Here is the caller graph for this function:



8.78 deallocatepure.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatpure
00023
00024 #ifdef DBCSR_ON
00025
00026     USE dbcsr_var_mod
00027
00028 #endif
00029
00030     USE constants_mod
00031     USE purearray
00032     USE sparsearray
00033
00034     IMPLICIT NONE
00035     IF (existerror) RETURN
00036
00037     IF (sparseon .EQ. 0) THEN
00038
00039         IF (spinon .EQ. 0) THEN
00040             DEALLOCATE(x2)
00041         ELSE
00042             DEALLOCATE(x2up, x2down)
00043         ENDIF
00044
00045     ELSE
00046
00047 #ifdef DBCSR_ON
00048
00049         DEALLOCATE(bo_padded)
00050
00051 #elif defined(DBCSR_OFF)
00052
00053         DEALLOCATE(rx, rxtmp, work, xb)
00054
00055 #endif
00056
00057     ENDIF
00058
00059     IF ( control .EQ. 5 ) THEN
00060         DEALLOCATE( signlist )
00061     ENDIF
00062
00063     RETURN
00064
00065 END SUBROUTINE deallocatpure

```

8.79 deallocatesubgraph.f90 File Reference

Functions/Subroutines

- subroutine [deallocatesubgraph](#)

8.79.1 Function/Subroutine Documentation

8.79.1.1 subroutine deallocatesubgraph ()

Definition at line 23 of file [deallocatesubgraph.f90](#).

8.80 deallocatesubgraph.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatesubgraph
00023
00024   USE subgraph
00025   USE constants_mod, ONLY: existerror
00026
00027   IMPLICIT NONE
00028
00029   IF (existerror) RETURN
00030
00031   DEALLOCATE(g)
00032   !DEALLOCATE(G,G0,G1,G2,G3,G4,G5,G6)
00033
00034   RETURN
00035
00036 END SUBROUTINE deallocatesubgraph

```

8.81 deallocatexbo.f90 File Reference

Functions/Subroutines

- subroutine [deallocatexbo](#)

8.81.1 Function/Subroutine Documentation

8.81.1.1 subroutine deallocatexbo ()

Definition at line 23 of file [deallocatexbo.f90](#).

[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was made available
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
00004 ! National Laboratory (LANL), which is operated by Los Alamos National Security, LLC
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has the right
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.     !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of     !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deallocatexbo
00023
00024   USE constants_mod
00025   USE xboarray
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   DEALLOCATE(pnk)
00031
00032   IF (control .EQ. 1 .OR. control .EQ. 3) THEN
00033     DEALLOCATE(chempot_pnk)
00034   ENDIF
00035
00036   IF (spinon .EQ. 1) THEN
00037     DEALLOCATE(spin_pnk)
00038   ENDIF
00039
00040   RETURN
00041
00042 END SUBROUTINE deallocatexbo

```

8.83 deorthomyrho.f90 File Reference

Functions/Subroutines

- subroutine [deorthomyrho](#)

8.83.1 Function/Subroutine Documentation

8.83.1.1 subroutine deorthomyrho ()

Definition at line 23 of file [deorthomyrho.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deorthomyrho
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE nonoarray
00027     USE spinarray
00028     USE myprecision
00029
00030     IMPLICIT NONE
00031     IF (existerror) RETURN
00032
00033     !
00034     ! RHO = X ORTHORHO X^dag
00035     !
00036
00037     IF (spinon .EQ. 0) THEN
00038
00039 #ifdef DOUBLEPREC
00040
00041         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00042             xmat, hdim, bo, hdim, zero, nonotmp, hdim)
00043
00044         CALL dgemm('N', 'C', hdim, hdim, hdim, one, &
00045             nonotmp, hdim, xmat, hdim, zero, bo, hdim)
00046
00047 #elif defined(SINGLEPREC)
00048
00049         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00050             xmat, hdim, bo, hdim, zero, nonotmp, hdim)
00051
00052         CALL sgemm('N', 'C', hdim, hdim, hdim, one, &
00053             nonotmp, hdim, xmat, hdim, zero, bo, hdim)
00054
00055 #endif
00056
00057     ELSE
00058
00059 #ifdef DOUBLEPREC
00060
00061         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00062             xmat, hdim, rhoup, hdim, zero, nonotmp,
00063             hdim)
00064
00065         CALL dgemm('N', 'T', hdim, hdim, hdim, one, &
00066             nonotmp, hdim, xmat, hdim, zero, rhoup,
00067             hdim)
00068
00069         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00070             xmat, hdim, rhodown, hdim, zero, nonotmp,
00071             hdim)
00072
00073         CALL dgemm('N', 'T', hdim, hdim, hdim, one, &
00074             nonotmp, hdim, xmat, hdim, zero, rhodown,
00075             hdim)
00076
00077 #elif defined(SINGLEPREC)
00078
00079         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00080             xmat, hdim, rhoup, hdim, zero, nonotmp,
00081             hdim)
00082
00083         CALL sgemm('N', 'T', hdim, hdim, hdim, one, &
00084             nonotmp, hdim, xmat, hdim, zero, rhoup,
00085             hdim)
00086
00087         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00088             xmat, hdim, rhodown, hdim, zero, nonotmp,
00089             hdim)
00090
00091         CALL sgemm('N', 'T', hdim, hdim, hdim, one, &
00092             nonotmp, hdim, xmat, hdim, zero, rhodown,
00093             hdim)
00094
00095 #endif
00096
00097     ENDIF
00098
00099 END SUBROUTINE deorthomyrho

```

```
00086
00087 #endif
00088
00089     ENDIF
00090
00091     RETURN
00092
00093 END SUBROUTINE deorthomyrho
00094
```

8.85 deorthomyrhoprogress.f90 File Reference

Functions/Subroutines

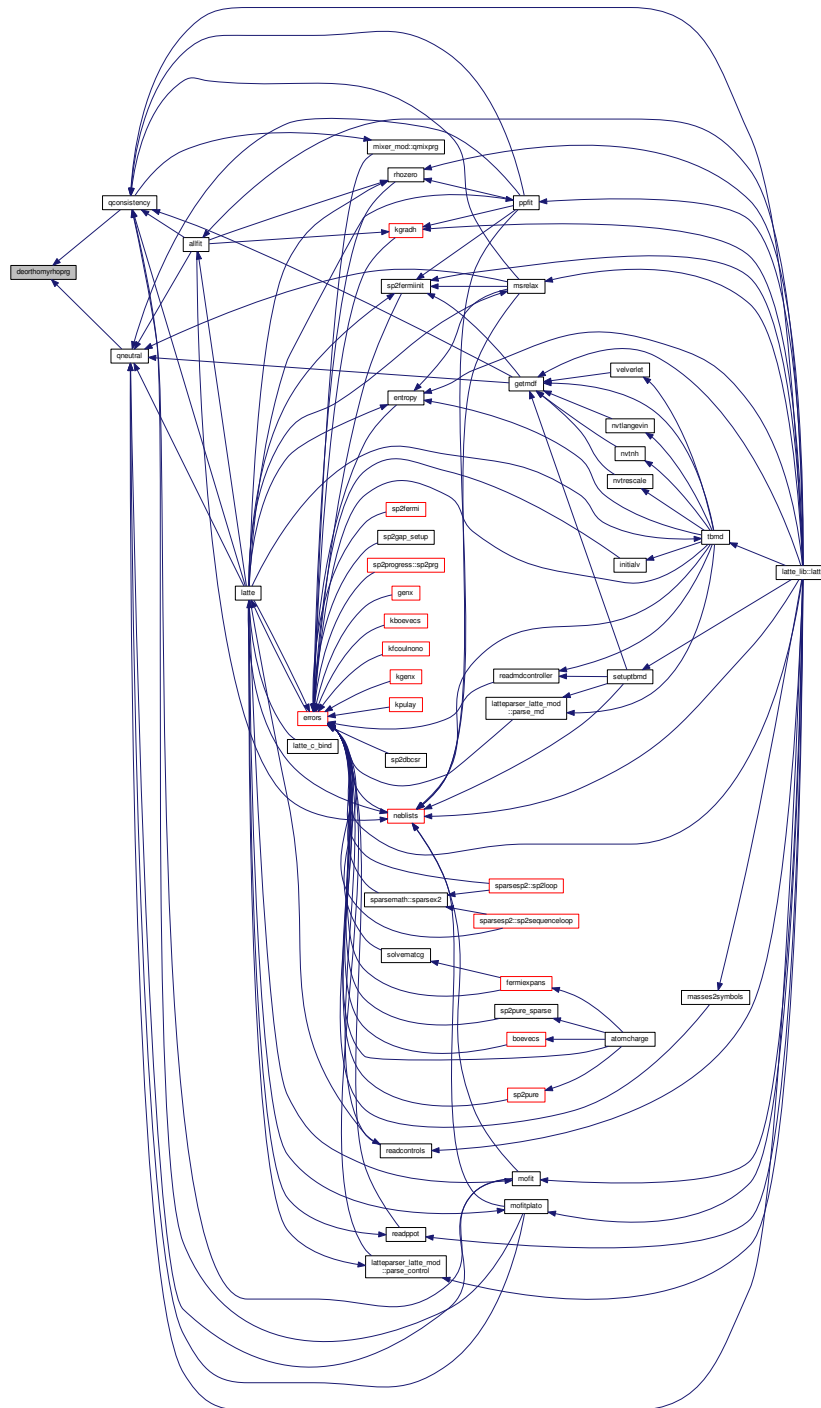
- subroutine [deorthomyrhoprg](#)

8.85.1 Function/Subroutine Documentation

8.85.1.1 subroutine deorthomyrhoprg ()

Definition at line 23 of file [deorthomyrhoprogress.f90](#).

Here is the caller graph for this function:



8.86 deorthomythoprogress.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE deorthomyrhoprg
00023
00024 #ifdef PROGRESSON
00025
00026     USE constants_mod
00027     USE setuparray
00028     USE nonoarray
00029     USE spinarray
00030     USE myprecision
00031     USE latteparser_latte_mod
00032
00033     USE bml
00034     USE prg_nonortho_mod
00035     USE genxprogress
00036
00037     IMPLICIT NONE
00038
00039     TYPE(bml_matrix_t) :: ORTHOBO_BML, BO_BML
00040
00041     IF (existerror) RETURN
00042
00043     !
00044     ! RHO = X ORTHORHO X^dag
00045     !
00046
00047     IF (spinon .EQ. 0) THEN
00048
00049         !! Convert Density matrix to bml format
00050         CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00051             latteprec, hdim, lt%MDIM, orthobo_bml)
00052
00053         CALL bml_import_from_dense(lt%BML_TYPE, &
00054             bo, bo_bml, zero, lt%MDIM)
00055
00056         CALL prg_deorthogonalize(bo_bml, zmat_bml, orthobo_bml, &
00057             lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)
00058
00059         CALL bml_export_to_dense(orthobo_bml, bo)
00060
00061         CALL bml_deallocate(bo_bml)
00062         CALL bml_deallocate(orthobo_bml)
00063
00064     ELSE
00065
00066         !! For rhoup
00067         CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00068             latteprec, hdim, lt%MDIM, orthobo_bml)
00069
00070         CALL bml_import_from_dense(lt%BML_TYPE, &
00071             rhoup, bo_bml, zero, lt%MDIM)
00072
00073         CALL prg_deorthogonalize(bo_bml, zmat_bml, orthobo_bml, &
00074             lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)
00075
00076         CALL bml_export_to_dense(orthobo_bml, rhoup)
00077
00078         CALL bml_deallocate(bo_bml)
00079         CALL bml_deallocate(orthobo_bml)
00080
00081         !! For rhodown
00082         CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00083             latteprec, hdim, lt%MDIM, orthobo_bml)
00084
00085         CALL bml_import_from_dense(lt%BML_TYPE, &
00086             rhodown, bo_bml, zero, lt%MDIM)
00087
00088         CALL prg_deorthogonalize(bo_bml, zmat_bml, orthobo_bml, &
00089             lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)
00090
00091         CALL bml_export_to_dense(orthobo_bml, rhodown)
00092
00093     END IF
00094
00095 END SUBROUTINE deorthomyrhoprg

```

```

00094      CALL bml_deallocate(bo_bml)
00095      CALL bml_deallocate(orthobo_bml)
00096
00097  ENDIF
00098
00099  RETURN
00100
00101 #endif
00102
00103 END SUBROUTINE deorthomyrhopro

```

8.87 dfda.f90 File Reference

Functions/Subroutines

- `real(latteprec)` function `dfda` (`I`, `J`, `L1`, `L2`, `M1`, `M2`, `R`, `ALPHA`, `COSBETA`, `WHICHINT`)

8.87.1 Function/Subroutine Documentation

8.87.1.1 `real(latteprec)` function `dfda` (`integer I`, `integer J`, `integer L1`, `integer L2`, `integer M1`, `integer M2`, `real(latteprec) R`, `real(latteprec) ALPHA`, `real(latteprec) COSBETA`, `character(len=1) WHICHINT`)

Definition at line 23 of file `dfda.f90`.

8.88 dfda.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it       !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it     !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be     !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of   !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dfda(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)
00023
00024     ! Build derivative defined in PRB 72 165107 eq. (13)
00025
00026     USE myprecision
00027
00028     IMPLICIT NONE
00029
00030     INTEGER :: I, J, L1, L2, M1, M2, MP
00031     REAL(LATTEPREC) :: ALPHA, COSBETA, R, DFDA
00032     REAL(LATTEPREC), EXTERNAL :: SLMP, TLMMP, AM, BM, WIGNERD, UNIVSCALE
00033     REAL(LATTEPREC), EXTERNAL :: DSLMPDA, DTLMPDA
00034     CHARACTER(LEN=1) :: WHICHINT
00035
00036     dfda = two * wignerd(l1, abs(m1), 0, cosbeta) * &
00037           wignerd(l2, abs(m2), 0, cosbeta) * &
00038           univscale(i, j, l1, l2, 0, r, whichint) * &
00039           (abs(m1) * bm(m1, alpha) * am(m2, alpha) + abs(m2) * &
00040            am(m1, alpha) * bm(m2, alpha))
00041

```



```

00042 DO mp = 1, min(l1, l2)
00043
00044     dfda = dfda + (dslmmpda(l1, m1, mp, alpha, cosbeta) * &
00045         slmmp(l2, m2, mp, alpha, cosbeta) + &
00046         slmmp(l1, m1, mp, alpha, cosbeta) * &
00047         dslmmpda(l2, m2, mp, alpha, cosbeta) + &
00048         dtlmpda(l1, m1, mp, alpha, cosbeta) * &
00049         tlmmp(l2, m2, mp, alpha, cosbeta) + &
00050         tlmmp(l1, m1, mp, alpha, cosbeta) * &
00051         dtlmpda(l2, m2, mp, alpha, cosbeta)) * &
00052         univscale(i, j, l1, l2, mp, r, whichint)
00053
00054 ENDDO
00055
00056 RETURN
00057
00058 END FUNCTION dfda

```

8.89 dfdb.f90 File Reference

Functions/Subroutines

- `real(latteprec)` function `dfdb` (`I`, `J`, `L1`, `L2`, `M1`, `M2`, `R`, `ALPHA`, `COSBETA`, `WHICHINT`)

8.89.1 Function/Subroutine Documentation

8.89.1.1 `real(latteprec)` function `dfdb` (`integer I`, `integer J`, `integer L1`, `integer L2`, `integer M1`, `integer M2`, `real(latteprec) R`, `real(latteprec) ALPHA`, `real(latteprec) COSBETA`, `character(len=1) WHICHINT`)

Definition at line 23 of file `dfdb.f90`.

8.90 dfdb.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be     !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of   !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dfdb(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)
00023
00024 ! Build derivative defined in PRB 72 165107 eq. (14)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: I, J, L1, L2, M1, M2, MP
00031 REAL(LATTEPREC) :: ALPHA, COSBETA, R, DFDB
00032 REAL(LATTEPREC), EXTERNAL :: SLMMP, TLMMP, AM, WIGNERD, UNIVSCALE
00033 REAL(LATTEPREC), EXTERNAL :: DSLMMPDB, DTLMPDB, DWIGNERDDB
00034 CHARACTER(LEN=1) :: WHICHINT

```

```

00035
00036   dfdb = two * am(m1, alpha) * am(m2, alpha) * &
00037         univscale(i, j, l1, l2, 0, r, whichint) * &
00038         (dwignerddb(l1, abs(m1), 0, cosbeta) * wignerdd(l2, abs(m2), 0, cosbeta) + &
00039         wignerdd(l1, abs(m1), 0, cosbeta) * dwignerddb(l2, abs(m2), 0, cosbeta))
00040
00041   DO mp = 1, min(l1, l2)
00042
00043       dfdb = dfdb + (dslmmpdb(l1, m1, mp, alpha, cosbeta) * &
00044                   slmmp(l2, m2, mp, alpha, cosbeta) + &
00045                   slmmp(l1, m1, mp, alpha, cosbeta) * &
00046                   dslmmpdb(l2, m2, mp, alpha, cosbeta) + &
00047                   dtlmmpdb(l1, m1, mp, alpha, cosbeta) * &
00048                   tlmmp(l2, m2, mp, alpha, cosbeta) + &
00049                   tlmmp(l1, m1, mp, alpha, cosbeta) * &
00050                   dtlmmpdb(l2, m2, mp, alpha, cosbeta)) * &
00051                   univscale(i, j, l1, l2, mp, r, whichint)
00052
00053   ENDDO
00054
00055   RETURN
00056
00057 END FUNCTION dfdb

```

8.91 dfdr.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dfdr](#) (I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)

8.91.1 Function/Subroutine Documentation

8.91.1.1 real(latteprec) function [dfdr](#) (integer *I*, integer *J*, integer *L1*, integer *L2*, integer *M1*, integer *M2*, real(latteprec) *R*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*, character(len=1) *WHICHINT*)

Definition at line 23 of file [dfdr.f90](#).

8.92 dfdr.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dfdr(I, J, L1, L2, M1, M2, R, ALPHA, COSBETA, WHICHINT)
00023
00024   ! Build derivative defined in PRB 72 165107 eq. (15)
00025
00026   USE myprecision
00027
00028   IMPLICIT NONE

```

```

00029
00030  INTEGER :: I, J, L1, L2, M1, M2, MP
00031  REAL(LATTEPREC) :: ALPHA, COSBETA, R, DFDR
00032  REAL(LATTEPREC), EXTERNAL :: SLMMP, TLMMP, AM, WIGNERD, DUNIVSCALE
00033  CHARACTER(LEN=1) :: WHICHINT
00034
00035  dfdr = two * am(m1, alpha) * am(m2, alpha) * &
00036         wignerd(l1, abs(m1), 0, cosbeta) * &
00037         wignerd(l2, abs(m2), 0, cosbeta) * &
00038         dunivscale(i, j, l1, l2, 0, r, whichint)
00039
00040  DO mp = 1, min(l1, l2)
00041
00042     dfdr = dfdr + (slmmp(l1, m1, mp, alpha, cosbeta) * &
00043                  slmmp(l2, m2, mp, alpha, cosbeta) + &
00044                  tlmmp(l1, m1, mp, alpha, cosbeta) * &
00045                  tlmmp(l2, m2, mp, alpha, cosbeta)) * &
00046                  dunivscale(i, j, l1, l2, mp, r, whichint)
00047
00048  ENDDO
00049
00050  RETURN
00051
00052  END FUNCTION dfdr

```

8.93 dgspdr.f90 File Reference

Functions/Subroutines

- `real(latteprec)` function `dgspdr` (`I`, `J`, `L1`, `L2`, `M`, `MAGR`)

8.93.1 Function/Subroutine Documentation

8.93.1.1 `real(latteprec)` function `dgspdr` (`integer I`, `integer J`, `integer L1`, `integer L2`, `integer M`, `real(latteprec) MAGR`)

Definition at line 23 of file `dgspdr.f90`.

8.94 dgspdr.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022  FUNCTION dgspdr(I, J, L1, L2, M, MAGR)
00023
00024     ! Derivative of Goodwin-Skinner-Pettifor function
00025     ! Note that -d/dR is returned
00026
00027     USE gsparray
00028     USE myprecision
00029     USE setuparray

```

```

00030
00031     IMPLICIT NONE
00032
00033     INTEGER :: IC, IGS, I, J, L1, L2, M, IP1, IP2
00034     REAL(LATTEPREC) :: MAGR, DGSPDR, DC
00035     REAL(LATTEPREC) :: RMINUSR1, RMINUSR12, RMINUSR14
00036     REAL(LATTEPREC), EXTERNAL :: GSP
00037     CHARACTER (LEN = 3) :: IGLTYPE
00038     LOGICAL BINGO
00039
00040     ! can't test directly on L values because basis strings always list
00041     ! lower L values first
00042     IF (l1 > l2) THEN
00043         ip1 = l2
00044         ip2 = l1
00045     ELSE
00046         ip1 = l1
00047         ip2 = l2
00048     ENDIF
00049
00050     ! build basis string from L and M values - pure hackery
00051     SELECT CASE(ip1)
00052     CASE(0)
00053         igltype = "s"
00054     CASE(1)
00055         igltype = "p"
00056     CASE(2)
00057         igltype = "d"
00058     CASE(3)
00059         igltype = "f"
00060     END SELECT
00061     SELECT CASE(ip2)
00062     CASE(0)
00063         igltype = trim(igltype)//"s"
00064     CASE(1)
00065         igltype = trim(igltype)//"p"
00066     CASE(2)
00067         igltype = trim(igltype)//"d"
00068     CASE(3)
00069         igltype = trim(igltype)//"f"
00070     END SELECT
00071     SELECT CASE(m)
00072     CASE(0)
00073         igltype = trim(igltype)//"s"
00074     CASE(1)
00075         igltype = trim(igltype)//"p"
00076     CASE(2)
00077         igltype = trim(igltype)//"d"
00078     CASE(3)
00079         igltype = trim(igltype)//"f"
00080     END SELECT
00081
00082     ic = 0
00083     bingo = .false.
00084     DO WHILE (.NOT. bingo)
00085         ic = ic + 1
00086         IF (((atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic)) .OR. &
00087             (atele(j) .EQ. ele1(ic) .AND. atele(i) .EQ. ele2(ic))) .AND. &
00088             (igltype .EQ. btype(ic))) THEN
00089             igs = ic
00090             bingo = .true.
00091         ENDIF
00092     ENDDO
00093
00094     IF (magr .LE. gsp1(igs)) THEN
00095
00096         dgspdr = gspn(igs) * gsp(i, j, l1, l2, m, magr) * (gspnc(igs)* &
00097             ((magr/gsprc(igs))*gspnc(igs)) + one) / magr
00098
00099     ELSEIF (magr .LT. gsprcut(igs)) THEN
00100
00101         rminusr1 = magr - gsp1(igs)
00102         rminusr12 = rminusr1*rminusr1
00103         rminusr14 = rminusr12*rminusr12
00104
00105         dgspdr = minusone * (b(2,igs) + two * b(3,igs) * rminusr1 + &
00106             three * b(4,igs) * rminusr12 + four * b(5,igs) * rminusr12 * &
00107             rminusr1 + five * b(6,igs) * rminusr14)
00108         dgspdr = dgspdr * hr0(igs)
00109
00110         ! note that permutation symmetry when GSPR1 < MAGR < GSPRCUT
00111         ! is accounted for in GSP
00112         IF (l1 > l2) THEN
00113             IF (mod(l1 + l2, 2) /= 0) dgspdr = minusone * dgspdr
00114         ENDIF
00115
00116     ELSE

```

```

00117
00118     dgspdr = zero
00119
00120     ENDIF
00121
00122     RETURN
00123
00124 END FUNCTION dgspdr

```

8.95 diagarray.f90 File Reference

Modules

- module [diagarray](#)

Variables

- real(latteprec), dimension(:), allocatable [diagarray::cplist](#)
- integer, dimension(:), allocatable [diagarray::diag_iwork](#)
- integer [diagarray::diag_liwork](#)
- integer [diagarray::diag_lrwork](#)
- integer [diagarray::diag_lwork](#)
- integer [diagarray::diag_lzwork](#)
- real(latteprec), dimension(:), allocatable [diagarray::diag_rwork](#)
- real(latteprec), dimension(:), allocatable [diagarray::diag_work](#)
- complex(latteprec), dimension(:), allocatable [diagarray::diag_zwork](#)
- real(latteprec), dimension(:), allocatable [diagarray::downevals](#)
- real(latteprec), dimension(:, :), allocatable [diagarray::downvecs](#)
- real(latteprec), dimension(:), allocatable [diagarray::evals](#)
- real(latteprec), dimension(:, :), allocatable [diagarray::evecs](#)
- real(latteprec), parameter [diagarray::exptol](#) = 30.0
- integer, dimension(:), allocatable [diagarray::ifail](#)
- real(latteprec), dimension(:, :), allocatable [diagarray::kevals](#)
- complex(latteprec), dimension(:, :, :), allocatable [diagarray::kevecs](#)
- complex(latteprec), dimension(:, :), allocatable [diagarray::khtmlp](#)
- real(latteprec) [diagarray::numlimit](#)
- real(latteprec), dimension(:), allocatable [diagarray::upevals](#)
- real(latteprec), dimension(:, :), allocatable [diagarray::upevecs](#)
- complex(latteprec), dimension(:, :), allocatable [diagarray::zbo](#)
- integer, dimension(:), allocatable [diagarray::zheevd_iwork](#)
- integer [diagarray::zheevd_liwork](#)
- integer [diagarray::zheevd_lrwork](#)
- integer [diagarray::zheevd_lwork](#)
- real(latteprec), dimension(:), allocatable [diagarray::zheevd_rwork](#)
- complex(latteprec), dimension(:), allocatable [diagarray::zheevd_work](#)

8.96 diagarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE diagarray
00023
00024     USE myprecision
00025
00026     IMPLICIT NONE
00027     SAVE
00028
00029     INTEGER :: diag_lwork, diag_liwork, diag_lzwork,
00030               diag_lrwork
00031     INTEGER :: zheevd_lwork, zheevd_lrwork, zheevd_liwork
00032     INTEGER, ALLOCATABLE :: diag_iwork(:), ifail(:)
00033     INTEGER, ALLOCATABLE :: zheevd_iwork(:)
00034     REAL(LATTEPREC), ALLOCATABLE :: diag_work(:), diag_rwork(:)
00035     REAL(LATTEPREC), ALLOCATABLE :: zheevd_rwork(:)
00036     COMPLEX(LATTEPREC), ALLOCATABLE :: diag_zwork(:)
00037     COMPLEX(LATTEPREC), ALLOCATABLE :: zheevd_work(:)
00038     REAL(LATTEPREC), ALLOCATABLE :: evals(:), evecs(:, :)
00039     REAL(LATTEPREC), ALLOCATABLE :: upevals(:), upevecs(:, :)
00040     REAL(LATTEPREC), ALLOCATABLE :: downevals(:), downevecs(:, :)
00041     COMPLEX(LATTEPREC), ALLOCATABLE :: zbo(:, :), khtmp(:, :)
00042     REAL(LATTEPREC), ALLOCATABLE :: kevals(:, :), cplist(:)
00043     COMPLEX(LATTEPREC), ALLOCATABLE :: kevecs(:, :, :)
00044     REAL(LATTEPREC), PARAMETER :: exptol = 30.0
00045     REAL(LATTEPREC) :: numlimit
00046     ! NUMLIMIT = EXP(-EXPTOL)
00047
00048 END MODULE diagarray

```

8.97 diagmyh.f90 File Reference

Functions/Subroutines

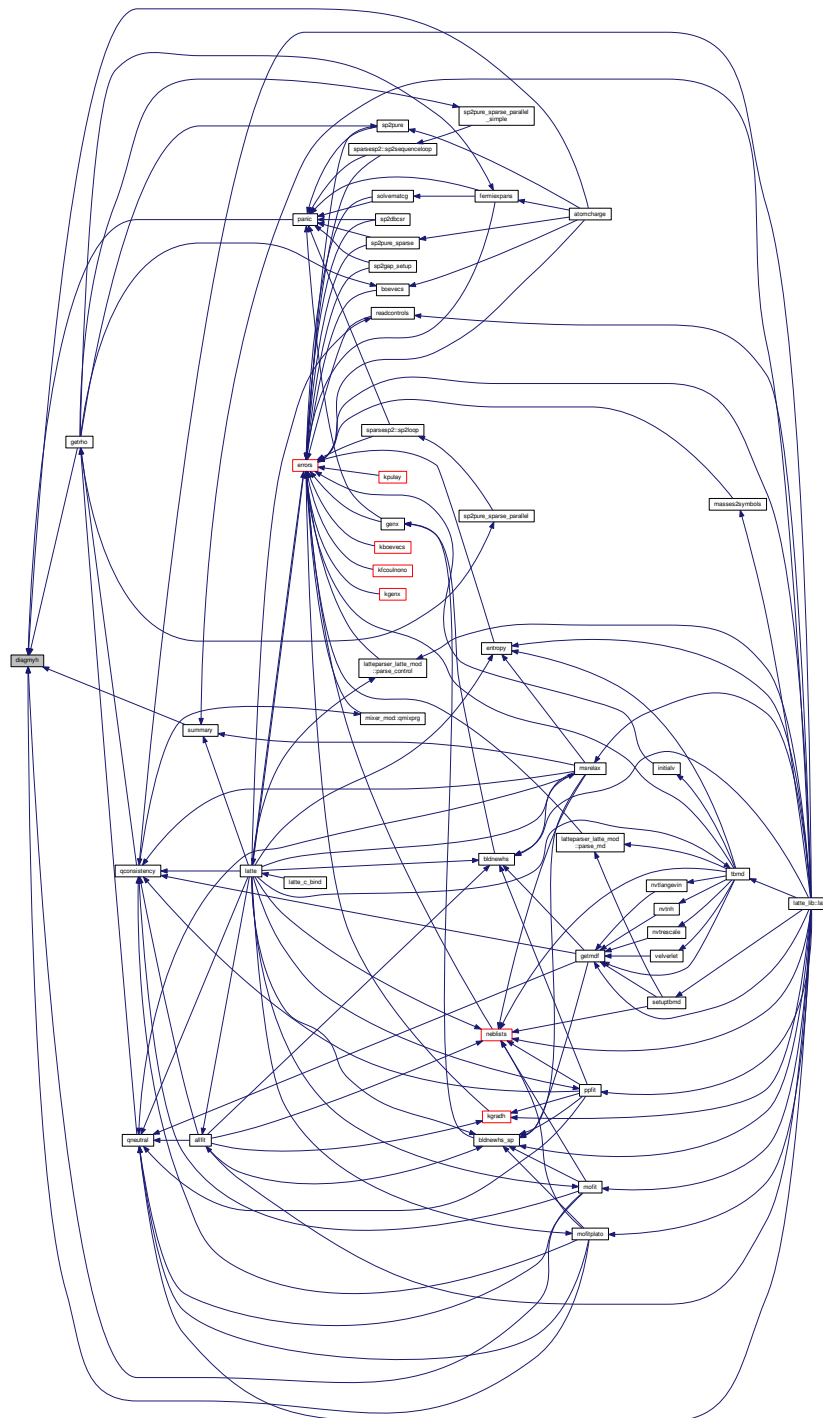
- subroutine [diagmyh](#) ()

8.97.1 Function/Subroutine Documentation

8.97.1.1 subroutine [diagmyh](#) ()

Definition at line 23 of file [diagmyh.f90](#).

Here is the caller graph for this function:



8.98 diagmyh.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE diagmyh()
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE spinarray
00028   USE nonoarray
00029   USE kspacearray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: INFO
00035   INTEGER :: I, J, K, M
00036   ! REAL(8), ALLOCATABLE :: TMPHMAT(:, :), TMPEVALS(:), DWORK(:)
00037   CHARACTER(LEN=1), PARAMETER :: JOBZ = "V", uplo = "U"
00038   IF (existerror) RETURN
00039
00040   IF (spinon .EQ. 0) THEN
00041
00042     IF (basistype .EQ. "ORTHO") THEN
00043       evecs = h
00044     ELSE
00045       evecs = orthoh
00046     ENDIF
00047
00048 #ifdef XSYEV
00049
00050 #ifdef DOUBLEPREC
00051     CALL dsyev(jobz, uplo, hdim, evecs, hdim, evals, &
00052              diag_work, diag_lwork, info)
00053 #elif defined(SINGLEPREC)
00054
00055     ! ALLOCATE(TMPHMAT(HDIM, HDIM), TMPEVALS(HDIM), DWORK(3*HDIM - 1))
00056     ! TMPHMAT = DBLE(H)
00057
00058     CALL ssyev(jobz, uplo, hdim, evecs, hdim, evals, &
00059              diag_work, diag_lwork, info)
00060     ! CALL DSYEV(JOBZ, UPLO, HDIM, TMPHMAT, HDIM, TMPEVALS, &
00061     !          DWORK, DIAG_LWORK, INFO)
00062
00063     ! EVECS = REAL(TMPHMAT)
00064     ! EVALS = REAL(TMPEVALS)
00065     ! DEALLOCATE(TMPHMAT, TMPEVALS, DWORK)
00066
00067 #endif
00068 #endif
00069
00070 #elif defined(XSYEVD)
00071
00072 #ifdef DOUBLEPREC
00073     CALL dsyevd(jobz, uplo, hdim, evecs, hdim, evals, &
00074              diag_work, diag_lwork, diag_iwork,
00075              diag_liwork, info)
00076 #elif defined(SINGLEPREC)
00077     CALL ssyevd(jobz, uplo, hdim, evecs, hdim, evals, &
00078              diag_work, diag_lwork, diag_iwork,
00079              diag_liwork, info)
00080 #endif
00081 #endif
00082
00083   ELSE
00084
00085     IF (basistype .EQ. "ORTHO") THEN
00086       upevecs = hup
00087       downevecs = hdown
00088     ELSE
00089       upevecs = orthohup
00090       downevecs = orthohdown
00091     ENDIF

```



```

00092
00093 #ifdef XSYEV
00094
00095 #ifdef DOUBLEPREC
00096     CALL dsyev(jobz, uplo, hdim, upevecs, hdim, upevals, &
00097              diag_work, diag_lwork, info)
00098 #elif defined(SINGLEPREC)
00099     CALL ssyev(jobz, uplo, hdim, upevecs, hdim, upevals, &
00100              diag_work, diag_lwork, info)
00101 #endif
00102
00103 #ifdef DOUBLEPREC
00104     CALL dsyev(jobz, uplo, hdim, downevecs, hdim, downevals, &
00105              diag_work, diag_lwork, info)
00106 #elif defined(SINGLEPREC)
00107     CALL ssyev(jobz, uplo, hdim, downevecs, hdim, downevals, &
00108              diag_work, diag_lwork, info)
00109 #endif
00110
00111 #elif defined(XSYEVD)
00112
00113 #ifdef DOUBLEPREC
00114     CALL dsyevd(jobz, uplo, hdim, upevecs, hdim, upevals, &
00115              diag_work, diag_lwork, diag_iwork,
00116              diag_liwork, info)
00117 #elif defined(SINGLEPREC)
00118     CALL ssyevd(jobz, uplo, hdim, upevecs, hdim, upevals, &
00119              diag_work, diag_lwork, diag_iwork,
00120              diag_liwork, info)
00121 #endif
00122
00123 #ifdef DOUBLEPREC
00124     CALL dsyevd(jobz, uplo, hdim, downevecs, hdim, downevals, &
00125              diag_work, diag_lwork, diag_iwork,
00126              diag_liwork, info)
00127 #elif defined(SINGLEPREC)
00128     CALL ssyevd(jobz, uplo, hdim, downevecs, hdim, downevals, &
00129              diag_work, diag_lwork, diag_iwork,
00130              diag_liwork, info)
00131 #endif
00132 #endif
00133
00134     RETURN
00135 END SUBROUTINE diagmyh

```

8.99 dosfit.f90 File Reference

Functions/Subroutines

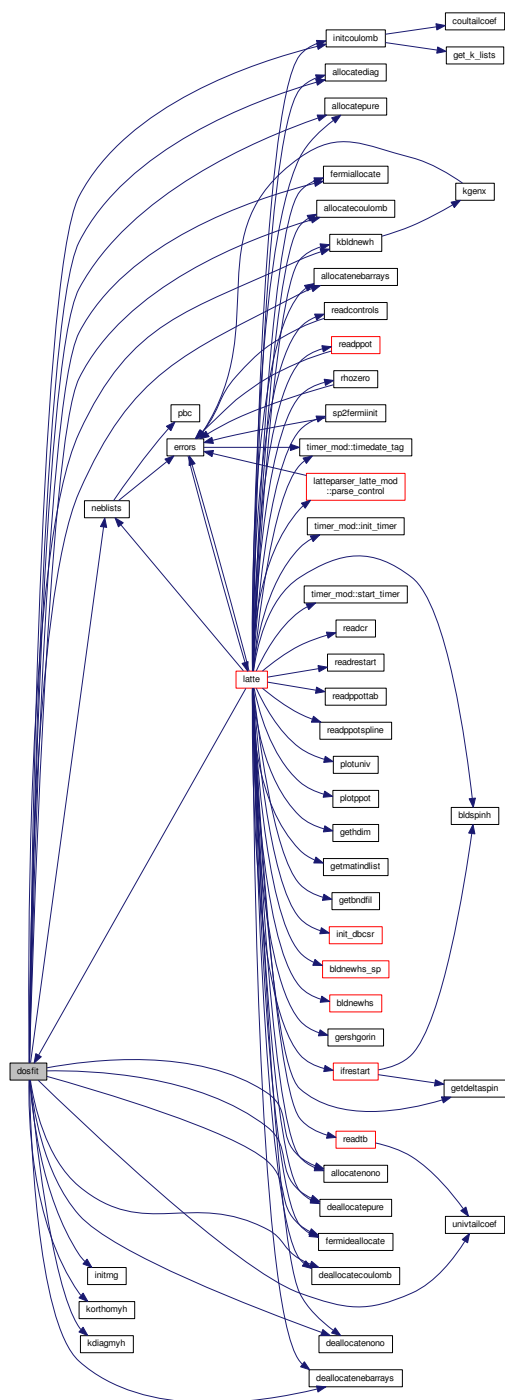
- subroutine [dosfit](#)

8.99.1 Function/Subroutine Documentation

8.99.1.1 subroutine dosfit ()

Definition at line 23 of file [dosfit.f90](#).

Here is the call graph for this function:



[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was made available
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
00004 ! National Laboratory (LANL), which is operated by Los Alamos National Security, LLC
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has the right
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE dosfit
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE kspacearray
00028   USE univarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, JUNK, ACC, II, COUNT
00034   INTEGER :: NSTEP, PICK, CPLOC(1), LOOPTARGET, AB
00035   REAL(LATTEPREC) :: EMIN, EMAX, MERIT, OLDMERIT, RN, ESTEP, EF
00036   REAL(LATTEPREC), ALLOCATABLE :: DFTDOS(:), DFTINTDOS(:)
00037   REAL(LATTEPREC), ALLOCATABLE :: TBDOS(:), TBOLD(:), TBORIG(:)
00038   REAL(LATTEPREC), ALLOCATABLE :: TBSCLOLD(:), TBSCLOLORIG(:)
00039   REAL(LATTEPREC), ALLOCATABLE :: TBBEST(:)
00040   REAL(LATTEPREC), PARAMETER :: ETA = 0.05, dosweight = 1.0d0
00041   REAL(LATTEPREC) :: ENERGY, INTDOS, NUME, JUNKNUM, MINERR
00042   COMPLEX(LATTEPREC) :: CMPARG
00043   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00044   IF (existerror) RETURN
00045
00046   minerr = 1.0d6
00047
00048   ! Read in the DOS from a VASP calculation
00049
00050   OPEN(unit=20, status="OLD", file="DOSCAR2fit.dat")
00051
00052   READ(20,*) emax, emin, nstep, ef, junknum
00053
00054   emax = emax - ef
00055   emin = emin - ef
00056
00057   looptarget = int(bndfil*REAL(hdim*nktot))
00058
00059   ALLOCATE(dftdos(nstep), dftintdos(nstep), tbdos(nstep))
00060   ALLOCATE(tbold(int2fit), tborig(int2fit))
00061   ALLOCATE(tbsclold(int2fit), tbsclorig(int2fit))
00062   ALLOCATE(tbbest(int2fit))
00063   DO i = 1, nstep
00064
00065     READ(20,*) junknum, dftdos(i), dftintdos(i)
00066
00067   ENDDO
00068
00069   CLOSE(20)
00070
00071   ! Initialize stuff
00072
00073
00074   IF (control .EQ. 1) THEN
00075     CALL allocateddiag
00076   ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00077     CALL allocatepure
00078   ELSEIF (control .EQ. 3) THEN
00079     CALL fermiallocate
00080   ENDIF
00081
00082   IF (electro .EQ. 1) THEN
00083     CALL allocatecoulomb
00084     CALL initcoulomb
00085   ENDIF
00086
00087   !
00088   ! Allocate stuff for building the neighbor lists
00089   !
00090
00091   CALL allocatenebarrays
00092
00093   CALL neblists(0)

```

```

00094
00095 IF (basistype .EQ. "NONORTHO") CALL allocatenono
00096
00097 ! Compute DOS from LATTE and compare
00098
00099 CALL initrng
00100
00101 acc = 0
00102
00103 DO i = 1, int2fit
00104     tborig(i) = bond(1,i)
00105     tbsclorig(i) = bond(2,i)
00106 ENDDO
00107
00108 nume = zero
00109 DO i = 1, nats
00110     nume = nume + atocc(elempointer(i))
00111 ENDDO
00112
00113 DO i = 1, int2fit
00114     tbbest(i) = bond(1,i)
00115 ENDDO
00116
00117
00118
00119 DO ii = 1, nfitstep
00120
00121     ! Change set of TB parameters
00122
00123     DO i = 1, int2fit
00124
00125         tbold(i) = bond(1,i)
00126         tbsclold(i) = bond(2,i)
00127     ENDDO
00128
00129     ! Pick one to change
00130
00131
00132
00133     IF (ii .GT. 1) THEN
00134         CALL random_number(rn)
00135
00136         pick = int(rn*REAL(int2fit)) + 1
00137
00138         CALL random_number(rn)
00139
00140         ! AB = INT(RN*2.0D0) + 1
00141         ab = 1
00142         CALL random_number(rn)
00143
00144         ! Change by +/- 20%
00145
00146         bond(ab,pick) = bond(ab,pick) * gaussrn(one, mcsigma)
00147
00148         bond(ab,pick) = sign(bond(ab,pick), tborig(pick))
00149
00150         IF (abs(bond(1,pick)) .LT. 0.01) &
00151             bond(1,pick) = sign(0.01d0, tborig(pick))
00152
00153         ! IF (BOND(1,PICK)/TBORIG(PICK) .LT. 0.5D0) THEN
00154         !     BOND(1,PICK) = 0.5D0*TBORIG(PICK)
00155         ! ELSEIF (BOND(1,PICK)/TBORIG(PICK) .GT. 1.5D0) THEN
00156         !     BOND(1,PICK) = 1.5D0*TBORIG(PICK)
00157         ! ENDIF
00158
00159         ! IF (BOND(2,PICK)/TBSCLOLIG(PICK) .LT. 0.5D0) THEN
00160         !     BOND(2,PICK) = 0.5D0*TBSCLOLIG(PICK)
00161         ! ELSEIF (BOND(2,PICK)/TBSCLOLIG(PICK) .GT. 1.5D0) THEN
00162         !     BOND(2,PICK) = 1.5D0*TBSCLOLIG(PICK)
00163         ! ENDIF
00164
00165         CALL univtailcoef(bond(:,pick))
00166
00167     ENDDO
00168
00169     ! Re-build cut-off tails
00170
00171     ! Build H
00172
00173     CALL kbldnewh
00174
00175     IF (basistype .EQ. "NONORTHO") CALL korthomyh
00176
00177     IF (qfit .EQ. 0) THEN
00178
00179
00180

```

```

00181      CALL kdiagmyh
00182
00183      ! Find the chemical potential - we need the looptarget'th lowest
00184      ! eigenvalue
00185
00186      count = 0
00187      DO i = 1, nktot
00188          DO j = 1, hdim
00189              count = count + 1
00190              cplist(count) = kevals(j,i)
00191          ENDDO
00192      ENDDO
00193
00194      DO i = 1, looptarget
00195          chempot = minval(cplist)
00196          cploc = minloc(cplist)
00197          cplist(cploc) = 1.0d12 ! We do this so we don't get this eigenvalue again on the next loop
00198      ENDDO
00199
00200      ELSE
00201
00202
00203          !      IF (ELECTRO .EQ. 0) THEN
00204          !          CALL QNEUTRAL(0, 1) ! Local charge neutrality
00205          !      ELSE
00206          !          CALL QCONSISTENCY(0,1) ! Self-consistent charges
00207          !      ENDIF
00208
00209
00210      ENDIF
00211
00212      kevals = kevals - chempot
00213
00214      ! Compute DOS
00215
00216      tbdos = zero
00217
00218      estep = (emax - emin)/REAL(nstep)
00219
00220      DO i = 1, nstep
00221
00222          energy = emin + REAL(i-1)*ESTEP
00223
00224          DO k = 1, nktot
00225              DO j = 1, hdim
00226
00227                  cmparg = one/(energy - kevals(j,k) + cmplx(zero,eta))
00228
00229                  tbdos(i) = tbdos(i) - (one/pi)*aimag(cmparg)
00230
00231              ENDDO
00232          ENDDO
00233
00234      ENDDO
00235
00236      tbdos = tbdos/REAL(nktot)
00237
00238      ! Normalize the DOS -> integral to Ef = Ne
00239
00240
00241      intdos = zero
00242      count = 0
00243      DO i = 1, nstep
00244
00245          energy = emin + REAL(i-1)*ESTEP
00246
00247          IF (energy .LE. zero) THEN
00248              count = count + 1
00249              IF (mod(i,2) .EQ. 0) THEN
00250                  intdos = intdos + four*tbdos(i)
00251              ELSE
00252                  intdos = intdos + two*tbdos(i)
00253              ENDIF
00254          ENDIF
00255      ENDDO
00256
00257      intdos = intdos - tbdos(1) - tbdos(count)
00258
00259      intdos = intdos*estep/three
00260
00261      tbdos = tbdos*nume/intdos
00262
00263      DO i = 1, nstep
00264
00265
00266
00267

```

```

00268         tbdos(i) = abs(tbdos(i) - dftdos(i))
00269
00270     ENDDO
00271
00272     intdos = zero
00273     DO i = 1, nstep
00274
00275         energy = emin + REAL(i-1)*ESTEP
00276
00277         IF (mod(i,2) .EQ. 0) THEN
00278             intdos = intdos + four*tbdos(i)
00279         ELSE
00280             intdos = intdos + two*tbdos(i)
00281         ENDIF
00282
00283     ENDDO
00284
00285     intdos = intdos - tbdos(1) - tbdos(nstep)
00286
00287     intdos = intdos*estep/three
00288
00289     merit = intdos
00290
00291     IF (ii .EQ. 1) THEN
00292         oldmerit = merit
00293         minerr = merit
00294     ENDIF
00295
00296
00297     IF (merit .LT. oldmerit) THEN
00298
00299         acc = acc + 1
00300         oldmerit = merit
00301
00302         IF (merit .LT. minerr) THEN
00303
00304             minerr = merit
00305             DO i = 1, int2fit
00306                 tbbest(i) = bond(1,i)
00307             ENDDO
00308
00309         ENDIF
00310
00311     ELSE
00312
00313         CALL random_number(rn)
00314
00315         IF ( exp( -(merit - oldmerit)*mcbeta) .GT. rn ) THEN
00316
00317             acc = acc + 1
00318             oldmerit = merit
00319
00320         ELSE
00321
00322             DO i = 1, int2fit
00323
00324                 bond(1,i) = tbold(i)
00325                 bond(2,i) = tbsclold(i)
00326
00327             ENDDO
00328
00329
00330         ENDIF
00331
00332     ENDIF
00333
00334     WRITE(*,11) ii, acc, merit, oldmerit, &
00335         (bond(1,j), j = 1, int2fit)
00336 11  FORMAT(2i9, 2f12.6, 50f12.6)
00337
00338     ENDDO
00339
00340     DO i = 1, int2fit
00341         bond(1,i) = tbbest(i)
00342     ENDDO
00343
00344
00345     DO i = 1, int2fit
00346         CALL univtailcoef(bond(:,i))
00347     ENDDO
00348
00349     CALL kbldnewh
00350
00351     IF (basistype .EQ. "NONORTHO") CALL korthomyh
00352
00353     IF (qfit .EQ. 0) THEN
00354

```

```

00355     CALL kdiagmyh
00356
00357     ! Find the chemical potential - we need the looptarget'th lowest
00358     ! eigenvalue
00359
00360     count = 0
00361     DO i = 1, nktot
00362         DO j = 1, hdim
00363             count = count + 1
00364             cplist(count) = kevals(j,i)
00365         ENDDO
00366     ENDDO
00367
00368     DO i = 1, looptarget
00369         chempot = minval(cplist)
00370         cploc = minloc(cplist)
00371         cplist(cploc) = 1.0d12 ! We do this so we don't get this eigenvalue again on the next loop
00372     ENDDO
00373
00374     ELSE
00375
00376         !         IF (ELECTRO .EQ. 0) THEN
00377         !             CALL QNEUTRAL(0, 1) ! Local charge neutrality
00378         !         ELSE
00379         !             CALL QCONSISTENCY(0,1) ! Self-consistent charges
00380         !         ENDIF
00381
00382     ENDF
00383
00384     kevals = kevals - chempot
00385
00386
00387     ! Build H
00388
00389
00390
00391     ! Compute and normalize the DOS one more time
00392
00393     tbdos = zero
00394
00395     estep = (emax - emin)/REAL(nstep)
00396
00397     DO i = 1, nstep
00398
00399         energy = emin + REAL(i-1)*ESTEP
00400
00401         DO k = 1, nktot
00402             DO j = 1, hdim
00403
00404                 cmparg = one/(energy - kevals(j,k) + cmplx(zero,eta))
00405
00406                 tbdos(i) = tbdos(i) - (one/pi)*aimag(cmparg)
00407
00408             ENDDO
00409         ENDDO
00410     ENDDO
00411
00412     tbdos = tbdos/REAL(nktot)
00413
00414
00415     ! Normalize the DOS -> integral to Ef = Ne
00416
00417
00418
00419     intdos = zero
00420     count = 0
00421     DO i = 1, nstep
00422
00423         energy = emin + REAL(i-1)*ESTEP
00424
00425         IF (energy .LE. zero) THEN
00426             count = count + 1
00427             IF (mod(i,2) .EQ. 0) THEN
00428                 intdos = intdos + four*tbdos(i)
00429             ELSE
00430                 intdos = intdos + two*tbdos(i)
00431             ENDIF
00432         ENDIF
00433     ENDDO
00434
00435     intdos = intdos - tbdos(1) - tbdos(count)
00436
00437     intdos = intdos*estep/three
00438
00439     tbdos = tbdos*nume/intdos
00440
00441

```



```

00442 OPEN(unit=20, status="UNKNOWN", file="checkdosfit.dat")
00443
00444 DO i = 1, nstep
00445     energy = emin + REAL(i-1)*ESTEP
00446     WRITE(20,10) energy, tbdos(i), dftdos(i)
00447
00448     WRITE(20,10) energy, tbdos(i), dftdos(i)
00449
00450 ENDDO
00451
00452 DO i = 1, int2fit
00453     print*, "# ", bond(1,i)
00454     WRITE(20,*) "# ", bond(1,i)
00455
00456 ENDDO
00457
00458 CLOSE(20)
00459
00460 10 FORMAT(f12.3, 2g18.9)
00461
00462 DEALLOCATE(tbdos, dftdos, dftintdos, tbsclorig, tbsclold, tbold, tborig)
00463 DEALLOCATE(tbbest)
00464
00465 IF (control .EQ. 1) THEN
00466     ! CALL DEALLOCATEDIAG
00467 ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00468     CALL deallocatepure
00469 ELSEIF (control .EQ. 3) THEN
00470     CALL fermideallocate
00471 ENDIF
00472
00473 IF (electro .EQ. 1) CALL deallocatecoulomb
00474
00475 IF (basistype .EQ. "NONORTHO") CALL deallocateenono
00476
00477 CALL deallocateenobarrays
00478
00479
00480 RETURN
00481
00482 END SUBROUTINE dosfit

```

8.101 dslmmpda.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dslmmpda](#) (L, M, MP, ALPHA, COSBETA)

8.101.1 Function/Subroutine Documentation

8.101.1.1 real(latteprec) function [dslmmpda](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*)

Definition at line 23 of file [dslmmpda.f90](#).

8.102 dslmmpda.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dslmmpda(L, M, MP, ALPHA, COSBETA)
00023
00024 ! Build derivative defined in PRB 72 165107 eq. (16)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: L, M, MP
00031 REAL(LATTEPREC) :: ALPHA, COSBETA, DSLMMPDA
00032 REAL(LATTEPREC), EXTERNAL :: BM, WIGNERD
00033
00034 dslmmpda = abs(m) * bm(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00035     WIGNERD(1, abs(m), mp, cosbeta) + WIGNERD(1, abs(m), -mp, cosbeta))
00036
00037 RETURN
00038
00039 END FUNCTION dslmmpda

```

8.103 dslmmpdb.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dslmmpdb](#) (L, M, MP, ALPHA, COSBETA)

8.103.1 Function/Subroutine Documentation

8.103.1.1 real(latteprec) function [dslmmpdb](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*)

Definition at line 23 of file [dslmmpdb.f90](#).

8.104 dslmmpdb.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos    !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has    !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE        !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,        !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS           !
00009 ! SOFTWARE. If software is modified to produce derivative works, such        !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                       !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021

```

```

00022 FUNCTION dslmpdb(L, M, MP, ALPHA, COSBETA)
00023
00024 ! Build derivative defined in PRB 72 165107 eq. (17)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: L, M, MP
00031 REAL(LATTEPREC) :: ALPHA, COSBETA, DSLMPDB
00032 REAL(LATTEPREC), EXTERNAL :: AM, DWIGNERDDB
00033
00034 dslmpdb = am(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00035     DWIGNERDDB(1, abs(m), mp, cosbeta) + DWIGNERDDB(1, abs(m), -mp, cosbeta))
00036
00037 RETURN
00038
00039 END FUNCTION dslmpdb

```

8.105 dtlmpda.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dtlmpda](#) (L, M, MP, ALPHA, COSBETA)

8.105.1 Function/Subroutine Documentation

8.105.1.1 real(latteprec) function [dtlmpda](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*)

Definition at line 23 of file [dtlmpda.f90](#).

8.106 dtlmpda.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dtlmpda(L, M, MP, ALPHA, COSBETA)
00023
00024 ! Build derivative defined in PRB 72 165107 eq. (18)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: L, M, MP
00031 REAL(LATTEPREC) :: ALPHA, COSBETA, DTLMPDA
00032 REAL(LATTEPREC), EXTERNAL :: AM, WIGNERD
00033

```

```

00034 IF (m .EQ. 0) THEN
00035   dtlmpda = zero
00036 ELSE
00037   dtlmpda = -abs(m) * am(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00038     WIGNERD(1, abs(m), mp, cosbeta) - WIGNERD(1, abs(m), -mp, cosbeta))
00039 ENDIF
00040
00041 RETURN
00042
00043 END FUNCTION dtlmpda

```

8.107 dtlmpdb.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dtlmpdb](#) (L, M, MP, ALPHA, COSBETA)

8.107.1 Function/Subroutine Documentation

8.107.1.1 real(latteprec) function [dtlmpdb](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*)

Definition at line 23 of file [dtlmpdb.f90](#).

8.108 dtlmpdb.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dtlmpdb(L, M, MP, ALPHA, COSBETA)
00023
00024 ! Build derivative defined in PRB 72 165107 eq. (19)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: L, M, MP
00031 REAL(LATTEPREC) :: ALPHA, COSBETA, DTLMPDB
00032 REAL(LATTEPREC), EXTERNAL :: BM, DWIGNERDDB
00033
00034 IF (m .EQ. 0) THEN
00035   dtlmpdb = zero
00036 ELSE
00037   dtlmpdb = bm(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00038     DWIGNERDDB(1, abs(m), mp, cosbeta) - DWIGNERDDB(1, abs(m), -mp, cosbeta))
00039 ENDIF
00040
00041 RETURN
00042
00043 END FUNCTION dtlmpdb

```

8.109 dunivscaling.f90 File Reference

Functions/Subroutines

- subroutine [dunivscale_sub](#) (R, A, DC, X, DXDR)

8.109.1 Function/Subroutine Documentation

8.109.1.1 subroutine `dunivscale_sub` (`real(latteprec)` *R*, `real(latteprec)`, `dimension(14)` *A*, `real(latteprec)`, `dimension(3)` *DC*, `real(latteprec)` *X*, `real(latteprec)`, `dimension(3)` *DXDR*)

Definition at line [23](#) of file [dunivscaling.f90](#).

Here is the call graph for this function:




```

00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE dunivscale_sub(R, A, DC, X, DXDR)
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027
00028   REAL(LATTEPREC) :: A(14)
00029   REAL(LATTEPREC) :: DC(3), DXDR(3), TMPDXDR, RMOD
00030   REAL(LATTEPREC) :: X, R, RMINUSR1, DPOLYNOM
00031
00032   CALL univscale_sub(r, a, x)
00033
00034   IF (r .LE. a(7)) THEN
00035
00036       rmod = r - a(6)
00037       dpolynom = a(2) + rmod*(two*a(3) + rmod*(three*a(4) + four*a(5)*rmod))
00038
00039       tmpdxdr = dpolynom*x
00040
00041   ELSEIF (r .GT. a(7) .AND. r .LT. a(8)) THEN
00042
00043       rminusr1 = r - a(7)
00044
00045       tmpdxdr = a(10) + rminusr1*(two*a(11) + &
00046           rminusr1*(three*a(12) + rminusr1*(four*a(13) + &
00047               rminusr1*five*a(14))))
00048
00049       tmpdxdr = a(1)*tmpdxdr
00050
00051   ELSE
00052
00053       tmpdxdr = zero
00054
00055   END IF
00056
00057   dxdr = -tmpdxdr*dc
00058
00059   RETURN
00060
00061 END SUBROUTINE dunivscale_sub

```

8.111 dunivscaling_function.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dunivscale](#) (I, J, L1, L2, MP, R, WHICHINT)

8.111.1 Function/Subroutine Documentation

8.111.1.1 real(latteprec) function [dunivscale](#) (integer *I*, integer *J*, integer *L1*, integer *L2*, integer *MP*, real(latteprec) *R*, character(len=1) *WHICHINT*)

Definition at line 38 of file [dunivscaling_function.f90](#).

8.112 dunivscaling_function.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 ! 1 = A0
00023 ! 2 = B1
00024 ! 3 = B2
00025 ! 4 = B3
00026 ! 5 = B4
00027 ! 6 = B5
00028 ! 7 = R1
00029 ! 8 = RCUT
00030 ! 9 = TAIL1
00031 ! 10 = TAIL2
00032 ! 11 = TAIL3
00033 ! 12 = TAIL4
00034 ! 13 = TAIL5
00035 ! 14 = TAIL6
00036
00037 FUNCTION dunivscale(I, J, L1, L2, MP, R, WHICHINT)
00038
00039     USE constants_mod
00040     USE setuparray
00041     USE univarray
00042     USE myprecision
00043
00044     IMPLICIT NONE
00045
00046     INTEGER :: I, J, L1, L2, IP1, IP2, MP, IC
00047     INTEGER :: BREAKLOOP
00048     REAL(LATTEPREC) :: DUNIVSCALE
00049     REAL(LATTEPREC) :: A(14), R, RMINUSR1, DPOLYNOM, RMOD
00050     CHARACTER(LEN=1) :: WHICHINT
00051     CHARACTER(LEN=3) :: IGLTYPE
00052     IF (existerror) RETURN
00053
00054     ! can't test directly on L values because basis strings always list
00055     ! lower L values first
00056
00057     IF (l1 .GT. 12) THEN
00058         ip1 = 12
00059         ip2 = 11
00060     ELSE
00061         ip1 = 11
00062         ip2 = 12
00063     ENDIF
00064
00065     ! build basis string from L and M values - pure hackery
00066
00067     SELECT CASE (ip1)
00068     CASE(0)
00069         igltype = "s"
00070     CASE(1)
00071         igltype = "p"
00072     CASE(2)
00073         igltype = "d"
00074     CASE(3)
00075         igltype = "f"
00076     END SELECT
00077
00078     SELECT CASE (ip2)
00079     CASE(0)
00080         igltype = trim(igltype)//"s"
00081     CASE(1)
00082         igltype = trim(igltype)//"p"
00083     CASE(2)
00084         igltype = trim(igltype)//"d"

```



```

00085     CASE(3)
00086         igltype = trim(igltype)//"f"
00087     END SELECT
00088
00089     SELECT CASE (mp)
00090     CASE(0)
00091         igltype = trim(igltype)//"s"
00092     CASE(1)
00093         igltype = trim(igltype)//"p"
00094     CASE(2)
00095         igltype = trim(igltype)//"d"
00096     CASE(3)
00097         igltype = trim(igltype)//"f"
00098     END SELECT
00099
00100     ! It makes a difference if our atoms are of the species or not...
00101
00102     ! Easier case first ATELE(I) = ATELE(J)
00103
00104     IF (atele(i) .EQ. atele(j)) THEN
00105
00106         breakloop = 0
00107         ic = 0
00108         DO WHILE (breakloop .EQ. 0)
00109
00110             ic = ic + 1
00111             IF (atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic) .AND. &
00112                 igltype .EQ. btype(ic)) THEN
00113
00114                 ! Now we've ID'ed our bond integral
00115
00116                 SELECT CASE(whichint)
00117                 CASE("H") ! We're doing the H matrix build
00118                     a = bond(:,ic)
00119                 CASE("S") ! We're doing the S matrix build
00120                     a = overl(:,ic)
00121                 END SELECT
00122
00123                 breakloop = 1
00124
00125             ENDIF
00126         ENDDO
00127
00128     ELSE
00129
00130         ! Elements are different - care must be taken with p-s, s-p etc.
00131
00132         IF (l1 .EQ. l2) THEN ! This is a special case sss, pps, ppp etc.
00133
00134             breakloop = 0
00135             ic = 0
00136             DO WHILE (breakloop .EQ. 0)
00137
00138                 ic = ic + 1
00139                 IF (((atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic)) .OR. &
00140                     (atele(i) .EQ. ele2(ic) .AND. atele(j) .EQ. ele1(ic))) .AND. &
00141                     igltype .EQ. btype(ic)) THEN
00142
00143                     ! Now we've ID'ed our bond integral
00144
00145                     SELECT CASE(whichint)
00146                     CASE("H") ! We're doing the H matrix build
00147                         a = bond(:,ic)
00148                     CASE("S") ! We're doing the S matrix build
00149                         a = overl(:,ic)
00150                     END SELECT
00151
00152                     breakloop = 1
00153
00154                 ENDIF
00155             ENDDO
00156
00157         ELSE ! L1 .NE. L2
00158
00159             IF (l1 .LT. l2) THEN
00160
00161                 DO ic = 1, noint
00162
00163                     IF ((atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic)) .AND. &
00164                         igltype .EQ. btype(ic)) THEN
00165
00166                         ! Now we've ID'ed our bond integral
00167
00168                         SELECT CASE(whichint)
00169                         CASE("H") ! We're doing the H matrix build
00170                             a = bond(:,ic)
00171                         CASE("S") ! We're doing the S matrix build
00172                             a = overl(:,ic)

```

```

00172             END SELECT
00173
00174             ENDIF
00175         ENDDO
00176
00177     ELSE
00178
00179         DO ic = 1, noint
00180
00181             IF ((atele(i) .EQ. ele2(ic) .AND. atele(j) .EQ. ele1(ic)) .AND. &
00182                 igltype .EQ. btype(ic)) THEN
00183
00184                 ! Now we've ID'ed our bond integral
00185
00186                 SELECT CASE(whichint)
00187                 CASE("H") ! We're doing the H matrix build
00188                     a = bond(:,ic)
00189                 CASE("S") ! We're doing the S matrix build
00190                     a = overl(:,ic)
00191                 END SELECT
00192
00193             ENDIF
00194         ENDDO
00195
00196     ENDIF
00197 ENDIF
00198
00199 ENDIF
00200
00201 IF (r .LE. a(7)) THEN
00202
00203     rmod = r - a(6)
00204
00205     dpolynom = a(2) + rmod*(two*a(3) + rmod*(three*a(4) + four*a(5)*rmod))
00206
00207     dunivscale = dpolynom * exp( rmod*(a(2) + &
00208         rmod*(a(3) + rmod*(a(4) + a(5)*rmod))) )
00209
00210 ELSEIF (r .GT. a(7) .AND. r .LT. a(8)) THEN
00211
00212     rminusr1 = r - a(7)
00213
00214     dunivscale = a(10) + rminusr1*(two*a(11) + &
00215         rminusr1*(three*a(12) + rminusr1*(four*a(13) + &
00216             rminusr1*five*a(14))))
00217
00218 ELSE
00219
00220     dunivscale = zero
00221
00222 END IF
00223
00224 dunivscale = -a(1)*dunivscale
00225
00226 ! permutation symmetry
00227
00228 IF (l1 .GT. 12 .AND. mod(l1 + 12, 2) .NE. 0) dunivscale = -dunivscale
00229
00230 ! PRINT*, UNIVSCALE
00231
00232 RETURN
00233
00234 END FUNCTION dunivscale
00235

```

8.113 dwignerddb.f90 File Reference

Functions/Subroutines

- real(latteprec) function [dwignerddb](#) (L, M, MP, COSBETA)

8.113.1 Function/Subroutine Documentation

8.113.1.1 real(latteprec) function [dwignerddb](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *COSBETA*)

Definition at line 23 of file [dwignerddb.f90](#).

8.114 dwignerddb.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION dwignerddb(L, M, MP, COSBETA)
00023
00024 ! Build derivative defined in PRB 72 165107 eqn. (21)
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 INTEGER :: L, M, MP
00031 REAL(LATTEPREC) :: COSBETA, DWIGNERDDB
00032 REAL(LATTEPREC), EXTERNAL :: WIGNERD
00033
00034 dwignerddb = half * sqrt(REAL((L + MP) * (1 - mp + 1))) * &
00035 WIGNERD(1, m, mp - 1, cosbeta) - HALF * SQRT(REAL((L - MP) * &
00036 (1 + mp + 1))) * wignerD(1, m, mp + 1, cosbeta)
00037
00038 RETURN
00039
00040 END FUNCTION dwignerddb

```

8.115 entropy.f90 File Reference

Functions/Subroutines

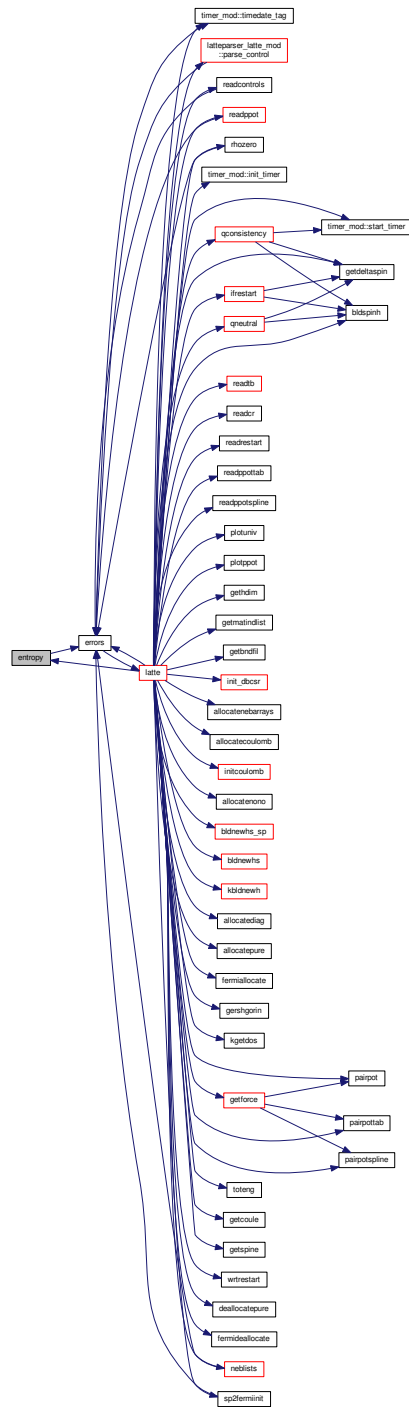
- subroutine [entropy](#)

8.115.1 Function/Subroutine Documentation

8.115.1.1 subroutine [entropy](#) ()

Definition at line 23 of file [entropy.f90](#).

Here is the call graph for this function:



[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was made available
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
00004 ! National Laboratory (LANL), which is operated by Los Alamos National Security, LLC
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has the right
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE entropy
00023
00024 !
00025 ! This subroutine compute the electronic entropy if we're running
00026 ! with a finite electron temperature
00027 !
00028
00029 USE constants_mod
00030 USE setuparray
00031 USE spinarray
00032 USE myprecision
00033
00034 IMPLICIT NONE
00035
00036 INTEGER :: I, J
00037 INTEGER :: LIWORK, LWORK, INFO
00038 INTEGER, ALLOCATABLE :: IWORK(:)
00039 REAL(LATTEPREC), ALLOCATABLE :: WORK(:), S_EVEC(:, :), S_EVAL(:)
00040 REAL(LATTEPREC), ALLOCATABLE :: MAT2(:, :), Y(:, :)
00041 REAL(LATTEPREC) :: OCC, S, NPOWER
00042 REAL(LATTEPREC) :: OCCLOGOCC_HOLES, OCCLOGOCC_ELECTRONS
00043 REAL(LATTEPREC) :: CLOSE2ZERO = 1.0d-16
00044 REAL(LATTEPREC) :: ELECTRON, HOLE, TMPELEC, TMPHOLE
00045 REAL(LATTEPREC) :: LN2, C4(2), C8(4)
00046 CHARACTER(LEN=1), PARAMETER :: JOBZ = "N", uplo = "U"
00047
00048 IF (existerror) RETURN
00049
00050 !
00051 ! Options:
00052 !
00053 ! ENTROPYKIND = 0: S=0 (for testing purposes)
00054 !
00055 ! Sanville's algorithm:
00056 !
00057 ! ENTROPYKIND = 1 : S = XlnX + (1-X)ln(1-X) (exact for Fermi-Dirac smearing)
00058 !
00059 ! THE NEXT 3 OPTIONS WERE INVENTED BY ANDERS NIKLASSON:
00060 !
00061 ! ENTROPYKIND = 2 : An option for use with running finite T SP2
00062 !
00063 ! N = 2^NORECS
00064 !
00065 ! S = 2N*[X(X^(1/N) - 1)/(X^(1/N)+1) + (1-X)*((1-X)^(1/N)-1)/((1-X)^(1/N)+1)]
00066 !
00067 ! ENTROPYKIND = 3: 4th-order expansion: approximate but no diagonalization
00068 !
00069 ! Y = X(X-1)
00070 ! S = Tr(Y(C1 + C2*Y))
00071 !
00072 ! ENTROPYKIND = 4: As above, but with an 8th-order expansion
00073 !
00074 ! Y = X(X-1)
00075 ! S = Tr[C1*Y + Y^2*(C2*I + C3*Y + c4*Y^2)]
00076 !
00077
00078 s = zero
00079
00080 IF (entropykind .EQ. 0) THEN
00081     s = zero
00082
00083 ELSEIF (entropykind .EQ. 1) THEN
00084     ALLOCATE(s_evec(hdim, hdim), s_eval(hdim))
00085
00086 #ifdef XSYEV
00087     lwork = 3*hdim - 1
00088     ALLOCATE(work(lwork))
00089 #elif defined(XSYEVD)
00090     lwork = 1 + 6*hdim + 2*hdim*hdim
00091     liwork = 3 + 5*hdim

```

```

00094      ALLOCATE(work(lwork), iwork(liwork))
00095 #endif
00096
00097      IF (spinon .EQ. 0) THEN
00098
00099          s_evec = half*bo
00100
00101 #ifdef XSYEV
00102      ! Pre-processing to select precision
00103
00104 #ifdef DOUBLEPREC
00105      CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00106 #elif defined(SINGLEPREC)
00107      CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00108 #endif
00109
00110 #elif defined(XSYEVD)
00111
00112 #ifdef DOUBLEPREC
00113      CALL dsyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00114                  iwork, liwork, info)
00115 #elif defined(SINGLEPREC)
00116      CALL ssyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00117                  iwork, liwork, info)
00118 #endif
00119 #endif
00120 #endif
00121
00122      DO i = 1, hdim
00123
00124          occ = s_eval(i)
00125
00126          IF (occ .LE. close2zero .OR. occ .GE. (one - close2zero)) THEN
00127
00128              occlogocc_electrons = zero
00129              occlogocc_holes = zero
00130
00131          ELSE
00132
00133              occlogocc_electrons = occ * log(occ)
00134              occlogocc_holes = (one - occ) * log(one - occ)
00135
00136          ENDIF
00137
00138          s = s + two*(occlogocc_electrons + occlogocc_holes)
00139
00140      ENDDO
00141
00142      ELSE
00143
00144          !
00145          ! Do both density matrices separately - this is correct
00146          ! when we're doing spin-polarized calculations
00147          !
00148
00149          s_evec = rhoup
00150
00151 #ifdef XSYEV
00152      ! Pre-processing to select precision
00153
00154 #ifdef DOUBLEPREC
00155      CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00156 #elif defined(SINGLEPREC)
00157      CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00158 #endif
00159
00160 #elif defined(XSYEVD)
00161
00162 #ifdef DOUBLEPREC
00163      CALL dsyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00164                  iwork, liwork, info)
00165 #elif defined(SINGLEPREC)
00166      CALL ssyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00167                  iwork, liwork, info)
00168 #endif
00169 #endif
00170 #endif
00171
00172      DO i = 1, hdim
00173
00174          occ = s_eval(i)
00175
00176          IF (occ .LE. close2zero .OR. occ .GE. (one - close2zero)) THEN
00177
00178              occlogocc_electrons = zero
00179              occlogocc_holes = zero
00180

```

```

00181
00182         ELSE
00183
00184             occlogocc_electrons = occ * log(occ)
00185             occlogocc_holes = (one - occ) * log(one - occ)
00186
00187         ENDIF
00188
00189         s = s + occlogocc_electrons + occlogocc_holes
00190
00191     ENDDO
00192
00193     s_evec = rhodown
00194
00195 #ifdef XSYEV
00196     ! Pre-processing to select precision
00197
00198 #ifdef DOUBLEPREC
00199     CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00200 #elif defined(SINGLEPREC)
00201     CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00202 #endif
00203
00204 #elif defined(XSYEVD)
00205
00206 #ifdef DOUBLEPREC
00207     CALL dsyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00208                 iwork, liwork, info)
00209 #elif defined(SINGLEPREC)
00210     CALL ssyevd(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork, &
00211                 iwork, liwork, info)
00212 #endif
00213
00214 #endif
00215
00216     DO i = 1, hdim
00217
00218         occ = s_eval(i)
00219
00220         IF (occ .LE. close2zero .OR. occ .GE. (one - close2zero)) THEN
00221
00222             occlogocc_electrons = zero
00223             occlogocc_holes = zero
00224
00225         ELSE
00226
00227             occlogocc_electrons = occ * log(occ)
00228             occlogocc_holes = (one - occ) * log(one - occ)
00229
00230         ENDIF
00231
00232         s = s + occlogocc_electrons + occlogocc_holes
00233
00234     ENDDO
00235
00236 ENDIF
00237
00238 DEALLOCATE(s_evec, s_eval)
00239
00240 #ifdef XSYEV
00241     DEALLOCATE(work)
00242 #elif defined(XSYEVD)
00243     DEALLOCATE(work, iwork)
00244 #endif
00245
00246 ELSEIF (entropykind .EQ. 2) THEN
00247
00248     IF (control .NE. 5) THEN
00249         CALL errors("entropy", "Only use ENTROPYKIND = 2 if you're using CONTROL = 5")
00250     ENDIF
00251
00252     lwork = 3*hdim - 1
00253
00254     ALLOCATE(s_evec(hdim, hdim), s_eval(hdim), work(lwork))
00255
00256     npower = two**norecs
00257
00258     IF (spinon .EQ. 0) THEN
00259
00260         s_evec = half*bo
00261
00262     #ifdef DOUBLEPREC
00263         CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00264     #elif defined(SINGLEPREC)
00265         CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00266     #endif
00267

```



```

00268
00269      DO i = 1, hdim
00270
00271          occ = s_eval(i)
00272
00273          IF (occ .LE. close2zero) THEN
00274              electron = zero
00275          ENDIF
00276
00277          IF (one - occ .LE. close2zero) THEN
00278              hole = zero
00279          ENDIF
00280
00281          IF (occ .GT. close2zero .AND. &
00282              one - occ .GT. close2zero) THEN
00283
00284              tmpelec = occ**(one/npower)
00285              tmphole = (one - occ)**(one/npower)
00286
00287              electron = occ*(tmpelec - one)/(tmpelec + one)
00288              hole = (one - occ)*(tmphole - one)/(tmphole + one)
00289
00290          ENDIF
00291
00292          s = s + four*npower*(electron + hole)
00293
00294      ENDDO
00295
00296      ELSE
00297
00298          !
00299          ! Spin polarized
00300          !
00301
00302          s_evec = rhoup
00303
00304      #ifdef DOUBLEPREC
00305          CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00306      #elif defined(SINGLEPREC)
00307          CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00308      #endif
00309
00310      DO i = 1, hdim
00311
00312          occ = s_eval(i)
00313
00314          IF (occ .LE. close2zero) THEN
00315              electron = zero
00316          ENDIF
00317
00318          IF (one - occ .LE. close2zero) THEN
00319              hole = zero
00320          ENDIF
00321
00322          IF (occ .GT. close2zero .AND. &
00323              one - occ .GT. close2zero) THEN
00324
00325              tmpelec = occ**(one/npower)
00326              tmphole = (one - occ)**(one/npower)
00327
00328              electron = occ*(tmpelec - one)/(tmpelec + one)
00329              hole = (one - occ)*(tmphole - one)/(tmphole + one)
00330
00331          ENDIF
00332
00333          s = s + two*npower*(electron + hole)
00334
00335      ENDDO
00336
00337          s_evec = rhodown
00338
00339      #ifdef DOUBLEPREC
00340          CALL dsyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00341      #elif defined(SINGLEPREC)
00342          CALL ssyev(jobz, uplo, hdim, s_evec, hdim, s_eval, work, lwork,info)
00343      #endif
00344
00345      DO i = 1, hdim
00346
00347          occ = s_eval(i)
00348
00349          IF (occ .LE. close2zero) THEN
00350              electron = zero
00351          ENDIF
00352
00353          IF (one - occ .LE. close2zero) THEN
00354              hole = zero

```

```

00355         ENDIF
00356
00357         IF (occ .GT. close2zero .AND. &
00358             one - occ .GT. close2zero) THEN
00359
00360             tmpelec = occ*(one/npower)
00361             tmphole = (one - occ)*(one/npower)
00362
00363             electron = occ*(tmpelec - one)/(tmpelec + one)
00364             hole = (one - occ)*(tmphole - one)/(tmphole + one)
00365
00366         ENDIF
00367
00368         s = s + two*npower*(electron + hole)
00369
00370     ENDDO
00371
00372 ENDIF
00373
00374 DEALLOCATE(s_evec, s_eval, work)
00375
00376 ELSEIF (entropykind .EQ. 3) THEN
00377
00378     !
00379     ! 4th-order approximation
00380     !
00381
00382     ln2 = log(two)
00383
00384     c4(1) = eight*ln2 - two
00385     c4(2) = sixteen*ln2 - eight
00386
00387     ALLOCATE(mat2(hdim, hdim))
00388
00389     IF (spinon .EQ. 0) THEN
00390
00391         !
00392         ! Here we make use of *GEMM to do half*BO*half*BO
00393         !
00394
00395     #ifdef DOUBLEPREC
00396         CALL dgemm('N', 'N', hdim, hdim, hdim, 0.25d0, &
00397             bo, hdim, bo, hdim, 0.0d0, mat2, hdim)
00398     #elif defined(SINGLEPREC)
00399         CALL sgemm('N', 'N', hdim, hdim, hdim, 0.25, &
00400             bo, hdim, bo, hdim, 0.0, mat2, hdim)
00401     #endif
00402
00403     DO i = 1, hdim
00404         DO j = i, hdim
00405
00406             IF (i .EQ. j) THEN
00407
00408                 s = s + (mat2(i,i) - half*bo(i,i)) * &
00409                     (c4(1) + c4(2)*(mat2(i,i) - half*bo(i,i)))
00410
00411             ELSE
00412
00413                 s = s + two*c4(2) * (mat2(j,i) - half*bo(j,i)) * &
00414                     (mat2(j,i) - half*bo(j,i))
00415
00416             ENDIF
00417
00418         ENDDO
00419     ENDDO
00420
00421     s = two*s
00422
00423 ELSE
00424
00425     !
00426     ! First spin up:
00427     !
00428
00429     #ifdef DOUBLEPREC
00430         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00431             rhoup, hdim, rhoup, hdim, 0.0d0, mat2, hdim)
00432     #elif defined(SINGLEPREC)
00433         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00434             rhoup, hdim, rhoup, hdim, 0.0, mat2, hdim)
00435     #endif
00436
00437     DO i = 1, hdim
00438         DO j = i, hdim
00439
00440             IF (i .EQ. j) THEN
00441

```

```

00442         s = s + (mat2(i,i) - rhoup(i,i)) * &
00443             (c4(1) + c4(2)*(mat2(i,i) - rhoup(i,i)))
00444
00445         ELSE
00446
00447             s = s + two*c4(2) * (mat2(j,i) - rhoup(j,i)) * &
00448                 (mat2(j,i) - rhoup(j,i))
00449
00450         ENDIF
00451
00452     ENDDO
00453 ENDDO
00454
00455     !
00456     ! Spin down
00457     !
00458
00459 #ifdef DOUBLEPREC
00460     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00461         rhodown, hdim, rhodown, hdim, 0.0d0, mat2,
00462         hdim)
00463 #elif defined(SINGLEPREC)
00464     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00465         rhodown, hdim, rhodown, hdim, 0.0, mat2, hdim)
00466 #endif
00467
00468     DO i = 1, hdim
00469         DO j = i, hdim
00470
00471             IF (i .EQ. j) THEN
00472
00473                 s = s + (mat2(i,i) - rhodown(i,i)) * &
00474                     (c4(1) + c4(2)*(mat2(i,i) - rhodown(i,i)))
00475
00476             ELSE
00477
00478                 s = s + two*c4(2) * (mat2(j,i) - rhodown(j,i)) * &
00479                     (mat2(j,i) - rhodown(j,i))
00480
00481             ENDIF
00482
00483         ENDDO
00484     ENDDO
00485
00486     ENDF
00487
00488     DEALLOCATE(mat2)
00489
00490     ELSEIF (entropykind .EQ. 4) THEN
00491
00492         !
00493         ! 8th order expansion
00494         !
00495         ln2 = log(two)
00496
00497         c8(1) = 16.0d0*ln2 - (34.0d0/5.0d0)
00498         c8(2) = 96.0d0*ln2 - (844.0d0/15.0d0)
00499         c8(3) = 256.0d0*ln2 - (2336.0d0/15.0d0)
00500         c8(4) = 256.0d0*ln2 - (2368.0d0/15.0d0)
00501
00502         ALLOCATE(mat2(hdim, hdim), y(hdim, hdim))
00503
00504         IF (spinon .EQ. 0) THEN
00505
00506             !
00507             ! MAT2 = half*BO*half*BO, temporarily (note we take
00508             ! care of the 'halfs' in *GEMM
00509             !
00510
00511 #ifdef DOUBLEPREC
00512         CALL dgemm('N', 'N', hdim, hdim, hdim, 0.25d0, &
00513             bo, hdim, bo, hdim, 0.0d0, mat2, hdim)
00514 #elif defined(SINGLEPREC)
00515         CALL sgemm('N', 'N', hdim, hdim, hdim, 0.25, &
00516             bo, hdim, bo, hdim, 0.0, mat2, hdim)
00517 #endif
00518
00519         y = mat2 - half*bo
00520
00521         !
00522         ! MAT2 = Y*Y
00523         !
00524
00525 #ifdef DOUBLEPREC
00526         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00527             y, hdim, y, hdim, 0.0d0, mat2, hdim)

```

```

00528 #elif defined(SINGLEPREC)
00529     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00530             y, hdim, y, hdim, 0.0, mat2, hdim)
00531 #endif
00532
00533     DO i = 1, hdim
00534         DO j = i, hdim
00535
00536             IF (i .EQ. j) THEN
00537
00538                 s = s + c8(1)*y(i,i) + mat2(i,i) * &
00539                     (c8(2) + c8(3)*y(i,i) + c8(4)*mat2(i,i))
00540
00541             ELSE
00542
00543                 s = s + c8(3)*mat2(j,i)*y(j,i) + &
00544                     c8(4)*mat2(j,i)*mat2(j,i)
00545
00546             ENDIF
00547
00548         ENDDO
00549     ENDDO
00550
00551     s = two*s
00552
00553 ELSE
00554
00555     !
00556     ! For the up-spins:
00557     !
00558     ! MAT2 = X*X
00559
00560 #ifdef DOUBLEPREC
00561     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00562             rhoup, hdim, rhoup, hdim, 0.0d0, mat2, hdim)
00563 #elif defined(SINGLEPREC)
00564     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00565             rhoup, hdim, rhoup, hdim, 0.0, mat2, hdim)
00566 #endif
00567
00568     y = mat2 - rhoup
00569
00570     ! MAT2 = Y*Y
00571
00572 #ifdef DOUBLEPREC
00573     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00574             y, hdim, y, hdim, 0.0d0, mat2, hdim)
00575 #elif defined(SINGLEPREC)
00576     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00577             y, hdim, y, hdim, 0.0, mat2, hdim)
00578 #endif
00579
00580     DO i = 1, hdim
00581         DO j = i, hdim
00582
00583             IF (i .EQ. j) THEN
00584
00585                 s = s + c8(1)*y(i,i) + mat2(i,i) * &
00586                     (c8(2) + c8(3)*y(i,i) + c8(4)*mat2(i,i))
00587
00588             ELSE
00589
00590                 s = s + c8(3)*mat2(j,i)*y(j,i) + &
00591                     c8(4)*mat2(j,i)*mat2(j,i)
00592
00593             ENDIF
00594
00595         ENDDO
00596     ENDDO
00597
00598     !
00599     ! Now for spin-down
00600     !
00601
00602     ! MAT2 = X*X
00603
00604 #ifdef DOUBLEPREC
00605     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00606             rhodown, hdim, rhodown, hdim, 0.0d0, mat2,
00607             hdim)
00608 #elif defined(SINGLEPREC)
00609     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00610             rhodown, hdim, rhodown, hdim, 0.0, mat2, hdim)
00611 #endif
00612
00613     y = mat2 - rhodown

```

```

00614
00615      ! MAT2 = Y*Y
00616
00617 #ifdef DOUBLEPREC
00618      CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00619              y, hdim, y, hdim, 0.0d0, mat2, hdim)
00620 #elif defined(SINGLEPREC)
00621      CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00622              y, hdim, y, hdim, 0.0, mat2, hdim)
00623 #endif
00624
00625      DO i = 1, hdim
00626          DO j = i, hdim
00627
00628              IF (i .EQ. j) THEN
00629
00630                  s = s + c8(1)*y(i,i) + mat2(i,i) * &
00631                      (c8(2) + c8(3)*y(i,i) + c8(4)*mat2(i,i))
00632
00633              ELSE
00634
00635                  s = s + c8(3)*mat2(j,i)*y(j,i) + &
00636                      c8(4)*mat2(j,i)*mat2(j,i)
00637
00638              ENDIF
00639
00640          ENDDO
00641      ENDDO
00642
00643      ENDF
00644
00645      DEALLOCATE(mat2, y)
00646
00647      ENDF
00648
00649      ente = minusone*kbt*s
00650
00651      RETURN
00652
00653 END SUBROUTINE entropy

```

8.117 errors.f90 File Reference

Functions/Subroutines

- subroutine [errors](#) (SUB, TAG)
Subroutine to handle errors.

8.117.1 Function/Subroutine Documentation

8.117.1.1 subroutine errors (character(*), intent(in) SUB, character(*), intent(in) TAG)

Subroutine to handle errors.

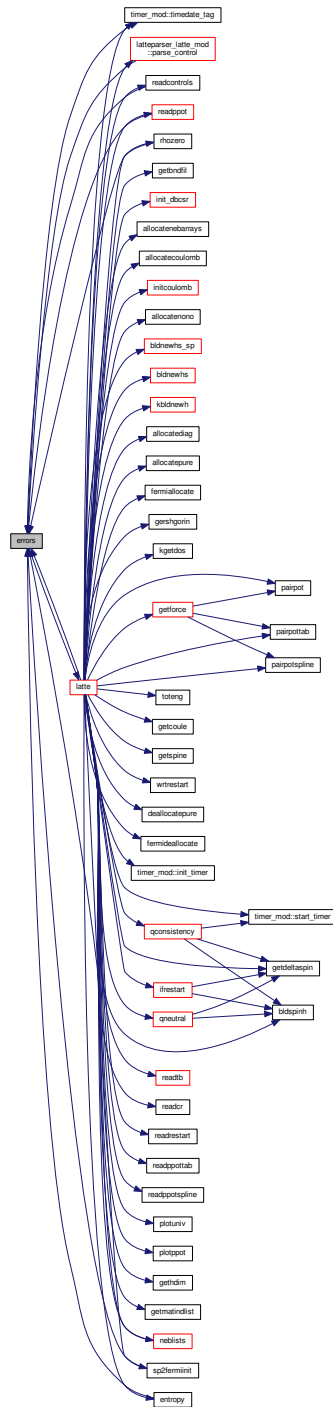
This will decide what to do if the program has to stop for some reason. It will either stop the program or send an error flag up.

Parameters

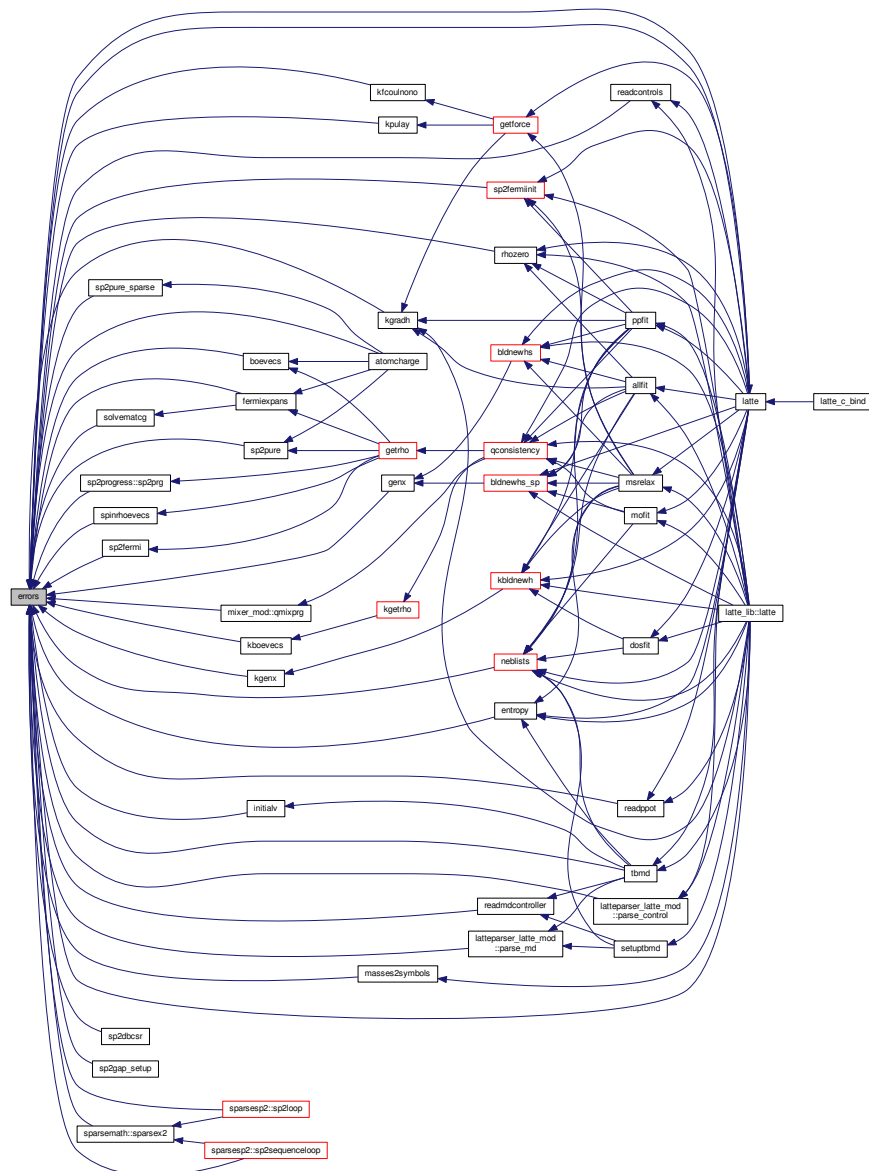
<i>SUB</i>	Subroutine where the program would stop.
<i>TAG</i>	Error message to be passed.

Definition at line 29 of file [errors.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.118 errors.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !

```

```

00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00028 SUBROUTINE errors(SUB,TAG)
00029   USE constants_mod, ONLY: librun, existerror,
libcalls
00030   USE timer_mod, ONLY: timedate_tag
00031   USE myprecision, ONLY: latteprec
00032   USE latte_lib, ONLY: latte
00033   CHARACTER(*), INTENT(IN) :: SUB,TAG
00034
00035   IF(librun)THEN
00036     WRITE(*,*) ""
00037     WRITE(*,*) "# *****ERROR*****"
00038     WRITE(*,*) "# LATTE stopped due to the following error at subroutine ",sub,":"
00039     WRITE(*,*) "# ",tag
00040     CALL timedate_tag("The error occurred at: ")
00041     WRITE(*,*) "# The error will be reported back to the host code"
00042     WRITE(*,*) "# *****"
00043     WRITE(*,*) ""
00044     CALL flush(6)
00045     existerror = .true.
00046   ELSE
00047     WRITE(*,*) ""
00048     WRITE(*,*) "# *****ERROR*****"
00049     WRITE(*,*) "# LATTE stopped due to the following error at subroutine ",sub,":"
00050     WRITE(*,*) "# ",tag
00051     CALL timedate_tag("The error occurred at: ")
00052     WRITE(*,*) "# *****"
00053     WRITE(*,*) ""
00054     stop
00055   ENDIF
00056
00057 END SUBROUTINE errors

```

8.119 factorial.f90 File Reference

Functions/Subroutines

- integer function [factorial](#) (I1)

8.119.1 Function/Subroutine Documentation

8.119.1.1 integer function factorial (integer I1)

Definition at line 23 of file [factorial.f90](#).

8.120 factorial.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !

```



```
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION factorial(I1)
00023
00024 ! factorial function
00025
00026 ! USE MYPRECISION
00027
00028 IMPLICIT NONE
00029 INTEGER :: I1, I, FACTORIAL
00030
00031 factorial = 1
00032
00033 DO i = 2, i1
00034     factorial = factorial * i
00035 ENDDO
00036
00037 RETURN
00038
00039 END FUNCTION factorial
```

8.121 fcoulnono.f90 File Reference

Functions/Subroutines

- subroutine [fcoulnono](#)

8.121.1 Function/Subroutine Documentation

8.121.1.1 subroutine [fcoulnono](#) ()

Definition at line 23 of file [fcoulnono.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fcoulnono
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE neblistarray
00030   USE spinarray
00031   USE virialarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00037   INTEGER :: LBRA, MBRA, LKET, MKET
00038   INTEGER :: PREVJ, NEWJ
00039   INTEGER :: PBCI, PBCJ, PBCK
00040   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00041   REAL(LATTEPREC) :: ALPHA, BETA, PHI, RHO, COSBETA
00042   REAL(LATTEPREC) :: RIJ(3), DC(3)
00043   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2, FTMP(3)
00044   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00045   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00046   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00047   LOGICAL PATH
00048   IF (existererror) RETURN
00049
00050
00051   fscoul = zero
00052   virscoul = zero
00053
00054   !$OMP PARALLEL DO DEFAULT (NONE) &
00055   !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00056   !$OMP SHARED(CR, BOX, BO, RHOU, RHODOWN, SPINON, NOINT, ATELE, ELE1, ELE2) &
00057   !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00058   !$OMP SHARED(HUBBARDU, DELTAQ, COULOMBV) &
00059   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00060   !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00061   !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00062   !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB) &
00063   !$OMP REDUCTION(+:FSCOUL, VIRSCOUL)
00064
00065   DO i = 1, nats
00066
00067     ! Build list of orbitals on atom I
00068     SELECT CASE(basis(elempointer(i)))
00069
00070     CASE("s")
00071       basisi(1) = 0
00072       basisi(2) = -1
00073     CASE("p")
00074       basisi(1) = 1
00075       basisi(2) = -1
00076     CASE("d")
00077       basisi(1) = 2
00078       basisi(2) = -1
00079     CASE("f")
00080       basisi(1) = 3
00081       basisi(2) = -1
00082     CASE("sp")
00083       basisi(1) = 0
00084       basisi(2) = 1
00085       basisi(3) = -1
00086     CASE("sd")
00087       basisi(1) = 0
00088       basisi(2) = 2
00089       basisi(3) = -1
00090     CASE("sf")
00091       basisi(1) = 0
00092       basisi(2) = 3
00093       basisi(3) = -1

```

```

00094      CASE("pd")
00095          basisi(1) = 1
00096          basisi(2) = 2
00097          basisi(3) = -1
00098      CASE("pf")
00099          basisi(1) = 1
00100          basisi(2) = 3
00101          basisi(3) = -1
00102      CASE("df")
00103          basisi(1) = 2
00104          basisi(2) = 3
00105          basisi(3) = -1
00106      CASE("spd")
00107          basisi(1) = 0
00108          basisi(2) = 1
00109          basisi(3) = 2
00110          basisi(4) = -1
00111      CASE("spf")
00112          basisi(1) = 0
00113          basisi(2) = 1
00114          basisi(3) = 3
00115          basisi(4) = -1
00116      CASE("sdf")
00117          basisi(1) = 0
00118          basisi(2) = 2
00119          basisi(3) = 3
00120          basisi(4) = -1
00121      CASE("pdf")
00122          basisi(1) = 1
00123          basisi(2) = 2
00124          basisi(3) = 3
00125          basisi(4) = -1
00126      CASE("spdf")
00127          basisi(1) = 0
00128          basisi(2) = 1
00129          basisi(3) = 2
00130          basisi(4) = 3
00131          basisi(5) = -1
00132      END SELECT
00133
00134      indi = matindlist(i)
00135
00136      DO newj = 1, totnebtb(i)
00137
00138          j = nebtb(1, newj, i)
00139          pbci = nebtb(2, newj, i)
00140          pbcj = nebtb(3, newj, i)
00141          pbck = nebtb(4, newj, i)
00142
00143          rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00144              REAL(pbck)*BOX(3,1) - CR(1,i)
00145
00146          rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00147              REAL(pbck)*BOX(3,2) - CR(2,i)
00148
00149          rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00150              REAL(pbck)*BOX(3,3) - CR(3,i)
00151
00152          magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00153
00154          rcuttb = zero
00155          DO k = 1, noint
00156
00157              IF ( (atele(i) .EQ. ele1(k) .AND. &
00158                  atele(j) .EQ. ele2(k)) .OR. &
00159                  (atele(j) .EQ. ele1(k) .AND. &
00160                  atele(i) .EQ. ele2(k)) ) THEN
00161
00162                  IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00163
00164                  IF (basistype .EQ. "NONORTHO") THEN
00165                      IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00166                  ENDIF
00167
00168              ENDIF
00169
00170          ENDDO
00171
00172          IF (magr2 .LT. rcuttb*rcuttb) THEN
00173
00174              magr = sqrt(magr2)
00175
00176              ! Build list of orbitals on atom J
00177
00178              SELECT CASE(basis(elempointer(j)))
00179              CASE("s")
00180                  basisj(1) = 0

```

```

00181         basisj(2) = -1
00182     CASE("p")
00183         basisj(1) = 1
00184         basisj(2) = -1
00185     CASE("d")
00186         basisj(1) = 2
00187         basisj(2) = -1
00188     CASE("f")
00189         basisj(1) = 3
00190         basisj(2) = -1
00191     CASE("sp")
00192         basisj(1) = 0
00193         basisj(2) = 1
00194         basisj(3) = -1
00195     CASE("sd")
00196         basisj(1) = 0
00197         basisj(2) = 2
00198         basisj(3) = -1
00199     CASE("sf")
00200         basisj(1) = 0
00201         basisj(2) = 3
00202         basisj(3) = -1
00203     CASE("pd")
00204         basisj(1) = 1
00205         basisj(2) = 2
00206         basisj(3) = -1
00207     CASE("pf")
00208         basisj(1) = 1
00209         basisj(2) = 3
00210         basisj(3) = -1
00211     CASE("df")
00212         basisj(1) = 2
00213         basisj(2) = 3
00214         basisj(3) = -1
00215     CASE("spd")
00216         basisj(1) = 0
00217         basisj(2) = 1
00218         basisj(3) = 2
00219         basisj(4) = -1
00220     CASE("spf")
00221         basisj(1) = 0
00222         basisj(2) = 1
00223         basisj(3) = 3
00224         basisj(4) = -1
00225     CASE("sdf")
00226         basisj(1) = 0
00227         basisj(2) = 2
00228         basisj(3) = 3
00229         basisj(4) = -1
00230     CASE("pdf")
00231         basisj(1) = 1
00232         basisj(2) = 2
00233         basisj(3) = 3
00234         basisj(4) = -1
00235     CASE("spdf")
00236         basisj(1) = 0
00237         basisj(2) = 1
00238         basisj(3) = 2
00239         basisj(4) = 3
00240         basisj(5) = -1
00241     END SELECT
00242
00243     indj = matindlist(j)
00244
00245     magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00246     magrp = sqrt(magrp2)
00247
00248     ! transform to system in which z-axis is aligned with RIJ
00249
00250     path = .false.
00251     IF (abs(rij(1)) .GT. 1e-12) THEN
00252         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00253             phi = zero
00254         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00255             phi = two * pi
00256         ELSE
00257             phi = pi
00258         ENDIF
00259         alpha = atan(rij(2) / rij(1)) + phi
00260     ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00261         IF (rij(2) .GT. 1e-12) THEN
00262             alpha = pi / two
00263         ELSE
00264             alpha = three * pi / two
00265         ENDIF
00266     ELSE
00267         ! pathological case: pole in alpha at beta=0

```

```

00268         path = .true.
00269     ENDIF
00270
00271     cosbeta = rij(3)/magr
00272     beta = acos(rij(3) / magr)
00273
00274     dc = rij/magr
00275
00276     ! build forces using PRB 72 165107 eq. (12) - the sign of the
00277     ! dfda contribution seems to be wrong, but gives the right
00278     ! answer(?)
00279
00280     ftmp = zero
00281     k = indi
00282
00283     lbrainc = 1
00284     DO WHILE (basisi(lbrainc) .NE. -1)
00285
00286         lbra = basisi(lbrainc)
00287         lbrainc = lbrainc + 1
00288
00289         DO mbra = -lbra, lbra
00290
00291             k = k + 1
00292             l = indj
00293
00294             lketinc = 1
00295             DO WHILE (basisj(lketinc) .NE. -1)
00296
00297                 lket = basisj(lketinc)
00298                 lketinc = lketinc + 1
00299
00300                 DO mket = -lket, lket
00301
00302                     l = l + 1
00303
00304                     SELECT CASE(spinon)
00305                     CASE(0)
00306                         rho = bo(l, k)
00307                     CASE(1)
00308                         rho = rhoup(l, k) + rhodown(l, k)
00309                     END SELECT
00310
00311                     IF (.NOT. path) THEN
00312
00313                         ! Unroll loops and pre-compute
00314
00315                         mydfda = dfda(i, j, lbra, lket, mbra, &
00316                             mket, magr, alpha, cosbeta, "S")
00317
00318                         mydfdb = dfdb(i, j, lbra, lket, mbra, &
00319                             mket, magr, alpha, cosbeta, "S")
00320
00321                         mydfdr = dfdr(i, j, lbra, lket, mbra, &
00322                             mket, magr, alpha, cosbeta, "S")
00323
00324                         !
00325                         ! d/d_alpha
00326                         !
00327
00328                         ftmp(1) = ftmp(1) + rho * &
00329                             (-rij(2) / magrp2 * mydfda)
00330
00331                         ftmp(2) = ftmp(2) + rho * &
00332                             (rij(1)/ magrp2 * mydfda)
00333
00334                         !
00335                         ! d/d_beta
00336                         !
00337
00338                         ftmp(1) = ftmp(1) + rho * &
00339                             (((rij(3) * rij(1)) / &
00340                                 magr2)) / magrp) * mydfdb)
00341
00342                         ftmp(2) = ftmp(2) + rho * &
00343                             (((rij(3) * rij(2)) / &
00344                                 magr2)) / magrp) * mydfdb)
00345
00346                         ftmp(3) = ftmp(3) - rho * &
00347                             ((one - (rij(3) * rij(3)) / &
00348                                 magr2)) / magrp) * mydfdb)
00349
00350                         !
00351                         ! d/dR
00352                         !
00353
00354                         ftmp(1) = ftmp(1) - rho * dc(1) * &

```

```

00355             mydfdr
00356
00357             ftmp(2) = ftmp(2) - rho * dc(2) * &
00358             mydfdr
00359
00360             ftmp(3) = ftmp(3) - rho * dc(3) * &
00361             mydfdr
00362
00363
00364             ELSE
00365
00366             ! pathological configuration in which beta=0
00367             ! or pi => alpha undefined
00368
00369             ! fixed: MJC 12/17/13
00370
00371             mydfdb = dfdb(i, j, lbra, lket, &
00372             mbra, mket, magr, zero, cosbeta, "S") / magr
00373
00374             ftmp(1) = ftmp(1) - rho * (cosbeta * mydfdb)
00375
00376             mydfdb = dfdb(i, j, lbra, lket, &
00377             mbra, mket, magr, pi/two, cosbeta, "S") / magr
00378
00379             ftmp(2) = ftmp(2) - rho * (cosbeta * mydfdb)
00380
00381             mydfdr = dfdr(i, j, lbra, lket, mbra, &
00382             mket, magr, zero, cosbeta, "S")
00383
00384             ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00385
00386             ENDIF
00387
00388             ENDDO
00389             ENDDO
00390             ENDDO
00391             ENDDO
00392
00393             ftmp = ftmp * ( hubbardu(elempointer(j))*deltaq(j) +
00394 coulombv(j) &
00395             +hubbardu(elempointer(i))*deltaq(i) +
00396 coulombv(i))
00397
00398             fscoul(1,i) = fscoul(1,i) + ftmp(1)
00399             fscoul(2,i) = fscoul(2,i) + ftmp(2)
00400             fscoul(3,i) = fscoul(3,i) + ftmp(3)
00401
00402             ! with the factor of 2...
00403
00404             virscoul(1) = virscoul(1) + rij(1)*ftmp(1)
00405             virscoul(2) = virscoul(2) + rij(2)*ftmp(2)
00406             virscoul(3) = virscoul(3) + rij(3)*ftmp(3)
00407             virscoul(4) = virscoul(4) + rij(1)*ftmp(2)
00408             virscoul(5) = virscoul(5) + rij(2)*ftmp(3)
00409             virscoul(6) = virscoul(6) + rij(3)*ftmp(1)
00410
00411             ENDIF
00412             ENDDO
00413
00414             ENDDO
00415
00416             !$OMP END PARALLEL DO
00417
00418             virscoul = virscoul/two
00419
00420             ! DO I = 1, NATS
00421             ! WRITE(6,10) I, FSCOU(1,I), FSCOU(2,I), FSCOU(3,I)
00422             ! ENDDO
00423
00424             !10 FORMAT(I4, 3F12.6)
00425
00426             RETURN
00427
00428 END SUBROUTINE fcoulnono

```

8.123 fcoulnono_sp.f90 File Reference

Functions/Subroutines

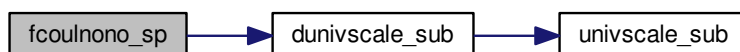
- subroutine [fcoulnono_sp](#)

8.123.1 Function/Subroutine Documentation

8.123.1.1 subroutine `fcoulnono_sp ()`

Definition at line 23 of file [fcoulnono_sp.f90](#).

Here is the call graph for this function:



[illegible]

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!  
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !  
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !  
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !  
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !  
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !  
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !  
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !  
00009 ! SOFTWARE. If software is modified to produce derivative works, such !  
00010 ! modified software should be clearly marked, so as not to confuse it !  
00011 ! with the version available from LANL. !  
00012 ! !  
00013 ! Additionally, this program is free software; you can redistribute it !  
00014 ! and/or modify it under the terms of the GNU General Public License as
```

```

00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fcoulnono_sp
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE neblistarray
00030   USE spinarray
00031   USE virialarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, KK, INDI, INDJ
00037   INTEGER :: NEWJ
00038   INTEGER :: PBCI, PBCJ, PBCK
00039   REAL(LATTEPREC) :: HSSS, HSPS, HPSS, HPPS, HPPP
00040   REAL(LATTEPREC) :: RIJ(3), DC(3)
00041   REAL(LATTEPREC) :: L, M, N, L2, M2, N2, LM, LN, MN, LMN
00042   REAL(LATTEPREC) :: DSSSDR(3), DSPSDR(3), DPSSDR(3), DPPSDR(3), DPPPDR(3)
00043   REAL(LATTEPREC) :: MAGR, INVR, FTMP(3)
00044   REAL(LATTEPREC) :: PPSMPPP, PPSUBINVR
00045   REAL(LATTEPREC) :: VIRTMP(6), CHECKP
00046   CHARACTER(LEN=2) :: BASISI, BASISJ
00047   IF (existerror) RETURN
00048
00049   fscoul = zero
00050   virscoul = zero
00051
00052   !
00053   ! We are computing the contribution to the forces from the
00054   ! S dependence of the Mulliken partial charges when we use
00055   ! the non-orthogonal basis
00056   !
00057   ! F_scoul_k = Sum_i dq_i/dR_k [ U q_i + Sum_{j != i} q_j gamma_ij ]
00058   !
00059
00060   !
00061   ! q_i = Z_i - 2*Sum_{a = orbitals belonging to i} (rho S)_ia, ia
00062
00063   ! We need first dq_i/dR_k:
00064   !
00065   ! i != k    -2 Sum_{a, b} rho_{ia,kb} dS_{kb,ia} /dR_k
00066   !
00067   ! i == k    -2 Sum_{b, j c} rho_{kb,jc} dS_{jc, kb} / dR_k
00068
00069   !$OMP PARALLEL DO DEFAULT (NONE) &
00070   !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00071   !$OMP SHARED(CR, BOX, BO, RHOUP, RHODOWN, NOINT, ATELE, ELE1, ELE2) &
00072   !$OMP SHARED(BOND, OVERL, MATINDLIST, BTYPE, BASISTYPE, SPINON) &
00073   !$OMP SHARED(HUBBARDU, COULOMBV, DELTAQ)&
00074   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00075   !$OMP PRIVATE(RIJ, DC, MAGR, INVR, FTMP) &
00076   !$OMP PRIVATE(DSSSDR, DSPSDR, DPSSDR, DPPSDR, DPPPDR, PPSMPPP, PPSUBINVR)&
00077   !$OMP PRIVATE( L, M, N, L2, M2, N2, LM, LN, MN, LMN) &
00078   !$OMP PRIVATE(HSSS, HSPS, HPSS, HPPS, HPPP)&
00079   !$OMP REDUCTION(+:FSCOUL, VIRSCOUL)
00080
00081   DO i = 1, nats
00082
00083     basisi = basis(elempointer(i))
00084     indi = matindlist(i)
00085
00086     ! Loop over all neighbors of I
00087
00088     DO newj = 1, totnebtb(i)
00089
00090       j = nebtb(1, newj, i)
00091       pbcI = nebtb(2, newj, i)
00092       pbcj = nebtb(3, newj, i)
00093       pbck = nebtb(4, newj, i)
00094
00095       indj = matindlist(j)
00096       basisj = basis(elempointer(j))
00097
00098       rij(1) = cr(1,j) + REAL(pbcI)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00099         REAL(pbck)*BOX(3,1) - CR(1,i)
00100
00101       rij(2) = cr(2,j) + REAL(pbcI)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &

```

```

00102      REAL(pbck)*BOX(3,2) - CR(2,i)
00103
00104      rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00105      REAL(pbck)*BOX(3,3) - CR(3,i)
00106
00107
00108      magr = sqrt(rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3))
00109
00110      invr = one/magr
00111
00112      ftmp = zero
00113
00114      !
00115      ! Direction cosines (DC)
00116      !
00117
00118      dc = rij/magr
00119
00120      l = dc(1)
00121      m = dc(2)
00122      n = dc(3)
00123
00124      ! Let's compute rho * dS/dR
00125
00126      IF (basisi .EQ. "s") THEN
00127
00128          IF (basisj .EQ. "s") THEN
00129
00130              DO k = 1, noint
00131                  IF ((atele(i) .EQ. ele1(k) .AND. &
00132                     atele(j) .EQ. ele2(k)) .OR. &
00133                     (atele(i) .EQ. ele2(k) .AND. &
00134                     atele(j) .EQ. ele1(k))) THEN
00135
00136                      IF (btype(k) .EQ. "sss") THEN
00137
00138                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssdr)
00139
00140                      ENDIF
00141
00142                  ENDIF
00143              ENDDO
00144
00145              IF (spinon .EQ. 0) THEN
00146
00147                  ftmp = ftmp - bo(indi+1, indj+1)* dssdr
00148
00149              ELSE
00150
00151                  ftmp = ftmp - dssdr*(rhoup(indi+1, indj+1) + &
00152                  rhodown(indi+1, indj+1))
00153
00154              ENDIF
00155
00156
00157              !FTMP = FTMP - BO(INDI+1, INDJ+1)* DSSDR
00158
00159          ELSEIF (basisj .EQ. "sp") THEN
00160
00161              DO k = 1, noint
00162
00163                  IF ((atele(i) .EQ. ele1(k) .AND. &
00164                     atele(j) .EQ. ele2(k)) .OR. &
00165                     (atele(i) .EQ. ele2(k) .AND. &
00166                     atele(j) .EQ. ele1(k))) THEN
00167
00168                      IF (btype(k) .EQ. "sss") THEN
00169
00170                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssdr)
00171
00172                      ELSEIF (btype(k) .EQ. "sps") THEN
00173
00174                          CALL dunivscale_sub(magr, overl(:,k), dc, hsps, dpsdr)
00175
00176                      ENDIF
00177                  ENDIF
00178              ENDDO
00179
00180              l2 = l*l
00181              m2 = m*m
00182              n2 = n*n
00183              lm = l*m
00184              ln = l*n
00185              mn = m*n
00186
00187              IF (spinon .EQ. 0) THEN
00188

```

```

00189         ! E_s1,s2
00190
00191         ftmp = ftmp - bo(indi+1, indj+1)*dssedr
00192
00193         ! E_s1,x2
00194
00195         ftmp(1) = ftmp(1) - bo(indi+1, indj+2) * &
00196             (l*dspedr(1) + (l2 - one)*invr*hsps)
00197
00198         ftmp(2) = ftmp(2) - bo(indi+1, indj+2) * &
00199             (l*dspedr(2) + lm*invr*hsps)
00200
00201         ftmp(3) = ftmp(3) - bo(indi+1, indj+2) * &
00202             (l*dspedr(3) + ln*invr*hsps)
00203
00204         ! E_s1,y2
00205
00206         ftmp(1) = ftmp(1) - bo(indi+1, indj+3) * &
00207             (m*dspedr(1) + lm*invr*hsps)
00208
00209         ftmp(2) = ftmp(2) - bo(indi+1, indj+3) * &
00210             (m*dspedr(2) + (m2 - one)*invr*hsps)
00211
00212         ftmp(3) = ftmp(3) - bo(indi+1, indj+3) * &
00213             (m*dspedr(3) + mn*invr*hsps)
00214
00215         ! E_s1,z2
00216
00217         ftmp(1) = ftmp(1) - bo(indi+1, indj+4) * &
00218             (n*dspedr(1) + ln*invr*hsps)
00219
00220         ftmp(2) = ftmp(2) - bo(indi+1, indj+4) * &
00221             (n*dspedr(2) + mn*invr*hsps)
00222
00223         ftmp(3) = ftmp(3) - bo(indi+1, indj+4) * &
00224             (n*dspedr(3) + (n2 - one)*invr*hsps)
00225
00226     ELSE
00227
00228         ! E_s1,s2
00229
00230         ftmp = ftmp - dssedr*(rhoup(indi+1, indj+1) + &
00231             rhodown(indi+1, indj+1))
00232
00233         ! E_s1,x2
00234
00235         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+2) + &
00236             rhodown(indi+1, indj+2)) * &
00237             (l*dspedr(1) + (l2 - one)*invr*hsps)
00238
00239         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+2) + &
00240             rhodown(indi+1, indj+2)) * &
00241             (l*dspedr(2) + lm*invr*hsps)
00242
00243         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+2) + &
00244             rhodown(indi+1, indj+2)) * &
00245             (l*dspedr(3) + ln*invr*hsps)
00246
00247         ! E_s1,y2
00248
00249         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+3) + &
00250             rhodown(indi+1, indj+3)) * &
00251             (m*dspedr(1) + lm*invr*hsps)
00252
00253         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+3) + &
00254             rhodown(indi+1, indj+3)) * &
00255             (m*dspedr(2) + (m2 - one)*invr*hsps)
00256
00257         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+3) + &
00258             rhodown(indi+1, indj+3)) * &
00259             (m*dspedr(3) + mn*invr*hsps)
00260
00261         ! E_s1,z2
00262
00263         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+4) + &
00264             rhodown(indi+1, indj+4)) * &
00265             (n*dspedr(1) + ln*invr*hsps)
00266
00267         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+4) + &
00268             rhodown(indi+1, indj+4)) * &
00269             (n*dspedr(2) + mn*invr*hsps)
00270
00271         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+4) + &
00272             rhodown(indi+1, indj+4)) * &
00273             (n*dspedr(3) + (n2 - one)*invr*hsps)
00274
00275     ENDIF

```

```

00276
00277         ENDIF
00278
00279     ELSEIF (basisi .EQ. "sp") THEN
00280
00281         IF (basisj .EQ. "s") THEN
00282
00283             DO k = 1, noint
00284
00285                 IF ((atele(i) .EQ. ele1(k) .AND. &
00286                     atele(j) .EQ. ele2(k)) .OR. &
00287                     (atele(i) .EQ. ele2(k) .AND. &
00288                     atele(j) .EQ. ele1(k))) THEN
00289
00290                     IF (btype(k) .EQ. "sss") THEN
00291
00292                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00293
00294                     ELSEIF (btype(k) .EQ. "sps") THEN
00295
00296                         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpssdr)
00297
00298                         hpss = -hpss
00299                         dpssdr = -dpssdr
00300
00301                     ENDIF
00302                 ENDIF
00303             ENDDO
00304
00305             l2 = l*l
00306             m2 = m*m
00307             n2 = n*n
00308             lm = l*m
00309             ln = l*n
00310             mn = m*n
00311
00312             IF (spinon .EQ. 0) THEN
00313
00314                 ! E_s1,s2
00315
00316                 ftmp = ftmp - dssssdr*bo(indi+1, indj+1)
00317
00318                 ! E_x1,s2
00319
00320                 ftmp(1) = ftmp(1) - bo(indi+2, indj+1) * &
00321                     (l*dpssdr(1) + (l2 - one)*invr*hpss)
00322
00323                 ftmp(2) = ftmp(2) - bo(indi+2, indj+1) * &
00324                     (l*dpssdr(2) + lm*invr*hpss)
00325
00326                 ftmp(3) = ftmp(3) - bo(indi+2, indj+1) * &
00327                     (l*dpssdr(3) + ln*invr*hpss)
00328
00329                 ! E_y1,s2
00330
00331                 ftmp(1) = ftmp(1) - bo(indi+3, indj+1) * &
00332                     (m*dpssdr(1) + lm*invr*hpss)
00333
00334                 ftmp(2) = ftmp(2) - bo(indi+3, indj+1) * &
00335                     (m*dpssdr(2) + (m2 - one)*invr*hpss)
00336
00337                 ftmp(3) = ftmp(3) - bo(indi+3, indj+1) * &
00338                     (m*dpssdr(3) + mn*invr*hpss)
00339
00340                 ! E_z1,s2
00341
00342                 ftmp(1) = ftmp(1) - bo(indi+4, indj+1) * &
00343                     (n*dpssdr(1) + ln*invr*hpss)
00344
00345                 ftmp(2) = ftmp(2) - bo(indi+4, indj+1) * &
00346                     (n*dpssdr(2) + mn*invr*hpss)
00347
00348                 ftmp(3) = ftmp(3) - bo(indi+4, indj+1) * &
00349                     (n*dpssdr(3) + (n2 - one)*invr*hpss)
00350
00351             ELSE
00352
00353                 ! E_s1,s2
00354
00355                 ftmp = ftmp - dssssdr*(rhoup(indi+1, indj+1) + &
00356                     rhodown(indi+1, indj+1))
00357
00358                 ! E_x1,s2
00359
00360                 ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+1) + &
00361                     rhodown(indi+2, indj+1)) * &
00362                     (l*dpssdr(1) + (l2 - one)*invr*hpss)

```

```

00363      ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+1) + &
00364      rhodown(indi+2, indj+1)) * &
00365      (l*dpssdr(2) + lm*invr*hpss)
00366
00367      ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+1) + &
00368      rhodown(indi+2, indj+1)) * &
00369      (l*dpssdr(3) + ln*invr*hpss)
00370
00371      ! E_y1,s2
00372
00373      ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+1) + &
00374      rhodown(indi+3, indj+1)) * &
00375      (m*dpssdr(1) + lm*invr*hpss)
00376
00377      ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+1) + &
00378      rhodown(indi+3, indj+1)) * &
00379      (m*dpssdr(2) + (m2 - one)*invr*hpss)
00380
00381      ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+1) + &
00382      rhodown(indi+3, indj+1)) * &
00383      (m*dpssdr(3) + mn*invr*hpss)
00384
00385      ! E_z1,s2
00386
00387      ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+1) + &
00388      rhodown(indi+4, indj+1)) * &
00389      (n*dpssdr(1) + ln*invr*hpss)
00390
00391      ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+1) + &
00392      rhodown(indi+4, indj+1)) * &
00393      (n*dpssdr(2) + mn*invr*hpss)
00394
00395      ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+1) + &
00396      rhodown(indi+4, indj+1)) * &
00397      (n*dpssdr(3) + (n2 - one)*invr*hpss)
00398
00399      ENDIF
00400
00401      ELSEIF (basisj .EQ. "sp") THEN
00402
00403      IF (atele(i) .EQ. atele(j)) THEN
00404
00405      DO k = 1, noint
00406
00407      IF (atele(i) .EQ. ele1(k) .AND. &
00408      atele(j) .EQ. ele2(k)) THEN
00409
00410      IF (btype(k) .EQ. "sss") THEN
00411
00412      CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00413
00414      ELSEIF (btype(k) .EQ. "sps") THEN
00415
00416      CALL dunivscale_sub(magr, overl(:,k), dc, hsps, dpspsdr)
00417
00418      dpssdr = -dpspsdr
00419
00420      hpss = -hsps
00421
00422      ELSEIF (btype(k) .EQ. "pps") THEN
00423
00424      CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dpppsdr)
00425
00426      ELSEIF (btype(k) .EQ. "ppp") THEN
00427
00428      CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dppppdr)
00429
00430      ENDIF
00431      ENDIF
00432      ENDDO
00433
00434      ELSEIF (atele(i) .NE. atele(j)) THEN
00435
00436      DO k = 1, noint
00437
00438      IF (atele(i) .EQ. ele1(k) .AND. &
00439      atele(j) .EQ. ele2(k)) THEN
00440
00441      IF (btype(k) .EQ. "sss") THEN
00442
00443      CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00444
00445      ELSEIF (btype(k) .EQ. "sps") THEN
00446
00447      CALL dunivscale_sub(magr, overl(:,k), dc, hsps, dpspsdr)
00448
00449

```

```

00450 ELSEIF (btype(k) .EQ. "pps") THEN
00451
00452     CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dpssdr)
00453
00454 ELSEIF (btype(k) .EQ. "ppp") THEN
00455
00456     CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00457
00458 ENDIF
00459
00460 ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00461     atele(j) .EQ. ele1(k)) THEN
00462
00463     IF (btype(k) .EQ. "sss") THEN
00464
00465         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00466
00467     ELSEIF (btype(k) .EQ. "sps") THEN
00468
00469         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpssdr)
00470
00471         dpssdr = -dpssdr
00472         hpss = -hpss
00473
00474     ELSEIF (btype(k) .EQ. "pps") THEN
00475
00476         CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dpssdr)
00477
00478     ELSEIF (btype(k) .EQ. "ppp") THEN
00479
00480         CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00481
00482     ENDIF
00483
00484 ENDIF
00485 ENDDO
00486
00487 ENDIF
00488
00489 ppsmppp = hpps - hppp
00490 ppsubinvr = ppsmppp * invr
00491
00492 l2 = l*l
00493 m2 = m*m
00494 n2 = n*n
00495 lm = l*m
00496 ln = l*n
00497 mn = m*n
00498 lmn = l*m*n
00499
00500 IF (spinon .EQ. 0) THEN
00501
00502     ! E_s1,s2
00503
00504     ftmp = ftmp - dssssdr*bo(indi+1, indj+1)
00505
00506     ! E_s1,x2
00507
00508     ftmp(1) = ftmp(1) - bo(indi+1, indj+2) * &
00509         (l*dspssdr(1) + (l2 - one)*invr*hsps)
00510
00511     ftmp(2) = ftmp(2) - bo(indi+1, indj+2) * &
00512         (l*dspssdr(2) + lm*invr*hsps)
00513
00514     ftmp(3) = ftmp(3) - bo(indi+1, indj+2) * &
00515         (l*dspssdr(3) + ln*invr*hsps)
00516
00517     ! E_s1,y2
00518
00519     ftmp(1) = ftmp(1) - bo(indi+1, indj+3) * &
00520         (m*dspssdr(1) + lm*invr*hsps)
00521
00522     ftmp(2) = ftmp(2) - bo(indi+1, indj+3) * &
00523         (m*dspssdr(2) + (m2 - one)*invr*hsps)
00524
00525     ftmp(3) = ftmp(3) - bo(indi+1, indj+3) * &
00526         (m*dspssdr(3) + mn*invr*hsps)
00527
00528     ! E_s1,z2
00529
00530     ftmp(1) = ftmp(1) - bo(indi+1, indj+4) * &
00531         (n*dspssdr(1) + ln*invr*hsps)
00532
00533     ftmp(2) = ftmp(2) - bo(indi+1, indj+4) * &
00534         (n*dspssdr(2) + mn*invr*hsps)
00535
00536     ftmp(3) = ftmp(3) - bo(indi+1, indj+4) * &

```

```

00537         (n*dpsdr(3) + (n2 - one)*invr*hsps)
00538
00539     ! E_x1,s2
00540
00541     ftmp(1) = ftmp(1) - bo(indi+2, indj+1) * &
00542         (1*dpsdr(1) + (l2 - one)*invr*hpss)
00543
00544     ftmp(2) = ftmp(2) - bo(indi+2, indj+1) * &
00545         (1*dpsdr(2) + lm*invr*hpss)
00546
00547     ftmp(3) = ftmp(3) - bo(indi+2, indj+1) * &
00548         (1*dpsdr(3) + ln*invr*hpss)
00549
00550     ! E_x1,x2
00551
00552     ftmp(1) = ftmp(1) - bo(indi+2, indj+2) * &
00553         (l2*dpsdr(1) + (one - l2)*dppdr(1) + &
00554         two*1*(l2 - one)*ppsubinvr)
00555
00556     ftmp(2) = ftmp(2) - bo(indi+2, indj+2) * &
00557         (l2*dpsdr(2) + (one - l2)*dppdr(2) + &
00558         two*l2*m*ppsubinvr)
00559
00560     ftmp(3) = ftmp(3) - bo(indi+2, indj+2) * &
00561         (l2*dpsdr(3) + (one - l2)*dppdr(3) + &
00562         two*l2*n*ppsubinvr)
00563
00564     ! E_x1,y2
00565
00566     ftmp(1) = ftmp(1) - bo(indi+2, indj+3) * &
00567         (lm*(dpsdr(1) - dppdr(1)) + &
00568         m*(two*l2 - one)*ppsubinvr)
00569
00570     ftmp(2) = ftmp(2) - bo(indi+2, indj+3) * &
00571         (lm*(dpsdr(2) - dppdr(2)) + &
00572         l*(two*m2 - one)*ppsubinvr)
00573
00574     ftmp(3) = ftmp(3) - bo(indi+2, indj+3) * &
00575         (lm*(dpsdr(3) - dppdr(3)) + &
00576         two*lmn*ppsubinvr)
00577
00578     ! E_x1,z2
00579
00580     ftmp(1) = ftmp(1) - bo(indi+2, indj+4) * &
00581         (ln*(dpsdr(1) - dppdr(1)) + &
00582         n*(two*l2 - one)*ppsubinvr)
00583
00584     ftmp(2) = ftmp(2) - bo(indi+2, indj+4) * &
00585         (ln*(dpsdr(2) - dppdr(2)) + &
00586         two*lmn*ppsubinvr)
00587
00588     ftmp(3) = ftmp(3) - bo(indi+2, indj+4) * &
00589         (ln*(dpsdr(3) - dppdr(3)) + &
00590         l*(two*n2 - one)*ppsubinvr)
00591
00592     ! E_y1,s2
00593
00594     ftmp(1) = ftmp(1) - bo(indi+3, indj+1) * &
00595         (m*dpsdr(1) + lm*invr*hpss)
00596
00597     ftmp(2) = ftmp(2) - bo(indi+3, indj+1) * &
00598         (m*dpsdr(2) + (m2 - one)*invr*hpss)
00599
00600     ftmp(3) = ftmp(3) - bo(indi+3, indj+1) * &
00601         (m*dpsdr(3) + mn*invr*hpss)
00602
00603     ! E_y1,x2
00604
00605     ftmp(1) = ftmp(1) - bo(indi+3, indj+2) * &
00606         (lm*(dpsdr(1) - dppdr(1)) + &
00607         m*(two*l2 - one)*ppsubinvr)
00608
00609     ftmp(2) = ftmp(2) - bo(indi+3, indj+2) * &
00610         (lm*(dpsdr(2) - dppdr(2)) + &
00611         l*(two*m2 - one)*ppsubinvr)
00612
00613     ftmp(3) = ftmp(3) - bo(indi+3, indj+2) * &
00614         (lm*(dpsdr(3) - dppdr(3)) + &
00615         two*lmn*ppsubinvr)
00616
00617     ! E_y1,y2
00618
00619     ftmp(1) = ftmp(1) - bo(indi+3, indj+3) * &
00620         (m2*dpsdr(1) + (one - m2)*dppdr(1) + &
00621         two*1*m2*ppsubinvr)
00622
00623     ftmp(2) = ftmp(2) - bo(indi+3, indj+3) * &

```



```

00624          (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00625          two*m*(m2 - one)*ppsbinvr)
00626
00627      ftmp(3) = ftmp(3) - bo(indi+3, indj+3) * &
00628          (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00629          two*n*m2*ppsbinvr)
00630
00631      ! E_y1,z2
00632
00633      ftmp(1) = ftmp(1) - bo(indi+3, indj+4) * &
00634          (mn*(dppsdr(1) - dpppdr(1)) + &
00635          two*lmn*ppsbinvr)
00636
00637      ftmp(2) = ftmp(2) - bo(indi+3, indj+4) * &
00638          (mn*(dppsdr(2) - dpppdr(2)) + &
00639          n*(two*m2 - one)*ppsbinvr)
00640
00641      ftmp(3) = ftmp(3) - bo(indi+3, indj+4) * &
00642          (mn*(dppsdr(3) - dpppdr(3)) + &
00643          m*(two*n2 - one)*ppsbinvr)
00644
00645      ! E_z1,s2
00646
00647      ftmp(1) = ftmp(1) - bo(indi+4, indj+1) * &
00648          (n*dpssdr(1) + ln*invr*hpss)
00649
00650      ftmp(2) = ftmp(2) - bo(indi+4, indj+1) * &
00651          (n*dpssdr(2) + mn*invr*hpss)
00652
00653      ftmp(3) = ftmp(3) - bo(indi+4, indj+1) * &
00654          (n*dpssdr(3) + (n2 - one)*invr*hpss)
00655
00656      ! E_z1,x2
00657
00658      ftmp(1) = ftmp(1) - bo(indi+4, indj+2) * &
00659          (ln*(dppsdr(1) - dpppdr(1)) + &
00660          n*(two*l2 - one)*ppsbinvr)
00661
00662      ftmp(2) = ftmp(2) - bo(indi+4, indj+2) * &
00663          (ln*(dppsdr(2) - dpppdr(2)) + &
00664          two*lmn*ppsbinvr)
00665
00666      ftmp(3) = ftmp(3) - bo(indi+4, indj+2) * &
00667          (ln*(dppsdr(3) - dpppdr(3)) + &
00668          l*(two*n2 - one)*ppsbinvr)
00669
00670      ! E_z1,y2
00671
00672      ftmp(1) = ftmp(1) - bo(indi+4, indj+3) * &
00673          (mn*(dppsdr(1) - dpppdr(1)) + &
00674          two*lmn*ppsbinvr)
00675
00676      ftmp(2) = ftmp(2) - bo(indi+4, indj+3) * &
00677          (mn*(dppsdr(2) - dpppdr(2)) + &
00678          n*(two*m2 - one)*ppsbinvr)
00679
00680      ftmp(3) = ftmp(3) - bo(indi+4, indj+3) * &
00681          (mn*(dppsdr(3) - dpppdr(3)) + &
00682          m*(two*n2 - one)*ppsbinvr)
00683
00684      ! E_z1,z2
00685
00686      ftmp(1) = ftmp(1) - bo(indi+4, indj+4) * &
00687          (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00688          two*l*n2*ppsbinvr)
00689
00690      ftmp(2) = ftmp(2) - bo(indi+4, indj+4) * &
00691          (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00692          two*m*n2*ppsbinvr)
00693
00694      ftmp(3) = ftmp(3) - bo(indi+4, indj+4) * &
00695          (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00696          two*n*(n2 - one)*ppsbinvr)
00697
00698      ELSE ! SPIN-POLARIZED CALCULATION
00699
00700      ! E_s1,s2
00701
00702      ftmp(1) = ftmp(1) - dssdr(1)*(rhoup(indi+1, indj+1) + &
00703          rhodown(indi+1, indj+1))
00704
00705      ftmp(2) = ftmp(2) - dssdr(2)*(rhoup(indi+1, indj+1) + &
00706          rhodown(indi+1, indj+1))
00707
00708      ftmp(3) = ftmp(3) - dssdr(3)*(rhoup(indi+1, indj+1) + &
00709          rhodown(indi+1, indj+1))
00710

```

```

00711         ! E_s1,x2
00712
00713         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+2) + &
00714             rhodown(indi+1, indj+2)) * &
00715             (l*dspdr(1) + (l2 - one)*invr*hsps)
00716
00717         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+2) + &
00718             rhodown(indi+1, indj+2)) * &
00719             (l*dspdr(2) + lm*invr*hsps)
00720
00721         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+2) + &
00722             rhodown(indi+1, indj+2)) * &
00723             (l*dspdr(3) + ln*invr*hsps)
00724
00725         ! E_s1,y2
00726
00727         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+3) + &
00728             rhodown(indi+1, indj+3)) * &
00729             (m*dspdr(1) + lm*invr*hsps)
00730
00731         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+3) + &
00732             rhodown(indi+1, indj+3)) * &
00733             (m*dspdr(2) + (m2 - one)*invr*hsps)
00734
00735         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+3) + &
00736             rhodown(indi+1, indj+3)) * &
00737             (m*dspdr(3) + mn*invr*hsps)
00738
00739         ! E_s1,z2
00740
00741         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+4) + &
00742             rhodown(indi+1, indj+4)) * &
00743             (n*dspdr(1) + ln*invr*hsps)
00744
00745         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+4) + &
00746             rhodown(indi+1, indj+4)) * &
00747             (n*dspdr(2) + mn*invr*hsps)
00748
00749         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+4) + &
00750             rhodown(indi+1, indj+4)) * &
00751             (n*dspdr(3) + (n2 - one)*invr*hsps)
00752
00753         ! E_x1,s2
00754
00755         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+1) + &
00756             rhodown(indi+2, indj+1)) * &
00757             (l*dpssdr(1) + (l2 - one)*invr*hpss)
00758
00759         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+1) + &
00760             rhodown(indi+2, indj+1)) * &
00761             (l*dpssdr(2) + lm*invr*hpss)
00762
00763         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+1) + &
00764             rhodown(indi+2, indj+1)) * &
00765             (l*dpssdr(3) + ln*invr*hpss)
00766
00767         ! E_x1,x2
00768
00769         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+2) + &
00770             rhodown(indi+2, indj+2)) * &
00771             (l2*dppsdr(1) + (one - l2)*dppdr(1) + &
00772             two*l*(l2 - one)*ppsubinvr)
00773
00774         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+2) + &
00775             rhodown(indi+2, indj+2)) * &
00776             (l2*dppsdr(2) + (one - l2)*dppdr(2) + &
00777             two*l2*m*ppsubinvr)
00778
00779         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+2) + &
00780             rhodown(indi+2, indj+2)) * &
00781             (l2*dppsdr(3) + (one - l2)*dppdr(3) + &
00782             two*l2*n*ppsubinvr)
00783
00784         ! E_x1,y2
00785
00786         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+3) + &
00787             rhodown(indi+2, indj+3)) * &
00788             (lm*(dppsdr(1) - dppdr(1)) + &
00789             m*(two*l2 - one)*ppsubinvr)
00790
00791         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+3) + &
00792             rhodown(indi+2, indj+3)) * &
00793             (lm*(dppsdr(2) - dppdr(2)) + &
00794             l*(two*m2 - one)*ppsubinvr)
00795
00796         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+3) + &
00797             rhodown(indi+2, indj+3)) * &

```

```

00798          (lm*(dppsdr(3) - dpppdr(3)) + &
00799          two*lmn*ppsubinvr)
00800
00801      ! E_x1,z2
00802
00803      ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+4) + &
00804      rhodown(indi+2, indj+4)) * &
00805      (ln*(dppsdr(1) - dpppdr(1)) + &
00806      n*(two*12 - one)*ppsubinvr)
00807
00808      ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+4) + &
00809      rhodown(indi+2, indj+4)) * &
00810      (ln*(dppsdr(2) - dpppdr(2)) + &
00811      two*lmn*ppsubinvr)
00812
00813      ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+4) + &
00814      rhodown(indi+2, indj+4)) * &
00815      (ln*(dppsdr(3) - dpppdr(3)) + &
00816      1*(two*n2 - one)*ppsubinvr)
00817
00818      ! E_y1,s2
00819
00820      ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+1) + &
00821      rhodown(indi+3, indj+1)) * &
00822      (m*dppsdr(1) + lm*invr*hpss)
00823
00824      ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+1) + &
00825      rhodown(indi+3, indj+1)) * &
00826      (m*dppsdr(2) + (m2 - one)*invr*hpss)
00827
00828      ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+1) + &
00829      rhodown(indi+3, indj+1)) * &
00830      (m*dppsdr(3) + mn*invr*hpss)
00831
00832      ! E_y1,x2
00833
00834      ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+2) + &
00835      rhodown(indi+3, indj+2)) * &
00836      (lm*(dppsdr(1) - dpppdr(1)) + &
00837      m*(two*12 - one)*ppsubinvr)
00838
00839      ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+2) + &
00840      rhodown(indi+3, indj+2)) * &
00841      (lm*(dppsdr(2) - dpppdr(2)) + &
00842      1*(two*m2 - one)*ppsubinvr)
00843
00844      ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+2) + &
00845      rhodown(indi+3, indj+2)) * &
00846      (lm*(dppsdr(3) - dpppdr(3)) + &
00847      two*lmn*ppsubinvr)
00848
00849      ! E_y1,y2
00850
00851      ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+3) + &
00852      rhodown(indi+3, indj+3)) * &
00853      (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00854      two*1*m2*ppsubinvr)
00855
00856      ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+3) + &
00857      rhodown(indi+3, indj+3)) * &
00858      (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00859      two*m*(m2 - one)*ppsubinvr)
00860
00861      ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+3) + &
00862      rhodown(indi+3, indj+3)) * &
00863      (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00864      two*n*m2*ppsubinvr)
00865
00866      ! E_y1,z2
00867
00868      ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+4) + &
00869      rhodown(indi+3, indj+4)) * &
00870      (mn*(dppsdr(1) - dpppdr(1)) + &
00871      two*lmn*ppsubinvr)
00872
00873      ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+4) + &
00874      rhodown(indi+3, indj+4)) * &
00875      (mn*(dppsdr(2) - dpppdr(2)) + &
00876      n*(two*m2 - one)*ppsubinvr)
00877
00878      ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+4) + &
00879      rhodown(indi+3, indj+4)) * &
00880      (mn*(dppsdr(3) - dpppdr(3)) + &
00881      m*(two*n2 - one)*ppsubinvr)
00882
00883      ! E_z1,s2
00884

```

```

00885      ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+1) + &
00886      rhodown(indi+4, indj+1)) * &
00887      (n*dppsdr(1) + ln*invr*hpss)
00888
00889      ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+1) + &
00890      rhodown(indi+4, indj+1)) * &
00891      (n*dppsdr(2) + mn*invr*hpss)
00892
00893      ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+1) + &
00894      rhodown(indi+4, indj+1)) * &
00895      (n*dppsdr(3) + (n2 - one)*invr*hpss)
00896
00897      ! E_z1,x2
00898
00899      ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+2) + &
00900      rhodown(indi+4, indj+2)) * &
00901      (ln*(dppsdr(1) - dpppdr(1)) + &
00902      n*(two*12 - one)*ppsubinvr)
00903
00904      ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+2) + &
00905      rhodown(indi+4, indj+2)) * &
00906      (ln*(dppsdr(2) - dpppdr(2)) + &
00907      two*lmn*ppsubinvr)
00908
00909      ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+2) + &
00910      rhodown(indi+4, indj+2)) * &
00911      (ln*(dppsdr(3) - dpppdr(3)) + &
00912      1*(two*n2 - one)*ppsubinvr)
00913
00914      ! E_z1,y2
00915
00916      ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+3) + &
00917      rhodown(indi+4, indj+3)) * &
00918      (mn*(dppsdr(1) - dpppdr(1)) + &
00919      two*lmn*ppsubinvr)
00920
00921      ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+3) + &
00922      rhodown(indi+4, indj+3)) * &
00923      (mn*(dppsdr(2) - dpppdr(2)) + &
00924      n*(two*m2 - one)*ppsubinvr)
00925
00926      ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+3) + &
00927      rhodown(indi+4, indj+3)) * &
00928      (mn*(dppsdr(3) - dpppdr(3)) + &
00929      m*(two*n2 - one)*ppsubinvr)
00930
00931      ! E_z1,z2
00932
00933      ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+4) + &
00934      rhodown(indi+4, indj+4)) * &
00935      (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00936      two*1*n2*ppsubinvr)
00937
00938      ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+4) + &
00939      rhodown(indi+4, indj+4)) * &
00940      (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00941      two*m*n2*ppsubinvr)
00942
00943      ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+4) + &
00944      rhodown(indi+4, indj+4)) * &
00945      (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00946      two*n*(n2 - one)*ppsubinvr)
00947
00948      ENDIF
00949
00950      ENDIF
00951
00952      ENDIF
00953
00954      ftmp = ftmp * ( hubbardu(elempointer(j))*deltaq(j) +
coulombv(j) &
00955      +hubbardu(elempointer(i))*deltaq(i) +
coulombv(i))
00956
00957      !      FTMP = FTMP * ( HUBBARDU(ELEMPOINTER(J))*DELTAQ(J) + COULOMBV(J))
00958
00959      fscoul(1,i) = fscoul(1,i) + ftmp(1)
00960      fscoul(2,i) = fscoul(2,i) + ftmp(2)
00961      fscoul(3,i) = fscoul(3,i) + ftmp(3)
00962
00963      ! with the factor of 2...
00964
00965      virscoul(1) = virscoul(1) + rij(1)*ftmp(1)
00966      virscoul(2) = virscoul(2) + rij(2)*ftmp(2)
00967      virscoul(3) = virscoul(3) + rij(3)*ftmp(3)
00968      virscoul(4) = virscoul(4) + rij(1)*ftmp(2)
00969      virscoul(5) = virscoul(5) + rij(2)*ftmp(3)

```

```
00970      virscoul(6) = virscoul(6) + rij(3)*ftmp(1)
00971
00972      ENDDO
00973
00974      ENDDO
00975
00976
00977      !$OMP END PARALLEL DO
00978
00979      ! PRINT*, VIRSCOUL(1)
00980      virscoul = virscoul/two
00981      ! PRINT*, "Check p = ", CHECKP
00982
00983      RETURN
00984
00985 END SUBROUTINE fcoulnono_sp
```

8.125 fermiallocate.f90 File Reference

Functions/Subroutines

- subroutine [fermiallocate](#)

8.125.1 Function/Subroutine Documentation

8.125.1.1 subroutine [fermiallocate](#) ()

Definition at line 23 of file [fermiallocate.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fermiallocate
00023
00024   USE constants_mod
00025   USE fermicommon
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   IF (cgorlib .EQ. 0) THEN
00031     ALLOCATE( x2(hdim,hdim), a(hdim,hdim) )
00032   ELSEIF (cgorlib .EQ. 1) THEN
00033     ALLOCATE( a(hdim,hdim) )
00034     ALLOCATE( r0(hdim, hdim), p0(hdim,hdim), tmpmat(
00035       hdim, hdim) )
00035   ENDIF
00036
00037 END SUBROUTINE fermiallocate

```

8.127 fermicommon.f90 File Reference

Modules

- module [fermicommon](#)

Variables

- real(latteprec), dimension(:,:), allocatable [fermicommon::a](#)
- real(latteprec) [fermicommon::cgtol](#)
- real(latteprec) [fermicommon::cgtol2](#)
- integer [fermicommon::fermim](#)
- real(latteprec), dimension(:,:), allocatable [fermicommon::p0](#)
- real(latteprec), dimension(:,:), allocatable [fermicommon::r0](#)
- real(latteprec), dimension(:,:), allocatable [fermicommon::tmpmat](#)
- real(latteprec), dimension(:), allocatable [fermicommon::vala](#)
- real(latteprec), dimension(:), allocatable [fermicommon::valp0](#)
- real(latteprec), dimension(:), allocatable [fermicommon::valr0](#)
- real(latteprec), dimension(:), allocatable [fermicommon::valrho](#)
- real(latteprec), dimension(:), allocatable [fermicommon::valtmp](#)
- real(latteprec), dimension(:,:), allocatable [fermicommon::x2](#)

8.128 fermicommon.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE fermicommon
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027
00028   SAVE
00029
00030   INTEGER :: fermim
00031   REAL(LATTEPREC) :: cgtol, cgtol2
00032
00033   ! These are for the dense matrix version...
00034
00035   ! REAL(LATTEPREC), ALLOCATABLE :: X(:, :)
00036   REAL(LATTEPREC), ALLOCATABLE :: r0(:, :), p0(:, :)
00037   REAL(LATTEPREC), ALLOCATABLE :: a(:, :)
00038   REAL(LATTEPREC), ALLOCATABLE :: tmpmat(:, :), x2(:, :)
00039
00040   ! And these are for the sparse matrix one
00041
00042   REAL(LATTEPREC), ALLOCATABLE :: valrho(:, ), valr0(:, ), valp0(:, )
00043   REAL(LATTEPREC), ALLOCATABLE :: vala(:, ), valtmp(:, )
00044
00045 END MODULE fermicommon

```

8.129 fermideallocate.f90 File Reference

Functions/Subroutines

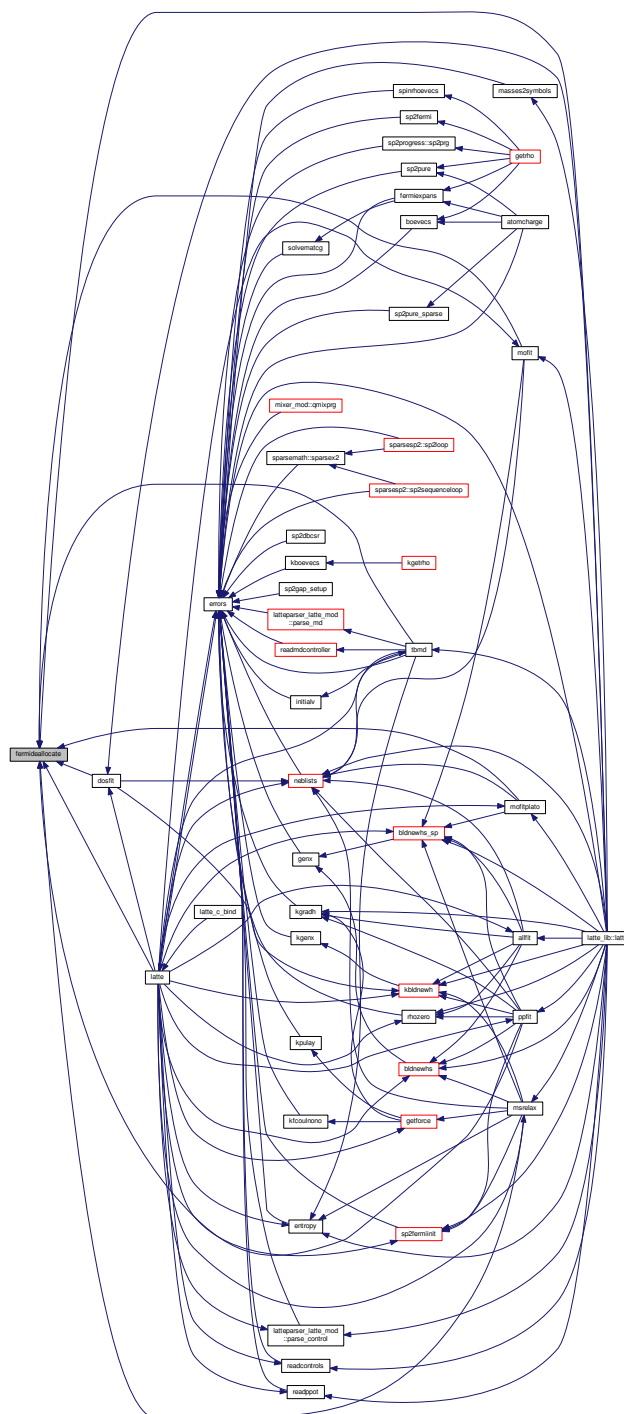
- subroutine [fermideallocate](#)

8.129.1 Function/Subroutine Documentation

8.129.1.1 subroutine fermideallocate ()

Definition at line 23 of file [fermideallocate.f90](#).

Here is the caller graph for this function:



8.130 fermideallocate.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was made available
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
00004 ! National Laboratory (LANL), which is operated by Los Alamos National Security, LLC
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has the right
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fermideallocate
00023
00024   USE constants_mod
00025   USE fermicommon
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   IF (cgorlib .EQ. 0) THEN
00031     DEALLOCATE(x2, a)
00032   ELSEIF (cgorlib .EQ. 1) THEN
00033     DEALLOCATE(a, r0, p0, tmpmat)
00034   ENDIF
00035
00036 END SUBROUTINE fermideallocate

```

8.131 fermiexpans.f90 File Reference

Functions/Subroutines

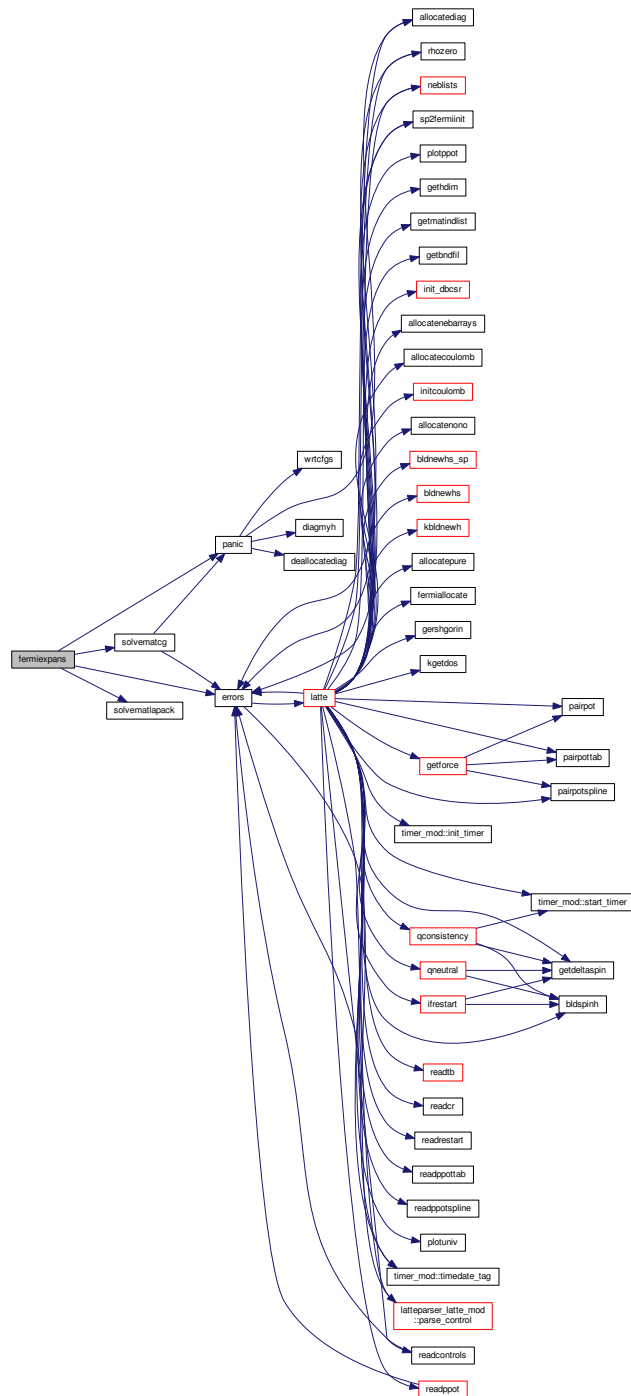
- subroutine [fermiexpans](#)

8.131.1 Function/Subroutine Documentation

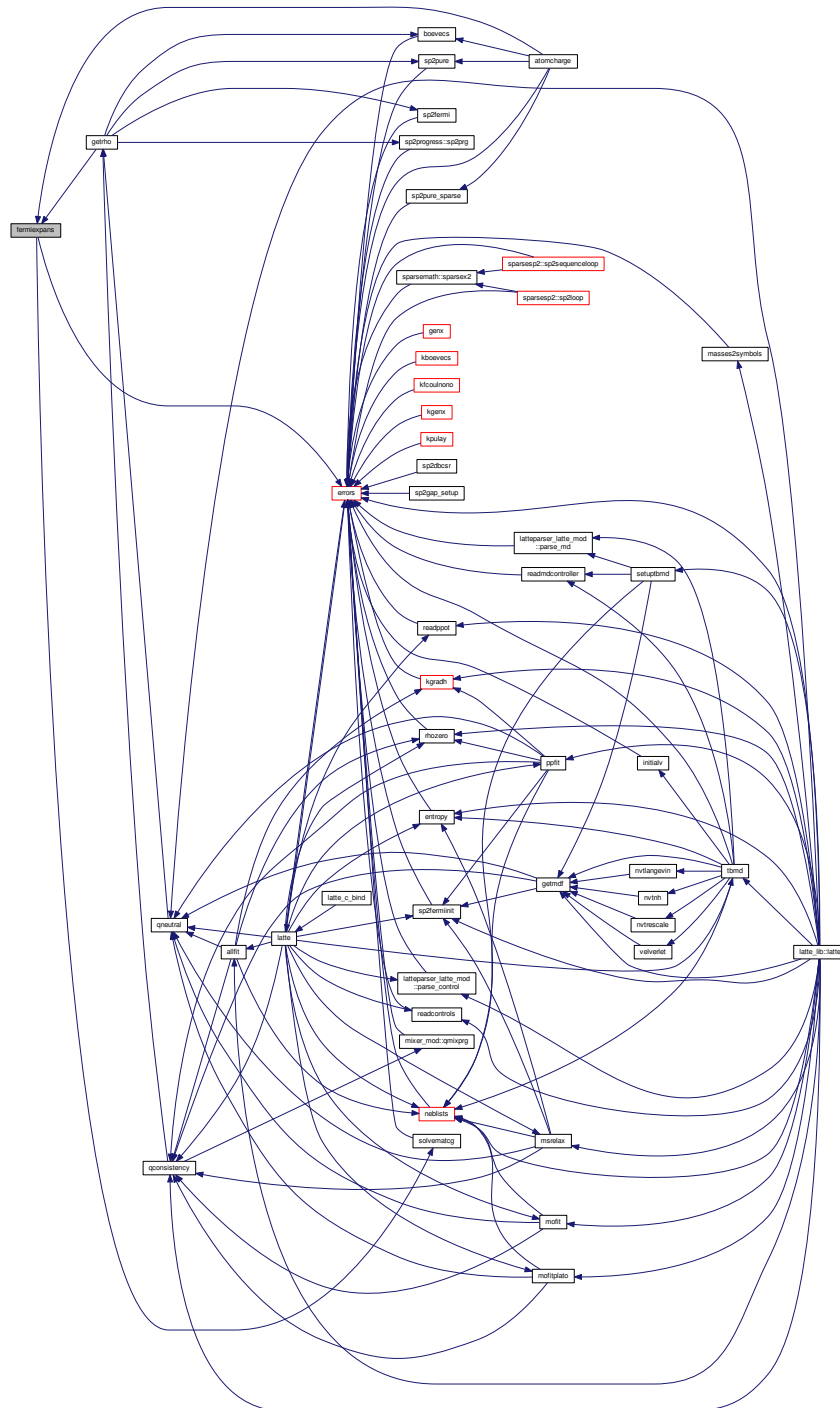
8.131.1.1 subroutine [fermiexpans](#) ()

Definition at line 23 of file [fermiexpans.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.132 fermiexpans.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fermiexpans
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE fermicommon
00027   USE spinarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, ITER
00033   INTEGER :: BREAKLOOP
00034   REAL(LATTEPREC) :: OCC, OCCERROR
00035   REAL(LATTEPREC) :: PREVEERROR, PREVEERROR2, PREVEERROR3
00036   REAL(LATTEPREC) :: BOVER2M, TRX, TRXOMX
00037   REAL(LATTEPREC) :: SHIFTCPC, HFACCT
00038   REAL(LATTEPREC), PARAMETER :: MAXSHIFT = one
00039   IF (existerror) RETURN
00040
00041   occ = bndfil*float(hdim)
00042
00043   iter = 0
00044
00045   breakloop = 0
00046
00047   bover2m = one/(kbt*(two**(2 + fermim)))
00048
00049   scfs = 0
00050
00051   preveerror = zero
00052   preveerror2 = zero
00053   preveerror3 = zero
00054
00055   DO WHILE (breakloop .EQ. 0)
00056
00057     scfs = scfs + 1
00058
00059     iter = iter + 1
00060
00061     IF (iter .EQ. 100) THEN
00062       CALL panic
00063       CALL errors("fermiexpans","Fermi expansion is not converging: STOP!")
00064     ENDIF
00065
00066     IF (spinon .EQ. 0) THEN
00067
00068       bo = -bover2m*h
00069
00070       hfact = half + bover2m*chempot
00071
00072       DO i = 1, hdim
00073         bo(i,i) = hfact + bo(i,i)
00074       ENDDO
00075
00076       ELSE
00077
00078         rhoup = -bover2m*hup
00079         rhodown = -bover2m*hdown
00080
00081         hfact = half + bover2m*chempot
00082
00083         DO i = 1, hdim
00084           rhoup(i,i) = hfact + rhoup(i,i)
00085           rhodown(i,i) = hfact + rhodown(i,i)
00086         ENDDO
00087
00088       ENDIF
00089 #ifdef GPUOFF
00090
00091       DO i = 1, fermim
00092
00093         IF (cgorlib .EQ. 0) THEN

```

```

00094
00095         ! Call GESV-based solver
00096
00097         CALL solvematlpack
00098
00099     ELSE
00100
00101         ! Call the conjugate gradient solver on the CPU
00102
00103         CALL solvemacg
00104
00105     ENDIF
00106
00107     ENDDO
00108
00109 #elif defined(GPUON)
00110
00111     !
00112     ! This calls Sanville's CUDA routines on the GPU
00113     ! Now modified by MJC to send down FERMIM too
00114     !
00115
00116     CALL solve_matrix_cg(bo, rhoup, rhodown, hdim, cgtol2, &
00117         spinon, latteprec, fermim)
00118
00119 #endif
00120
00121     ! Modifying chemical potential
00122
00123     trx = zero
00124     trxomx = zero
00125
00126     IF (spinon .EQ. 0) THEN
00127
00128         DO i = 1, hdim
00129             DO j = i, hdim
00130
00131                 IF (i .EQ. j) THEN
00132
00133                     trxomx = trxomx + bo(i,i)*(one - bo(i,i))
00134
00135                 ELSE
00136
00137                     trxomx = trxomx - two*bo(j,i)*bo(j,i)
00138
00139                 ENDIF
00140
00141             ENDDO
00142
00143             trx = trx + bo(i,i)
00144
00145         ENDDO
00146
00147         shiftcp = kbt*(occ - trx)/trxomx
00148
00149         preverror3 = preverror2
00150         preverror2 = preverror
00151         preverror = occerror
00152         occerror = abs(occ - trx)
00153         ! PRINT*, ITER, OCCERROR
00154
00155     ELSE
00156
00157         DO i = 1, hdim
00158             DO j = i, hdim
00159
00160                 IF (i .EQ. j) THEN
00161
00162                     trxomx = trxomx + rhoup(i,i)*(one - rhoup(i,i)) + &
00163                         rhodown(i,i)*(one - rhodown(i,i))
00164
00165                 ELSE
00166
00167                     trxomx = trxomx - two*(rhoup(j,i)*rhoup(j,i) + &
00168                         rhodown(j,i)*rhodown(j,i))
00169
00170                 ENDIF
00171
00172             ENDDO
00173
00174             trx = trx + rhoup(i,i) + rhodown(i,i)
00175
00176         ENDDO
00177
00178         shiftcp = kbt*(totne - trx)/trxomx
00179
00180         preverror3 = preverror2

```

```

00181      preverror2 = preverror
00182      preverror = occerror
00183      occerror = abs(totne - trx)
00184
00185      ENDIF
00186
00187      !      PRINT*, ITER, OCCERROR
00188
00189      IF (abs(shiftcp) .GT. maxshift) THEN
00190        shiftcp = sign(maxshift, shiftcp)
00191      ENDIF
00192
00193      chempot = chempot + shiftcp
00194
00195 #ifdef DOUBLEPREC
00196
00197      IF (iter .GE. 3 .AND. occerror .LT. breaktol) THEN
00198        breakloop = 1
00199      ENDIF
00200
00201 #elif defined(SINGLEPREC)
00202
00203      IF ( iter .GE. 3 .AND. ((occerror .LT. breaktol) .AND. &
00204        (occerror .EQ. preverror .OR. &
00205        occerror .EQ. preverror2 .OR. &
00206        occerror .EQ. preverror3)) .OR. &
00207        iter .EQ. 25) THEN
00208
00209        breakloop = 1
00210
00211      ENDIF
00212
00213 #endif
00214
00215      ENDDO
00216
00217      IF (spinon .EQ. 0) THEN
00218        bo = two*bo
00219      ENDIF
00220
00221      RETURN
00222
00223 END SUBROUTINE fermiexpans

```

8.133 fittingoutput.f90 File Reference

Functions/Subroutines

- subroutine [fittingoutput](#) (BECLEAN)

8.133.1 Function/Subroutine Documentation

8.133.1.1 subroutine fittingoutput (integer, intent(in) *BECLEAN*)

Definition at line 23 of file [fittingoutput.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fittingoutput (BECLEAN)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE purearray
00028   USE sparsearray
00029   USE coulombarray
00030   USE spinarray
00031   USE myprecision
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I
00036   INTEGER, INTENT(IN) :: BECLEAN
00037   REAL(LATTEPREC) :: MYDIPOLE
00038   REAL(LATTEPREC) :: BIGNO = 9.9e9
00039   IF (existerror) RETURN
00040
00041   OPEN(unit=66, status="UNKNOWN", file="fittingoutput.dat")
00042
00043   IF (beclean .EQ. 0) THEN
00044
00045     IF (spinon .EQ. 0) THEN
00046       tote = trrhoh + erep - ecoul - ente
00047     ELSE
00048       tote = trrhoh + erep - ecoul - ente + espin
00049     ENDIF
00050
00051     WRITE(66,10) tote
00052     DO i = 1, nats
00053       WRITE(66,11) ftot(1,i), ftot(2,i), ftot(3,i)
00054     ENDDO
00055
00056     CALL getdipole(mydipole)
00057
00058     WRITE(66,12) mydipole
00059
00060     WRITE(66,12) egap
00061
00062   ELSEIF (beclean .EQ. 1) THEN ! There was a problem with the calculation
00063
00064     WRITE(66,10) bigno
00065     DO i = 1, nats
00066       WRITE(66,11) bigno, bigno, bigno
00067     ENDDO
00068
00069     WRITE(66,12) bigno
00070
00071   ENDIF
00072
00073 10 FORMAT(g18.9)
00074 11 FORMAT(3g18.9)
00075 12 FORMAT(g18.9)
00076
00077   CLOSE(66)
00078
00079 END SUBROUTINE fittingoutput

```

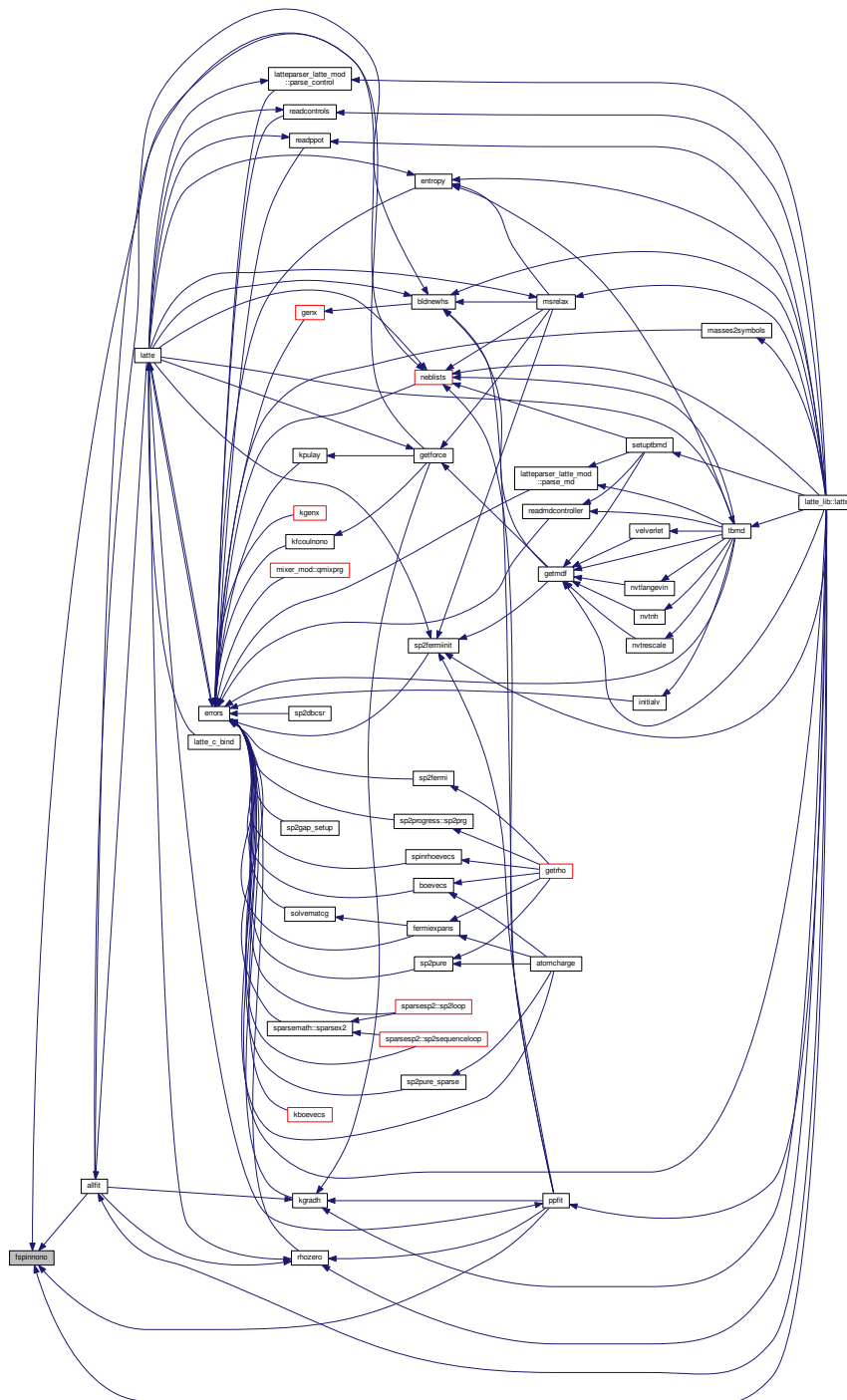
8.135 fspinnono.f90 File Reference

Functions/Subroutines

- subroutine [fspinnono](#)

8.135.1.1 subroutine fspinno ()

Here is the caller graph for this function:



8.136 fspinnono.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fspinnono
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE neblastarray
00030   USE spinarray
00031   USE virialarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00037   INTEGER :: LBRA, MBRA, LKET, MKET
00038   INTEGER :: PREVJ, NEWJ
00039   INTEGER :: PBCI, PBCJ, PBCK
00040   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00041   INTEGER :: SPININDI, SPININDJ
00042   REAL(LATTEPREC) :: ALPHA, BETA, PHI, RHO, COSBETA
00043   REAL(LATTEPREC) :: RIJ(3), DC(3)
00044   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2, FTMP(3)
00045   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00046   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00047   REAL(LATTEPREC) :: WSPINI, WSPINJ
00048   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00049   LOGICAL PATH
00050   IF (existerror) RETURN
00051
00052   fsspin = zero
00053   virsspin = zero
00054
00055   !$OMP PARALLEL DO DEFAULT (NONE) &
00056   !$OMP SHARED(NATS, BASIS, ELEMPINTER, TOTNEBTB, NEBTB) &
00057   !$OMP SHARED(CR, BOX, BO, RHOU, RHODOWN, SPINON, NOINT, ATELE, ELE1, ELE2) &
00058   !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00059   !$OMP SHARED(DELTA SPIN, WSS, WPP, WDD, WFF, SPININDLIST) &
00060   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00061   !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00062   !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00063   !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB, WSPINI, WSPINJ) &
00064   !$OMP PRIVATE(SPININDI, SPININDJ) &
00065   !$OMP REDUCTION(+:FSSPIN, VIRSSPIN)
00066
00067   DO i = 1, nats
00068
00069     ! Build list of orbitals on atom I
00070     SELECT CASE(basis(elempointer(i)))
00071
00072     CASE("s")
00073       basisi(1) = 0
00074       basisi(2) = -1
00075     CASE("p")
00076       basisi(1) = 1
00077       basisi(2) = -1
00078     CASE("d")
00079       basisi(1) = 2
00080       basisi(2) = -1
00081     CASE("f")
00082       basisi(1) = 3
00083       basisi(2) = -1
00084     CASE("sp")

```

```

00085      basisi(1) = 0
00086      basisi(2) = 1
00087      basisi(3) = -1
00088      CASE("sd")
00089      basisi(1) = 0
00090      basisi(2) = 2
00091      basisi(3) = -1
00092      CASE("sf")
00093      basisi(1) = 0
00094      basisi(2) = 3
00095      basisi(3) = -1
00096      CASE("pd")
00097      basisi(1) = 1
00098      basisi(2) = 2
00099      basisi(3) = -1
00100      CASE("pf")
00101      basisi(1) = 1
00102      basisi(2) = 3
00103      basisi(3) = -1
00104      CASE("df")
00105      basisi(1) = 2
00106      basisi(2) = 3
00107      basisi(3) = -1
00108      CASE("spd")
00109      basisi(1) = 0
00110      basisi(2) = 1
00111      basisi(3) = 2
00112      basisi(4) = -1
00113      CASE("spf")
00114      basisi(1) = 0
00115      basisi(2) = 1
00116      basisi(3) = 3
00117      basisi(4) = -1
00118      CASE("sdf")
00119      basisi(1) = 0
00120      basisi(2) = 2
00121      basisi(3) = 3
00122      basisi(4) = -1
00123      CASE("pdf")
00124      basisi(1) = 1
00125      basisi(2) = 2
00126      basisi(3) = 3
00127      basisi(4) = -1
00128      CASE("spdf")
00129      basisi(1) = 0
00130      basisi(2) = 1
00131      basisi(3) = 2
00132      basisi(4) = 3
00133      basisi(5) = -1
00134      END SELECT
00135
00136      indi = matindlist(i)
00137      spinindi = spinindlist(i)
00138
00139      DO newj = 1, totnebtb(i)
00140
00141      j = nebtb(1, newj, i)
00142      pbci = nebtb(2, newj, i)
00143      pbcj = nebtb(3, newj, i)
00144      pbck = nebtb(4, newj, i)
00145
00146      rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00147      REAL(pbck)*BOX(3,1) - CR(1,i)
00148
00149      rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00150      REAL(pbck)*BOX(3,2) - CR(2,i)
00151
00152      rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00153      REAL(pbck)*BOX(3,3) - CR(3,i)
00154
00155      magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00156
00157      rcuttb = zero
00158      DO k = 1, noint
00159
00160      IF ( (atele(i) .EQ. ele1(k) .AND. &
00161      atele(j) .EQ. ele2(k)) .OR. &
00162      (atele(j) .EQ. ele1(k) .AND. &
00163      atele(i) .EQ. ele2(k)) ) THEN
00164
00165      IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00166
00167      IF (basistype .EQ. "NONORTHO") THEN
00168      IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00169      ENDIF
00170
00171      ENDIF

```

```

00172
00173      ENDDO
00174
00175      IF (magr2 .LT. rcuttb*rcuttb) THEN
00176
00177          magr = sqrt(magr2)
00178
00179          ! Build list of orbitals on atom J
00180
00181          SELECT CASE(basis(elempointer(j)))
00182          CASE("s")
00183              basisj(1) = 0
00184              basisj(2) = -1
00185          CASE("p")
00186              basisj(1) = 1
00187              basisj(2) = -1
00188          CASE("d")
00189              basisj(1) = 2
00190              basisj(2) = -1
00191          CASE("f")
00192              basisj(1) = 3
00193              basisj(2) = -1
00194          CASE("sp")
00195              basisj(1) = 0
00196              basisj(2) = 1
00197              basisj(3) = -1
00198          CASE("sd")
00199              basisj(1) = 0
00200              basisj(2) = 2
00201              basisj(3) = -1
00202          CASE("sf")
00203              basisj(1) = 0
00204              basisj(2) = 3
00205              basisj(3) = -1
00206          CASE("pd")
00207              basisj(1) = 1
00208              basisj(2) = 2
00209              basisj(3) = -1
00210          CASE("pf")
00211              basisj(1) = 1
00212              basisj(2) = 3
00213              basisj(3) = -1
00214          CASE("df")
00215              basisj(1) = 2
00216              basisj(2) = 3
00217              basisj(3) = -1
00218          CASE("spd")
00219              basisj(1) = 0
00220              basisj(2) = 1
00221              basisj(3) = 2
00222              basisj(4) = -1
00223          CASE("spdf")
00224              basisj(1) = 0
00225              basisj(2) = 1
00226              basisj(3) = 3
00227              basisj(4) = -1
00228          CASE("sdf")
00229              basisj(1) = 0
00230              basisj(2) = 2
00231              basisj(3) = 3
00232              basisj(4) = -1
00233          CASE("pdf")
00234              basisj(1) = 1
00235              basisj(2) = 2
00236              basisj(3) = 3
00237              basisj(4) = -1
00238          CASE("spdf")
00239              basisj(1) = 0
00240              basisj(2) = 1
00241              basisj(3) = 2
00242              basisj(4) = 3
00243              basisj(5) = -1
00244          END SELECT
00245
00246          indj = matindlist(j)
00247          spinindj = spinindlist(j)
00248
00249          magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00250          magrp = sqrt(magrp2)
00251
00252          ! transform to system in which z-axis is aligned with RIJ
00253
00254          path = .false.
00255          IF (abs(rij(1)) .GT. 1e-12) THEN
00256              IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00257                  phi = zero
00258              ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN

```

```

00259         phi = two * pi
00260     ELSE
00261         phi = pi
00262     ENDIF
00263     alpha = atan(rij(2) / rij(1)) + phi
00264 ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00265     IF (rij(2) .GT. 1e-12) THEN
00266         alpha = pi / two
00267     ELSE
00268         alpha = three * pi / two
00269     ENDIF
00270 ELSE
00271     ! pathological case: pole in alpha at beta=0
00272     path = .true.
00273 ENDIF
00274
00275 cosbeta = rij(3)/magr
00276 beta = acos(rij(3) / magr)
00277
00278 dc = rij/magr
00279
00280 ! build forces using PRB 72 165107 eq. (12) - the sign of the
00281 ! dfda contribution seems to be wrong, but gives the right
00282 ! answer(?)
00283
00284 ftmp = zero
00285 k = indi
00286
00287 lbrainc = 1
00288 DO WHILE (basisi(lbrainc) .NE. -1)
00289
00290     lbra = basisi(lbrainc)
00291     lbrainc = lbrainc + 1
00292
00293     SELECT CASE(lbra)
00294     CASE(0)
00295         wspini = wss(elempointer(i))
00296     CASE(1)
00297         wspini = wpp(elempointer(i))
00298     CASE(2)
00299         wspini = wdd(elempointer(i))
00300     CASE(3)
00301         wspini = wff(elempointer(i))
00302     END SELECT
00303
00304     wspini = wspini*deltaspin(spinindi + lbrainc - 1)
00305
00306     DO mbra = -lbra, lbra
00307
00308         k = k + 1
00309         l = indj
00310
00311         lketinc = 1
00312         DO WHILE (basisj(lketinc) .NE. -1)
00313
00314             lket = basisj(lketinc)
00315             lketinc = lketinc + 1
00316
00317             SELECT CASE(lket)
00318             CASE(0)
00319                 wspinj = wss(elempointer(j))
00320             CASE(1)
00321                 wspinj = wpp(elempointer(j))
00322             CASE(2)
00323                 wspinj = wdd(elempointer(j))
00324             CASE(3)
00325                 wspinj = wff(elempointer(j))
00326             END SELECT
00327
00328             wspinj = wspinj*deltaspin(spinindj + lketinc - 1)
00329
00330             DO mket = -lket, lket
00331
00332                 l = l + 1
00333
00334                 rho = rhoup(l, k) - rhodown(l, k)
00335
00336                 IF (.NOT. path) THEN
00337
00338                     ! Unroll loops and pre-compute
00339
00340                     mydfda = dfda(i, j, lbra, lket, mbra, &
00341                         mket, magr, alpha, cosbeta, "S")
00342
00343                     mydfdb = dfdb(i, j, lbra, lket, mbra, &
00344                         mket, magr, alpha, cosbeta, "S")
00345

```

```

00346      mydfdr = dfdr(i, j, lbra, lket, mbra, &
00347      mket, magr, alpha, cosbeta, "S")
00348
00349      mydfda = mydfda * (wspini + wspinj)
00350      mydfdb = mydfdb * (wspini + wspinj)
00351      mydfdr = mydfdr * (wspini + wspinj)
00352
00353      !
00354      ! d/d_alpha
00355      !
00356
00357      ftmp(1) = ftmp(1) + rho * &
00358      (-rij(2) / magrp2 * mydfda)
00359
00360      ftmp(2) = ftmp(2) + rho * &
00361      (rij(1) / magrp2 * mydfda)
00362
00363      !
00364      ! d/d_beta
00365      !
00366
00367      ftmp(1) = ftmp(1) + rho * &
00368      (((rij(3) * rij(1)) / &
00369      magr2)) / magrp) * mydfdb)
00370
00371      ftmp(2) = ftmp(2) + rho * &
00372      (((rij(3) * rij(2)) / &
00373      magr2)) / magrp) * mydfdb)
00374
00375      ftmp(3) = ftmp(3) - rho * &
00376      ((one - (rij(3) * rij(3)) / &
00377      magr2)) / magrp) * mydfdb)
00378
00379      !
00380      ! d/dR
00381      !
00382
00383      ftmp(1) = ftmp(1) - rho * dc(1) * &
00384      mydfdr
00385
00386      ftmp(2) = ftmp(2) - rho * dc(2) * &
00387      mydfdr
00388
00389      ftmp(3) = ftmp(3) - rho * dc(3) * &
00390      mydfdr
00391
00392
00393      ELSE
00394
00395      ! pathological configuration in which beta=0
00396      ! or pi => alpha undefined
00397
00398      ! fixed: MJC 12/17/13
00399
00400      mydfdb = dfdb(i, j, lbra, lket, &
00401      mbra, mket, magr, zero, cosbeta, "S") / magr
00402
00403      mydfdb = mydfdb * (wspini + wspinj)
00404
00405      ftmp(1) = ftmp(1) - rho * (cosbeta * mydfdb)
00406
00407      mydfdb = dfdb(i, j, lbra, lket, &
00408      mbra, mket, magr, pi/two, cosbeta, "S") / magr
00409
00410      mydfdb = mydfdb * (wspini + wspinj)
00411
00412      ftmp(2) = ftmp(2) - rho * (cosbeta * mydfdb)
00413
00414      mydfdr = dfdr(i, j, lbra, lket, mbra, &
00415      mket, magr, zero, cosbeta, "S")
00416
00417      mydfdr = mydfdr * (wspini + wspinj)
00418
00419      ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00420
00421      ENDIF
00422
00423      ENDDO
00424      ENDDO
00425      ENDDO
00426      ENDDO
00427
00428      !      FTMP = FTMP * ( HUBBARDU(ELEMPINTER(J))*DELTAQ(J) + COULOMBV(J) &
00429      !      +HUBBARDU(ELEMPINTER(I))*DELTAQ(I) + COULOMBV(I))
00430
00431
00432      fsspin(1,i) = fsspin(1,i) + ftmp(1)

```

```

00433         fsspin(2,i) = fsspin(2,i) + ftmp(2)
00434         fsspin(3,i) = fsspin(3,i) + ftmp(3)
00435
00436         ! with the factor of 2...
00437
00438         virsspin(1) = virsspin(1) + rij(1)*ftmp(1)
00439         virsspin(2) = virsspin(2) + rij(2)*ftmp(2)
00440         virsspin(3) = virsspin(3) + rij(3)*ftmp(3)
00441         virsspin(4) = virsspin(4) + rij(1)*ftmp(2)
00442         virsspin(5) = virsspin(5) + rij(2)*ftmp(3)
00443         virsspin(6) = virsspin(6) + rij(3)*ftmp(1)
00444
00445
00446         ENDF
00447     ENDDO
00448
00449 ENDDO
00450
00451 !$OMP END PARALLEL DO
00452
00453 virsspin = virsspin/two
00454
00455 ! PRINT*, FSSPIN(1,1)
00456
00457 ! DO I = 1, NATS
00458 !     WRITE(6,10) I, FSSPIN(1,I), FSSPIN(2,I), FSSPIN(3,I)
00459 ! ENDDO
00460
00461 !10 FORMAT(I4, 3F12.6)
00462
00463 RETURN
00464
00465 END SUBROUTINE fspinnono

```

8.137 fspinnono_sp.f90 File Reference

Functions/Subroutines

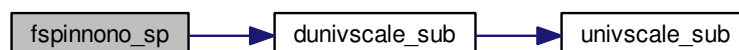
- subroutine [fspinnono_sp](#)

8.137.1 Function/Subroutine Documentation

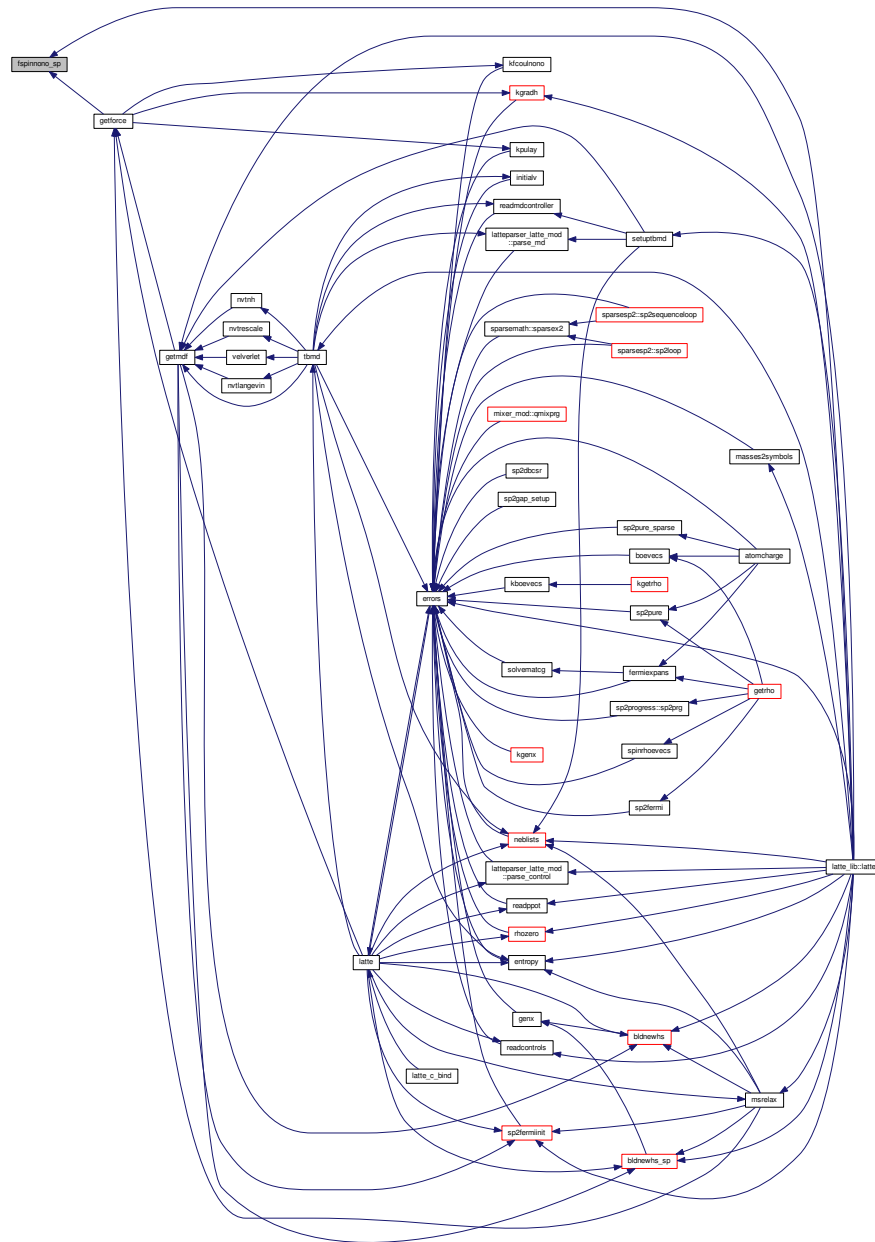
8.137.1.1 subroutine [fspinnono_sp](#) ()

Definition at line 23 of file [fspinnono_sp.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.138 fspinnono_sp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !

```

```

00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be    !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                         !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE fspinmono_sp
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE neblistarray
00030   USE spinarray
00031   USE virialarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, KK, INDI, INDJ
00037   INTEGER :: NEWJ
00038   INTEGER :: PBCI, PBCJ, PBCK
00039   INTEGER :: SPININDI, SPININDJ
00040   REAL(LATTEPREC) :: HSSS, HSPPS, HPSS, HPPS, HPPP
00041   REAL(LATTEPREC) :: RIJ(3), DC(3)
00042   REAL(LATTEPREC) :: L, M, N, L2, M2, N2, LM, LN, MN, LMN
00043   REAL(LATTEPREC) :: DSSSDR(3), DSPSDR(3), DPSSDR(3), DPPSDR(3), DPPPDR(3)
00044   REAL(LATTEPREC) :: MAGR, INVR, FTMP(3), FTMP(3), FTMP(3)
00045   REAL(LATTEPREC) :: PPSMPPP, PPSUBINVR
00046   REAL(LATTEPREC) :: VIRTMP(6), VIRTMP(6), VIRTMP(6)
00047   CHARACTER(LEN=2) :: BASISI, BASISJ
00048   IF (existerror) RETURN
00049
00050   ! INDI = 0
00051
00052   fsspin = zero
00053
00054   virsspin = zero
00055
00056   !
00057   ! We are computing the contribution to the forces from the
00058   ! S dependence of the Mulliken spin densities when we use
00059   ! the non-orthogonal basis
00060
00061   DO i = 1, nats
00062
00063     basisi = basis(elempointer(i))
00064     indi = matindlist(i)
00065     spinindi = spinindlist(i)
00066     ! Loop over all neighbors of I
00067
00068     DO newj = 1, totnebtb(i)
00069
00070       j = nebtb(1, newj, i)
00071       pbcj = nebtb(2, newj, i)
00072       pbcj = nebtb(3, newj, i)
00073       pbck = nebtb(4, newj, i)
00074
00075       indj = matindlist(j)
00076       spinindj = spinindlist(j)
00077
00078       basisj = basis(elempointer(j))
00079
00080       rij(1) = cr(1,j) + REAL(pbcj)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00081         REAL(pbck)*BOX(3,1) - CR(1,i)
00082
00083       rij(2) = cr(2,j) + REAL(pbcj)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00084         REAL(pbck)*BOX(3,2) - CR(2,i)
00085
00086       rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00087         REAL(pbck)*BOX(3,3) - CR(3,i)
00088
00089       magr = sqrt(rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3))
00090
00091       invr = one/magr
00092
00093       ftmps = zero
00094       ftmpp = zero
00095
00096       !
00097       ! Direction cosines (DC)
00098       !
00099
00100       dc = rij/magr
00101

```

```

00102      l = dc(1)
00103      m = dc(2)
00104      n = dc(3)
00105
00106      ! Let's compute rho * dS/dR
00107
00108      IF (basisi .EQ. "s") THEN
00109
00110          IF (basisj .EQ. "s") THEN
00111
00112              DO k = 1, noint
00113                  IF ((atele(i) .EQ. ele1(k) .AND. &
00114                      atele(j) .EQ. ele2(k)) .OR. &
00115                      (atele(i) .EQ. ele2(k) .AND. &
00116                      atele(j) .EQ. ele1(k))) THEN
00117
00118                      IF (btype(k) .EQ. "sss") THEN
00119
00120                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssedr)
00121
00122                      ENDIF
00123
00124                  ENDIF
00125              ENDDO
00126
00127              ftmps = ftmps - dssedr*(rhoup(indi+1, indj+1) - &
00128                  rhodown(indi+1, indj+1))
00129
00130          ELSEIF (basisj .EQ. "sp") THEN
00131
00132              DO k = 1, noint
00133
00134                  IF ((atele(i) .EQ. ele1(k) .AND. &
00135                      atele(j) .EQ. ele2(k)) .OR. &
00136                      (atele(i) .EQ. ele2(k) .AND. &
00137                      atele(j) .EQ. ele1(k))) THEN
00138
00139                      IF (btype(k) .EQ. "sss") THEN
00140
00141                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssedr)
00142
00143                      ELSEIF (btype(k) .EQ. "sps") THEN
00144
00145                          CALL dunivscale_sub(magr, overl(:,k), dc, hsps, dpsedr)
00146
00147                      ENDIF
00148                  ENDIF
00149              ENDDO
00150
00151              l2 = l+1
00152              m2 = m+m
00153              n2 = n+n
00154              lm = l*m
00155              ln = l*n
00156              mn = m*n
00157
00158              ! E_s1,s2
00159
00160              ftmps = ftmps - dssedr*(rhoup(indi+1, indj+1) - &
00161                  rhodown(indi+1, indj+1))
00162
00163              ! E_s1,x2
00164
00165              ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+2) - &
00166                  rhodown(indi+1, indj+2)) * &
00167                  (l*dpsedr(1) + (l2 - one)*invr*hsps)
00168
00169              ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+2) - &
00170                  rhodown(indi+1, indj+2)) * &
00171                  (l*dpsedr(2) + lm*invr*hsps)
00172
00173              ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+2) - &
00174                  rhodown(indi+1, indj+2)) * &
00175                  (l*dpsedr(3) + ln*invr*hsps)
00176
00177              ! E_s1,y2
00178
00179              ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+3) - &
00180                  rhodown(indi+1, indj+3)) * &
00181                  (m*dpsedr(1) + lm*invr*hsps)
00182
00183              ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+3) - &
00184                  rhodown(indi+1, indj+3)) * &
00185                  (m*dpsedr(2) + m2*invr*hsps)
00186
00187              ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+3) - &
00188                  rhodown(indi+1, indj+3)) * &

```

```

00189          (m*dspssdr(3) + mn*invr*hsps)
00190
00191      ! E_s1,z2
00192
00193      ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+4) - &
00194          rhodown(indi+1, indj+4)) * &
00195          (n*dspssdr(1) + ln*invr*hsps)
00196
00197      ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+4) - &
00198          rhodown(indi+1, indj+4)) * &
00199          (n*dspssdr(2) + mn*invr*hsps)
00200
00201      ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+4) - &
00202          rhodown(indi+1, indj+4)) * &
00203          (n*dspssdr(3) + (n2 - one)*invr*hsps)
00204
00205      ENDIF
00206
00207      ELSEIF (basisi .EQ. "sp") THEN
00208
00209          IF (basisj .EQ. "s") THEN
00210
00211              DO k = 1, noint
00212
00213                  IF ((atele(i) .EQ. ele1(k) .AND. &
00214                      atele(j) .EQ. ele2(k)) .OR. &
00215                      (atele(i) .EQ. ele2(k) .AND. &
00216                          atele(j) .EQ. ele1(k))) THEN
00217
00218                      IF (btype(k) .EQ. "sss") THEN
00219
00220                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00221
00222                      ELSEIF (btype(k) .EQ. "sps") THEN
00223
00224                          CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpssdr)
00225
00226                          hpss = -hpss
00227                          dpssdr = -dpssdr
00228
00229                      ENDIF
00230                  ENDIF
00231              ENDDO
00232
00233              l2 = l*1
00234              m2 = m*m
00235              n2 = n*n
00236              lm = l*m
00237              ln = l*n
00238              mn = m*n
00239
00240              ! E_s1,s2
00241
00242              ftmps = ftmps - dssssdr*(rhoup(indi+1, indj+1) - &
00243                  rhodown(indi+1, indj+1))
00244
00245              ! E_x1,s2
00246
00247              ftmps(1) = ftmps(1) - (rhoup(indi+2, indj+1) - &
00248                  rhodown(indi+2, indj+1)) * &
00249                  (l*dpssdr(1) + (l2 - one)*invr*hpss)
00250
00251              ftmps(2) = ftmps(2) - (rhoup(indi+2, indj+1) - &
00252                  rhodown(indi+2, indj+1)) * &
00253                  (l*dpssdr(2) + lm*invr*hpss)
00254
00255              ftmps(3) = ftmps(3) - (rhoup(indi+2, indj+1) - &
00256                  rhodown(indi+2, indj+1)) * &
00257                  (l*dpssdr(3) + ln*invr*hpss)
00258
00259              ! E_y1,s2
00260
00261              ftmps(1) = ftmps(1) - (rhoup(indi+3, indj+1) - &
00262                  rhodown(indi+3, indj+1)) * &
00263                  (m*dpssdr(1) + lm*invr*hpss)
00264
00265              ftmps(2) = ftmps(2) - (rhoup(indi+3, indj+1) - &
00266                  rhodown(indi+3, indj+1)) * &
00267                  (m*dpssdr(2) + (m2 - one)*invr*hpss)
00268
00269              ftmps(3) = ftmps(3) - (rhoup(indi+3, indj+1) - &
00270                  rhodown(indi+3, indj+1)) * &
00271                  (m*dpssdr(3) + mn*invr*hpss)
00272
00273              ! E_z1,s2
00274
00275              ftmps(1) = ftmps(1) - (rhoup(indi+4, indj+1) - &

```

```

00276         rhodown(indi+4, indj+1)) * &
00277         (n*dpssdr(1) + ln*invr*hpss)
00278
00279         ftmps(2) = ftmps(2) - (rhoup(indi+4, indj+1) - &
00280         rhodown(indi+4, indj+1)) * &
00281         (n*dpssdr(2) + mn*invr*hpss)
00282
00283         ftmps(3) = ftmps(3) - (rhoup(indi+4, indj+1) - &
00284         rhodown(indi+4, indj+1)) * &
00285         (n*dpssdr(3) + (n2 - one)*invr*hpss)
00286
00287     ELSEIF (basisj .EQ. "sp") THEN
00288
00289         IF (atele(i) .EQ. atele(j)) THEN
00290
00291             DO k = 1, noint
00292
00293                 IF (atele(i) .EQ. ele1(k) .AND. &
00294                     atele(j) .EQ. ele2(k)) THEN
00295
00296                     IF (btype(k) .EQ. "sss") THEN
00297
00298                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dsssdrr)
00299
00300                     ELSEIF (btype(k) .EQ. "sps") THEN
00301
00302                         CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsdrr)
00303
00304                         dpssdr = -dpsdrr
00305
00306                         hpss = -hspss
00307
00308                     ELSEIF (btype(k) .EQ. "pps") THEN
00309
00310                         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpsdrr)
00311
00312                     ELSEIF (btype(k) .EQ. "ppp") THEN
00313
00314                         CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00315
00316                     ENDIF
00317                 ENDIF
00318             ENDDO
00319
00320         ELSEIF (atele(i) .NE. atele(j)) THEN
00321
00322             DO k = 1, noint
00323
00324                 IF (atele(i) .EQ. ele1(k) .AND. &
00325                     atele(j) .EQ. ele2(k)) THEN
00326
00327                     IF (btype(k) .EQ. "sss") THEN
00328
00329                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dsssdrr)
00330
00331                     ELSEIF (btype(k) .EQ. "sps") THEN
00332
00333                         CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsdrr)
00334
00335                     ELSEIF (btype(k) .EQ. "pps") THEN
00336
00337                         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpsdrr)
00338
00339                     ELSEIF (btype(k) .EQ. "ppp") THEN
00340
00341                         CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00342
00343                     ENDIF
00344
00345                 ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00346                     atele(j) .EQ. ele1(k)) THEN
00347
00348                     IF (btype(k) .EQ. "sss") THEN
00349
00350                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dsssdrr)
00351
00352                     ELSEIF (btype(k) .EQ. "sps") THEN
00353
00354                         CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsdrr)
00355
00356                         dpssdr = -dpsdrr
00357                         hpss = -hpss
00358
00359                     ELSEIF (btype(k) .EQ. "pps") THEN
00360
00361                         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpsdrr)
00362

```

```

00363             ELSEIF (btype(k) .EQ. "ppp") THEN
00364
00365                 CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00366
00367             ENDIF
00368
00369         ENDIF
00370     ENDDO
00371
00372 ENDIF
00373
00374 ppsmppp = hpps - hppp
00375 ppsubinvr = ppsmppp * invr
00376
00377 l2 = l*l
00378 m2 = m*m
00379 n2 = n*n
00380 lm = l*m
00381 ln = l*n
00382 mn = m*n
00383 lmn = lm*n
00384
00385 ! E_s1,s2
00386
00387 ftmps = ftmps - dssedr*(rhoup(indi+1, indj+1) - &
00388     rhodown(indi+1, indj+1))
00389
00390 ! E_s1,x2
00391
00392 ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+2) - &
00393     rhodown(indi+1, indj+2)) * &
00394     (l*dpsedr(1) + (l2 - one)*invr*hsps)
00395
00396 ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+2) - &
00397     rhodown(indi+1, indj+2)) * &
00398     (l*dpsedr(2) + lm*invr*hsps)
00399
00400 ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+2) - &
00401     rhodown(indi+1, indj+2)) * &
00402     (l*dpsedr(3) + ln*invr*hsps)
00403
00404 ! E_s1,y2
00405
00406 ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+3) - &
00407     rhodown(indi+1, indj+3)) * &
00408     (m*dpsedr(1) + lm*invr*hsps)
00409
00410 ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+3) - &
00411     rhodown(indi+1, indj+3)) * &
00412     (m*dpsedr(2) + (m2 - one)*invr*hsps)
00413
00414 ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+3) - &
00415     rhodown(indi+1, indj+3)) * &
00416     (m*dpsedr(3) + mn*invr*hsps)
00417
00418 ! E_s1,z2
00419
00420 ftmpp(1) = ftmpp(1) - (rhoup(indi+1, indj+4) - &
00421     rhodown(indi+1, indj+4)) * &
00422     (n*dpsedr(1) + ln*invr*hsps)
00423
00424 ftmpp(2) = ftmpp(2) - (rhoup(indi+1, indj+4) - &
00425     rhodown(indi+1, indj+4)) * &
00426     (n*dpsedr(2) + mn*invr*hsps)
00427
00428 ftmpp(3) = ftmpp(3) - (rhoup(indi+1, indj+4) - &
00429     rhodown(indi+1, indj+4)) * &
00430     (n*dpsedr(3) + (n2 - one)*invr*hsps)
00431
00432 ! E_x1,s2
00433
00434 ftmps(1) = ftmps(1) - (rhoup(indi+2, indj+1) - &
00435     rhodown(indi+2, indj+1)) * &
00436     (l*dpssedr(1) + (l2 - one)*invr*hpss)
00437
00438 ftmps(2) = ftmps(2) - (rhoup(indi+2, indj+1) - &
00439     rhodown(indi+2, indj+1)) * &
00440     (l*dpssedr(2) + lm*invr*hpss)
00441
00442 ftmps(3) = ftmps(3) - (rhoup(indi+2, indj+1) - &
00443     rhodown(indi+2, indj+1)) * &
00444     (l*dpssedr(3) + ln*invr*hpss)
00445
00446 ! E_x1,x2
00447
00448 ftmpp(1) = ftmpp(1) - (rhoup(indi+2, indj+2) - &
00449     rhodown(indi+2, indj+2)) * &

```

```

00450          (12*dppsdr(1) + (one - 12)*dpppdr(1) + &
00451          two*1*(12 - one)*ppsubinvr)
00452
00453          ftmpp(2) = ftmpp(2) - (rhoup(indi+2, indj+2) - &
00454          rhodown(indi+2, indj+2)) * &
00455          (12*dppsdr(2) + (one - 12)*dpppdr(2) + &
00456          two*12*m*ppsubinvr)
00457
00458          ftmpp(3) = ftmpp(3) - (rhoup(indi+2, indj+2) - &
00459          rhodown(indi+2, indj+2)) * &
00460          (12*dppsdr(3) + (one - 12)*dpppdr(3) + &
00461          two*12*n*ppsubinvr)
00462
00463          ! E_x1,y2
00464
00465          ftmpp(1) = ftmpp(1) - (rhoup(indi+2, indj+3) - &
00466          rhodown(indi+2, indj+3)) * &
00467          (lm*(dppsdr(1) - dpppdr(1)) + &
00468          m*(two*12 - one)*ppsubinvr)
00469
00470          ftmpp(2) = ftmpp(2) - (rhoup(indi+2, indj+3) - &
00471          rhodown(indi+2, indj+3)) * &
00472          (lm*(dppsdr(2) - dpppdr(2)) + &
00473          l*(two*m2 - one)*ppsubinvr)
00474
00475          ftmpp(3) = ftmpp(3) - (rhoup(indi+2, indj+3) - &
00476          rhodown(indi+2, indj+3)) * &
00477          (lm*(dppsdr(3) - dpppdr(3)) + &
00478          two*lmn*ppsubinvr)
00479
00480          ! E_x1,z2
00481
00482          ftmpp(1) = ftmpp(1) - (rhoup(indi+2, indj+4) - &
00483          rhodown(indi+2, indj+4)) * &
00484          (ln*(dppsdr(1) - dpppdr(1)) + &
00485          n*(two*12 - one)*ppsubinvr)
00486
00487          ftmpp(2) = ftmpp(2) - (rhoup(indi+2, indj+4) - &
00488          rhodown(indi+2, indj+4)) * &
00489          (ln*(dppsdr(2) - dpppdr(2)) + &
00490          two*lmn*ppsubinvr)
00491
00492          ftmpp(3) = ftmpp(3) - (rhoup(indi+2, indj+4) - &
00493          rhodown(indi+2, indj+4)) * &
00494          (ln*(dppsdr(3) - dpppdr(3)) + &
00495          l*(two*n2 - one)*ppsubinvr)
00496
00497          ! E_y1,s2
00498
00499          ftmps(1) = ftmps(1) - (rhoup(indi+3, indj+1) - &
00500          rhodown(indi+3, indj+1)) * &
00501          (m*dppsdr(1) + lm*invr*hpss)
00502
00503          ftmps(2) = ftmps(2) - (rhoup(indi+3, indj+1) - &
00504          rhodown(indi+3, indj+1)) * &
00505          (m*dppsdr(2) + (m2 - one)*invr*hpss)
00506
00507          ftmps(3) = ftmps(3) - (rhoup(indi+3, indj+1) - &
00508          rhodown(indi+3, indj+1)) * &
00509          (m*dppsdr(3) + mn*invr*hpss)
00510
00511          ! E_y1,x2
00512
00513          ftmpp(1) = ftmpp(1) - (rhoup(indi+3, indj+2) - &
00514          rhodown(indi+3, indj+2)) * &
00515          (lm*(dppsdr(1) - dpppdr(1)) + &
00516          m*(two*12 - one)*ppsubinvr)
00517
00518          ftmpp(2) = ftmpp(2) - (rhoup(indi+3, indj+2) - &
00519          rhodown(indi+3, indj+2)) * &
00520          (lm*(dppsdr(2) - dpppdr(2)) + &
00521          l*(two*m2 - one)*ppsubinvr)
00522
00523          ftmpp(3) = ftmpp(3) - (rhoup(indi+3, indj+2) - &
00524          rhodown(indi+3, indj+2)) * &
00525          (lm*(dppsdr(3) - dpppdr(3)) + &
00526          two*lmn*ppsubinvr)
00527
00528          ! E_y1,y2
00529
00530          ftmpp(1) = ftmpp(1) - (rhoup(indi+3, indj+3) - &
00531          rhodown(indi+3, indj+3)) * &
00532          (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00533          two*1*m2*ppsubinvr)
00534
00535          ftmpp(2) = ftmpp(2) - (rhoup(indi+3, indj+3) - &
00536          rhodown(indi+3, indj+3)) * &

```

```

00537         (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00538         two*m*(m2 - one)*ppsubinvr)
00539
00540     ftmpp(3) = ftmpp(3) - (rhoup(indi+3, indj+3) - &
00541     rhodown(indi+3, indj+3)) * &
00542     (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00543     two*n*m2*ppsubinvr)
00544
00545     ! E_y1,z2
00546
00547     ftmpp(1) = ftmpp(1) - (rhoup(indi+3, indj+4) - &
00548     rhodown(indi+3, indj+4)) * &
00549     (mn*(dppsdr(1) - dpppdr(1)) + &
00550     two*lmn*ppsubinvr)
00551
00552     ftmpp(2) = ftmpp(2) - (rhoup(indi+3, indj+4) - &
00553     rhodown(indi+3, indj+4)) * &
00554     (mn*(dppsdr(2) - dpppdr(2)) + &
00555     n*(two*m2 - one)*ppsubinvr)
00556
00557     ftmpp(3) = ftmpp(3) - (rhoup(indi+3, indj+4) - &
00558     rhodown(indi+3, indj+4)) * &
00559     (mn*(dppsdr(3) - dpppdr(3)) + &
00560     m*(two*n2 - one)*ppsubinvr)
00561
00562     ! E_z1,s2
00563
00564     ftmps(1) = ftmps(1) - (rhoup(indi+4, indj+1) - &
00565     rhodown(indi+4, indj+1)) * &
00566     (n*dpsdr(1) + ln*invr*hpss)
00567
00568     ftmps(2) = ftmps(2) - (rhoup(indi+4, indj+1) - &
00569     rhodown(indi+4, indj+1)) * &
00570     (n*dpsdr(2) + mn*invr*hpss)
00571
00572     ftmps(3) = ftmps(3) - (rhoup(indi+4, indj+1) - &
00573     rhodown(indi+4, indj+1)) * &
00574     (n*dpsdr(3) + (n2 - one)*invr*hpss)
00575
00576     ! E_z1,x2
00577
00578     ftmpp(1) = ftmpp(1) - (rhoup(indi+4, indj+2) - &
00579     rhodown(indi+4, indj+2)) * &
00580     (ln*(dppsdr(1) - dpppdr(1)) + &
00581     n*(two*l2 - one)*ppsubinvr)
00582
00583     ftmpp(2) = ftmpp(2) - (rhoup(indi+4, indj+2) - &
00584     rhodown(indi+4, indj+2)) * &
00585     (ln*(dppsdr(2) - dpppdr(2)) + &
00586     two*lmn*ppsubinvr)
00587
00588     ftmpp(3) = ftmpp(3) - (rhoup(indi+4, indj+2) - &
00589     rhodown(indi+4, indj+2)) * &
00590     (ln*(dppsdr(3) - dpppdr(3)) + &
00591     l*(two*n2 - one)*ppsubinvr)
00592
00593     ! E_z1,y2
00594
00595     ftmpp(1) = ftmpp(1) - (rhoup(indi+4, indj+3) - &
00596     rhodown(indi+4, indj+3)) * &
00597     (mn*(dppsdr(1) - dpppdr(1)) + &
00598     two*lmn*ppsubinvr)
00599
00600     ftmpp(2) = ftmpp(2) - (rhoup(indi+4, indj+3) - &
00601     rhodown(indi+4, indj+3)) * &
00602     (mn*(dppsdr(2) - dpppdr(2)) + &
00603     n*(two*m2 - one)*ppsubinvr)
00604
00605     ftmpp(3) = ftmpp(3) - (rhoup(indi+4, indj+3) - &
00606     rhodown(indi+4, indj+3)) * &
00607     (mn*(dppsdr(3) - dpppdr(3)) + &
00608     m*(two*n2 - one)*ppsubinvr)
00609
00610     ! E_z1,z2
00611
00612     ftmpp(1) = ftmpp(1) - (rhoup(indi+4, indj+4) - &
00613     rhodown(indi+4, indj+4)) * &
00614     (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00615     two*l*n2*ppsubinvr)
00616
00617     ftmpp(2) = ftmpp(2) - (rhoup(indi+4, indj+4) - &
00618     rhodown(indi+4, indj+4)) * &
00619     (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00620     two*m*n2*ppsubinvr)
00621
00622     ftmpp(3) = ftmpp(3) - (rhoup(indi+4, indj+4) - &
00623     rhodown(indi+4, indj+4)) * &

```



```

00624          (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00625          two*n*(n2 - one)*ppsubinvr)
00626
00627      ENDIF
00628
00629  ENDIF
00630
00631  IF (basisj .EQ. "s") THEN
00632
00633      ftmp = ftmps*deltaspin(spinindj+1)*wss(elempointer(j))
00634
00635  ELSEIF (basisj .EQ. "sp") THEN
00636
00637      ftmp = ftmps*deltaspin(spinindj+1)*wss(elempointer(j)) + &
00638      ftmpp*deltaspin(spinindj+2)*wpp(elempointer(j))
00639
00640  ENDIF
00641
00642  IF (basisi .EQ. "s") THEN
00643
00644      ftmp = ftmp + ftmps*deltaspin(spinindi+1)*wss(elempointer(i))
00645
00646  ELSEIF (basisi .EQ. "sp") THEN
00647
00648      ftmp = ftmp + ftmps*deltaspin(spinindi+1)*wss(elempointer(i)) + &
00649      ftmpp*deltaspin(spinindi+2)*wpp(elempointer(i))
00650
00651  ENDIF
00652
00653  fsspin(1,i) = fsspin(1,i) + ftmp(1)
00654  fsspin(2,i) = fsspin(2,i) + ftmp(2)
00655  fsspin(3,i) = fsspin(3,i) + ftmp(3)
00656
00657  ! with the factor of 2...
00658
00659  virsspin(1) = virsspin(1) + rij(1)*ftmp(1)
00660  virsspin(2) = virsspin(2) + rij(2)*ftmp(2)
00661  virsspin(3) = virsspin(3) + rij(3)*ftmp(3)
00662  virsspin(4) = virsspin(4) + rij(1)*ftmp(2)
00663  virsspin(5) = virsspin(5) + rij(2)*ftmp(3)
00664  virsspin(6) = virsspin(6) + rij(3)*ftmp(1)
00665
00666  ENDDO
00667
00668  ENDDO
00669
00670  virsspin = half*virsspin
00671
00672  RETURN
00673
00674  END SUBROUTINE fspinono_sp

```

8.139 gaussrn.f90 File Reference

Functions/Subroutines

- real(latteprec) function [gaussrn](#) (MEAN, STDDEV)

8.139.1 Function/Subroutine Documentation

8.139.1.1 real(latteprec) function [gaussrn](#) (real(latteprec) *MEAN*, real(latteprec) *STDDEV*)

Definition at line 2 of file [gaussrn.f90](#).

8.140 gaussrn.f90

```

00001 FUNCTION gaussrn(MEAN,STDDEV)
00002
00003   USE mdarray
00004   USE myprecision
00005
00006   IMPLICIT NONE
00007
00008   !
00009   ! Based on GASDEV from Numerical Recipes
00010   !
00011   ! We generate 2 random numbers at a time so we have to be
00012   ! a little careful.
00013   !
00014
00015   REAL(LATTEPREC) :: MEAN, STDDEV
00016   REAL(LATTEPREC) :: GAUSSRN
00017   REAL(LATTEPREC) :: V1, V2, RN(2), R, FAC, Y1, Y2
00018   REAL(LATTEPREC), SAVE :: G2
00019
00020   IF (setth .EQ. 0) THEN
00021
00022       r = two
00023
00024       DO WHILE (r .GE. one)
00025
00026           CALL random_number(rn)
00027
00028           v1 = two*rn(1) - one
00029           v2 = two*rn(2) - one
00030
00031           r = v1*v1 + v2*v2
00032
00033       ENDDO
00034
00035       fac = sqrt( -two * log(r) / r )
00036       y1 = v1 * fac
00037       y2 = v2 * fac
00038
00039       g2 = (y1 * stddev) + mean
00040       gaussrn = (y2 * stddev) + mean
00041
00042       setth = 1
00043
00044   ELSE
00045
00046       gaussrn = g2
00047
00048       setth = 0
00049
00050   ENDIF
00051
00052   RETURN
00053
00054 END FUNCTION gaussrn

```

8.141 gendiag.f90 File Reference

Functions/Subroutines

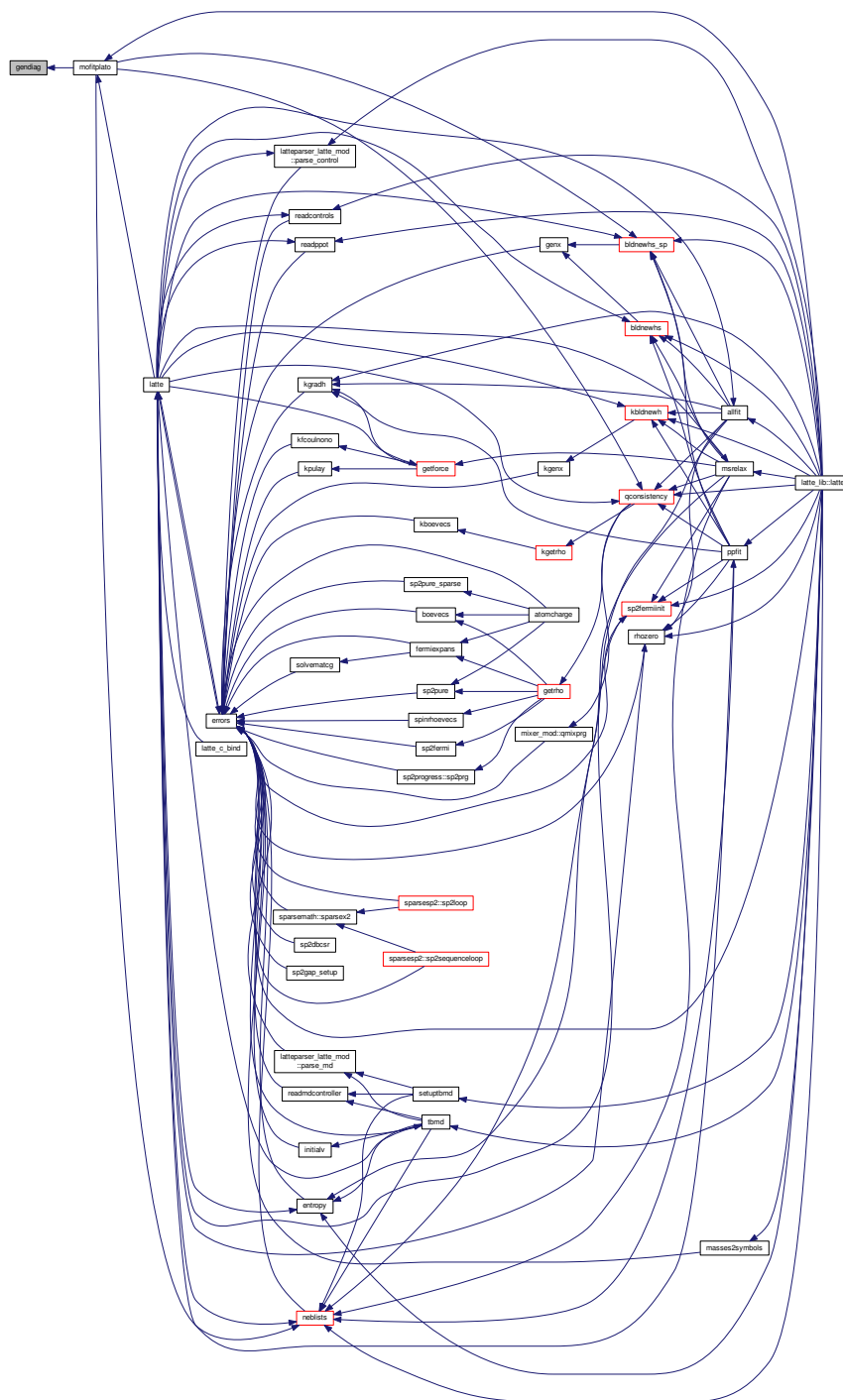
- subroutine [gendiag](#)

8.141.1 Function/Subroutine Documentation

8.141.1.1 subroutine gendiag ()

Definition at line 23 of file [gendiag.f90](#).

Here is the caller graph for this function:



8.142 gendiag.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gendiag
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE nonoarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: INFO
00033   INTEGER :: I, J, K, M, LWORK
00034   REAL(LATTEPREC), ALLOCATABLE :: GENWORK(:), STMP(:, :)
00035   IF (existererror) RETURN
00036
00037   lwork = 3*hdim - 1
00038   ALLOCATE(genwork(lwork), stmp(hdim, hdim))
00039
00040   evecs = h
00041   stmp = smat
00042
00043   CALL dsygv(1,'V','U', hdim, evecs, hdim, stmp, hdim, evals, genwork, &
00044             lwork, info)
00045
00046   IF (debugon .EQ. 1) THEN
00047
00048     DO i = 1, hdim
00049       print*, i, evals(i)
00050     ENDDO
00051
00052     DO i = 1, hdim
00053       WRITE(6,'(100F12.6)') (evecs(j,i), j = 1, hdim)
00054     ENDDO
00055
00056   ENDIF
00057
00058   DEALLOCATE(genwork, stmp)
00059
00060   RETURN
00061
00062 END SUBROUTINE gendiag

```

8.143 genX.f90 File Reference

Functions/Subroutines

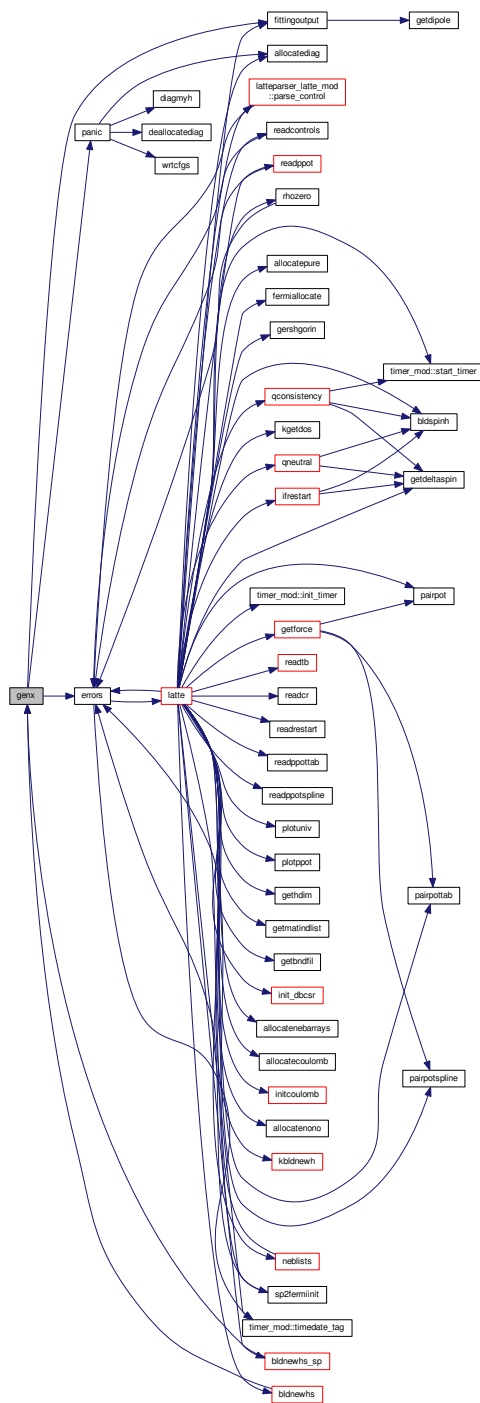
- subroutine [genx](#)

8.143.1 Function/Subroutine Documentation

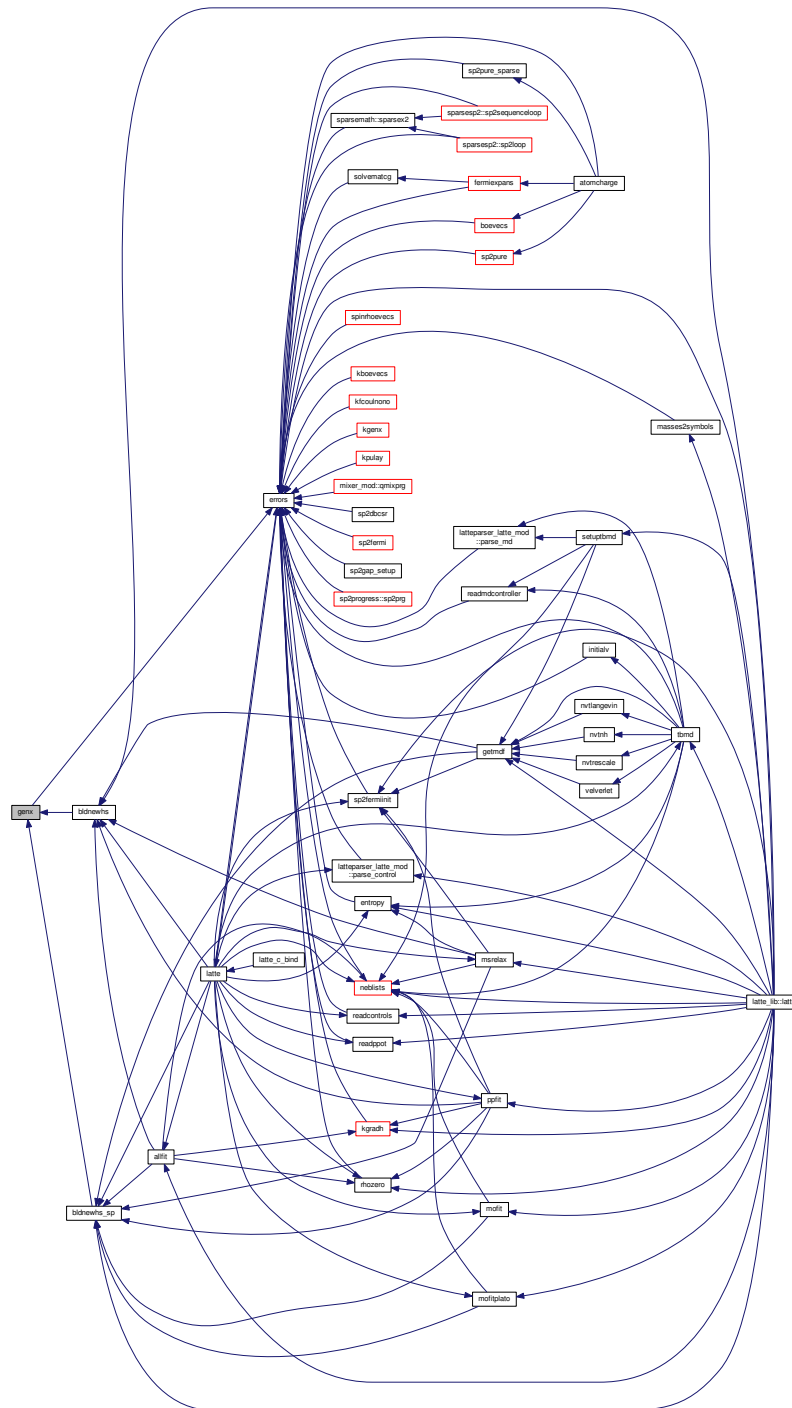
8.143.1.1 subroutine [genx](#) ()

Definition at line 23 of file [genX.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.144 genX.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE genx
00023
00024   USE constants_mod
00025   USE nonoarray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I, J, K, INFO
00031   REAL(LATTEPREC) :: INVSQRT
00032   REAL(LATTEPREC), ALLOCATABLE :: IDENTITY(:, :)
00033
00034   IF (existerror) RETURN
00035
00036   !
00037   !  $X = U s^{-1/2} U^{\dagger}$ 
00038   !
00039
00040   ! Eigenvectors overwrite S (S = U)
00041
00042   umat = smat
00043
00044   #ifdef XSYEV
00045
00046   #ifdef DOUBLEPREC
00047     CALL dsyev("V", "U", hdim, umat, hdim, nono_evals,
00048               nono_work, &
00049               nono_lwork, info)
00050   #elif defined(SINGLEPREC)
00051     CALL ssyev("V", "U", hdim, umat, hdim, nono_evals,
00052               nono_work, &
00053               nono_lwork, info)
00054   #endif
00055
00056   #elif defined(XSYEVD)
00057   #ifdef DOUBLEPREC
00058     CALL dsyevd("V", "U", hdim, umat, hdim, nono_evals,
00059                nono_work, &
00060                nono_lwork, nono_iwork, nono_liwork, info)
00061   #elif defined(SINGLEPREC)
00062     CALL ssyevd("V", "U", hdim, umat, hdim, nono_evals,
00063                nono_work, &
00064                nono_lwork, nono_iwork, nono_liwork, info)
00065   #endif
00066
00067   #endif
00068
00069   IF ( nono_evals(1) .LE. zero ) THEN
00070     CALL fittingoutput(1)
00071     CALL panic
00072     CALL errors("genX", "Eigenvalues of overlap matrix <= 0")
00073     IF (existerror) RETURN
00074   ENDIF
00075
00076   ! PRINT*, MINVAL(NONO_EVALS), MAXVAL(NONO_EVALS)
00077
00078   DO i = 1, hdim
00079     invsqrt = one/sqrt(nono_evals(i))
00080
00081     DO j = 1, hdim
00082       nonotmp(j,i) = umat(j,i) * invsqrt
00083     ENDDO
00084   ENDDO
00085
00086   #ifdef DOUBLEPREC
00087     CALL dgemm('N', 'T', hdim, hdim, hdim, one, &
00088               nonotmp, hdim, umat, hdim, zero, xmat, hdim)
00089   #elif defined(SINGLEPREC)

```

```

00090    CALL sgemm('N', 'T', hdim, hdim, hdim, one, &
00091              nonotmp, hdim, umat, hdim, zero, xmat, hdim)
00092  #endif
00093
00094  IF (debugon .EQ. 1) THEN
00095
00096    ALLOCATE(identity(hdim, hdim))
00097
00098    print*, "Caution - you're writing to file the X matrix!"
00099
00100    OPEN(unit=31, status="UNKNOWN", file="myX.dat")
00101
00102    DO i = 1, hdim
00103      WRITE(31,10) (xmat(i,j), j = 1, hdim)
00104    ENDDO
00105
00106    CLOSE(31)
00107
00108 10  FORMAT(100g18.8)
00109
00110    ! Let's also check the inverse
00111
00112    !    CALL DGEMM( 'N', 'N', HDIM, HDIM, HDIM, ONE, &
00113    !              XMAT, HDIM, XMAT, HDIM, ZERO, XSQ, HDIM)
00114    !    CALL DGEMM( 'N', 'N', HDIM, HDIM, HDIM, ONE, &
00115    !              XSQ, HDIM, IDENTITY, HDIM, ZERO, NONOTMP, HDIM)
00116
00117  #ifdef DOUBLEPREC
00118
00119    CALL dgemm( 'T', 'N', hdim, hdim, hdim, one, &
00120              xmat, hdim, smat, hdim, zero, nonotmp, hdim)
00121    CALL dgemm( 'N', 'N', hdim, hdim, hdim, one, &
00122              nonotmp, hdim, xmat, hdim, zero, identity, hdim)
00123
00124  #elif defined(SINGLEPREC)
00125
00126    CALL sgemm( 'T', 'N', hdim, hdim, hdim, one, &
00127              xmat, hdim, smat, hdim, zero, nonotmp, hdim)
00128    CALL sgemm( 'N', 'N', hdim, hdim, hdim, one, &
00129              nonotmp, hdim, xmat, hdim, zero, identity, hdim)
00130
00131  #endif
00132
00133    OPEN(unit=31, status="UNKNOWN", file="myXcheck.dat")
00134
00135    DO i = 1, hdim
00136      WRITE(31,10) (identity(i,j), j = 1, hdim)
00137    ENDDO
00138
00139    CLOSE(31)
00140
00141    DEALLOCATE (identity)
00142
00143  ENDIF
00144
00145  RETURN
00146
00147  END SUBROUTINE genx

```

8.145 genXprogress.f90 File Reference

Modules

- module [genxprogress](#)
To produce a matrix Z which is needed to orthogonalize H .

Functions/Subroutines

- subroutine, public [genxprogress::genxbml](#)

Variables

- integer, public `genxprogress::igenx` = 0
- type(`bml_matrix_t`), public `genxprogress::over_bml`
- type(`bml_matrix_t`), public `genxprogress::zk1_bml`
- type(`bml_matrix_t`), public `genxprogress::zk2_bml`
- type(`bml_matrix_t`), public `genxprogress::zk3_bml`
- type(`bml_matrix_t`), public `genxprogress::zk4_bml`
- type(`bml_matrix_t`), public `genxprogress::zk5_bml`
- type(`bml_matrix_t`), public `genxprogress::zk6_bml`
- type(`bml_matrix_t`), public `genxprogress::zmat_bml`
- type(`genzspinp`), public `genxprogress::zsp`

8.146 genXprogress.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00027 MODULE genxprogress
00028
00029 #ifdef PROGRESSON
00030
00031 USE bml
00032 USE prg_genz_mod
00033
00034 USE constants_mod
00035 USE nonoarray
00036 USE myprecision
00037
00038 PRIVATE
00039
00040 PUBLIC :: genxbml
00041
00042 TYPE(bml_matrix_t), PUBLIC :: over_bml !FOR STORING S IN BML FORMAT
00043 TYPE(bml_matrix_t), PUBLIC :: zk1_bml, zk2_bml,
00044 zk3_bml
00045 TYPE(bml_matrix_t), PUBLIC :: zk4_bml, zk5_bml,
00046 zk6_bml, zmat_bml
00047
00048 INTEGER, PUBLIC :: igenx = 0!COUNTER TO KEEP TRACK OF THE TIMES ZMAT IS
00049 COMPUTED.
00050 TYPE(genzspinp), PUBLIC :: zsp
00051
00052 CONTAINS
00053
00054 SUBROUTINE genxbml
00055
00056 IMPLICIT NONE
00057
00058 IF(igenx == 0) THEN
00059 CALL prg_parse_zsp(zsp,"latte.in")
00060 IF(zsp%BML_TYPE == bml_matrix_dense .AND. sparseon == 1) stop 'If CONTROL{ SPARSEON= 1 }
00061 then ZSP{ BMLType= Ellpack }'
00062 IF(zsp%BML_TYPE .EQ. bml_matrix_ellpack) THEN
00063 IF(zsp%ZSP .EQV. .false.) stop 'If ZSP{ ZSP= F } then ZSP{ BMLType= Dense }'
00064 IF(sparseon == 0) stop 'If CONTROL{ SPARSEON= 0 } then ZSP{ BMLType= Dense }'
00065 ENDIF
00066 ENDIF

```

```

00063
00064     IF (bml_allocated(over_bml)) CALL bml_deallocate(over_bml)
00065
00066     IF (zsp%VERBOSE.GE.1) WRITE(*,*) "Inside genx ..."
00067
00068     IF (zsp%MDIM < 0) zsp%MDIM = hdim
00069
00070
00071     igenx = igenx + 1 !Counter to keep track of the iterations (md and optimization)
00072
00073     IF (verbose >= 1) WRITE(*,*) "IGENX =", igenx
00074
00075     CALL bml_zero_matrix(zsp%BML_TYPE, bml_element_real, latteprec,
00076                          hdim, zsp%MDIM, over_bml)
00077
00078     CALL bml_import_from_dense(zsp%BML_TYPE, &
00079                               smat, over_bml, zero, zsp%MDIM)
00079
00080     ! CALL BML_PRINT_MATRIX("OVER_BML", OVER_BML, 1, 10, 1, 10)
00081
00082     IF (zsp%ZSP) THEN !Congruence transformation.
00083
00084         IF (igenx == 0) THEN
00085             CALL prg_init_zspmat(igenx, zk1_bml, zk2_bml, zk3_bml&
00086                                , zk4_bml, zk5_bml, zk6_bml, zsp%MDIM, zsp%BML_TYPE)
00087         ENDIF
00088
00089         CALL prg_buildzsparse(over_bml, zmat_bml, igenx, zsp%MDIM, &
00090                              zsp%BML_TYPE, zk1_bml, zk2_bml, zk3_bml&
00091                              , zk4_bml, zk5_bml, zk6_bml, zsp%NFIRST, zsp%NREFI,
00092                              zsp%NREFF, &
00093                              zsp%NUMTHRESI, zsp%NUMTHRESF, zsp%INTEGRATION, zsp%VERBOSE)
00094
00095     ELSE
00096
00097         !Build z matrix using diagonalization (usual method).
00098         CALL prg_buildzdiag(over_bml, zmat_bml, zsp%NUMTHRESF,
00099                             zsp%MDIM, zsp%BML_TYPE)
00098
00099     ENDIF
00100
00101     CALL flush(6)
00102
00103     CALL bml_export_to_dense(zmat_bml, xmat)
00104
00105     IF (zsp%VERBOSE.GE.1) WRITE(*,*) "Out of genx ..."
00106
00107     END SUBROUTINE genxbml
00108
00109 #endif
00110
00111 END MODULE genxprogress

```

8.147 gershgorin.f90 File Reference

Functions/Subroutines

- subroutine [gershgorin](#)

8.147.1 Function/Subroutine Documentation

8.147.1.1 subroutine gershgorin ()

Definition at line 23 of file [gershgorin.f90](#).

[illegible]

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 !
```

```

00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gershgorin
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE sparsearray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J
00034   REAL(LATTEPREC) :: RADIUS, DOWNRAD, UPRAD, ABSHAM
00035   IF (existerror) RETURN
00036
00037   mineval = 1000000000000.0
00038   maxeval = -1000000000000.0
00039
00040   nnz = 0
00041
00042   IF (spinon .EQ. 0) THEN
00043
00044     IF (basistype .EQ. "ORTHO") THEN
00045
00046       ! No spin, orthogonal
00047
00048       DO i = 1, hdim
00049
00050         radius = zero
00051
00052         DO j = 1, hdim
00053
00054           absham = abs( h(j,i) )
00055           radius = radius + absham
00056
00057           IF ( absham .GT. hthresh ) nnz = nnz + 1
00058
00059         ENDDO
00060
00061         radius = radius - abs(h(i,i))
00062
00063         maxeval = max(maxeval, h(i,i) + radius)
00064         mineval = min(mineval, h(i,i) - radius)
00065
00066       ENDDO
00067
00068     ELSE
00069
00070       ! No spin, non-orthogonal
00071
00072       DO i = 1, hdim
00073
00074         radius = zero
00075
00076         DO j = 1, hdim
00077
00078           absham = abs( orthoh(j,i) )
00079           radius = radius + absham
00080
00081           IF ( absham .GT. hthresh ) nnz = nnz + 1
00082
00083         ENDDO
00084
00085         radius = radius - abs(orthoh(i,i))
00086
00087         maxeval = max(maxeval, orthoh(i,i) + radius)
00088         mineval = min(mineval, orthoh(i,i) - radius)
00089
00090       ENDDO
00091
00092     ENDIF
00093
00094   ELSE
00095
00096     IF (basistype .EQ. "ORTHO") THEN
00097
00098       ! Spin polarized, orthogonal
00099

```

```

00100      DO i = 1, hdim
00101
00102          uprad = zero
00103          downrad = zero
00104
00105      DO j = 1, hdim
00106
00107          uprad = uprad + abs( hup(j,i) )
00108          downrad = downrad + abs( hdown(j,i) )
00109
00110      ENDDO
00111
00112      uprad = uprad - abs(hup(i,i))
00113      downrad = downrad - abs(hdown(i,i))
00114
00115      maxeval = max(maxeval, hup(i,i) + uprad, &
00116                  hdown(i,i) + downrad)
00117      mineval = min(mineval, hup(i,i) - uprad, &
00118                  hdown(i,i) - downrad)
00119
00120      ENDDO
00121
00122  ELSE
00123
00124      ! Spin-polarized, non-orthogonal
00125
00126      DO i = 1, hdim
00127
00128          uprad = zero
00129          downrad = zero
00130
00131      DO j = 1, hdim
00132
00133          uprad = uprad + abs( orthohup(j,i) )
00134          downrad = downrad + abs( orthohdown(j,i) )
00135
00136      ENDDO
00137
00138      uprad = uprad - abs(orthohup(i,i))
00139      downrad = downrad - abs(orthohdown(i,i))
00140
00141      maxeval = max(maxeval, orthohup(i,i) + uprad, &
00142                  orthohdown(i,i) + downrad)
00143      mineval = min(mineval, orthohup(i,i) - uprad, &
00144                  orthohdown(i,i) - downrad)
00145
00146      ENDDO
00147
00148  ENDIF
00149
00150  ENDIF
00151
00152  maxminusmin = maxeval - mineval
00153
00154  RETURN
00155
00156  END SUBROUTINE gershgorin

```

8.149 get_end_scope.py File Reference

Namespaces

- [get_end_scope](#)

Functions

- def [get_end_scope.get_end](#) (fin)
- def [get_end_scope.main](#) ()

8.150 get_end_scope.py

```

00001 #!/usr/bin/env python
00002
00003 import numpy as np
00004 def get_end(fin):
00005     count = 0
00006     nsubs = 0
00007     fd = open(fin, "r")
00008     lines_split = {}
00009     lines_after = np.zeros(6)
00010     for lines in fd:
00011         count = count + 1
00012         lines_split = lines.split()
00013         if (len(lines_split[:]) >= 1):
00014             if ((lines_split[0][0:10] == "SUBROUTINE") or (lines_split[0][0:8] == "FUNCTION")):
00015                 nsubs = nsubs + 1
00016             if ((lines_split[0][0:15] == "REAL(LATTEPREC)") or (lines_split[0][0:8] == "INTEGER") or
00017                 (lines_split[0][0:9] == "CHARACTER") or (lines_split[0][0:18] == "COMPLEX(LATTEPREC)") or
00018                 (lines_split[0][0:4] == "TYPE") or (lines_split[0][0:7] == "LOGICAL") or
00019                 (lines_split[0][0:8] == "IMPLICIT")):
00020                 lines_after[nsubs] = count
00021             lines_after[nsubs] = count
00022
00023     for subs in range(1, nsubs+1):
00024         print int(lines_after[subs])
00025         #print "Lines after scope =", int(lines_after[subs]), count, nsubs
00026
00027
00028 def main():
00029     """The main function.
00030     """
00031     import argparse, os, sys
00032
00033     parser = argparse.ArgumentParser(description="Script to get the line
00034     number right after the scope")
00035
00036     parser.add_argument("--fin",
00037                         help="The input file")
00038     options = parser.parse_args()
00039
00040     get_end(options.fin)
00041
00042 if __name__ == "__main__":
00043     main()

```

8.151 getbndfil.f90 File Reference

Functions/Subroutines

- subroutine [getbndfil](#) ()

8.151.1 Function/Subroutine Documentation

8.151.1.1 subroutine getbndfil ()

Definition at line 23 of file [getbndfil.f90](#).

[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getbndfil()
00023
00024   USE constants_mod
00025   USE setuparray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J
00030   INTEGER :: SUMBASIS, NUMORB
00031   IF (existerror) RETURN
00032
00033   totne = zero
00034   sumbasis = 0
00035
00036   DO i = 1, nats
00037
00038     totne = totne + atocc(elempointer(i))
00039
00040     SELECT CASE(basis(elempointer(i)))
00041
00042     CASE("s")
00043
00044       numorb = 1
00045
00046     CASE("p")
00047
00048       numorb = 3
00049
00050     CASE("d")
00051
00052       numorb = 5
00053
00054     CASE("f")
00055
00056       numorb = 7
00057
00058     CASE("sp")
00059
00060       numorb = 4
00061
00062     CASE("sd")
00063
00064       numorb = 6
00065
00066     CASE("sf")
00067
00068       numorb = 8
00069
00070     CASE("pd")
00071
00072       numorb = 8
00073
00074     CASE("pf")
00075
00076       numorb = 10
00077
00078     CASE("df")
00079
00080       numorb = 12
00081
00082     CASE("spd")
00083
00084       numorb = 9
00085
00086     CASE("spf")
00087
00088       numorb = 11
00089
00090     CASE("sdf")
00091
00092       numorb = 13
00093

```



```
00094      CASE("pdf")
00095
00096      numorb = 15
00097
00098      CASE("spdf")
00099
00100      numorb = 16
00101
00102      END_SELECT
00103
00104      sumbasis = sumbasis + numorb
00105
00106      ENDDO
00107
00108      sumbasis = 2*sumbasis
00109
00110      !
00111      ! TOTNE = total number of electrons = tr(rho_up) + tr(rho_down) = tr(bo)
00112      !
00113
00114      totne = totne - REAL(charge)
00115
00116      bndfil = totne/REAL(sumbasis)
00117      ! print*, bndfil
00118      RETURN
00119
00120 END SUBROUTINE getbndfil
```

8.153 getcoule.f90 File Reference

Functions/Subroutines

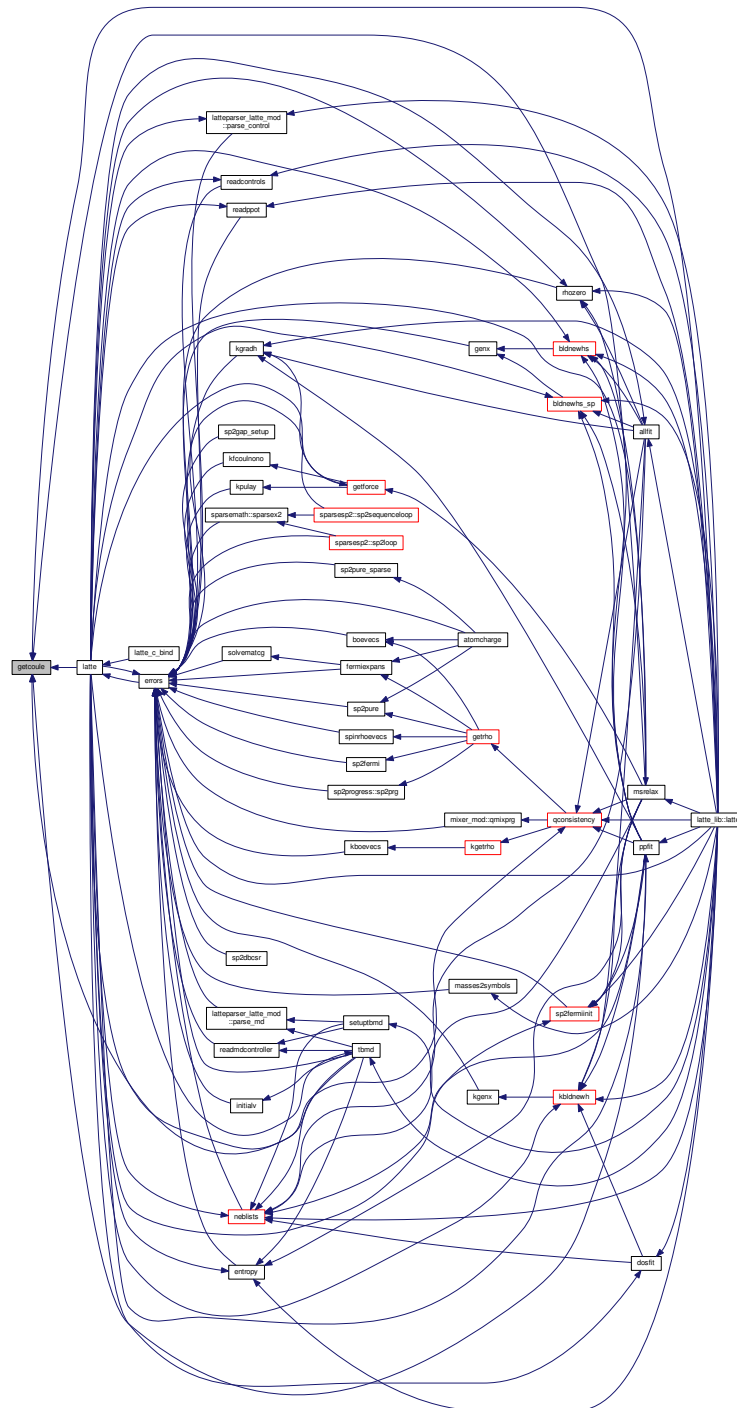
- subroutine [getcoule](#)

8.153.1 Function/Subroutine Documentation

8.153.1.1 subroutine [getcoule](#) ()

Definition at line 23 of file [getcoule.f90](#).

Here is the caller graph for this function:



8.154 getcoule.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getcoule
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I
00032   IF (existerror) RETURN
00033
00034   ecoul = zero
00035
00036   DO i = 1, nats
00037
00038       ecoul = ecoul + deltaq(i) * &
00039           (hubbardu(elempointer(i))*deltaq(i) +
00040            coulombv(i))
00041
00042   ENDDO
00043
00044   ecoul = half*ecoul
00045
00046   RETURN
00047 END SUBROUTINE getcoule
00048

```

8.155 getdeltaq.f90 File Reference

Functions/Subroutines

- subroutine [getdeltaq](#)

8.155.1 Function/Subroutine Documentation

8.155.1.1 subroutine [getdeltaq](#) ()

Definition at line 23 of file [getdeltaq.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getdeltaq
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE myprecision
00030   USE kspacearray
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, INDEX, NUMORB
00035   COMPLEX(LATTEPREC) :: QTMP
00036   IF (existerror) RETURN
00037   ! COMPLEX(LATTEPREC), ALLOCATABLE :: TMPQ(:)
00038
00039   ! ALLOCATE (TMPQ (HDIM) )
00040
00041   index = 0
00042
00043   qlist = zero
00044   mycharge = zero
00045
00046   IF (kon .EQ. 0 ) THEN
00047
00048     IF (basistype .EQ. "ORTHO") THEN
00049
00050       IF (spinon .EQ. 0) THEN
00051
00052         DO i = 1, hdim
00053           qlist(i) = bo(i,i)
00054         ENDDO
00055
00056       ELSE
00057
00058         DO i = 1, hdim
00059           qlist(i) = rhoup(i,i) + rhodown(i,i)
00060         ENDDO
00061
00062       ENDIF
00063
00064     ELSEIF (basistype .EQ. "NONORTHO") THEN
00065
00066       IF (spinon .EQ. 0) THEN
00067
00068         DO i = 1, hdim
00069           DO j = 1, hdim
00070             qlist(i) = qlist(i) + bo(j,i)*smat(j,i)
00071           ENDDO
00072         ENDDO
00073
00074       ELSE
00075
00076         DO i = 1, hdim
00077           DO j = 1, hdim
00078             qlist(i) = qlist(i) + (rhoup(j,i)+rhodown(j,i))*
00079 smat(j,i)
00080           ENDDO
00081         ENDDO
00082       ENDIF
00083
00084     ENDIF
00085
00086   ENDIF
00087
00088 ELSE
00089
00090   ! Loop over k-points
00091
00092   IF (basistype .EQ. "ORTHO") THEN

```

```

00093
00094      DO k = 1, nktot
00095          DO i = 1, hdim
00096              qlist(i) = qlist(i) + REAL(kbo(i,i,k))
00097          ENDDO
00098      ENDDO
00099
00100      qlist = qlist/REAL(nktot)
00101
00102  ELSE
00103
00104      !          TMPQ = (ZERO, ZERO)
00105      DO k = 1, nktot
00106          DO i = 1, hdim
00107              DO j = 1, hdim
00108                  !          TMPQ(I) = TMPQ(I) + KBO(J,I,K)*CONJG(SK(I,J,K))
00109                  qtmp = half*((kbo(j,i,k))*sk(i,j,k) + &
00110                      (kbo(i,j,k))*sk(j,i,k))
00111                  !          QTMP = HALF*((KBO(J,I,K))*SK(I,J,K) + &
00112                      (KBO(J,I,K))*SK(I,J,K))
00113                  qlist(i) = qlist(i) + REAL(qtmp)
00114                  !          QLIST(I) = QLIST(I) + REAL(KBO(J,I,K)*CONJG(SK(J,I,K)))
00115              ENDDO
00116          ENDDO
00117      ENDDO
00118      !          TMPQ = TMPQ/REAL(NKTOT)
00119
00120      qlist = qlist/REAL(nktot)
00121      !          DO I =1, HDIM
00122      !              PRINT*, I, TMPQ(I), QLIST(I)
00123      !          ENDDO
00124
00125      !          PRINT*, "K-space not yet implemented with non-orthogonal basis"
00126      !          STOP
00127
00128
00129  ENDIF
00130
00131  ENDIF
00132
00133  DO i = 1, nats
00134      SELECT CASE(basis(elempointer(i)))
00135
00136      CASE("s")
00137
00138          numorb = 1
00139
00140
00141      CASE("p")
00142
00143          numorb = 3
00144
00145      CASE("d")
00146
00147          numorb = 5
00148
00149      CASE("f")
00150
00151          numorb = 7
00152
00153      CASE("sp")
00154
00155          numorb = 4
00156
00157      CASE("sd")
00158
00159          numorb = 6
00160
00161      CASE("sf")
00162
00163          numorb = 8
00164
00165      CASE("pd")
00166
00167          numorb = 8
00168
00169      CASE("pf")
00170
00171          numorb = 10
00172
00173      CASE("df")
00174
00175          numorb = 12
00176
00177      CASE("spd")
00178
00179

```

```
00180         numorb = 9
00181
00182         CASE("spf")
00183
00184         numorb = 11
00185
00186         CASE("sdf")
00187
00188         numorb = 13
00189
00190         CASE("pdf")
00191
00192         numorb = 15
00193
00194         CASE("spdf")
00195
00196         numorb = 16
00197
00198         END SELECT
00199
00200         !      MYCHARGE = ZERO
00201         DO j = 1, numorb
00202
00203             index = index + 1
00204             mycharge(i) = mycharge(i) + qlist(index)
00205
00206         ENDDO
00207
00208         deltaq(i) = mycharge(i) - atocc(elempointer(i))
00209
00210         ENDDO
00211         !      deallocate(tmpq)
00212
00213         RETURN
00214
00215 END SUBROUTINE getdeltaq
```

8.157 getdeltaspin.f90 File Reference

Functions/Subroutines

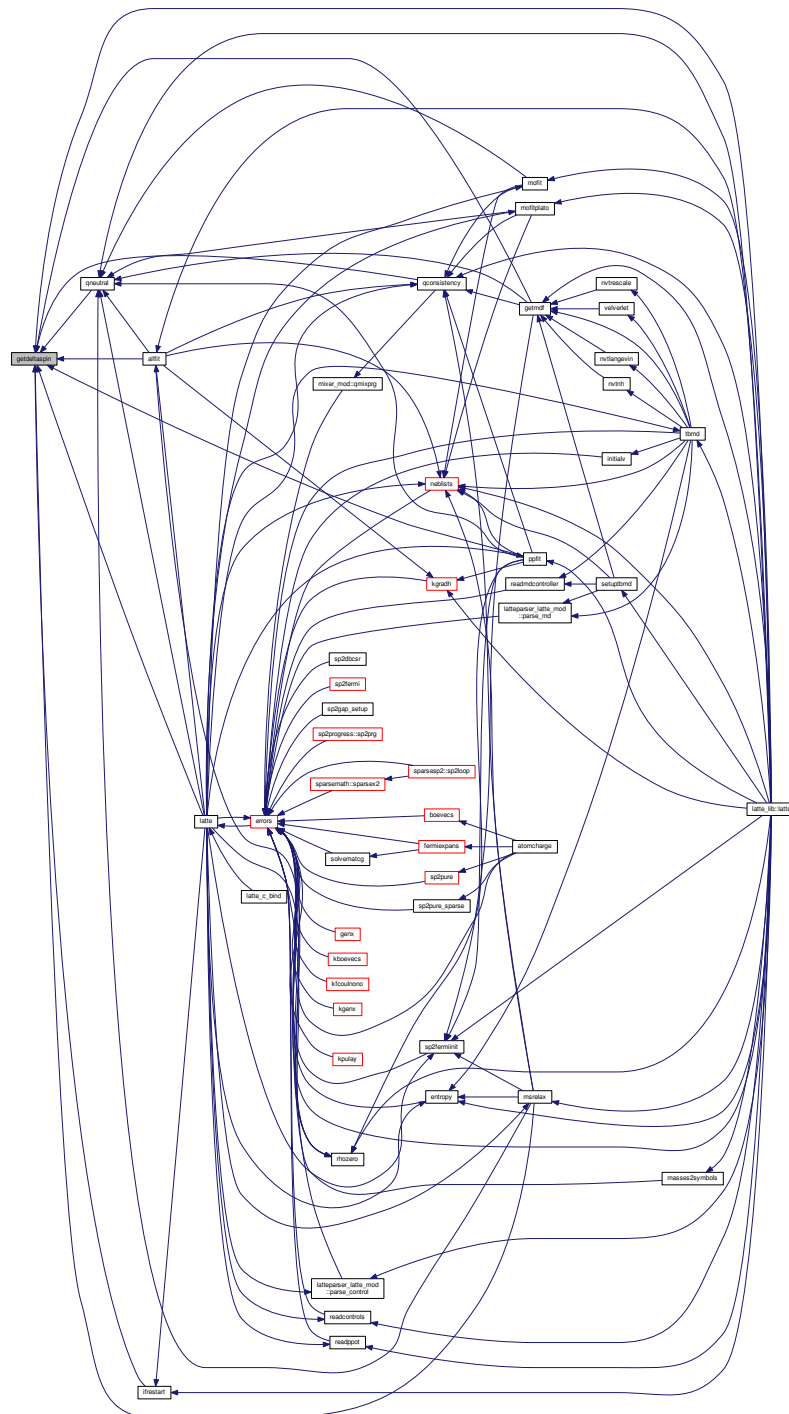
- subroutine [getdeltaspin](#)

8.157.1 Function/Subroutine Documentation

8.157.1.1 subroutine [getdeltaspin](#) ()

Definition at line 23 of file [getdeltaspin.f90](#).

Here is the caller graph for this function:



8.158 getdeltaspin.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getdeltaspin
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, INDEX, DINDEX
00033   IF (existerror) RETURN
00034
00035   index = 0
00036   dindex = 0
00037
00038   ! deltaspin = number of spin ups - number of spin downs
00039
00040   IF (basistype .EQ. "ORTHO") THEN
00041
00042     DO i = 1, nats
00043
00044       SELECT CASE(basis(elempointer(i)))
00045
00046       CASE("s")
00047
00048         index = index + 1
00049         dindex = dindex + 1
00050
00051         ! s
00052
00053         deltaspin(dindex) = rhoup(index, index) - rhodown(index,index)
00054
00055       CASE("p")
00056
00057         dindex = dindex + 1
00058
00059         deltaspin(dindex) = &
00060           rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00061           rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00062           rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00063
00064         index = index + 3
00065
00066       CASE("d")
00067
00068         dindex = dindex + 1
00069
00070         deltaspin(dindex) = &
00071           rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00072           rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00073           rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00074           rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00075           rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00076
00077         index = index + 5
00078
00079       CASE("f")
00080
00081         dindex = dindex + 1
00082
00083         deltaspin(dindex) = &
00084           rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00085           rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00086           rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00087           rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00088           rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00089           rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00090           rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00091
00092         index = index + 7
00093

```

```

00094
00095     CASE("sp")
00096
00097         dindex = dindex + 1
00098
00099         deltaspin(dindex) = &
00100             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00101
00102         index = index + 1
00103
00104         dindex = dindex + 1
00105
00106         deltaspin(dindex) = &
00107             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00108             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00109             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00110
00111         index = index + 3
00112
00113     CASE("sd")
00114
00115         dindex = dindex + 1
00116
00117         deltaspin(dindex) = &
00118             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00119
00120         index = index + 1
00121
00122         dindex = dindex + 1
00123
00124         deltaspin(dindex) = &
00125             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00126             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00127             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00128             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00129             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00130
00131         index = index + 5
00132
00133     CASE("sf")
00134
00135         dindex = dindex + 1
00136
00137         deltaspin(dindex) = &
00138             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00139
00140         index = index + 1
00141
00142         dindex = dindex + 1
00143
00144         deltaspin(dindex) = &
00145             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00146             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00147             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00148             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00149             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00150             rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00151             rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00152
00153         index = index + 7
00154
00155     CASE("pd")
00156
00157         dindex = dindex + 1
00158
00159         deltaspin(dindex) = &
00160             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00161             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00162             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00163
00164         index = index + 3
00165
00166         dindex = dindex + 1
00167
00168         deltaspin(dindex) = &
00169             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00170             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00171             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00172             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00173             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00174
00175         index = index + 5
00176
00177     CASE("pf")
00178
00179         dindex = dindex + 1
00180

```

```

00181
00182     deltaspin(dindex) = &
00183         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00184         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00185         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00186
00187     index = index + 3
00188
00189     dindex = dindex + 1
00190
00191     deltaspin(dindex) = &
00192         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00193         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00194         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00195         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00196         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00197         rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00198         rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00199
00200     index = index + 7
00201
00202     CASE("df")
00203
00204         dindex = dindex + 1
00205
00206         deltaspin(dindex) = &
00207             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00208             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00209             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00210             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00211             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00212
00213         index = index + 5
00214
00215         dindex = dindex + 1
00216
00217         deltaspin(dindex) = &
00218             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00219             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00220             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00221             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00222             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00223             rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00224             rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00225
00226         index = index + 7
00227
00228     CASE("spd")
00229
00230         dindex = dindex + 1
00231
00232         deltaspin(dindex) = &
00233             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00234
00235         index = index + 1
00236
00237         dindex = dindex + 1
00238
00239         deltaspin(dindex) = &
00240             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00241             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00242             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00243
00244         index = index + 3
00245
00246         dindex = dindex + 1
00247
00248         deltaspin(dindex) = &
00249             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00250             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00251             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00252             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00253             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00254
00255         index = index + 5
00256
00257     CASE("spf")
00258
00259         dindex = dindex + 1
00260
00261         deltaspin(dindex) = &
00262             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00263
00264         index = index + 1
00265
00266         dindex = dindex + 1
00267

```

```

00268     deltaspin(dindex) = &
00269         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00270         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00271         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00272
00273     index = index + 3
00274
00275     dindex = dindex + 1
00276
00277     deltaspin(dindex) = &
00278         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00279         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00280         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00281         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00282         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00283         rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00284         rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00285
00286     index = index + 7
00287
00288 CASE("sdf")
00289
00290     dindex = dindex + 1
00291
00292     deltaspin(dindex) = &
00293         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00294
00295     index = index + 1
00296
00297     dindex = dindex + 1
00298
00299     deltaspin(dindex) = &
00300         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00301         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00302         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00303         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00304         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00305
00306     index = index + 5
00307
00308     dindex = dindex + 1
00309
00310     deltaspin(dindex) = &
00311         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00312         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00313         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00314         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00315         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00316         rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00317         rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00318
00319     index = index + 7
00320
00321 CASE("pdf")
00322
00323     dindex = dindex + 1
00324
00325     deltaspin(dindex) = &
00326         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00327         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00328         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00329
00330     index = index + 3
00331
00332     dindex = dindex + 1
00333
00334     deltaspin(dindex) = &
00335         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00336         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00337         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00338         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00339         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00340
00341     index = index + 5
00342
00343     dindex = dindex + 1
00344
00345     deltaspin(dindex) = &
00346         rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00347         rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00348         rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00349         rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00350         rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00351         rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00352         rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00353
00354

```

```

00355         index = index + 7
00356
00357     CASE("spdf")
00358
00359         dindex = dindex + 1
00360
00361         deltaspin(dindex) = &
00362             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1)
00363
00364         index = index + 1
00365
00366         dindex = dindex + 1
00367
00368         deltaspin(dindex) = &
00369             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00370             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00371             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3)
00372
00373         index = index + 3
00374
00375         dindex = dindex + 1
00376
00377         deltaspin(dindex) = &
00378             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00379             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00380             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00381             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00382             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5)
00383
00384         index = index + 5
00385
00386         dindex = dindex + 1
00387
00388         deltaspin(dindex) = &
00389             rhoup(index + 1, index + 1) - rhodown(index + 1, index + 1) + &
00390             rhoup(index + 2, index + 2) - rhodown(index + 2, index + 2) + &
00391             rhoup(index + 3, index + 3) - rhodown(index + 3, index + 3) + &
00392             rhoup(index + 4, index + 4) - rhodown(index + 4, index + 4) + &
00393             rhoup(index + 5, index + 5) - rhodown(index + 5, index + 5) + &
00394             rhoup(index + 6, index + 6) - rhodown(index + 6, index + 6) + &
00395             rhoup(index + 7, index + 7) - rhodown(index + 7, index + 7)
00396
00397         index = index + 7
00398
00399     END SELECT
00400
00401     ENDDO
00402
00403     ELSEIF (basistype .EQ. "NONORTHO") THEN
00404
00405         ! Mulliken spin densities in non-orthogonal basis:
00406
00407         ! m = n_up - n_down
00408
00409         ! n_sigma = partial_trace(rho_sigma S)
00410
00411         spinlist = zero
00412
00413         DO i = 1, hdim
00414             DO j = 1, hdim
00415
00416                 spinlist(i) = spinlist(i) + (rhoup(j,i) -
00417                     rhodown(j,i))*smat(j,i)
00418
00419             ENDDO
00420         ENDDO
00421
00422         DO i = 1, nats
00423
00424             SELECT CASE(basis(elempointer(i)))
00425
00426             CASE("s")
00427
00428                 dindex = dindex + 1
00429
00430                 deltaspin(dindex) = spinlist(index + 1)
00431
00432                 index = index + 1
00433
00434             CASE("p")
00435
00436                 dindex = dindex + 1
00437
00438                 deltaspin(dindex) = spinlist(index + 1) + &
00439                     spinlist(index + 2) + spinlist(index + 3)
00440
00441                 index = index + 3

```

```
00441
00442     CASE("d")
00443
00444         dindex = dindex + 1
00445
00446         deltaspin(dindex) = spinlist(index + 1) + &
00447             spinlist(index + 2) + spinlist(index + 3) + &
00448             spinlist(index + 4) + spinlist(index + 5)
00449
00450         index = index + 5
00451
00452     CASE("f")
00453
00454         dindex = dindex + 1
00455
00456         deltaspin(dindex) = spinlist(index + 1) + &
00457             spinlist(index + 2) + spinlist(index + 3) + &
00458             spinlist(index + 4) + spinlist(index + 5) + &
00459             spinlist(index + 6) + spinlist(index + 7)
00460
00461         index = index + 7
00462
00463     CASE("sp")
00464
00465         dindex = dindex + 1
00466
00467         deltaspin(dindex) = spinlist(index + 1)
00468
00469         index = index + 1
00470
00471         dindex = dindex + 1
00472
00473         deltaspin(dindex) = spinlist(index + 1) + &
00474             spinlist(index + 2) + spinlist(index + 3)
00475
00476         index = index + 3
00477
00478     CASE("sd")
00479
00480         dindex = dindex + 1
00481
00482         deltaspin(dindex) = spinlist(index + 1)
00483
00484         index = index + 1
00485
00486         dindex = dindex + 1
00487
00488         deltaspin(dindex) = spinlist(index + 1) + &
00489             spinlist(index + 2) + spinlist(index + 3) + &
00490             spinlist(index + 4) + spinlist(index + 5)
00491
00492         index = index + 5
00493
00494     CASE("sf")
00495
00496         dindex = dindex + 1
00497
00498         deltaspin(dindex) = spinlist(index + 1)
00499
00500         index = index + 1
00501
00502         dindex = dindex + 1
00503
00504         deltaspin(dindex) = spinlist(index + 1) + &
00505             spinlist(index + 2) + spinlist(index + 3) + &
00506             spinlist(index + 4) + spinlist(index + 5) + &
00507             spinlist(index + 6) + spinlist(index + 7)
00508
00509         index = index + 7
00510
00511
00512     CASE("pd")
00513
00514         dindex = dindex + 1
00515
00516         deltaspin(dindex) = spinlist(index + 1) + &
00517             spinlist(index + 2) + spinlist(index + 3)
00518
00519         index = index + 3
00520
00521         dindex = dindex + 1
00522
00523         deltaspin(dindex) = spinlist(index + 1) + &
00524             spinlist(index + 2) + spinlist(index + 3) + &
00525             spinlist(index + 4) + spinlist(index + 5)
00526
00527         index = index + 5
```

```

00528
00529     CASE("pf")
00530
00531         dindex = dindex + 1
00532
00533         deltaspin(dindex) = spinlist(index + 1) + &
00534             spinlist(index + 2) + spinlist(index + 3)
00535
00536         index = index + 3
00537
00538         dindex = dindex + 1
00539
00540         deltaspin(dindex) = spinlist(index + 1) + &
00541             spinlist(index + 2) + spinlist(index + 3) + &
00542             spinlist(index + 4) + spinlist(index + 5) + &
00543             spinlist(index + 6) + spinlist(index + 7)
00544
00545         index = index + 7
00546
00547     CASE("df")
00548
00549         dindex = dindex + 1
00550
00551         deltaspin(dindex) = spinlist(index + 1) + &
00552             spinlist(index + 2) + spinlist(index + 3) + &
00553             spinlist(index + 4) + spinlist(index + 5)
00554
00555         index = index + 5
00556
00557         dindex = dindex + 1
00558
00559         deltaspin(dindex) = spinlist(index + 1) + &
00560             spinlist(index + 2) + spinlist(index + 3) + &
00561             spinlist(index + 4) + spinlist(index + 5) + &
00562             spinlist(index + 6) + spinlist(index + 7)
00563
00564         index = index + 7
00565
00566
00567     CASE("spd")
00568
00569         dindex = dindex + 1
00570
00571         deltaspin(dindex) = spinlist(index + 1)
00572
00573         index = index + 1
00574
00575         dindex = dindex + 1
00576
00577         deltaspin(dindex) = spinlist(index + 1) + &
00578             spinlist(index + 2) + spinlist(index + 3)
00579
00580         index = index + 3
00581
00582         dindex = dindex + 1
00583
00584         deltaspin(dindex) = spinlist(index + 1) + &
00585             spinlist(index + 2) + spinlist(index + 3) + &
00586             spinlist(index + 4) + spinlist(index + 5)
00587
00588         index = index + 5
00589
00590     CASE("spf")
00591
00592         dindex = dindex + 1
00593
00594         deltaspin(dindex) = spinlist(index + 1)
00595
00596         index = index + 1
00597
00598         dindex = dindex + 1
00599
00600         deltaspin(dindex) = spinlist(index + 1) + &
00601             spinlist(index + 2) + spinlist(index + 3)
00602
00603         index = index + 3
00604
00605         dindex = dindex + 1
00606
00607         deltaspin(dindex) = spinlist(index + 1) + &
00608             spinlist(index + 2) + spinlist(index + 3) + &
00609             spinlist(index + 4) + spinlist(index + 5) + &
00610             spinlist(index + 6) + spinlist(index + 7)
00611
00612         index = index + 7
00613
00614     CASE("sdf")

```

```

00615
00616         dindex = dindex + 1
00617
00618         deltaspin(dindex) = spinlist(index + 1)
00619
00620         index = index + 1
00621
00622         dindex = dindex + 1
00623
00624         deltaspin(dindex) = spinlist(index + 1) + &
00625             spinlist(index + 2) + spinlist(index + 3) + &
00626             spinlist(index + 4) + spinlist(index + 5)
00627
00628         index = index + 5
00629
00630         dindex = dindex + 1
00631
00632         deltaspin(dindex) = spinlist(index + 1) + &
00633             spinlist(index + 2) + spinlist(index + 3) + &
00634             spinlist(index + 4) + spinlist(index + 5) + &
00635             spinlist(index + 6) + spinlist(index + 7)
00636
00637         index = index + 7
00638
00639     CASE("pdf")
00640
00641         dindex = dindex + 1
00642
00643         deltaspin(dindex) = spinlist(index + 1) + &
00644             spinlist(index + 2) + spinlist(index + 3)
00645
00646         index = index + 3
00647
00648         dindex = dindex + 1
00649
00650         deltaspin(dindex) = spinlist(index + 1) + &
00651             spinlist(index + 2) + spinlist(index + 3) + &
00652             spinlist(index + 4) + spinlist(index + 5)
00653
00654         index = index + 5
00655
00656         dindex = dindex + 1
00657
00658         deltaspin(dindex) = spinlist(index + 1) + &
00659             spinlist(index + 2) + spinlist(index + 3) + &
00660             spinlist(index + 4) + spinlist(index + 5) + &
00661             spinlist(index + 6) + spinlist(index + 7)
00662
00663         index = index + 7
00664
00665     CASE("spdf")
00666
00667         dindex = dindex + 1
00668
00669         deltaspin(dindex) = spinlist(index + 1)
00670
00671         index = index + 1
00672
00673         dindex = dindex + 1
00674
00675         deltaspin(dindex) = spinlist(index + 1) + &
00676             spinlist(index + 2) + spinlist(index + 3)
00677
00678         index = index + 3
00679
00680         dindex = dindex + 1
00681
00682         deltaspin(dindex) = spinlist(index + 1) + &
00683             spinlist(index + 2) + spinlist(index + 3) + &
00684             spinlist(index + 4) + spinlist(index + 5)
00685
00686         index = index + 5
00687
00688         dindex = dindex + 1
00689
00690         deltaspin(dindex) = spinlist(index + 1) + &
00691             spinlist(index + 2) + spinlist(index + 3) + &
00692             spinlist(index + 4) + spinlist(index + 5) + &
00693             spinlist(index + 6) + spinlist(index + 7)
00694
00695         index = index + 7
00696
00697     END SELECT
00698
00699 ENDDO
00700
00701 ENDIF

```



```
00702  
00703     RETURN  
00704  
00705 END SUBROUTINE getdeltaspin
```

8.159 getdensity.f90 File Reference

Functions/Subroutines

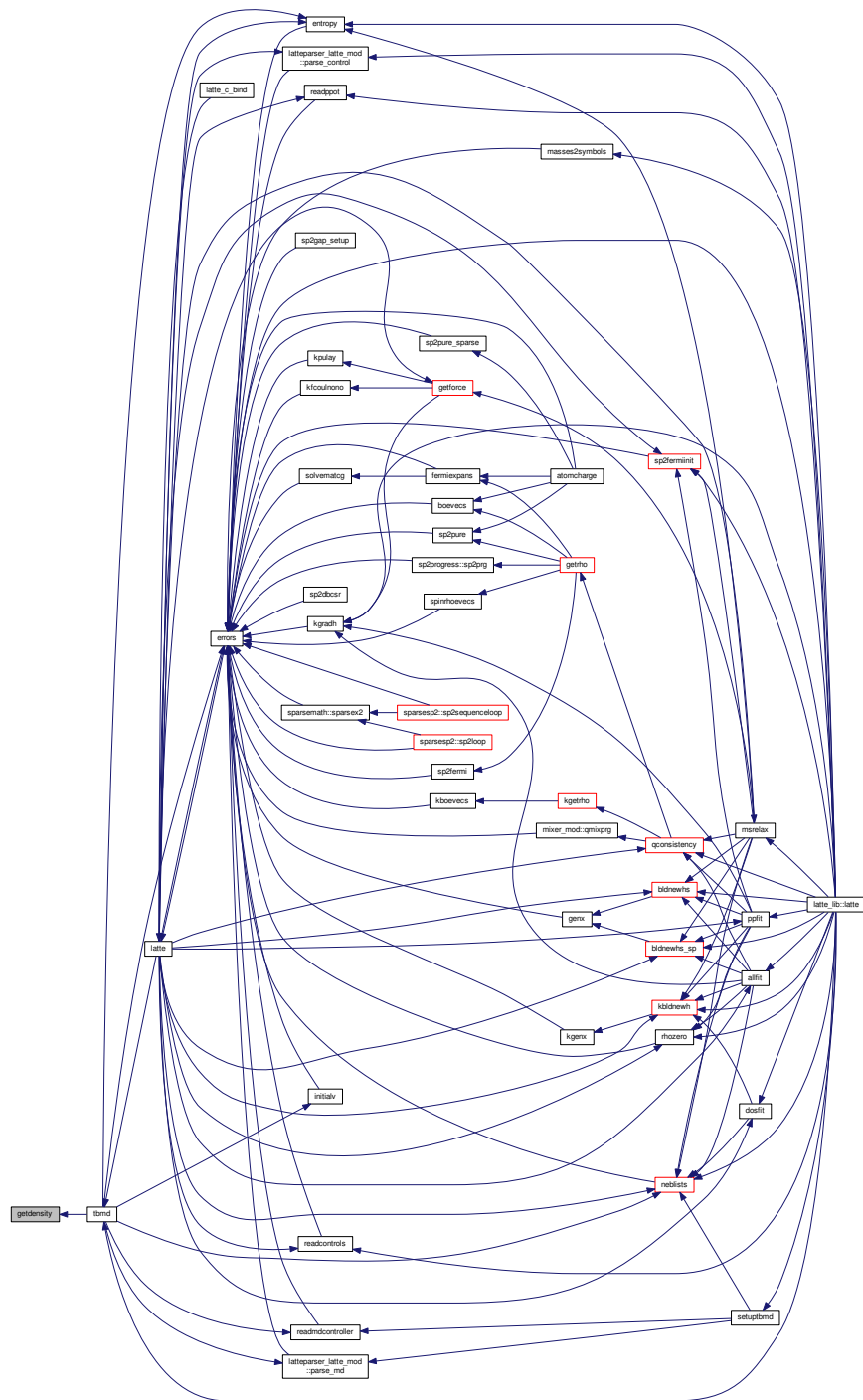
- subroutine [getdensity](#)

8.159.1 Function/Subroutine Documentation

8.159.1.1 subroutine [getdensity](#) ()

Definition at line 23 of file [getdensity.f90](#).

Here is the caller graph for this function:



8.160 getdensity.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getdensity
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   REAL(LATTEPREC) :: SYSVOL
00031   IF (existerror) RETURN
00032
00033   sysvol = abs(box(1,1)*(box(2,2)*box(3,3) - box(3,2)*box(2,3)) + &
00034             box(1,2)*(box(2,1)*box(3,3) - box(3,1)*box(2,3)) + &
00035             box(1,3)*(box(2,1)*box(3,2) - box(3,1)*box(2,2)))
00036
00037   ! Let's have the mass density in g/cm3
00038
00039   massden = 1.660538921*summass/sysvol
00040
00041   ! print*, summass, massden
00042
00043   RETURN
00044
00045 END SUBROUTINE getdensity
00046
00047

```

8.161 getdipole.f90 File Reference

Functions/Subroutines

- subroutine [getdipole](#) (DIPOLEMAG)

8.161.1 Function/Subroutine Documentation

8.161.1.1 subroutine getdipole (real(latteprec), intent(out) DIPOLEMAG)

Definition at line 23 of file [getdipole.f90](#).


```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getdipole(DIPOLEMAG)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I
00032   REAL(LATTEPREC) :: DIPOLEVEC(3)
00033   REAL(LATTEPREC), INTENT(OUT) :: DIPOLEMAG
00034   IF (existerror) RETURN
00035
00036   dipolevec = zero
00037
00038   DO i = 1, nats
00039
00040     dipolevec(1) = dipolevec(1) + cr(1,i)*deltaq(i)
00041     dipolevec(2) = dipolevec(2) + cr(2,i)*deltaq(i)
00042     dipolevec(3) = dipolevec(3) + cr(3,i)*deltaq(i)
00043
00044   ENDDO
00045
00046   dipolemag = sqrt(dipolevec(1)*dipolevec(1) + &
00047     dipolevec(2)*dipolevec(2) + dipolevec(3)*dipolevec(3))
00048
00049   RETURN
00050
00051 END SUBROUTINE getdipole
00052

```

8.163 getforce.f90 File Reference

Functions/Subroutines

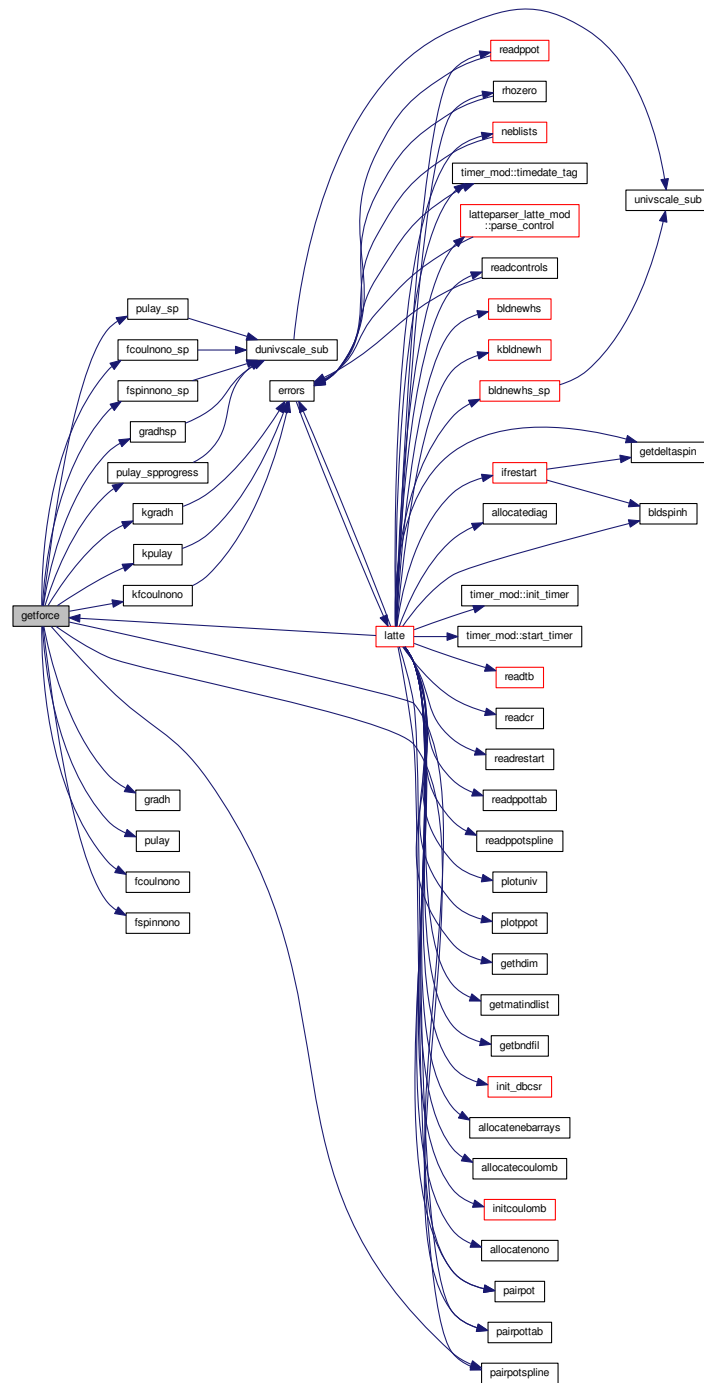
- subroutine [getforce](#)

8.163.1 Function/Subroutine Documentation

8.163.1.1 subroutine [getforce](#) ()

Definition at line 23 of file [getforce.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getforce
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029   IF (existerror) RETURN
00030
00031   ftot = zero
00032
00033   IF (kon .EQ. 0) THEN
00034
00035     IF (sponly .EQ. 0) THEN
00036       CALL gradhsp
00037     ELSE
00038       CALL gradh
00039     ENDIF
00040
00041     ftot = two * f
00042
00043     IF (basistype .EQ. "NONORTHO") THEN
00044
00045       IF (sponly .EQ. 0) THEN
00046
00047         ! s/sp orbitals only so we can use the analytic code
00048
00049         #ifdef PROGRESSON
00050           IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00051             CALL pulay_spprogress
00052           ELSE
00053             CALL pulay_sp
00054           ENDIF
00055         #else
00056           CALL pulay_sp
00057         #endif
00058
00059         IF (electro .EQ. 1) CALL fcoulnono_sp
00060         IF (spinon .EQ. 1) CALL fspinnono_sp
00061
00062       ELSE
00063
00064         ! Otherwise use the complex but general expansions Josh Coe implemented
00065         CALL pulay
00066         IF (electro .EQ. 1) CALL fcoulnono
00067         IF (spinon .EQ. 1) CALL fspinnono
00068
00069       ENDIF
00070
00071       ftot = ftot - two*fpul
00072
00073       IF (electro .EQ. 1) ftot = ftot + fscoul
00074
00075       IF (spinon .EQ. 1) ftot = ftot + fsspin
00076
00077     ENDIF
00078
00079   ELSEIF (kon .EQ. 1) THEN
00080
00081     CALL kgradh
00082
00083     ftot = two*f
00084
00085     IF (basistype .EQ. "NONORTHO") THEN
00086
00087       CALL kpulay
00088
00089       ftot = ftot - two*fpul
00090
00091       IF (electro .EQ. 1) THEN
00092         CALL kfcoulnono
00093

```



```
00094         ftot = ftot + fscoul
00095     ENDIF
00096
00097     ENDIF
00098
00099 ENDIF
00100
00101 IF (ppoton .EQ. 1) THEN
00102     CALL pairpot
00103     ftot = ftot + fpp
00104 ENDIF
00105
00106 IF (ppoton .EQ. 2) THEN
00107     CALL pairpottab
00108     ftot = ftot + fpp
00109 ENDIF
00110
00111 IF (ppoton .EQ. 3) THEN
00112     CALL pairpotspline
00113     ftot = ftot + fpp
00114 ENDIF
00115
00116
00117 IF (electro .EQ. 1) ftot = ftot + fcoul
00118
00119 RETURN
00120
00121 END SUBROUTINE getforce
```

8.165 gethdim.f90 File Reference

Functions/Subroutines

- subroutine [gethdim](#)

8.165.1 Function/Subroutine Documentation

8.165.1.1 subroutine gethdim ()

Definition at line 23 of file [gethdim.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gethdim
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE restartarray
00028   USE kspacearray
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, NUMORB
00033   IF (existerror) RETURN
00034
00035   deltadim = 0
00036
00037   IF (restart .EQ. 0) THEN
00038
00039     hdim = 0
00040
00041     DO i = 1, nats
00042
00043       SELECT CASE(basis(elempointer(i)))
00044
00045         CASE("s")
00046           numorb = 1
00047         CASE("p")
00048           numorb = 3
00049         CASE("d")
00050           numorb = 5
00051         CASE("f")
00052           numorb = 7
00053         CASE("sp")
00054           numorb = 4
00055         CASE("sd")
00056           numorb = 6
00057         CASE("sf")
00058           numorb = 8
00059         CASE("pd")
00060           numorb = 8
00061         CASE("pf")
00062           numorb = 10
00063         CASE("df")
00064           numorb = 12
00065         CASE("spd")
00066           numorb = 9
00067         CASE("spf")
00068           numorb = 11
00069         CASE("sdf")
00070           numorb = 13
00071         CASE("pdf")
00072           numorb = 15
00073         CASE("spdf")
00074           numorb = 16
00075       END SELECT
00076
00077       hdim = hdim + numorb
00078
00079     ENDDO
00080
00081   ELSEIF (restart .EQ. 1) THEN
00082
00083     hdim = tmphdim
00084
00085   ENDIF
00086
00087   ! For use when getting the partial charges
00088
00089   ALLOCATE(qlist(hdim))
00090
00091   ! Allocate the Hamiltonian matrix
00092
00093   IF (kon .EQ. 0) THEN

```

```

00094
00095     ! Real space
00096
00097     ALLOCATE(h(hdim, hdim), hdiag(hdim))
00098
00099 ELSE ! k-space
00100
00101     ALLOCATE(hk(hdim, hdim, nktot), hkdiag(hdim, nktot))
00102
00103 ENDIF
00104
00105 ! If we're using a non-orthogonal basis:
00106
00107 IF (basistype .EQ. "NONORTHO") THEN
00108     IF (kon .EQ. 0) THEN
00109         ALLOCATE(h0(hdim, hdim))
00110     ELSE
00111         ALLOCATE(hk0(hdim, hdim, nktot))
00112     ENDIF
00113 ENDIF
00114
00115 IF (spinon .EQ. 0) THEN
00116
00117     ! No spins: allocate 1 double-occupied bond order matrix
00118
00119     IF (kon .EQ. 0) THEN
00120
00121         ALLOCATE(bo(hdim, hdim))
00122
00123     ELSE
00124
00125         ALLOCATE(kbo(hdim, hdim, nktot))
00126
00127     ENDIF
00128
00129 ELSEIF (spinon .EQ. 1) THEN
00130
00131     ! We're going to have two Hamiltonians because I can't
00132     ! figure out a more elegant way to do it just yet...
00133
00134     ! With spins, we need spin-up and spin-down density matrices
00135
00136     ALLOCATE(hup(hdim, hdim), hdown(hdim, hdim))
00137     ALLOCATE(rhoup(hdim, hdim), rhodown(hdim, hdim))
00138     ALLOCATE(h2vect(hdim))
00139
00140     hup = zero
00141     hdown = zero
00142     rhoup = zero
00143     rhodown = zero
00144
00145     ! And our spin-dependent H_(2) matrix
00146
00147     DO i = 1, nats
00148
00149         SELECT CASE(basis(elempointer(i)))
00150
00151             CASE("s")
00152                 numorb = 1
00153             CASE("p")
00154                 numorb = 1
00155             CASE("d")
00156                 numorb = 1
00157             CASE("f")
00158                 numorb = 1
00159
00160             CASE("sp")
00161                 numorb = 2
00162
00163             CASE("sd")
00164                 numorb = 2
00165
00166             CASE("sf")
00167                 numorb = 2
00168
00169             CASE("pd")
00170                 numorb = 2
00171             CASE("pf")
00172                 numorb = 2
00173             CASE("df")
00174                 numorb = 2
00175             CASE("spd")
00176                 numorb = 3
00177
00178         ENDSELECT
00179     END DO
00180

```

```

00181      CASE("spf")
00182          numorb = 3
00183      CASE("sdf")
00184          numorb = 3
00185      CASE("pdf")
00186          numorb = 3
00187      CASE("spdf")
00188          numorb = 4
00189      END SELECT
00190
00191      deltadim = deltadim + numorb
00192
00193      ENDDO
00194
00195      ! This array is required when we calculate Mulliken spin densities
00196      ! with a non-orthogonal basis
00197
00198      IF (basistype .EQ. "NONORTHO") ALLOCATE(spinlist(hdim))
00199
00200      ALLOCATE(deltaspin(deltadim), olddeltaspin(
deltadim))
00201
00202      deltaspin = zero
00203      olddeltaspin = zero
00204
00205      ENDIF
00206
00207      RETURN
00208
00209 END SUBROUTINE gethdim

```

8.167 getke.f90 File Reference

Functions/Subroutines

- subroutine [getke](#)

8.167.1 Function/Subroutine Documentation

8.167.1.1 subroutine [getke](#) ()

Definition at line 23 of file [getke.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getke
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031
00032   INTEGER :: I, J
00033   IF (existerror) RETURN
00034
00035   kee = zero
00036
00037   DO i = 1, nats
00038
00039     kee = kee + mass(elempointer(i)) * &
00040       (v(1,i)*v(1,i) + v(2,i)*v(2,i) + v(3,i)*v(3,i))
00041
00042   ENDDO
00043
00044   kee = half*mvv2ke*kee
00045
00046   temperature = (two/three)*ke2t*kee/float(nats)
00047
00048   RETURN
00049
00050 END SUBROUTINE getke
00051
00052

```

8.169 getmatindlist.f90 File Reference

Functions/Subroutines

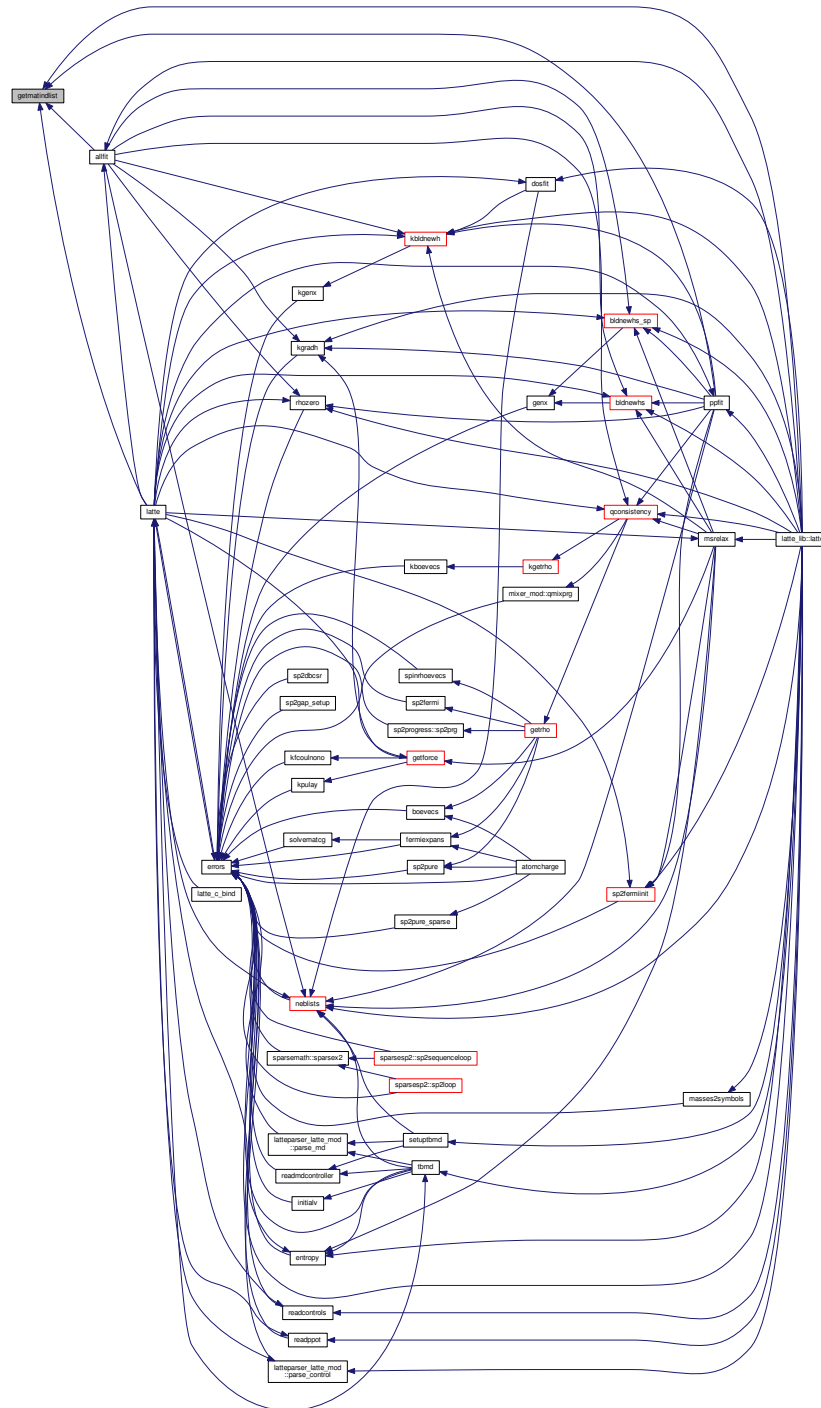
- subroutine [getmatindlist](#)

8.169.1 Function/Subroutine Documentation

8.169.1.1 subroutine [getmatindlist](#) ()

Definition at line 23 of file [getmatindlist.f90](#).

Here is the caller graph for this function:



8.170 getmatindlist.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getmatindlist
00023
00024   USE constants_mod
00025   USE setuparray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J, INDI
00030   IF (existerror) RETURN
00031
00032   ! This arrays contains and index for the position of atoms
00033   ! in the Hamiltonian matrix. We only need to compute this once
00034
00035   IF (.NOT. ALLOCATED(matindlist)) ALLOCATE(matindlist(nats))
00036
00037   DO i = 1, nats
00038     indi = 0
00039     DO j = 1, i - 1
00040
00041       SELECT CASE(basis(elempointer(j)))
00042       CASE("s")
00043         indi = indi + 1
00044       CASE("p")
00045         indi = indi + 3
00046       CASE("d")
00047         indi = indi + 5
00048       CASE("f")
00049         indi = indi + 7
00050       CASE("sp")
00051         indi = indi + 4
00052       CASE("sd")
00053         indi = indi + 6
00054       CASE("sf")
00055         indi = indi + 8
00056       CASE("pd")
00057         indi = indi + 8
00058       CASE("pf")
00059         indi = indi + 10
00060       CASE("df")
00061         indi = indi + 12
00062       CASE("spd")
00063         indi = indi + 9
00064       CASE("spf")
00065         indi = indi + 11
00066       CASE("sdf")
00067         indi = indi + 13
00068       CASE("pdf")
00069         indi = indi + 15
00070       CASE("spdf")
00071         indi = indi + 16
00072       END SELECT
00073
00074     ENDDO
00075
00076     matindlist(i) = indi
00077
00078   ENDDO
00079
00080   IF (spinon .EQ. 1) THEN
00081     ALLOCATE(spinindlist(nats))
00082
00083     DO i = 1, nats
00084       indi = 0
00085       DO j = 1, i - 1
00086
00087         ! The number of spins = number of orbitals
00088
00089         SELECT CASE(basis(elempointer(j)))
00090         CASE("s")
00091           indi = indi + 1
00092         CASE("p")

```

```

00094         indi = indi + 1
00095         CASE ("d")
00096         indi = indi + 1
00097         CASE ("f")
00098         indi = indi + 1
00099         CASE ("sp")
00100         indi = indi + 2
00101         CASE ("sd")
00102         indi = indi + 2
00103         CASE ("sf")
00104         indi = indi + 2
00105         CASE ("pd")
00106         indi = indi + 2
00107         CASE ("pf")
00108         indi = indi + 2
00109         CASE ("df")
00110         indi = indi + 2
00111         CASE ("spd")
00112         indi = indi + 3
00113         CASE ("spf")
00114         indi = indi + 3
00115         CASE ("sdf")
00116         indi = indi + 3
00117         CASE ("pdf")
00118         indi = indi + 3
00119         CASE ("spdf")
00120         indi = indi + 4
00121         END SELECT
00122
00123         ENDDO
00124
00125         spinindlist(i) = indi
00126
00127         ENDDO
00128
00129     ENDIF
00130
00131     RETURN
00132
00133 END SUBROUTINE getmatindlist

```

8.171 getmaxf.f90 File Reference

Functions/Subroutines

- subroutine [getmaxf](#) (MAXF)

8.171.1 Function/Subroutine Documentation

8.171.1.1 subroutine [getmaxf](#) (real(latteprec) *MAXF*)

Definition at line 23 of file [getmaxf.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getmaxf(MAXF)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I
00031   REAL(LATTEPREC) :: MAXF, ABSF
00032   IF (existerror) RETURN
00033
00034   maxf = zero
00035
00036   DO i = 1, nats
00037
00038     absf = ftot(1,i)*ftot(1,i) + ftot(2,i)*ftot(2,i) + &
00039           ftot(3,i)*ftot(3,i)
00040
00041     IF (absf .GT. maxf) maxf = absf
00042
00043
00044   ENDDO
00045
00046   maxf = sqrt(maxf)
00047
00048   RETURN
00049
00050 END SUBROUTINE getmaxf

```

8.173 getmdf.f90 File Reference

Functions/Subroutines

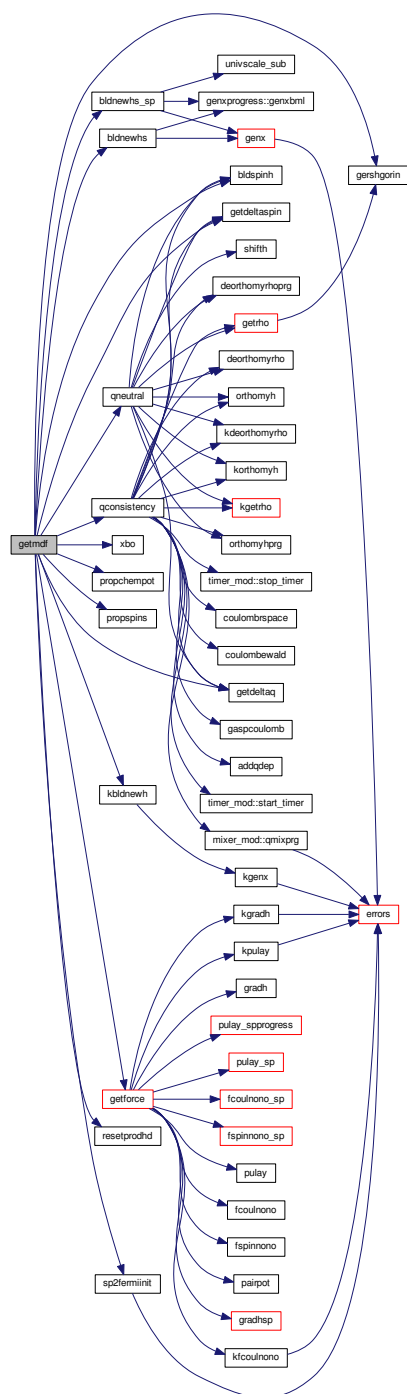
- subroutine [getmdf](#) (SWITCH, CURRITER)

8.173.1 Function/Subroutine Documentation

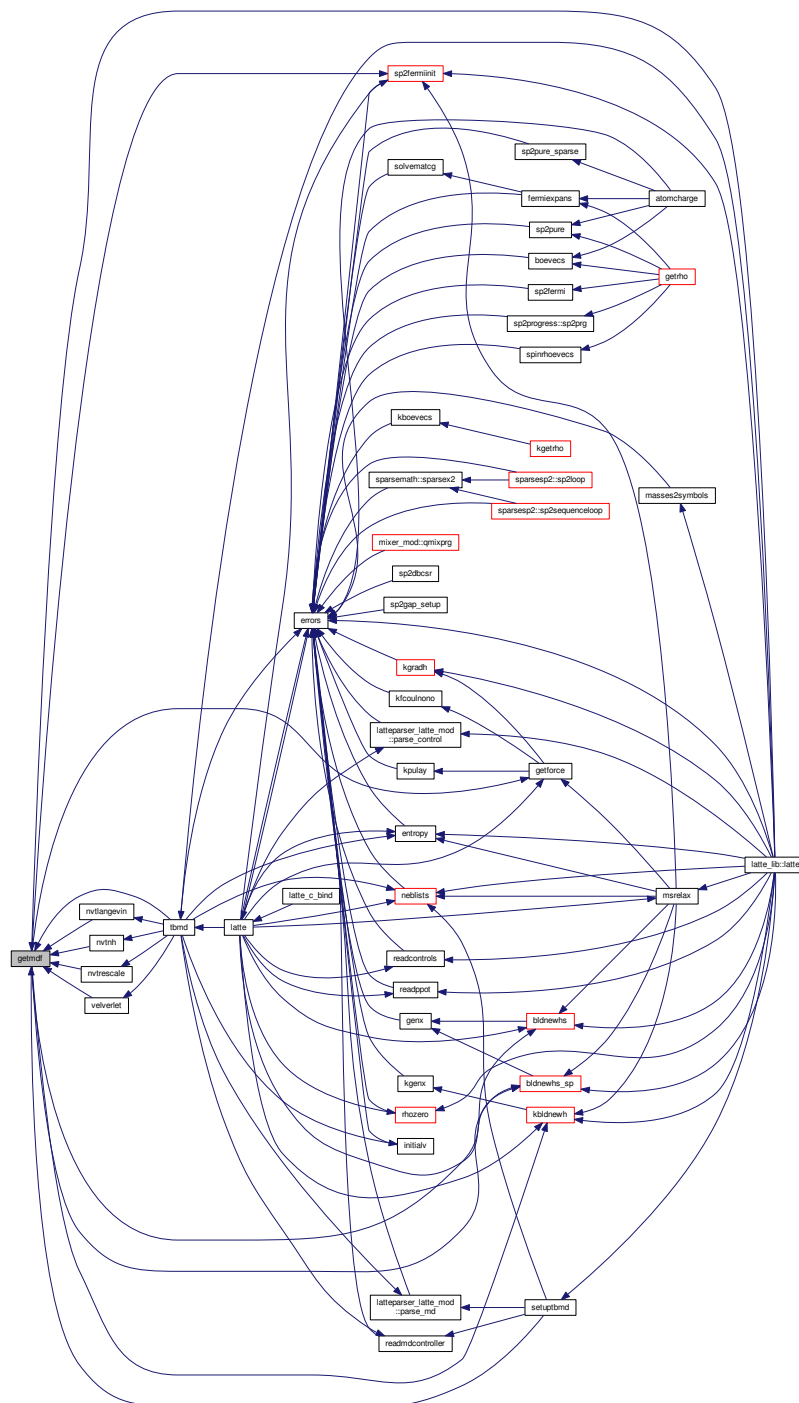
8.173.1.1 subroutine [getmdf](#) (integer SWITCH, integer CURRITER)

Definition at line 23 of file [getmdf.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.174 getmdf.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it       !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getmdf(SWITCH, CURRITER)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE xboarray
00028   USE myprecision
00029   USE coulombarray
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: SWITCH, CURRITER, I
00034   REAL(LATTEPREC) :: ZEROSECFMOD
00035   IF (existerror) RETURN
00036   !
00037   ! The atoms have moved, so we have to build a new H (and overlap matrix)
00038   !
00039   IF (kon .EQ. 0) THEN
00040     IF(verbose >= 1)WRITE(*,*)"KON = 0 ..."
00041     IF (sponly .EQ. 0) THEN
00042       CALL bldnewhs_sp
00043     ELSE
00044       CALL bldnewhs
00045     ENDIF
00046
00047   ELSE
00048     IF(verbose >= 1)WRITE(*,*)"KON = 1 ..."
00049     CALL kbldnewh
00050
00051   ENDIF
00052   CALL flush(6)
00053
00054   ! Broken?
00055
00056   ! IF (SWITCH .EQ. 0 .AND. RESTART .EQ. 1) CALL IFRESTART
00057
00058   IF (spinon .EQ. 1 .AND. curriter .EQ. 1) THEN
00059     CALL getdeltaspin
00060     CALL bldspinh
00061   ENDIF
00062
00063   IF (control .EQ. 5 .AND. curriter .EQ. 1) THEN
00064
00065     ! We do this to initialize SP2 Fermi
00066
00067     !   CONTROL = 2
00068     !   CALL QCONSISTENCY(0,1)
00069     !   CONTROL = 5
00070     CALL gershgorin
00071     CALL sp2fermiinit
00072   ENDIF
00073
00074   ! IF (ELECTRO .EQ. 1) THEN ! Self-consistent charge transfer on
00075   !
00076   ! If we're running with XBO on, we start to propagate things
00077   ! after the first iteration
00078   !
00079   !
00080
00081   IF (xboon .GT. 0 .AND. curriter .GT. 1 .AND. switch .NE. 0) THEN
00082
00083     IF(verbose >= 1)WRITE(*,*)".GT..AND..GT..AND..NE."XBOON 0 CURRITER 1 SWITCH 0 ..."
00084     IF (xboon .EQ. 1) THEN ! We will add other types of XBO later
00085
00086       ! Propagating partial charges or diagonal elements of H
00087       IF(verbose >= 1)WRITE(*,*)"Doing XBO ..."
00088       CALL xbo(curriter) ! Propagate q's
00089
00090       !
00091       ! If we are also propagating the chemical potential
00092       !
00093

```

```

00094         IF (control .EQ. 1 .OR. control .EQ. 3 .OR. control .EQ. 5) THEN
00095
00096             CALL propchempot(curriter) ! Propagate mu
00097
00098         ENDIF
00099
00100         IF (spinon .EQ. 1) CALL propspins(curriter) ! Propagate m's
00101
00102         CALL flush(6)
00103
00104     ENDIF
00105
00106     CALL flush(6)
00107 ENDIF
00108
00109 !
00110 ! If SWITCH = 0, then we don't have a set of partials charges
00111 ! yet and we'll have to get them from the charge-independent
00112 ! H-matrix
00113 !
00114 ! Whether we're running full self consistency at each MD time step
00115 ! or only a user-specified number of iterations is determined by the
00116 ! value of FULLQCONV
00117 !
00118
00119 IF (electro .EQ. 0) THEN
00120     IF(verbose >= 1)WRITE(*,*)"Doing QNEUTRAL ..."
00121     CALL qneutral(switch, curriter) ! Local charge neutrality
00122 ELSE
00123     IF(verbose >= 1)WRITE(*,*)"Doing QCONSISTENCY ..."
00124     CALL qconsistency(switch, curriter) ! Self consistent charge transfer
00125 ENDIF
00126
00127 ! Run to self-consistency QITER = 0 -> only H(P) + D calculated ANDERS
00128
00129 !
00130 ! Setting up our XBO arrays after the first iteration
00131 !
00132
00133 ! We initialize once we have our first set of self-consistent q's, mu, and m's
00134
00135 IF (xboon .GT. 0 .AND. curriter .EQ. 1) THEN
00136
00137     IF (xboon .EQ. 1) THEN ! Other cases to come
00138
00139         IF(verbose >= 1)WRITE(*,*)"Doing XBO ..."
00140         CALL xbo(1)
00141
00142         IF (control .EQ. 1 .OR. control .EQ. 3 &
00143             .OR. control .EQ. 5) CALL propchempot(1)
00144
00145         IF (spinon .EQ. 1) CALL propspins(1)
00146
00147
00148     ENDIF
00149
00150 ENDIF
00151
00152 ! Setting up the XBO arrays
00153
00154 IF (xboon .EQ. 1 .AND. electro .EQ. 0) THEN
00155
00156     CALL resetprodhd
00157
00158     ! IF (CURRITER .EQ. 1) CALL XBO(1)
00159
00160 ENDIF
00161
00162 CALL flush(6)
00163 !
00164 ! Get the forces from the covalent part
00165 !
00166
00167 ! When we're done with qconsistency we have the non-orthogonal density
00168 ! matrix so we don't have to de-orthogonalize again here
00169
00170 IF(verbose >= 1)WRITE(*,*)"Getting forces ..."
00171 CALL getforce
00172
00173 IF (electro .EQ. 1 .AND. qiter .EQ. 0) THEN
00174
00175     olddeltaqs = deltaq ! save the propagated charges
00176     CALL getdeltaq ! Get updated set of partial charges
00177
00178     ecoul = zero
00179
00180     ftot = ftot - fcoul ! We're going to correct the electrostatic force

```



```

00181
00182     DO i = 1, nats
00183
00184         zeroscfmod = (two*deltaq(i) - olddeltags(i))/
00185         olddeltags(i)
00186         fcoul(1,i) = fcoul(1,i)*zeroscfmod
00187         fcoul(2,i) = fcoul(2,i)*zeroscfmod
00188         fcoul(3,i) = fcoul(3,i)*zeroscfmod
00189         ecoul = ecoul + (two*deltaq(i) - olddeltags(i)) * &
00190         coulombv(i)
00191     ENDDO
00192
00193     ecoul = ecoul/two
00194
00195     ftot = ftot + fcoul
00196
00197 ENDIF
00198
00199 CALL flush(6)
00200
00201 RETURN
00202
00203 END SUBROUTINE getmdf

```

8.175 getpressure.f90 File Reference

Functions/Subroutines

- subroutine [getpressure](#)

8.175.1 Function/Subroutine Documentation

8.175.1.1 subroutine [getpressure](#) ()

Definition at line 23 of file [getpressure.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General  !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getpressure
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE virialarray
00027   USE mdarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J
00033   REAL(LATTEPREC) :: MYMASS, TPRESS
00034   REAL(LATTEPREC) :: MYP(5)
00035   IF (existerror) RETURN
00036   keten = zero
00037
00038   !   SYSVOL = BOXDIMS(1)*BOXDIMS(2)*BOXDIMS(3)
00039
00040   sysvol = abs(box(1,1)*(box(2,2)*box(3,3) - box(3,2)*box(2,3)) + &
00041             box(1,2)*(box(2,1)*box(3,3) - box(3,1)*box(2,3)) + &
00042             box(1,3)*(box(2,1)*box(3,2) - box(3,1)*box(2,2)))
00043
00044   ! Already have an extra factor of 2 in virbond
00045
00046   IF (electro .EQ. 0) vircoul = zero
00047
00048   !   MYP = ZERO
00049   !   DO J = 1, 3
00050   !       MYP(1) = MYP(1) + VIRBOND(J)
00051   !       MYP(2) = MYP(2) + VIRPAIR(J)
00052   !       MYP(3) = MYP(3) + VIRPUL(J)
00053   !       MYP(4) = MYP(4) + VIRCOUL(J)
00054   !       MYP(5) = MYP(5) + VIRSCOUL(J)
00055   !   ENDDO
00056
00057   !   MYP = MYP/THREE
00058   !   MYP = -MYP*TOGPA/SYSVOL
00059
00060   !   PRINT*, MYP(1), MYP(2), MYP(3), MYP(4), MYP(5), &
00061   !       MYP(1)+ MYP(2)- MYP(3)+ MYP(4)+ MYP(5)
00062
00063   !   PRINT*, VIRBOND(1), VIRPAIR(1), VIRPUL(1), VIRCOUL(1), VIRSCOUL(1), &
00064   !       VIRBOND(1) + VIRPAIR(1) - VIRPUL(1)+ VIRCOUL(1) + VIRSCOUL(1)
00065
00066   virial = virbond + virpair + vircoul
00067
00068   IF (basistype .EQ. "NONORTHO") THEN
00069
00070       IF (spinon .EQ. 0) THEN
00071           virial = virial - virpul + virscoul
00072       ELSE
00073           virial = virial - virpul + virscoul + virsspin
00074       ENDIF
00075
00076   ENDIF
00077
00078   IF ( mdon .EQ. 1 ) THEN
00079
00080       DO i = 1, nats
00081
00082           mymass = mass(elempointer(i))
00083
00084           keten(1) = keten(1) + mymass*v(1,i)*v(1,i)
00085           keten(2) = keten(2) + mymass*v(2,i)*v(2,i)
00086           keten(3) = keten(3) + mymass*v(3,i)*v(3,i)
00087           keten(4) = keten(4) + mymass*v(1,i)*v(2,i)
00088           keten(5) = keten(5) + mymass*v(2,i)*v(3,i)
00089           keten(6) = keten(6) + mymass*v(3,i)*v(1,i)
00090
00091       ENDDO
00092
00093   CALL getke

```

```

00094
00095      tpress = (REAL(nats)/SYSVOL)*temperature/ke2t
00096
00097  ENDIF
00098
00099  ! Minus the virial sum because we have Rij = Rj - Ri. See Allen
00100  ! and Tildesley.
00101
00102  ! STRTEN(1) = ( VIRCOUL(1) + KETEN(1)/F2V ) / SYSVOL
00103  ! STRTEN(2) = ( VIRCOUL(2) + KETEN(2)/F2V ) / SYSVOL
00104  ! STRTEN(3) = ( VIRCOUL(3) + KETEN(3)/F2V ) / SYSVOL
00105  ! STRTEN(4) = ( VIRBOND(4) + KETEN(4)/F2V ) / SYSVOL
00106  ! STRTEN(5) = ( VIRBOND(5) + KETEN(5)/F2V ) / SYSVOL
00107  ! STRTEN(6) = ( VIRBOND(6) + KETEN(6)/F2V ) / SYSVOL
00108
00109  strten(1) = ( -virial(1) + keten(1)/f2v ) / sysvol
00110  strten(2) = ( -virial(2) + keten(2)/f2v ) / sysvol
00111  strten(3) = ( -virial(3) + keten(3)/f2v ) / sysvol
00112  strten(4) = ( -virial(4) + keten(4)/f2v ) / sysvol
00113  strten(5) = ( -virial(5) + keten(5)/f2v ) / sysvol
00114  strten(6) = ( -virial(6) + keten(6)/f2v ) / sysvol
00115
00116
00117  !
00118  ! That's right folks: we're going to be reporting pressure in
00119  ! GPa since that's what some of us materials science/shock physics
00120  ! people like to deal with. STRTEN is still in eV/A^3
00121  !
00122
00123  strten = strten * togpa
00124
00125  pressure = (strten(1) + strten(2) + strten(3))/three
00126
00127  RETURN
00128
00129  END SUBROUTINE getpressure

```

8.177 getrespf.f90 File Reference

Functions/Subroutines

- subroutine [getrespf](#)

8.177.1 Function/Subroutine Documentation

8.177.1.1 subroutine getrespf ()

Definition at line 23 of file [getrespf.f90](#).

8.178 getrespf.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !

```

```

00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getrespf
00023
00024 ! Use our already computed eigenvalues and eigenvectors to compute
00025 ! the response function while enforcing LCN
00026
00027 USE constants_mod
00028 USE setuparray
00029 USE diagarray
00030 USE kspacearray
00031 USE myprecision
00032
00033 IMPLICIT NONE
00034
00035 INTEGER :: I, J, K, KK, INDEX, NUMORB
00036 REAL(LATTEPREC), ALLOCATABLE :: RESPF(:)
00037 IF (existerror) RETURN
00038
00039 ALLOCATE(respf(hdim))
00040
00041 respf = zero
00042
00043 IF (kon .EQ. 0) THEN
00044
00045     DO i = 1, hdim
00046
00047         IF (evals(i) .LE. chempot) THEN
00048
00049             DO j = i, hdim
00050
00051                 IF (evals(j) .GT. chempot .AND. abs(evals(i) -
00052 evals(j)) .GT. 1.0d-4) THEN
00053
00054                     DO k = 1, hdim
00055
00056                         respf(k) = respf(k) + &
00057                             two*evecs(k,i)*evecs(k,i)*evecs(k,j)*
00058                             evecs(k,j)/ &
00059                             (evals(i) - evals(j))
00060
00061                     ENDDO
00062
00063                 ENDIF
00064
00065             ENDDO
00066
00067         ENDIF
00068
00069     ELSE ! K-space integration now
00070
00071         DO kk = 1, nktot
00072
00073             DO i = 1, hdim
00074
00075                 IF (kevals(i, kk) .LE. chempot) THEN
00076
00077                     DO j = i, hdim
00078
00079                         IF (kevals(j, kk) .GT. chempot) THEN
00080
00081                             DO k = 1, hdim
00082
00083                                 respf(k) = respf(k) - &
00084                                     (four/(kevals(i, kk) - kevals(j, kk))) * &
00085                                     REAL(conjg(kevecs(k, i, kk)) * kevecs(k, i, kk) &
00086                                     * CONJG(kevecs(k, j, kk)) * KEVECS(k, j, kk))
00087
00088                             ENDDO
00089
00090                         ENDIF
00091
00092                     ENDDO
00093
00094                 ENDIF
00095
00096             ENDDO
00097
00098         ENDDO
00099
00100         respf = respf/REAL(nktot)
00101
00102     ENDIF

```

```

00103     respchi = zero
00104     index = 0
00105
00106     DO i = 1, nats
00107
00108         SELECT CASE(basis(elempointer(i)))
00109
00110             CASE("s")
00111                 numorb = 1
00112             CASE("p")
00113                 numorb = 3
00114             CASE("d")
00115                 numorb = 5
00116             CASE("f")
00117                 numorb = 7
00118             CASE("sp")
00119                 numorb = 4
00120             CASE("sd")
00121                 numorb = 6
00122             CASE("sf")
00123                 numorb = 8
00124             CASE("pd")
00125                 numorb = 8
00126             CASE("pf")
00127                 numorb = 10
00128             CASE("df")
00129                 numorb = 12
00130             CASE("spd")
00131                 numorb = 9
00132             CASE("spf")
00133                 numorb = 11
00134             CASE("sdf")
00135                 numorb = 13
00136             CASE("pdf")
00137                 numorb = 15
00138             CASE("spdf")
00139                 numorb = 16
00140             END SELECT
00141
00142         DO j = 1, numorb
00143             index = index + 1
00144             respchi(i) = respchi(i) + respf(index)
00145         ENDDO
00146
00147         respchi(i) = respchi(i)/REAL(numorb)
00148
00149     ENDDO
00150
00151     ! DO I = 1, NATS
00152     !     PRINT*, RESPCHI(I)
00153     ! ENDDO
00154
00155     DEALLOCATE(respf)
00156
00157     RETURN
00158
00159 END SUBROUTINE getrespf
00160
00161

```

8.179 getrho.f90 File Reference

Functions/Subroutines

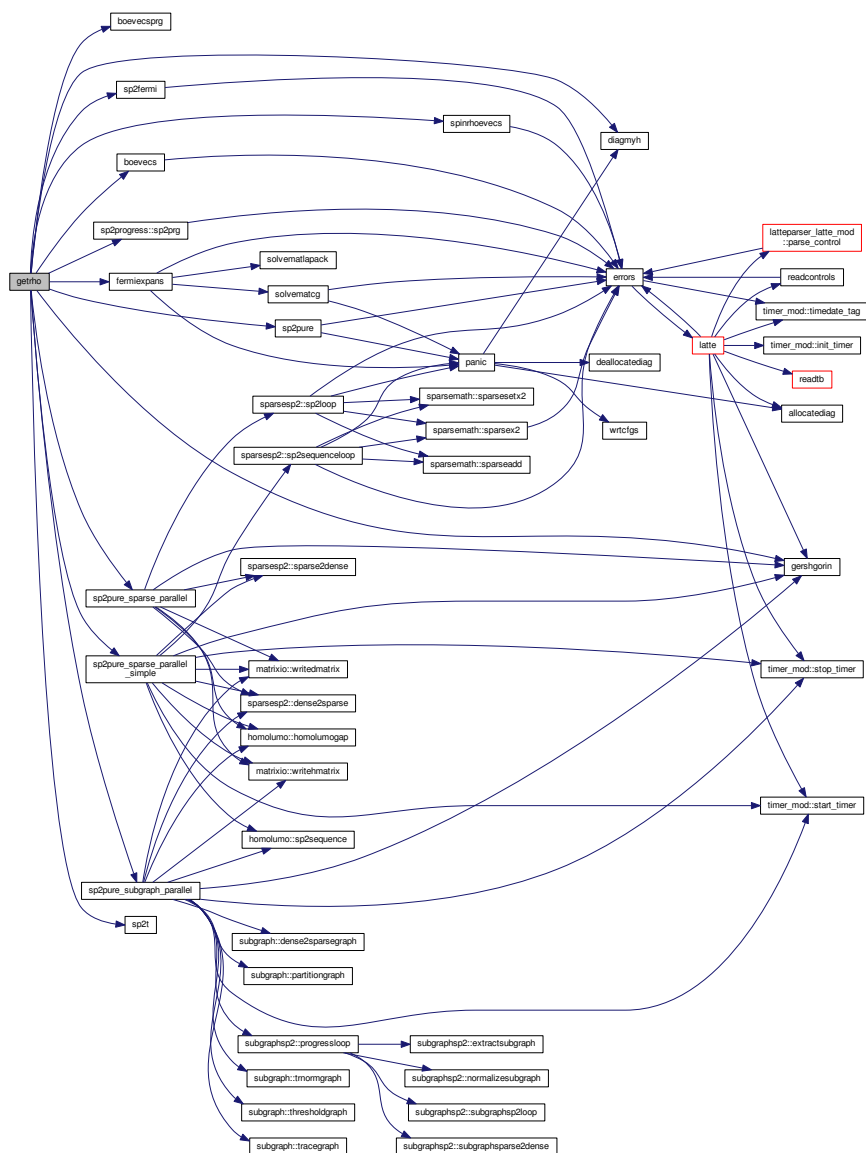
- subroutine [getrho](#) (MDITER)

8.179.1 Function/Subroutine Documentation

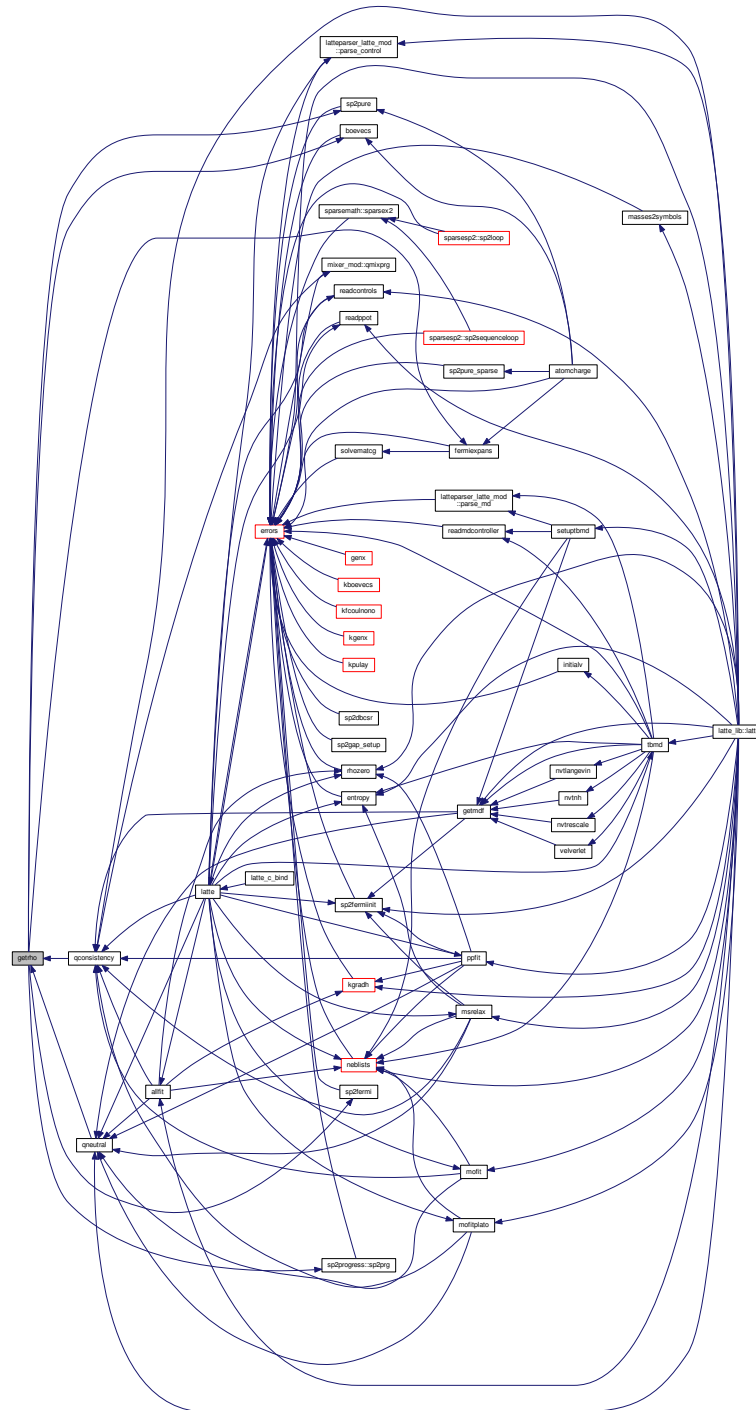
8.179.1.1 subroutine getrho (integer MDITER)

Definition at line 23 of file [getrho.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.180 getrho.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getrho(MDITER)
00023
00024   USE constants_mod
00025
00026   #ifdef PROGRESSON
00027     USE sp2progress
00028   #endif
00029
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: MDITER
00034   IF (existerror) RETURN
00035
00036   ! This subroutine selects and calls the subroutines used to
00037   ! compute the density matrix based on the value of CONTROL
00038
00039   ! CONTROL = 1 : DIAGONALIZATION
00040   ! CONTROL = 2 : SP2
00041   ! CONTROL = 3 : FERMI OPERATOR EXPANSION (FINITE T)
00042   ! CONTROL = 4, 5 : EXPERIMENTAL FINITE T SP2
00043
00044   IF (control .EQ. 1) THEN
00045     #ifdef PROGRESSON
00046       IF (spinon .EQ. 0) THEN
00047         CALL boevecsprg()
00048       ELSE
00049         CALL diagmyh()
00050         CALL spinrhoevcs
00051         WRITE(*,*)"This is using the original LATTE routine. Spin-polarized non yet with PROGRESS/BML"
00052       ENDIF
00053     #else
00054       CALL diagmyh()
00055       IF (spinon .EQ. 0) THEN
00056         CALL boevecs()
00057       ELSE
00058         CALL spinrhoevcs
00059       ENDIF
00060     #endif
00061
00062   ELSEIF (control .EQ. 2) THEN
00063
00064     IF (sparseon .EQ. 0) THEN
00065
00066       CALL gershgorin
00067
00068       CALL sp2pure
00069
00070       !           IF (MDITER .LE. 10) THEN
00071       !             CALL SP2GAP_SETUP
00072       !           ELSE
00073       !             CALL SP2GAP
00074       !           ENDIF
00075
00076     ELSEIF (sparseon .EQ. 1) THEN
00077
00078       !! CALL SP2PURE_PARSE
00079
00080     #ifdef PROGRESSON
00081       CALL sp2prg
00082
00083     #else
00084       IF (mditer .LE. 10) THEN
00085         CALL sp2pure_sparse_parallel(mditer)
00086       ELSE
00087         CALL sp2pure_sparse_parallel_simple(mditer)
00088       ENDIF
00089     #endif
00090
00091   #endif
00092
00093

```

```

00094      ELSEIF (sparseon .EQ. 2) THEN
00095
00096          IF (mditer .LE. 10) THEN
00097              CALL sp2pure_sparse_parallel(mditer)
00098          ELSE
00099              CALL sp2pure_subgraph_parallel(mditer)
00100          ENDIF
00101      ENDIF
00102  ENDIF
00103
00104      !      ELSEIF (SPARSEON .EQ. 1) THEN
00105      !
00106      !          CALL SP2PURE_SPARSE
00107      !          CALL SP2PURE_SPARSE_PARALLEL(MDITER)
00108      !      ELSEIF (SPARSEON .EQ. 2) THEN
00109      !          CALL SP2PURE_SPARSE_PARALLEL(MDITER)
00110      !      ENDIF
00111      !
00112
00113      ELSEIF (control .EQ. 3) THEN
00114
00115          CALL fermiexpans
00116
00117      ELSEIF (control .EQ. 4) THEN
00118
00119          CALL gershgorin
00120          CALL sp2t
00121
00122      ELSEIF (control .EQ. 5) THEN
00123
00124          CALL sp2fermi
00125
00126      ENDIF
00127
00128      RETURN
00129
00130 END SUBROUTINE getrho

```

8.181 getspinE.f90 File Reference

Functions/Subroutines

- subroutine [getspine](#)

8.181.1 Function/Subroutine Documentation

8.181.1.1 subroutine [getspine](#) ()

Definition at line 23 of file [getspinE.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getspine
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I, INDEX
00031   REAL(LATTEPREC) :: DSPINS, DSPINP, DSPIND, DSPINF
00032   IF (existerror) RETURN
00033
00034   espin = zero
00035   index = 0
00036
00037   DO i = 1, nats
00038
00039     SELECT CASE(basis(elempointer(i)))
00040
00041     CASE("s")
00042
00043       dspins = deltaspin(index + 1)
00044       index = index + 1
00045
00046       espin = espin + wss(elempointer(i))*dspins*dspins
00047
00048     CASE("p")
00049
00050       dspinp = deltaspin(index + 1)
00051       index = index + 1
00052
00053       espin = espin + wpp(elempointer(i))*dspinp*dspinp
00054
00055     CASE("d")
00056
00057       dspind = deltaspin(index + 1)
00058       index = index + 1
00059
00060       espin = espin + wdd(elempointer(i))*dspind*dspind
00061
00062     CASE("f")
00063
00064       dspinf = deltaspin(index + 1)
00065       index = index + 1
00066
00067       espin = espin + wff(elempointer(i))*dspinf*dspinf
00068
00069     CASE("sp")
00070
00071       dspins = deltaspin(index + 1)
00072       dspinp = deltaspin(index + 2)
00073       index = index + 2
00074
00075       espin = espin + wss(elempointer(i))*dspins*dspins + &
00076         wpp(elempointer(i))*dspinp*dspinp
00077
00078     CASE("sd")
00079
00080       dspins = deltaspin(index + 1)
00081       dspind = deltaspin(index + 2)
00082       index = index + 2
00083
00084       espin = espin + wss(elempointer(i))*dspins*dspins + &
00085         wdd(elempointer(i))*dspind*dspind
00086
00087     CASE("sf")
00088
00089       dspins = deltaspin(index + 1)
00090       dspinf = deltaspin(index + 2)
00091       index = index + 2
00092
00093       espin = espin + wss(elempointer(i))*dspins*dspins + &

```

```

00094         wff(elempointer(i))*dspinf*dspinf
00095
00096     CASE("pd")
00097
00098         dspinp = deltaspin(index + 1)
00099         dspind = deltaspin(index + 2)
00100         index = index + 2
00101
00102         espin = espin + wpp(elempointer(i))*dspinp*dspinp + &
00103             wdd(elempointer(i))*dspind*dspind
00104
00105     CASE("pf")
00106
00107         dspinp = deltaspin(index + 1)
00108         dspinf = deltaspin(index + 2)
00109         index = index + 2
00110
00111         espin = espin + wpp(elempointer(i))*dspinp*dspinp + &
00112             wff(elempointer(i))*dspinf*dspinf
00113
00114     CASE("df")
00115
00116         dspind = deltaspin(index + 1)
00117         dspinf = deltaspin(index + 2)
00118         index = index + 2
00119
00120         espin = espin + wdd(elempointer(i))*dspind*dspind + &
00121             wff(elempointer(i))*dspinf*dspinf
00122
00123     CASE("spd")
00124
00125         dspins = deltaspin(index + 1)
00126         dspinp = deltaspin(index + 2)
00127         dspind = deltaspin(index + 3)
00128         index = index + 3
00129
00130         espin = espin + wss(elempointer(i))*dspins*dspins + &
00131             wpp(elempointer(i))*dspinp*dspinp + &
00132             wdd(elempointer(i))*dspind*dspind
00133
00134     CASE("spf")
00135
00136         dspins = deltaspin(index + 1)
00137         dspinp = deltaspin(index + 2)
00138         dspinf = deltaspin(index + 3)
00139         index = index + 3
00140
00141         espin = espin + wss(elempointer(i))*dspins*dspins + &
00142             wpp(elempointer(i))*dspinp*dspinp + &
00143             wff(elempointer(i))*dspinf*dspinf
00144
00145     CASE("sdf")
00146
00147         dspins = deltaspin(index + 1)
00148         dspind = deltaspin(index + 2)
00149         dspinf = deltaspin(index + 3)
00150         index = index + 3
00151
00152         espin = espin + wss(elempointer(i))*dspins*dspins + &
00153             wdd(elempointer(i))*dspind*dspind + &
00154             wff(elempointer(i))*dspinf*dspinf
00155
00156     CASE("pdf")
00157
00158         dspinp = deltaspin(index + 1)
00159         dspind = deltaspin(index + 2)
00160         dspinf = deltaspin(index + 3)
00161         index = index + 3
00162
00163         espin = espin + wpp(elempointer(i))*dspinp*dspinp + &
00164             wdd(elempointer(i))*dspind*dspind + &
00165             wff(elempointer(i))*dspinf*dspinf
00166
00167     CASE("spdf")
00168
00169         dspins = deltaspin(index + 1)
00170         dspinp = deltaspin(index + 2)
00171         dspind = deltaspin(index + 3)
00172         dspinf = deltaspin(index + 4)
00173         index = index + 4
00174
00175         espin = espin + wss(elempointer(i))*dspins*dspins + &
00176             wpp(elempointer(i))*dspinp*dspinp + &
00177             wdd(elempointer(i))*dspind*dspind + &
00178             wff(elempointer(i))*dspinf*dspinf
00179
00180     END SELECT

```

```

00181
00182     ENDDO
00183
00184     espin = half*espin
00185
00186     RETURN
00187
00188 END SUBROUTINE getspine

```

8.183 getspineE_zero.f90 File Reference

Functions/Subroutines

- subroutine [getspinezero](#)

8.183.1 Function/Subroutine Documentation

8.183.1.1 subroutine getspinezero ()

Definition at line 23 of file [getspineE_zero.f90](#).

8.184 getspineE_zero.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE getspinezero
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE spinarray
00027
00028     IMPLICIT NONE
00029
00030     INTEGER :: I, J, K, MYINDEX
00031     REAL(LATTEPREC) :: DSPINS, DSPINP, DSPIND, DSPINF
00032
00033     espin_zero = zero
00034     myindex = 0
00035
00036     DO i = 1, nats
00037
00038         SELECT CASE(basis(elempointer(i)))
00039
00040             CASE("s")
00041
00042                 myindex = myindex + 1
00043                 dspins = rhoupzero(myindex) - rhodownzero(myindex)
00044                 espin_zero = espin_zero + wss(elempointer(i))*dspins*dspins
00045

```

```

00046      CASE("p")
00047
00048          myindex = myindex + 1
00049          dspins = rhoupzero(myindex) - rhodownzero(myindex)
00050
00051      IF (basis(elempointer(i)) .EQ. "s") THEN
00052
00053          myindex = myindex + 1
00054          dspins = rhoupzero(myindex) - rhodownzero(myindex)
00055
00056          espin_zero = espin_zero + wss(elempointer(i))*dspins*dspins
00057
00058      ELSEIF (basis(elempointer(i)) .EQ. "sp") THEN
00059
00060          myindex = myindex + 1
00061
00062          dspins = rhoupzero(myindex) - rhodownzero(myindex)
00063
00064          dspinp = zero
00065
00066          myindex = myindex + 1
00067          dspinp = dspinp + rhoupzero(myindex) - rhodownzero(myindex)
00068          myindex = myindex + 1
00069          dspinp = dspinp + rhoupzero(myindex) - rhodownzero(myindex)
00070          myindex = myindex + 1
00071          dspinp = dspinp + rhoupzero(myindex) - rhodownzero(myindex)
00072
00073          espin_zero = espin_zero + wss(elempointer(i))*dspins*dspins +
00074      &
00075          ! TWO*WSP(ELEMPONTER(I))*DSPINS*DSPINP + &
00076          wpp(elempointer(i))*dspinp*dspinp
00077
00078      ENDIF
00079
00080      ENDDO
00081
00082      espin_zero = half*espin_zero
00083
00084      ! PRINT*, "ESPIN_ZERO= ", ESPIN_ZERO, DSPINP
00085      RETURN
00086  END SUBROUTINE getspinezero

```

8.185 getsplinenr.f90 File Reference

Functions/Subroutines

- program [spline](#)
- subroutine [splint](#) (X, Y, Y2, R, NEWY, GRAD)

8.185.1 Function/Subroutine Documentation

8.185.1.1 program spline ()

Definition at line 1 of file [getsplinenr.f90](#).

Here is the call graph for this function:



8.185.1.2 subroutine splint (real, dimension(n) X, real, dimension(n) Y, real, dimension(n) Y2, real R, real NEWY, real GRAD)

Definition at line 51 of file [getsplinenr.f90](#).

Here is the caller graph for this function:



8.186 getsplinenr.f90

```

00001 PROGRAM spline
00002
00003 ! This is taken from Numerical recipes
00004 IMPLICIT NONE
00005
00006 INTEGER :: I, J, K
00007 INTEGER, PARAMETER :: N = 499
00008 REAL :: X(n), Y(n), Y2(n), P, QN, SIG, UN, U(n)
00009 REAL :: R, NEWY, GRAD
00010
00011 OPEN(unit=11, status="OLD", file="inputpoints.dat")
00012 OPEN(unit=12, status="UNKNOWN", file="checknumrep.dat")
00013
00014 DO i = 1, n
00015     READ(11,*) x(i), y(i)
00016 ENDDO
00017
00018 y2(1) = 0.0
00019 u(1) = 0.0
00020
00021 DO i = 2, n-1
00022     sig = (x(i) - x(i-1))/(x(i+1) - x(i-1))
00023     p = sig*y2(i-1) + 2.0
00024     y2(i) = (sig - 1.0)/p
00025     u(i) = (6.0*((y(i+1) - y(i))/(x(i+1) - x(i)) - (y(i) - y(i-1)) &
00026         / (x(i) - x(i-1)))/(x(i+1)-x(i-1)) - sig*u(i-1))/p
00027 ENDDO
00028
00029 qn = 0.0
00030 un = 0.0
00031
00032 y2(n) = (un-qn*u(n-1))/(qn*y2(n-1) + 1.0)
00033
00034 DO k = n-1, 1, -1
00035     y2(k) = y2(k)*y2(k+1) + u(k)
00036 ENDDO
00037
00038 DO i = 1, 1000
00039
00040     r = 0.5 + 0.5*REAL(i-1)/1000.0
00041
00042
00043     CALL splint(x, y, y2, r, newy, grad)
00044
00045     WRITE(12,*) r, newy, grad
00046 ENDDO
00047
00048 END PROGRAM spline
00049
00050 SUBROUTINE splint(X, Y, Y2, R, NEWY, GRAD)
00051
00052 IMPLICIT NONE
00053
00054 INTEGER :: K, KHI, KLO
00055 INTEGER, PARAMETER :: N = 499
  
```



```

00056  REAL :: X(n), Y(n), Y2(n), R, NEWY, A, B, H, GRAD
00057
00058  klo = 1
00059  khi = n
00060
00061  DO WHILE(khi - klo .GT. 1)
00062      k = (khi + klo)/2
00063      IF (x(k) .GT. r) THEN
00064          khi = k
00065      ELSE
00066          klo = k
00067      ENDIF
00068  ENDDO
00069
00070  h = x(khi) - x(klo)
00071
00072  a = (x(khi) - r)/h
00073  b = (r - x(klo))/h
00074
00075  newy = a*y(klo) + b*y(khi) + &
00076      ((a*a*a - a)*y2(klo) + (b*b*b - b)*y2(khi))*(h*h/6.0)
00077
00078  grad = -y(klo)/h + y(khi)/h + &
00079      ((3.0*a*a*(-1.0/h) + 1.0/h)*y2(klo) + &
00080      (3.0*b*b*(1.0/h) - 1.0/h)*y2(khi))*(h*h/6.0)
00081
00082
00083  RETURN
00084
00085  END SUBROUTINE splint
00086
00087

```

8.187 gradH.f90 File Reference

Functions/Subroutines

- subroutine [gradh](#)

8.187.1 Function/Subroutine Documentation

8.187.1.1 subroutine [gradh](#) ()

Definition at line 23 of file [gradH.f90](#).

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gradh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE univarray
00028   USE spinarray
00029   USE virialarray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00035   INTEGER :: LBRA, MBRA, LKET, MKET
00036   INTEGER :: PREVJ, NEWJ
00037   INTEGER :: PBCI, PBCJ, PBCK
00038   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00039   REAL(LATTEPREC) :: ALPHA, BETA, PHI, RHO, COSBETA
00040   REAL(LATTEPREC) :: RIJ(3), DC(3)
00041   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2, FTMP(3)
00042   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00043   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00044   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00045   LOGICAL PATH
00046   IF (existerror) RETURN
00047
00048   f = zero
00049   virbond = zero
00050
00051   !$OMP PARALLEL DO DEFAULT (NONE) &
00052   !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00053   !$OMP SHARED(CR, BOX, BO, RHOU, RHODOWN, SPINON, NOINT, ATELE, ELE1, ELE2) &
00054   !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00055   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00056   !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00057   !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00058   !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB) &
00059   !$OMP REDUCTION(+:F, VIRBOND)
00060
00061   DO i = 1, nats
00062
00063     ! Build list of orbitals on atom I
00064     SELECT CASE(basis(elempointer(i)))
00065
00066       CASE("s")
00067         basisi(1) = 0
00068         basisi(2) = -1
00069       CASE("p")
00070         basisi(1) = 1
00071         basisi(2) = -1
00072       CASE("d")
00073         basisi(1) = 2
00074         basisi(2) = -1
00075       CASE("f")
00076         basisi(1) = 3
00077         basisi(2) = -1
00078       CASE("sp")
00079         basisi(1) = 0
00080         basisi(2) = 1
00081         basisi(3) = -1
00082       CASE("sd")
00083         basisi(1) = 0
00084         basisi(2) = 2
00085         basisi(3) = -1
00086       CASE("sf")
00087         basisi(1) = 0
00088         basisi(2) = 3
00089         basisi(3) = -1
00090       CASE("pd")
00091         basisi(1) = 1
00092         basisi(2) = 2
00093         basisi(3) = -1

```

```

00094      CASE("pf")
00095          basisi(1) = 1
00096          basisi(2) = 3
00097          basisi(3) = -1
00098      CASE("df")
00099          basisi(1) = 2
00100          basisi(2) = 3
00101          basisi(3) = -1
00102      CASE("spd")
00103          basisi(1) = 0
00104          basisi(2) = 1
00105          basisi(3) = 2
00106          basisi(4) = -1
00107      CASE("spf")
00108          basisi(1) = 0
00109          basisi(2) = 1
00110          basisi(3) = 3
00111          basisi(4) = -1
00112      CASE("sdf")
00113          basisi(1) = 0
00114          basisi(2) = 2
00115          basisi(3) = 3
00116          basisi(4) = -1
00117      CASE("pdf")
00118          basisi(1) = 1
00119          basisi(2) = 2
00120          basisi(3) = 3
00121          basisi(4) = -1
00122      CASE("spdf")
00123          basisi(1) = 0
00124          basisi(2) = 1
00125          basisi(3) = 2
00126          basisi(4) = 3
00127          basisi(5) = -1
00128      END SELECT
00129
00130      indi = matindlist(i)
00131
00132      DO newj = 1, totnebtb(i)
00133
00134          j = nebtb(1, newj, i)
00135          pbci = nebtb(2, newj, i)
00136          pbcj = nebtb(3, newj, i)
00137          pbck = nebtb(4, newj, i)
00138
00139          rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00140              REAL(pbck)*BOX(3,1) - CR(1,i)
00141
00142          rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00143              REAL(pbck)*BOX(3,2) - CR(2,i)
00144
00145          rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00146              REAL(pbck)*BOX(3,3) - CR(3,i)
00147
00148          magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00149
00150          ! Building the forces is expensive - use the cut-off
00151
00152          rcuttb = zero
00153          DO k = 1, noint
00154
00155              IF ( (atele(i) .EQ. ele1(k) .AND. &
00156                  atele(j) .EQ. ele2(k)) .OR. &
00157                  (atele(j) .EQ. ele1(k) .AND. &
00158                  atele(i) .EQ. ele2(k) ) ) THEN
00159
00160                  IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00161
00162                  IF (basistype .EQ. "NONORTHO") THEN
00163                      IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00164                  ENDIF
00165
00166              ENDIF
00167
00168          ENDDO
00169
00170          IF (magr2 .LT. rcuttb*rcuttb) THEN
00171
00172              magr = sqrt(magr2)
00173
00174              ! Build list of orbitals on atom J
00175
00176              SELECT CASE(basis(elempointer(j)))
00177              CASE("s")
00178                  basisj(1) = 0
00179                  basisj(2) = -1
00180              CASE("p")

```

```

00181         basisj(1) = 1
00182         basisj(2) = -1
00183         CASE("d")
00184             basisj(1) = 2
00185             basisj(2) = -1
00186         CASE("f")
00187             basisj(1) = 3
00188             basisj(2) = -1
00189         CASE("sp")
00190             basisj(1) = 0
00191             basisj(2) = 1
00192             basisj(3) = -1
00193         CASE("sd")
00194             basisj(1) = 0
00195             basisj(2) = 2
00196             basisj(3) = -1
00197         CASE("sf")
00198             basisj(1) = 0
00199             basisj(2) = 3
00200             basisj(3) = -1
00201         CASE("pd")
00202             basisj(1) = 1
00203             basisj(2) = 2
00204             basisj(3) = -1
00205         CASE("pf")
00206             basisj(1) = 1
00207             basisj(2) = 3
00208             basisj(3) = -1
00209         CASE("df")
00210             basisj(1) = 2
00211             basisj(2) = 3
00212             basisj(3) = -1
00213         CASE("spd")
00214             basisj(1) = 0
00215             basisj(2) = 1
00216             basisj(3) = 2
00217             basisj(4) = -1
00218         CASE("spf")
00219             basisj(1) = 0
00220             basisj(2) = 1
00221             basisj(3) = 3
00222             basisj(4) = -1
00223         CASE("sdf")
00224             basisj(1) = 0
00225             basisj(2) = 2
00226             basisj(3) = 3
00227             basisj(4) = -1
00228         CASE("pdf")
00229             basisj(1) = 1
00230             basisj(2) = 2
00231             basisj(3) = 3
00232             basisj(4) = -1
00233         CASE("spdf")
00234             basisj(1) = 0
00235             basisj(2) = 1
00236             basisj(3) = 2
00237             basisj(4) = 3
00238             basisj(5) = -1
00239         END SELECT
00240
00241         indj = matindlist(j)
00242
00243         magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00244         magrp = sqrt(magrp2)
00245
00246
00247         ! transform to system in which z-axis is aligned with RIJ
00248
00249         path = .false.
00250         IF (abs(rij(1)) .GT. 1e-12) THEN
00251             IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00252                 phi = zero
00253             ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00254                 phi = two * pi
00255             ELSE
00256                 phi = pi
00257             ENDIF
00258             alpha = atan(rij(2) / rij(1)) + phi
00259         ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00260             IF (rij(2) .GT. 1e-12) THEN
00261                 alpha = pi / two
00262             ELSE
00263                 alpha = three * pi / two
00264             ENDIF
00265         ELSE
00266             ! pathological case: pole in alpha at beta=0
00267             path = .true.

```

```

00268         ENDIF
00269
00270         cosbeta = rij(3)/magr
00271         beta = acos(rij(3) / magr)
00272
00273         !           PRINT*, ALPHA, BETA
00274
00275         dc = rij/magr
00276
00277         ! build forces using PRB 72 165107 eq. (12) - the sign of the
00278         ! dfda contribution seems to be wrong, but gives the right
00279         ! answer(?)
00280
00281         ftmp = zero
00282         k = indi
00283
00284         lbrainc = 1
00285         DO WHILE (basisi(lbrainc) .NE. -1)
00286
00287             lbra = basisi(lbrainc)
00288             lbrainc = lbrainc + 1
00289
00290             DO mbra = -lbra, lbra
00291
00292                 k = k + 1
00293                 l = indj
00294
00295                 lketinc = 1
00296                 DO WHILE (basisj(lketinc) .NE. -1)
00297
00298                     lket = basisj(lketinc)
00299                     lketinc = lketinc + 1
00300
00301                     DO mket = -lket, lket
00302
00303                         l = l + 1
00304
00305                         SELECT CASE(spinon)
00306                         CASE(0)
00307                             rho = bo(l, k)
00308                         CASE(1)
00309                             rho = rhoup(l, k) + rhodown(l, k)
00310                         END SELECT
00311
00312                         IF (.NOT. path) THEN
00313
00314                             ! Unroll loops and pre-compute
00315
00316                             mydfda = dfda(i, j, lbra, lket, mbra, &
00317                                 mket, magr, alpha, cosbeta, "H")
00318
00319                             mydfdb = dfdb(i, j, lbra, lket, mbra, &
00320                                 mket, magr, alpha, cosbeta, "H")
00321
00322                             mydfdr = dfdr(i, j, lbra, lket, mbra, &
00323                                 mket, magr, alpha, cosbeta, "H")
00324
00325                             !
00326                             ! d/d_alpha
00327                             !
00328
00329                             ftmp(1) = ftmp(1) + rho * &
00330                                 (-rij(2) / magrp2 * mydfda)
00331
00332                             ftmp(2) = ftmp(2) + rho * &
00333                                 (rij(1)/ magrp2 * mydfda)
00334
00335                             !
00336                             ! d/d_beta
00337                             !
00338
00339                             ftmp(1) = ftmp(1) + rho * &
00340                                 (((rij(3) * rij(1)) / &
00341                                     magr2)) / magrp) * mydfdb)
00342
00343                             ftmp(2) = ftmp(2) + rho * &
00344                                 (((rij(3) * rij(2)) / &
00345                                     magr2)) / magrp) * mydfdb)
00346
00347                             ftmp(3) = ftmp(3) - rho * &
00348                                 (((one - (rij(3) * rij(3)) / &
00349                                     magr2)) / magrp) * mydfdb)
00350
00351                             !
00352                             ! d/dR
00353                             !
00354

```

```

00355             ftmp(1) = ftmp(1) - rho * dc(1) * &
00356             mydfdr
00357
00358             ftmp(2) = ftmp(2) - rho * dc(2) * &
00359             mydfdr
00360
00361             ftmp(3) = ftmp(3) - rho * dc(3) * &
00362             mydfdr
00363
00364
00365             ELSE
00366
00367             ! pathological configuration in which beta=0
00368             ! or pi => alpha undefined
00369
00370             ! fixed: MJC 12/17/13
00371
00372             mydfdb = dfdb(i, j, lbra, lket, &
00373             mbra, mket, magr, zero, cosbeta, "H") / magr
00374
00375             ftmp(1) = ftmp(1) - rho * (cosbeta * mydfdb)
00376
00377             mydfdb = dfdb(i, j, lbra, lket, &
00378             mbra, mket, magr, pi/two, cosbeta, "H") / magr
00379
00380             ftmp(2) = ftmp(2) - rho * (cosbeta * mydfdb)
00381
00382             mydfdr = dfdr(i, j, lbra, lket, mbra, &
00383             mket, magr, zero, cosbeta, "H")
00384
00385             ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00386
00387             ENDF
00388
00389             ENDDO
00390             ENDDO
00391             ENDDO
00392             ENDDO
00393
00394             f(1,i) = f(1,i) + ftmp(1)
00395             f(2,i) = f(2,i) + ftmp(2)
00396             f(3,i) = f(3,i) + ftmp(3)
00397
00398             virbond(1) = virbond(1) + rij(1) * ftmp(1)
00399             virbond(2) = virbond(2) + rij(2) * ftmp(2)
00400             virbond(3) = virbond(3) + rij(3) * ftmp(3)
00401             virbond(4) = virbond(4) + rij(1) * ftmp(2)
00402             virbond(5) = virbond(5) + rij(2) * ftmp(3)
00403             virbond(6) = virbond(6) + rij(3) * ftmp(1)
00404
00405             ENDF
00406
00407             ENDDO
00408
00409             !      INDI = INDI + NORBI
00410
00411             ENDDO
00412
00413             !$OMP END PARALLEL DO
00414
00415             RETURN
00416
00417 END SUBROUTINE gradh

```

8.189 gradH_sp.f90 File Reference

Functions/Subroutines

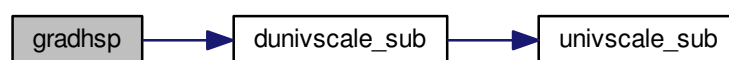
- subroutine [gradhsp](#)

8.189.1 Function/Subroutine Documentation

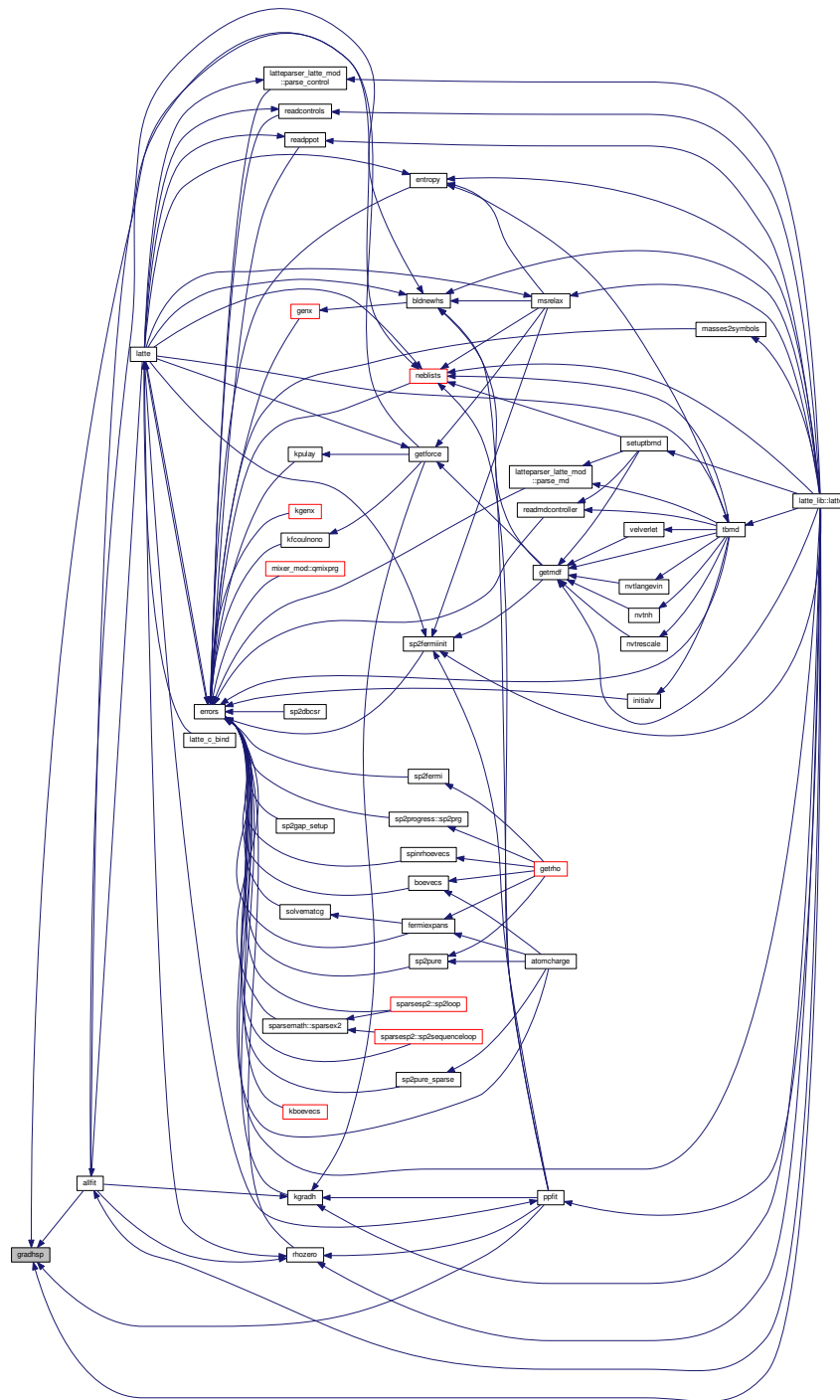
8.189.1.1 subroutine gradhsp ()

Definition at line 23 of file [gradH_sp.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.190 gradH_sp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE gradhsp
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE neblistarray
00028   USE spinarray
00029   USE virialarray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, KK, INDI, INDJ
00035   INTEGER :: NEWJ
00036   INTEGER :: PBCI, PBCJ, PBCK
00037   REAL(LATTEPREC) :: HSSS, HSPS, HPSS, HPPS, HPPP
00038   REAL(LATTEPREC) :: RIJ(3), DC(3), SCLGSP, DGSPDR(3)
00039   REAL(LATTEPREC) :: L, M, N, L2, M2, N2, LM, LN, MN, LMN
00040   REAL(LATTEPREC) :: DSSSDR(3), DSPSDR(3), DPSSDR(3), DPPSDR(3), DPPPDR(3)
00041   REAL(LATTEPREC) :: MAGR, INVR, FTMP(3)
00042   REAL(LATTEPREC) :: PPSMPPP, PPSUBINVR
00043   CHARACTER(LEN=2) :: BASISI, BASISJ
00044   IF (existerror) RETURN
00045
00046   f = zero
00047   virbond = zero
00048
00049   !$OMP PARALLEL DO DEFAULT (NONE) &
00050   !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00051   !$OMP SHARED(CR, BOX, BO, RHOUP, RHODOWN, NOINT, ATELE, ELE1, ELE2) &
00052   !$OMP SHARED(BOND, OVERL, MATINDLIST, BTYPE, BASISTYPE, SPINON) &
00053   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00054   !$OMP PRIVATE(RIJ, DC, MAGR, INVR, SCLGSP, DGSPDR, FTMP) &
00055   !$OMP PRIVATE(DSSSDR, DSPSDR, DPSSDR, DPPSDR, DPPPDR, PPSMPPP, PPSUBINVR)&
00056   !$OMP PRIVATE( L, M, N, L2, M2, N2, LM, LN, MN, LMN) &
00057   !$OMP PRIVATE(HSSS, HSPS, HPSS, HPPS, HPPP)&
00058   !$OMP REDUCTION(+:F, VIRBOND)
00059
00060
00061   DO i = 1, nats
00062
00063     basisi = basis(elempointer(i))
00064     indi = matindlist(i)
00065
00066
00067     ! Loop over all neighbors of I
00068
00069     DO newj = 1, totnebtb(i)
00070
00071       j = nebtb(1, newj, i)
00072       pbcj = nebtb(2, newj, i)
00073       pbcj = nebtb(3, newj, i)
00074       pbck = nebtb(4, newj, i)
00075
00076       indj = matindlist(j)
00077
00078       basisj = basis(elempointer(j))
00079
00080       rij(1) = cr(1,j) + REAL(pbcj)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00081       REAL(pbck)*BOX(3,1) - CR(1,i)
00082
00083       rij(2) = cr(2,j) + REAL(pbcj)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00084       REAL(pbck)*BOX(3,2) - CR(2,i)
00085
00086       rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00087       REAL(pbck)*BOX(3,3) - CR(3,i)
00088
00089       magr = sqrt(rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3))
00090
00091       invr = one/magr
00092
00093       ftmp = zero

```

```

00094
00095      !
00096      ! Direction cosines (DC)
00097      !
00098
00099      dc = rij/magr
00100
00101      l = dc(1)
00102      m = dc(2)
00103      n = dc(3)
00104
00105      !
00106      ! We'll need the GSP and gradient of the GSP
00107      !
00108
00109      IF (basisi .EQ. "s") THEN
00110
00111          IF (basisj .EQ. "s") THEN
00112
00113              DO k = 1, noint
00114                  IF ((atele(i) .EQ. ele1(k) .AND. &
00115                     atele(j) .EQ. ele2(k)) .OR. &
00116                     (atele(i) .EQ. ele2(k) .AND. &
00117                     atele(j) .EQ. ele1(k))) THEN
00118
00119                      IF (btype(k) .EQ. "sss") THEN
00120
00121                          CALL dunivscale_sub(magr, bond(:,k), dc, hsss, dssssdr)
00122
00123                      ENDIF
00124
00125                  ENDIF
00126              ENDDO
00127
00128              IF (spinon .EQ. 0) THEN
00129
00130                  ftmp = ftmp - bo(indi+1, indj+1)* dssssdr
00131
00132              ELSE
00133
00134                  ftmp = ftmp - dssssdr*(rhoup(indi+1, indj+1) + &
00135                     rhodown(indi+1, indj+1))
00136
00137              ENDIF
00138
00139          ELSEIF (basisj .EQ. "sp") THEN
00140
00141              DO k = 1, noint
00142
00143                  IF ((atele(i) .EQ. ele1(k) .AND. &
00144                     atele(j) .EQ. ele2(k)) .OR. &
00145                     (atele(i) .EQ. ele2(k) .AND. &
00146                     atele(j) .EQ. ele1(k))) THEN
00147
00148                      IF (btype(k) .EQ. "sss") THEN
00149
00150                          CALL dunivscale_sub(magr, bond(:,k), dc, hsss, dssssdr)
00151
00152                      ELSEIF (btype(k) .EQ. "sps") THEN
00153
00154                          CALL dunivscale_sub(magr, bond(:,k), dc, hsps, dspsdr)
00155
00156                      ENDIF
00157                  ENDIF
00158              ENDDO
00159
00160              l2 = l*l
00161              m2 = m*m
00162              n2 = n*n
00163              lm = l*m
00164              ln = l*n
00165              mn = m*n
00166
00167              IF (spinon .EQ. 0) THEN
00168
00169                  ! E_s1,s2
00170
00171                  ftmp = ftmp - bo(indi+1, indj+1)*dssssdr
00172
00173                  ! E_s1,x2
00174
00175                  ftmp(1) = ftmp(1) - bo(indi+1, indj+2) * &
00176                     (1*dspsdr(1) + (l2 - one)*invr*hsps)
00177
00178                  ftmp(2) = ftmp(2) - bo(indi+1, indj+2) * &
00179                     (1*dspsdr(2) + lm*invr*hsps)
00180

```

```

00181      ftmp(3) = ftmp(3) - bo(indi+1, indj+2) * &
00182      (l*dspdr(3) + ln*invr*hsps)
00183
00184      ! E_s1,y2
00185
00186      ftmp(1) = ftmp(1) - bo(indi+1, indj+3) * &
00187      (m*dspdr(1) + lm*invr*hsps)
00188
00189      ftmp(2) = ftmp(2) - bo(indi+1, indj+3) * &
00190      (m*dspdr(2) + (m2 - one)*invr*hsps)
00191
00192      ftmp(3) = ftmp(3) - bo(indi+1, indj+3) * &
00193      (m*dspdr(3) + mn*invr*hsps)
00194
00195      ! E_s1,z2
00196
00197      ftmp(1) = ftmp(1) - bo(indi+1, indj+4) * &
00198      (n*dspdr(1) + ln*invr*hsps)
00199
00200      ftmp(2) = ftmp(2) - bo(indi+1, indj+4) * &
00201      (n*dspdr(2) + mn*invr*hsps)
00202
00203      ftmp(3) = ftmp(3) - bo(indi+1, indj+4) * &
00204      (n*dspdr(3) + (n2 - one)*invr*hsps)
00205
00206  ELSE
00207
00208      ! E_s1,s2
00209
00210      ftmp = ftmp - dssdr*(rhoup(indi+1, indj+1) + &
00211      rhodown(indi+1, indj+1))
00212
00213      ! E_s1,x2
00214
00215      ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+2) + &
00216      rhodown(indi+1, indj+2)) * &
00217      (l*dspdr(1) + (l2 - one)*invr*hsps)
00218
00219      ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+2) + &
00220      rhodown(indi+1, indj+2)) * &
00221      (l*dspdr(2) + lm*invr*hsps)
00222
00223      ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+2) + &
00224      rhodown(indi+1, indj+2)) * &
00225      (l*dspdr(3) + ln*invr*hsps)
00226
00227      ! E_s1,y2
00228
00229      ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+3) + &
00230      rhodown(indi+1, indj+3)) * &
00231      (m*dspdr(1) + lm*invr*hsps)
00232
00233      ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+3) + &
00234      rhodown(indi+1, indj+3)) * &
00235      (m*dspdr(2) + (m2 - one)*invr*hsps)
00236
00237      ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+3) + &
00238      rhodown(indi+1, indj+3)) * &
00239      (m*dspdr(3) + mn*invr*hsps)
00240
00241      ! E_s1,z2
00242
00243      ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+4) + &
00244      rhodown(indi+1, indj+4)) * &
00245      (n*dspdr(1) + ln*invr*hsps)
00246
00247      ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+4) + &
00248      rhodown(indi+1, indj+4)) * &
00249      (n*dspdr(2) + mn*invr*hsps)
00250
00251      ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+4) + &
00252      rhodown(indi+1, indj+4)) * &
00253      (n*dspdr(3) + (n2 - one)*invr*hsps)
00254
00255  ENDIF
00256
00257  ENDIF
00258
00259  ELSEIF (basisi .EQ. "sp") THEN
00260
00261      IF (basisj .EQ. "s") THEN
00262
00263          DO k = 1, noint
00264
00265              IF ((atele(i) .EQ. ele1(k) .AND. &
00266              atele(j) .EQ. ele2(k)) .OR. &
00267              (atele(i) .EQ. ele2(k) .AND. &

```

```

00268         atele(j) .EQ. ele1(k)) THEN
00269
00270     IF (btype(k) .EQ. "sss") THEN
00271
00272         CALL dunivscale_sub(magr, bond(:,k), dc, hsss, dssssdr)
00273
00274     ELSEIF (btype(k) .EQ. "sps") THEN
00275
00276         CALL dunivscale_sub(magr, bond(:,k), dc, hpss, dpssdr)
00277
00278         hpss = -hpss
00279         dpssdr = -dpssdr
00280
00281     ENDIF
00282 ENDIF
00283 ENDDO
00284
00285 l2 = l*l
00286 m2 = m*m
00287 n2 = n*n
00288 lm = l*m
00289 ln = l*n
00290 mn = m*n
00291
00292 IF (spinon .EQ. 0) THEN
00293
00294     ! E_s1,s2
00295
00296     ftmp = ftmp - dssssdr*bo(indi+1, indj+1)
00297
00298     ! E_x1,s2
00299
00300     ftmp(1) = ftmp(1) - bo(indi+2, indj+1) * &
00301         (1*dpssdr(1) + (l2 - one)*invr*hpss)
00302
00303     ftmp(2) = ftmp(2) - bo(indi+2, indj+1) * &
00304         (1*dpssdr(2) + lm*invr*hpss)
00305
00306     ftmp(3) = ftmp(3) - bo(indi+2, indj+1) * &
00307         (1*dpssdr(3) + ln*invr*hpss)
00308
00309     ! E_y1,s2
00310
00311     ftmp(1) = ftmp(1) - bo(indi+3, indj+1) * &
00312         (m*dpssdr(1) + lm*invr*hpss)
00313
00314     ftmp(2) = ftmp(2) - bo(indi+3, indj+1) * &
00315         (m*dpssdr(2) + (m2 - one)*invr*hpss)
00316
00317     ftmp(3) = ftmp(3) - bo(indi+3, indj+1) * &
00318         (m*dpssdr(3) + mn*invr*hpss)
00319
00320     ! E_z1,s2
00321
00322     ftmp(1) = ftmp(1) - bo(indi+4, indj+1) * &
00323         (n*dpssdr(1) + ln*invr*hpss)
00324
00325     ftmp(2) = ftmp(2) - bo(indi+4, indj+1) * &
00326         (n*dpssdr(2) + mn*invr*hpss)
00327
00328     ftmp(3) = ftmp(3) - bo(indi+4, indj+1) * &
00329         (n*dpssdr(3) + (n2 - one)*invr*hpss)
00330
00331 ELSE
00332
00333     ! E_s1,s2
00334
00335     ftmp = ftmp - dssssdr*(rhoup(indi+1, indj+1) + &
00336         rhodown(indi+1, indj+1))
00337
00338     ! E_x1,s2
00339
00340     ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+1) + &
00341         rhodown(indi+2, indj+1)) * &
00342         (1*dpssdr(1) + (l2 - one)*invr*hpss)
00343
00344     ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+1) + &
00345         rhodown(indi+2, indj+1)) * &
00346         (1*dpssdr(2) + lm*invr*hpss)
00347
00348     ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+1) + &
00349         rhodown(indi+2, indj+1)) * &
00350         (1*dpssdr(3) + ln*invr*hpss)
00351
00352     ! E_y1,s2
00353
00354     ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+1) + &

```

```

00355         rhodown(indi+3, indj+1)) * &
00356         (m*dpssdr(1) + lm*invr*hpss)
00357
00358         ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+1) + &
00359         rhodown(indi+3, indj+1)) * &
00360         (m*dpssdr(2) + (m2 - one)*invr*hpss)
00361
00362         ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+1) + &
00363         rhodown(indi+3, indj+1)) * &
00364         (m*dpssdr(3) + mn*invr*hpss)
00365
00366         ! E_z1,s2
00367
00368         ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+1) + &
00369         rhodown(indi+4, indj+1)) * &
00370         (n*dpssdr(1) + ln*invr*hpss)
00371
00372         ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+1) + &
00373         rhodown(indi+4, indj+1)) * &
00374         (n*dpssdr(2) + mn*invr*hpss)
00375
00376         ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+1) + &
00377         rhodown(indi+4, indj+1)) * &
00378         (n*dpssdr(3) + (n2 - one)*invr*hpss)
00379
00380     ENDIF
00381
00382     ELSEIF (basisj .EQ. "sp") THEN
00383
00384         IF (atele(i) .EQ. atele(j)) THEN
00385
00386             DO k = 1, noint
00387
00388                 IF (atele(i) .EQ. ele1(k) .AND. &
00389                 atele(j) .EQ. ele2(k)) THEN
00390
00391                     IF (btype(k) .EQ. "sss") THEN
00392
00393                         CALL dunivscale_sub(magr, bond(:,k), dc, hsss, dsssdr)
00394
00395                     ELSEIF (btype(k) .EQ. "sps") THEN
00396
00397                         CALL dunivscale_sub(magr, bond(:,k), dc, hspss, dspssdr)
00398
00399                         dpssdr = -dspssdr
00400                         hpss = -hspss
00401
00402                     ELSEIF (btype(k) .EQ. "pps") THEN
00403
00404                         CALL dunivscale_sub(magr, bond(:,k), dc, hpps, dppsdr)
00405
00406                     ELSEIF (btype(k) .EQ. "ppp") THEN
00407
00408                         CALL dunivscale_sub(magr, bond(:,k), dc, hppp, dpppdr)
00409
00410                     ENDIF
00411                 ENDIF
00412             ENDDO
00413
00414         ELSEIF (atele(i) .NE. atele(j)) THEN
00415
00416             DO k = 1, noint
00417
00418                 IF (atele(i) .EQ. ele1(k) .AND. &
00419                 atele(j) .EQ. ele2(k)) THEN
00420
00421                     IF (btype(k) .EQ. "sss") THEN
00422
00423                         CALL dunivscale_sub(magr, bond(:,k), dc, hsss, dsssdr)
00424
00425                     ELSEIF (btype(k) .EQ. "sps") THEN
00426
00427                         CALL dunivscale_sub(magr, bond(:,k), dc, hspss, dspssdr)
00428
00429                     ELSEIF (btype(k) .EQ. "pps") THEN
00430
00431                         CALL dunivscale_sub(magr, bond(:,k), dc, hpps, dppsdr)
00432
00433                     ELSEIF (btype(k) .EQ. "ppp") THEN
00434
00435                         CALL dunivscale_sub(magr, bond(:,k), dc, hppp, dpppdr)
00436
00437                     ENDIF
00438
00439                 ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00440                 atele(j) .EQ. ele1(k)) THEN
00441

```

```

00442         IF (btype(k) .EQ. "sss") THEN
00443             CALL duniyscale_sub(magr, bond(:,k), dc, hsss, dssssdr)
00444
00445         ELSEIF (btype(k) .EQ. "sps") THEN
00446
00447             CALL duniyscale_sub(magr, bond(:,k), dc, hpss, dpssdr)
00448
00449             dpssdr = -dpssdr
00450             hpss = -hpss
00451
00452         ELSEIF (btype(k) .EQ. "pps") THEN
00453
00454             CALL duniyscale_sub(magr, bond(:,k), dc, hpps, dppssdr)
00455
00456         ELSEIF (btype(k) .EQ. "ppp") THEN
00457
00458             CALL duniyscale_sub(magr, bond(:,k), dc, hppp, dpppsdr)
00459
00460         ENDIF
00461     ENDIF
00462 ENDDO
00463
00464     ENDIF
00465
00466     ENDIF
00467
00468     ppsmppp = hpps - hppp
00469     ppsubinvr = ppsmppp * invr
00470
00471     l2 = l*l
00472     m2 = m*m
00473     n2 = n*n
00474     lm = l*m
00475     ln = l*n
00476     mn = m*n
00477     lmn = lm*n
00478
00479     IF (spinon .EQ. 0) THEN
00480
00481         ! E_s1,s2
00482
00483         ftmp = ftmp - dssssdr*bo(indi+1, indj+1)
00484
00485         ! E_s1,x2
00486
00487         ftmp(1) = ftmp(1) - bo(indi+1, indj+2) * &
00488             (l*dspssdr(1) + (l2 - one)*invr*hsps)
00489
00490         ftmp(2) = ftmp(2) - bo(indi+1, indj+2) * &
00491             (l*dspssdr(2) + lm*invr*hsps)
00492
00493         ftmp(3) = ftmp(3) - bo(indi+1, indj+2) * &
00494             (l*dspssdr(3) + ln*invr*hsps)
00495
00496         ! E_s1,y2
00497
00498         ftmp(1) = ftmp(1) - bo(indi+1, indj+3) * &
00499             (m*dspssdr(1) + lm*invr*hsps)
00500
00501         ftmp(2) = ftmp(2) - bo(indi+1, indj+3) * &
00502             (m*dspssdr(2) + (m2 - one)*invr*hsps)
00503
00504         ftmp(3) = ftmp(3) - bo(indi+1, indj+3) * &
00505             (m*dspssdr(3) + mn*invr*hsps)
00506
00507         ! E_s1,z2
00508
00509         ftmp(1) = ftmp(1) - bo(indi+1, indj+4) * &
00510             (n*dspssdr(1) + ln*invr*hsps)
00511
00512         ftmp(2) = ftmp(2) - bo(indi+1, indj+4) * &
00513             (n*dspssdr(2) + mn*invr*hsps)
00514
00515         ftmp(3) = ftmp(3) - bo(indi+1, indj+4) * &
00516             (n*dspssdr(3) + (n2 - one)*invr*hsps)
00517
00518         ! E_x1,s2
00519
00520         ftmp(1) = ftmp(1) - bo(indi+2, indj+1) * &
00521             (l*dpssdr(1) + (l2 - one)*invr*hpss)
00522
00523         ftmp(2) = ftmp(2) - bo(indi+2, indj+1) * &
00524             (l*dpssdr(2) + lm*invr*hpss)
00525
00526         ftmp(3) = ftmp(3) - bo(indi+2, indj+1) * &
00527             (l*dpssdr(3) + ln*invr*hpss)
00528

```

```

00529      ! E_x1,x2
00530
00531      ftmp(1) = ftmp(1) - bo(indi+2, indj+2) * &
00532          (l2*dppsdr(1) + (one - l2)*dpppdr(1) + &
00533          two*1*(l2 - one)*ppsubinvr)
00534
00535      ftmp(2) = ftmp(2) - bo(indi+2, indj+2) * &
00536          (l2*dppsdr(2) + (one - l2)*dpppdr(2) + &
00537          two*l2*m*ppsubinvr)
00538
00539      ftmp(3) = ftmp(3) - bo(indi+2, indj+2) * &
00540          (l2*dppsdr(3) + (one - l2)*dpppdr(3) + &
00541          two*l2*n*ppsubinvr)
00542
00543      ! E_x1,y2
00544
00545      ftmp(1) = ftmp(1) - bo(indi+2, indj+3) * &
00546          (lm*(dppsdr(1) - dpppdr(1)) + &
00547          m*(two*l2 - one)*ppsubinvr)
00548
00549      ftmp(2) = ftmp(2) - bo(indi+2, indj+3) * &
00550          (lm*(dppsdr(2) - dpppdr(2)) + &
00551          l*(two*m2 - one)*ppsubinvr)
00552
00553      ftmp(3) = ftmp(3) - bo(indi+2, indj+3) * &
00554          (lm*(dppsdr(3) - dpppdr(3)) + &
00555          two*lmn*ppsubinvr)
00556
00557      ! E_x1,z2
00558
00559      ftmp(1) = ftmp(1) - bo(indi+2, indj+4) * &
00560          (ln*(dppsdr(1) - dpppdr(1)) + &
00561          n*(two*l2 - one)*ppsubinvr)
00562
00563      ftmp(2) = ftmp(2) - bo(indi+2, indj+4) * &
00564          (ln*(dppsdr(2) - dpppdr(2)) + &
00565          two*lmn*ppsubinvr)
00566
00567      ftmp(3) = ftmp(3) - bo(indi+2, indj+4) * &
00568          (ln*(dppsdr(3) - dpppdr(3)) + &
00569          l*(two*n2 - one)*ppsubinvr)
00570
00571      ! E_y1,s2
00572
00573      ftmp(1) = ftmp(1) - bo(indi+3, indj+1) * &
00574          (m*dpssdr(1) + lm*invr*hpss)
00575
00576      ftmp(2) = ftmp(2) - bo(indi+3, indj+1) * &
00577          (m*dpssdr(2) + (m2 - one)*invr*hpss)
00578
00579      ftmp(3) = ftmp(3) - bo(indi+3, indj+1) * &
00580          (m*dpssdr(3) + mn*invr*hpss)
00581
00582      ! E_y1,x2
00583
00584      ftmp(1) = ftmp(1) - bo(indi+3, indj+2) * &
00585          (lm*(dppsdr(1) - dpppdr(1)) + &
00586          m*(two*l2 - one)*ppsubinvr)
00587
00588      ftmp(2) = ftmp(2) - bo(indi+3, indj+2) * &
00589          (lm*(dppsdr(2) - dpppdr(2)) + &
00590          l*(two*m2 - one)*ppsubinvr)
00591
00592      ftmp(3) = ftmp(3) - bo(indi+3, indj+2) * &
00593          (lm*(dppsdr(3) - dpppdr(3)) + &
00594          two*lmn*ppsubinvr)
00595
00596      ! E_y1,y2
00597
00598      ftmp(1) = ftmp(1) - bo(indi+3, indj+3) * &
00599          (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00600          two*1*m2*ppsubinvr)
00601
00602      ftmp(2) = ftmp(2) - bo(indi+3, indj+3) * &
00603          (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00604          two*m*(m2 - one)*ppsubinvr)
00605
00606      ftmp(3) = ftmp(3) - bo(indi+3, indj+3) * &
00607          (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00608          two*n*m2*ppsubinvr)
00609
00610      ! E_y1,z2
00611
00612      ftmp(1) = ftmp(1) - bo(indi+3, indj+4) * &
00613          (mn*(dppsdr(1) - dpppdr(1)) + &
00614          two*lmn*ppsubinvr)
00615

```



```

00616      ftmp(2) = ftmp(2) - bo(indi+3, indj+4) * &
00617      (mn*(dppsdr(2) - dpppdr(2)) + &
00618      n*(two*m2 - one)*ppsubinvr)
00619
00620      ftmp(3) = ftmp(3) - bo(indi+3, indj+4) * &
00621      (mn*(dppsdr(3) - dpppdr(3)) + &
00622      m*(two*n2 - one)*ppsubinvr)
00623
00624      ! E_z1,s2
00625
00626      ftmp(1) = ftmp(1) - bo(indi+4, indj+1) * &
00627      (n*dpssdr(1) + ln*invr*hpss)
00628
00629      ftmp(2) = ftmp(2) - bo(indi+4, indj+1) * &
00630      (n*dpssdr(2) + mn*invr*hpss)
00631
00632      ftmp(3) = ftmp(3) - bo(indi+4, indj+1) * &
00633      (n*dpssdr(3) + (n2 - one)*invr*hpss)
00634
00635      ! E_z1,x2
00636
00637      ftmp(1) = ftmp(1) - bo(indi+4, indj+2) * &
00638      (ln*(dppsdr(1) - dpppdr(1)) + &
00639      n*(two*l2 - one)*ppsubinvr)
00640
00641      ftmp(2) = ftmp(2) - bo(indi+4, indj+2) * &
00642      (ln*(dppsdr(2) - dpppdr(2)) + &
00643      two*lmn*ppsubinvr)
00644
00645      ftmp(3) = ftmp(3) - bo(indi+4, indj+2) * &
00646      (ln*(dppsdr(3) - dpppdr(3)) + &
00647      l*(two*n2 - one)*ppsubinvr)
00648
00649      ! E_z1,y2
00650
00651      ftmp(1) = ftmp(1) - bo(indi+4, indj+3) * &
00652      (mn*(dppsdr(1) - dpppdr(1)) + &
00653      two*lmn*ppsubinvr)
00654
00655      ftmp(2) = ftmp(2) - bo(indi+4, indj+3) * &
00656      (mn*(dppsdr(2) - dpppdr(2)) + &
00657      n*(two*m2 - one)*ppsubinvr)
00658
00659      ftmp(3) = ftmp(3) - bo(indi+4, indj+3) * &
00660      (mn*(dppsdr(3) - dpppdr(3)) + &
00661      m*(two*n2 - one)*ppsubinvr)
00662
00663      ! E_z1,z2
00664
00665      ftmp(1) = ftmp(1) - bo(indi+4, indj+4) * &
00666      (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00667      two*l*n2*ppsubinvr)
00668
00669      ftmp(2) = ftmp(2) - bo(indi+4, indj+4) * &
00670      (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00671      two*m*n2*ppsubinvr)
00672
00673      ftmp(3) = ftmp(3) - bo(indi+4, indj+4) * &
00674      (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00675      two*n*(n2 - one)*ppsubinvr)
00676
00677      ELSE ! SPIN-POLARIZED CALCULATION
00678
00679      ! E_s1,s2
00680
00681      ftmp(1) = ftmp(1) - dsssdr(1)*(rhoup(indi+1, indj+1) + &
00682      rhodown(indi+1, indj+1))
00683
00684      ftmp(2) = ftmp(2) - dsssdr(2)*(rhoup(indi+1, indj+1) + &
00685      rhodown(indi+1, indj+1))
00686
00687      ftmp(3) = ftmp(3) - dsssdr(3)*(rhoup(indi+1, indj+1) + &
00688      rhodown(indi+1, indj+1))
00689
00690      ! E_s1,x2
00691
00692      ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+2) + &
00693      rhodown(indi+1, indj+2)) * &
00694      (l*dspsdr(1) + (l2 - one)*invr*hsps)
00695
00696      ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+2) + &
00697      rhodown(indi+1, indj+2)) * &
00698      (l*dspsdr(2) + lm*invr*hsps)
00699
00700      ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+2) + &
00701      rhodown(indi+1, indj+2)) * &
00702      (l*dspsdr(3) + ln*invr*hsps)

```

```

00703
00704         ! E_s1,y2
00705
00706         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+3) + &
00707             rhodown(indi+1, indj+3)) * &
00708             (m*dspssdr(1) + lm*invr*hsps)
00709
00710         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+3) + &
00711             rhodown(indi+1, indj+3)) * &
00712             (m*dspssdr(2) + (m2 - one)*invr*hsps)
00713
00714         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+3) + &
00715             rhodown(indi+1, indj+3)) * &
00716             (m*dspssdr(3) + mn*invr*hsps)
00717
00718         ! E_s1,z2
00719
00720         ftmp(1) = ftmp(1) - (rhoup(indi+1, indj+4) + &
00721             rhodown(indi+1, indj+4)) * &
00722             (n*dspssdr(1) + ln*invr*hsps)
00723
00724         ftmp(2) = ftmp(2) - (rhoup(indi+1, indj+4) + &
00725             rhodown(indi+1, indj+4)) * &
00726             (n*dspssdr(2) + mn*invr*hsps)
00727
00728         ftmp(3) = ftmp(3) - (rhoup(indi+1, indj+4) + &
00729             rhodown(indi+1, indj+4)) * &
00730             (n*dspssdr(3) + (n2 - one)*invr*hsps)
00731
00732         ! E_x1,s2
00733
00734         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+1) + &
00735             rhodown(indi+2, indj+1)) * &
00736             (l*dpssdr(1) + (l2 - one)*invr*hpss)
00737
00738         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+1) + &
00739             rhodown(indi+2, indj+1)) * &
00740             (l*dpssdr(2) + lm*invr*hpss)
00741
00742         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+1) + &
00743             rhodown(indi+2, indj+1)) * &
00744             (l*dpssdr(3) + ln*invr*hpss)
00745
00746         ! E_x1,x2
00747
00748         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+2) + &
00749             rhodown(indi+2, indj+2)) * &
00750             (l2*dppsdr(1) + (one - l2)*dpppdr(1) + &
00751             two*l*(l2 - one)*ppsubinvr)
00752
00753         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+2) + &
00754             rhodown(indi+2, indj+2)) * &
00755             (l2*dppsdr(2) + (one - l2)*dpppdr(2) + &
00756             two*l2*m*ppsubinvr)
00757
00758         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+2) + &
00759             rhodown(indi+2, indj+2)) * &
00760             (l2*dppsdr(3) + (one - l2)*dpppdr(3) + &
00761             two*l2*n*ppsubinvr)
00762
00763         ! E_x1,y2
00764
00765         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+3) + &
00766             rhodown(indi+2, indj+3)) * &
00767             (lm*(dppsdr(1) - dpppdr(1)) + &
00768             m*(two*l2 - one)*ppsubinvr)
00769
00770         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+3) + &
00771             rhodown(indi+2, indj+3)) * &
00772             (lm*(dppsdr(2) - dpppdr(2)) + &
00773             l*(two*m2 - one)*ppsubinvr)
00774
00775         ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+3) + &
00776             rhodown(indi+2, indj+3)) * &
00777             (lm*(dppsdr(3) - dpppdr(3)) + &
00778             two*lmn*ppsubinvr)
00779
00780         ! E_x1,z2
00781
00782         ftmp(1) = ftmp(1) - (rhoup(indi+2, indj+4) + &
00783             rhodown(indi+2, indj+4)) * &
00784             (ln*(dppsdr(1) - dpppdr(1)) + &
00785             n*(two*l2 - one)*ppsubinvr)
00786
00787         ftmp(2) = ftmp(2) - (rhoup(indi+2, indj+4) + &
00788             rhodown(indi+2, indj+4)) * &
00789             (ln*(dppsdr(2) - dpppdr(2)) + &

```

```

00790         two*lmn*ppsubinvr)
00791
00792     ftmp(3) = ftmp(3) - (rhoup(indi+2, indj+4) + &
00793         rhodown(indi+2, indj+4)) * &
00794         (ln*(dppsdr(3) - dpppdr(3)) + &
00795         l*(two*n2 - one)*ppsubinvr)
00796
00797     ! E_y1,s2
00798
00799     ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+1) + &
00800         rhodown(indi+3, indj+1)) * &
00801         (m*dpssdr(1) + lm*invr*hpss)
00802
00803     ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+1) + &
00804         rhodown(indi+3, indj+1)) * &
00805         (m*dpssdr(2) + (m2 - one)*invr*hpss)
00806
00807     ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+1) + &
00808         rhodown(indi+3, indj+1)) * &
00809         (m*dpssdr(3) + mn*invr*hpss)
00810
00811     ! E_y1,x2
00812
00813     ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+2) + &
00814         rhodown(indi+3, indj+2)) * &
00815         (lm*(dppsdr(1) - dpppdr(1)) + &
00816         m*(two*l2 - one)*ppsubinvr)
00817
00818     ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+2) + &
00819         rhodown(indi+3, indj+2)) * &
00820         (lm*(dppsdr(2) - dpppdr(2)) + &
00821         l*(two*m2 - one)*ppsubinvr)
00822
00823     ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+2) + &
00824         rhodown(indi+3, indj+2)) * &
00825         (lm*(dppsdr(3) - dpppdr(3)) + &
00826         two*lmn*ppsubinvr)
00827
00828     ! E_y1,y2
00829
00830     ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+3) + &
00831         rhodown(indi+3, indj+3)) * &
00832         (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00833         two*l*m2*ppsubinvr)
00834
00835     ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+3) + &
00836         rhodown(indi+3, indj+3)) * &
00837         (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00838         two*m*(m2 - one)*ppsubinvr)
00839
00840     ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+3) + &
00841         rhodown(indi+3, indj+3)) * &
00842         (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00843         two*n*m2*ppsubinvr)
00844
00845     ! E_y1,z2
00846
00847     ftmp(1) = ftmp(1) - (rhoup(indi+3, indj+4) + &
00848         rhodown(indi+3, indj+4)) * &
00849         (mn*(dppsdr(1) - dpppdr(1)) + &
00850         two*lmn*ppsubinvr)
00851
00852     ftmp(2) = ftmp(2) - (rhoup(indi+3, indj+4) + &
00853         rhodown(indi+3, indj+4)) * &
00854         (mn*(dppsdr(2) - dpppdr(2)) + &
00855         n*(two*m2 - one)*ppsubinvr)
00856
00857     ftmp(3) = ftmp(3) - (rhoup(indi+3, indj+4) + &
00858         rhodown(indi+3, indj+4)) * &
00859         (mn*(dppsdr(3) - dpppdr(3)) + &
00860         m*(two*n2 - one)*ppsubinvr)
00861
00862     ! E_z1,s2
00863
00864     ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+1) + &
00865         rhodown(indi+4, indj+1)) * &
00866         (n*dpssdr(1) + ln*invr*hpss)
00867
00868     ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+1) + &
00869         rhodown(indi+4, indj+1)) * &
00870         (n*dpssdr(2) + mn*invr*hpss)
00871
00872     ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+1) + &
00873         rhodown(indi+4, indj+1)) * &
00874         (n*dpssdr(3) + (n2 - one)*invr*hpss)
00875
00876     ! E_z1,x2

```

```

00877
00878         ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+2) + &
00879             rhodown(indi+4, indj+2)) * &
00880             (ln*(dppsdr(1) - dpppdr(1)) + &
00881             n*(two*12 - one)*ppsubinvr)
00882
00883         ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+2) + &
00884             rhodown(indi+4, indj+2)) * &
00885             (ln*(dppsdr(2) - dpppdr(2)) + &
00886             two*lmn*ppsubinvr)
00887
00888         ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+2) + &
00889             rhodown(indi+4, indj+2)) * &
00890             (ln*(dppsdr(3) - dpppdr(3)) + &
00891             1*(two*n2 - one)*ppsubinvr)
00892
00893         ! E_z1,y2
00894
00895         ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+3) + &
00896             rhodown(indi+4, indj+3)) * &
00897             (mn*(dppsdr(1) - dpppdr(1)) + &
00898             two*lmn*ppsubinvr)
00899
00900         ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+3) + &
00901             rhodown(indi+4, indj+3)) * &
00902             (mn*(dppsdr(2) - dpppdr(2)) + &
00903             n*(two*m2 - one)*ppsubinvr)
00904
00905         ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+3) + &
00906             rhodown(indi+4, indj+3)) * &
00907             (mn*(dppsdr(3) - dpppdr(3)) + &
00908             m*(two*n2 - one)*ppsubinvr)
00909
00910         ! E_z1,z2
00911
00912         ftmp(1) = ftmp(1) - (rhoup(indi+4, indj+4) + &
00913             rhodown(indi+4, indj+4)) * &
00914             (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00915             two*1*n2*ppsubinvr)
00916
00917         ftmp(2) = ftmp(2) - (rhoup(indi+4, indj+4) + &
00918             rhodown(indi+4, indj+4)) * &
00919             (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00920             two*m*n2*ppsubinvr)
00921
00922         ftmp(3) = ftmp(3) - (rhoup(indi+4, indj+4) + &
00923             rhodown(indi+4, indj+4)) * &
00924             (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00925             two*n*(n2 - one)*ppsubinvr)
00926
00927     ENDIF
00928
00929     ENDIF
00930
00931     ENDIF
00932
00933     f(1,i) = f(1,i) + ftmp(1)
00934     f(2,i) = f(2,i) + ftmp(2)
00935     f(3,i) = f(3,i) + ftmp(3)
00936
00937     ! with the factor of 2...
00938
00939     virbond(1) = virbond(1) + rij(1)*ftmp(1)
00940     virbond(2) = virbond(2) + rij(2)*ftmp(2)
00941     virbond(3) = virbond(3) + rij(3)*ftmp(3)
00942     virbond(4) = virbond(4) + rij(1)*ftmp(2)
00943     virbond(5) = virbond(5) + rij(2)*ftmp(3)
00944     virbond(6) = virbond(6) + rij(3)*ftmp(1)
00945
00946     ENDDO
00947
00948     ENDDO
00949     !$OMP END PARALLEL DO
00950
00951
00952     RETURN
00953
00954 END SUBROUTINE gradhsp

```

8.191 homolumo.f90 File Reference

Modules

- module [homolumo](#)

Functions/Subroutines

- subroutine [homolumo::homolumogap](#) (ITERZ)
- subroutine [homolumo::sp2sequence](#) ()

8.192 homolumo.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE homolumo
00023
00024     IMPLICIT NONE
00025
00026     CONTAINS
00027
00028     SUBROUTINE homolumogap(ITERZ)
00029
00030         USE myprecision
00031         USE constants_mod
00032         USE purearray
00033         USE sparsearray
00034
00035         IMPLICIT NONE
00036         INTEGER, INTENT(IN) :: ITERZ
00037         INTEGER :: I, J
00038         REAL(8) :: PRECOMP, X_A, X_B, Y_A, Y_B, HGAMMA
00039
00040         !!!!!!!!!!!!!!!!!!!!!!! CALCULATE HOMO-LUMO ESTIMATES !!!!!!!!!!!!!!!
00041         x_a = 0.0d0
00042         x_b = 1.0d0
00043         y_a = 0.0d0
00044         y_b = 0.0d0
00045
00046         i = iterz
00047         hgamma = six - four*sqrt(two)
00048         hgamma = hgamma*( one - hgamma )
00049
00050         !write(*,*) "First I = ", ITERZ, " HGAMMA = ", HGAMMA, " VV = ", VV(I)
00051
00052         DO WHILE (dble(vv(i)) .LT. hgamma)
00053
00054             !write(*,*) "ITERZ = ", ITERZ, "I = ", I, "VV = ", VV(I)
00055
00056             precomp = sqrt( one - four*vv(i))
00057             y_a = half * ( one + precomp )
00058             y_b = half * ( one - precomp )
00059
00060             DO j = i-1,1,-1                !!! SHIFT -1 IN I
00061                 IF (pp(j) .GT. 0) THEN
00062                     y_a = sqrt(y_a)
00063                     y_b = sqrt(y_b)
00064                 ELSE
00065                     y_a = 1.0d0 - sqrt(one - y_a)

```

```

00066         y_b = 1.0d0 - sqrt(one - y_b)
00067     ENDIF
00068 ENDDO
00069
00070     x_a = max(x_a,y_a)
00071     x_b = min(x_b,y_b)
00072
00073     i = i - 1
00074     IF (i .LT. 1) THEN
00075         WRITE(*,*) "HomoLumo i = ", i
00076     ENDIF
00077 ENDDO
00078
00079     ehomo = maxeval - REAL(x_a)*(maxeval - mineval)
00080     elumo = maxeval - REAL(x_b)*(maxeval - mineval)
00081
00082     egap = elumo - ehomo
00083
00084     ! PRINT*, "EGAP = ", EGAP
00085
00086     !WRITE(*,*) '### EHOMO = ', EHOMO, ' ELUMO = ', ELUMO
00087     !WRITE(*,*) '### MAXEVAL = ', MAXEVAL, ' MINEVAL = ', MINEVAL
00088
00089     !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00090
00091 END SUBROUTINE homolumogap
00092
00093 SUBROUTINE sp2sequence()
00094
00095     USE myprecision
00096     USE constants_mod
00097     USE purearray
00098     USE sparsearray
00099
00100     IMPLICIT NONE
00101
00102     INTEGER :: IT
00103     REAL(8) :: EH, EL, ERR, SGM
00104     !REAL(LATTEPREC), PARAMETER :: ERRLIMIT=1E-20
00105     REAL(8), PARAMETER :: ERRLIMIT=1d-16
00106     REAL(8) :: DMAXEVAL, DMINEVAL, DEHOMO, DELUMO
00107
00108     dmaxeval = dble(maxeval)
00109     dmineval = dble(mineval)
00110     dehomo = dble(ehomo)
00111     delumo = dble(elumo)
00112
00113     !!! GET SEQUENCE OF X^2 AND 2X-X^2 -> PP(1:NIT) = 1 FOR X^2 AND 0 FOR 2X-X^2
00114
00115     ! WRITE(*,*) ' FORE NR_SP2_ITER = ',NR_SP2_ITER
00116
00117     eh = (dmaxeval - dehomo)/(dmaxeval - dmineval)
00118     el = (dmaxeval - delumo)/(dmaxeval - dmineval)
00119
00120     err = one
00121
00122     it = 0
00123
00124     DO WHILE (err .GT. errlimit)
00125
00126         it = it + 1
00127
00128         IF ( (abs(one - eh*eh) + abs(el*el)) .LT. &
00129             (abs(one - (two*eh - eh*eh) + abs(two*el - el*el))) ) THEN
00130
00131             pp(it) = 1
00132
00133             eh = eh*eh
00134             el = el*el
00135
00136         ELSE
00137
00138             pp(it) = 0
00139
00140             eh = two*eh - eh*eh
00141             el = two*el - el*el
00142
00143         ENDIF
00144
00145         err = abs(one - eh) + abs(el)
00146
00147         !write(*,*) "SGM = ", SGM, " EH = ", EH, " EL = ", EL, " ERR = ", ERR
00148
00149         IF (it.GE.100) THEN
00150             err = zero
00151             WRITE(*,*) 'SP2SEQUENCE WARNING NOT CONVERGING IN SP2'
00152         ENDIF

```

```
00153
00154     ENDDO
00155
00156     nr_sp2_iter = it
00157     !  WRITE(*,*) ' #SIMPLE NR_SP2_ITER = ', NR_SP2_ITER
00158
00159     END SUBROUTINE sp2sequence
00160
00161 END MODULE homolumo
```

8.193 hugrescale.f90 File Reference

Functions/Subroutines

- subroutine [hugrescale](#)

8.193.1 Function/Subroutine Documentation

8.193.1.1 subroutine hugrescale ()

Definition at line 24 of file [hugrescale.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS            !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such        !
00010 ! modified software should be clearly marked, so as not to confuse it         !
00011 ! with the version available from LANL.                                       !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it        !
00014 ! and/or modify it under the terms of the GNU General Public License as       !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of     !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022
00023 SUBROUTINE hugrescale
00024
00025   USE constants_mod
00026   USE setuparray
00027   USE mdarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, K
00033   REAL(LATTEPREC) :: PREFACTOR, PREPREFACT
00034   REAL(LATTEPREC) :: CHI, TMPTEMP, MYTEMP
00035   REAL(LATTEPREC) :: MAXCHI = 1.05d0, maxdbox = 0.001d0
00036   REAL(LATTEPREC) :: PSCALE = 100.0d0
00037   REAL(LATTEPREC) :: TMPPRESS, MYPRESS, DBOX, DTEMP
00038   REAL(LATTEPREC) :: TMPPRESSX, TMPPRESSY, TMPPRESSZ, TMPENERGY, AVEENERGY
00039   REAL(LATTEPREC) :: DBOXX, DBOXY, DBOXZ, MYPRESSX, MYPRESSY, MYPRESSZ
00040   REAL(LATTEPREC) :: AVEVOL, TMPVOL
00041   REAL(LATTEPREC), PARAMETER :: MAXDT = 10.0
00042   IF (existerror) RETURN
00043
00044   tmptemp = zero
00045   ! Isotropic
00046   tmppress = zero
00047   ! non-isotropic
00048   tmppressx = zero
00049   tmppressy = zero
00050   tmppressz = zero
00051
00052   ! Energy and volume too
00053
00054   tmpenergy = zero
00055   tmpvol = zero
00056
00057
00058   DO i = 1, aveper/wrtfreq
00059
00060     tmpenergy = tmpenergy + ehist(i)
00061     tmpvol = tmpvol + vhist(i)
00062     tmptemp = tmptemp + thist(i)
00063     IF (npttype .EQ. "ISO") THEN
00064       tmppress = tmppress + phist(i)
00065     ELSE
00066       tmppressx = tmppressx + phistx(i)
00067       tmppressy = tmppressy + phisty(i)
00068       tmppressz = tmppressz + phistz(i)
00069     ENDIF
00070
00071   ENDDO
00072
00073   mytemp = tmptemp/REAL(aveper/wrtfreq)
00074
00075   mypress = tmppress/REAL(aveper/wrtfreq)
00076
00077   mypressx = tmppressx/REAL(aveper/wrtfreq)
00078   mypressy = tmppressy/REAL(aveper/wrtfreq)
00079   mypressz = tmppressz/REAL(aveper/wrtfreq)
00080
00081   aveenergy = tmpenergy/REAL(aveper/wrtfreq)
00082
00083   avevol = tmpvol/REAL(aveper/wrtfreq)
00084
00085
00086
00087   ! Change box dimensions depending on the average pressure
00088
00089   IF (npttype .EQ. "ISO") THEN
00090
00091     dbox = min(abs(mypress - ptarget)*maxdbox, maxdbox)
00092
00093     dbox = sign(dbox, mypress - ptarget)

```

```

00094
00095     box = box * (one + dbox)
00096
00097     cr = cr * (one + dbox)
00098
00099     ELSE ! Allow the three vectors to change length independently
00100
00101         dboxx = min(abs(mypressx - ptarget)*maxdbox, maxdbox)
00102         dboxy = min(abs(mypressy - ptarget)*maxdbox, maxdbox)
00103         dboxz = min(abs(mypressz - ptarget)*maxdbox, maxdbox)
00104
00105         dboxx = sign(dboxx, mypressx - ptarget)
00106         dboxy = sign(dboxy, mypressy - ptarget)
00107         dboxz = sign(dboxz, mypressz - ptarget)
00108
00109         box(1,1) = box(1,1) * (one + dboxx)
00110         box(2,2) = box(2,2) * (one + dboxy)
00111         box(3,3) = box(3,3) * (one + dboxz)
00112
00113         DO i = 1, nats
00114
00115             cr(1,i) = cr(1,i) * (one + dboxx)
00116             cr(2,i) = cr(2,i) * (one + dboxy)
00117             cr(3,i) = cr(3,i) * (one + dboxz)
00118
00119         ENDDO
00120
00121     ENDIF
00122
00123     ! SOLVING HG = (E - E0) - 0.5*(P + P0)*(V0 - V)
00124
00125     hg = (aveenergy - e0) - half*(mypress + p0)*(v0 - avevol)/togpa
00126
00127     dtemp = 0.025d0*ttarget
00128
00129     dtemp = min(dtemp, maxdt)
00130
00131     IF (hg .LT. zero) THEN
00132         ttargt = ttargt + dtemp
00133     ELSE
00134         ttargt = ttargt - dtemp
00135     ENDIF
00136
00137     chi = sqrt(ttargt/mytemp)
00138
00139     v = v * chi
00140
00141     RETURN
00142
00143 END SUBROUTINE hugrescale
00144

```

8.195 ifrestart.f90 File Reference

Functions/Subroutines

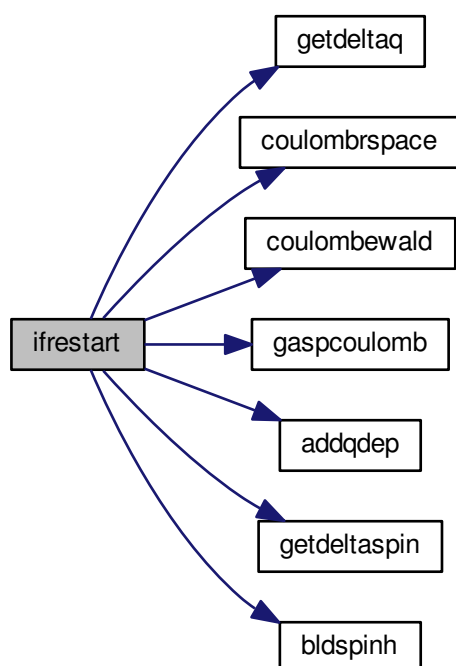
- subroutine [ifrestart](#)

8.195.1 Function/Subroutine Documentation

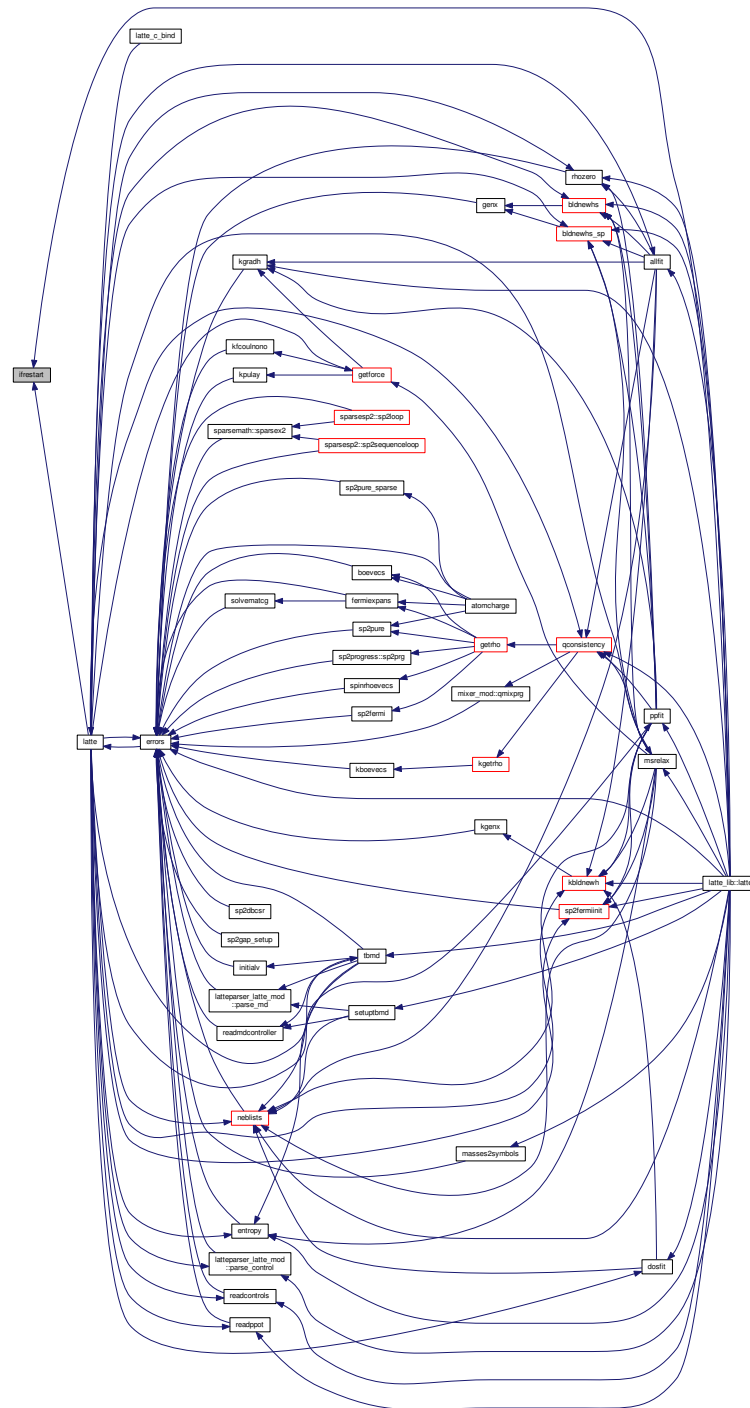
8.195.1.1 subroutine ifrestart ()

Definition at line 23 of file [ifrestart.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.196 ifrestart.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE ifrestart
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE restartarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031   IF (existerror) RETURN
00032
00033   !
00034   ! Now we've got to get the Hamiltonian(s) consistent with
00035   ! the density matrix we just read in
00036   !
00037   !
00038   !
00039   ! After this we should head into qconsistency and the rest
00040   !
00041
00042   IF (electro .EQ. 1) THEN
00043
00044     CALL getdeltaq
00045
00046     IF (elecmeth .EQ. 0) THEN
00047
00048       CALL coulombspace
00049       CALL coulombewald
00050
00051     ELSE
00052
00053       CALL gaspcoulomb
00054
00055     ENDIF
00056
00057     CALL addqdep
00058
00059   ENDIF
00060
00061   IF (spinon .EQ. 1) THEN
00062
00063     CALL getdeltaspin
00064
00065     CALL bldspinh
00066
00067   ENDIF
00068
00069 END SUBROUTINE ifrestart
00070
00071
00072

```

8.197 init_dbcsr.f90 File Reference

Functions/Subroutines

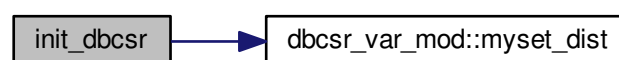
- subroutine [init_dbcsr](#)

8.197.1 Function/Subroutine Documentation

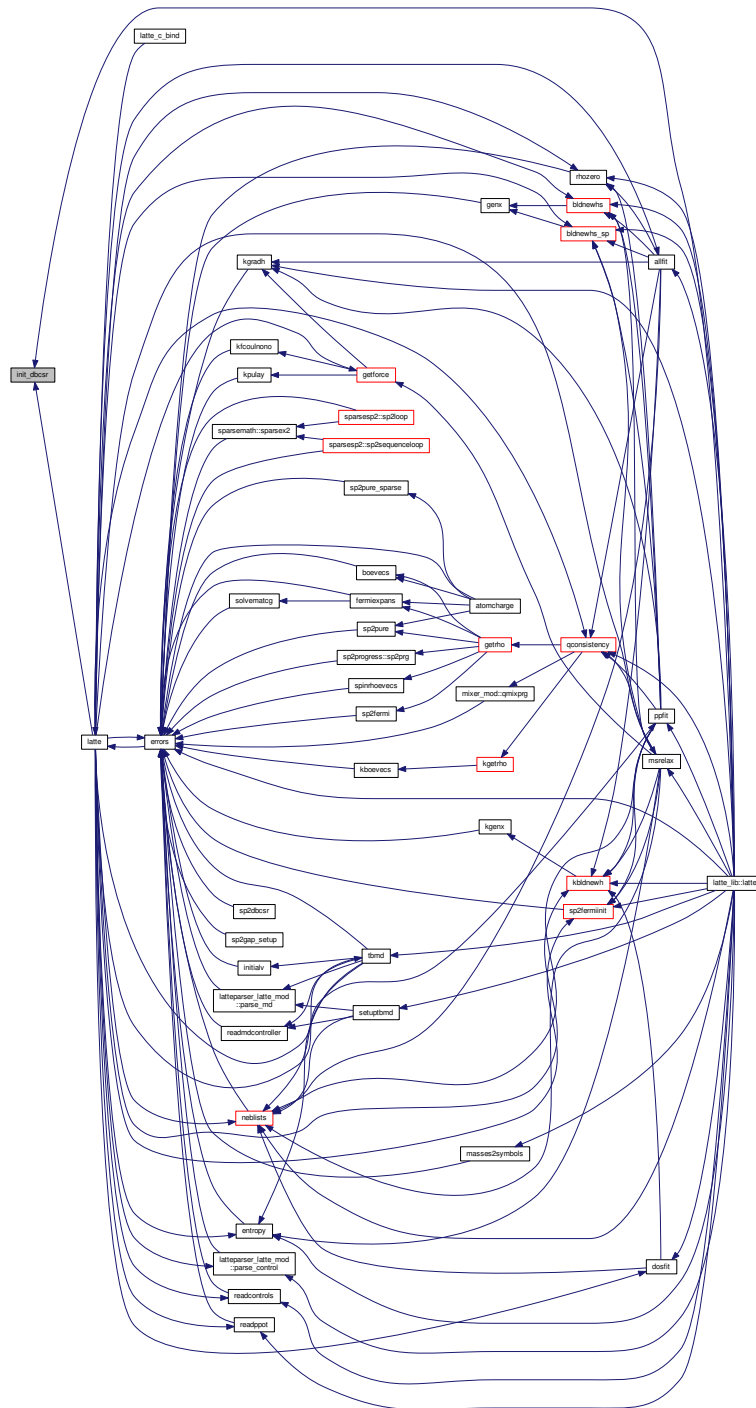
8.197.1.1 subroutine `init_dbcsr ()`

Definition at line 23 of file [init_dbcsr.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.198 init_dbcsr.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE init_dbcsr
00023
00024   USE dbcsr_var_mod
00025   USE dbcsr_config
00026   USE dbcsr_types
00027   USE dbcsr_methods
00028   USE dbcsr_error_handling
00029   USE array_types,          ONLY: array_data,&
00030       array_ild_obj,&
00031       array_new,&
00032       array_nullify,&
00033       array_release,&
00034       array_size
00035   USE dbcsr_io
00036   USE dbcsr_operations
00037   USE dbcsr_ptr_util
00038   USE dbcsr_transformations
00039   USE dbcsr_util
00040   USE dbcsr_work_operations
00041   USE dbcsr_message_passing
00042
00043   USE dbcsr_block_access
00044   USE dbcsr_iterator_operations,  ONLY: dbcsr_iterator_blocks_left,&
00045       dbcsr_iterator_next_block,&
00046       dbcsr_iterator_start,&
00047       dbcsr_iterator_stop
00048
00049   USE dbcsr_dist_operations,      ONLY: create_bl_distribution,&
00050       dbcsr_get_stored_coordinates
00051
00052   USE constants_mod
00053
00054   IMPLICIT NONE
00055
00056   !sets mpi
00057
00058   !sets up dbcsr matrix
00059   ! the matrix will contain nblkrows_total row blocks and nblkcols_total column blocks
00060
00061
00062   !initiallizing mpi
00063
00064   CALL mp_world_init(mp_comm)
00065
00066   npdims(:) = 0
00067
00068   CALL mp_cart_create (mp_comm, 2, npdims, myploc, group)
00069
00070   CALL mp_envIRON (numnodes, mynode, group)
00071
00072   ALLOCATE (pgrid(0:npdims(1)-1, 0:npdims(2)-1))
00073
00074   DO prow = 0, npdims(1)-1
00075     DO pcol = 0, npdims(2)-1
00076       CALL mp_cart_rank (group, (/ prow, pcol /), pgrid(prow,
00077         pcol))
00078     ENDDO
00079   ENDDO
00080
00081   ! Create the dbcsr_mp_obj
00082   CALL dbcsr_mp_new (mp_env, pgrid, group, mynode,
00083     numnodes,&
00084     myprow=mysploc(1), mypcol=mysploc(2))
00085
00086   DEALLOCATE(pgrid)
00087
00088   ! Use BLAS rather than the SMM
00089   CALL dbcsr_set_conf_mm_driver(2, error=error)
00090
00091   ! Now with padding

```



```

00092  nblkrows_total=(hdim-1)/blkksz + blkksz
00093  nblkcols_total=(hdim-1)/blkksz + blkksz
00094
00095
00096  !sets the block size for each row and column
00097  ALLOCATE(rbs(nblkrows_total))
00098  ALLOCATE(cbs(nblkcols_total))
00099  rbs(:)=blkksz
00100  cbs(:)=blkksz
00101
00102  CALL array_nullify (row_blk_sizes)
00103  CALL array_nullify (col_blk_sizes)
00104  CALL array_new (row_blk_sizes, rbs, gift=.true.)
00105  CALL array_new (col_blk_sizes, cbs, gift=.true.)
00106
00107
00108  !sets distribution to processors
00109  CALL myset_dist (row_dist_a, nblkrows_total,
npdims(1))
00110  CALL myset_dist (col_dist_a, nblkcols_total,
npdims(2))
00111
00112
00113  !Sets the distribution object
00114  CALL dbcsr_distribution_new (dist_a, mp_env, row_dist_a,
col_dist_a)
00115
00116 END SUBROUTINE init_dbcsr

```

8.199 initcoulomb.f90 File Reference

Functions/Subroutines

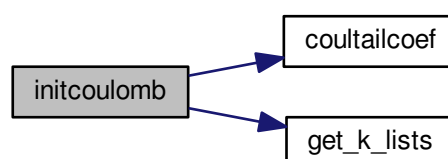
- subroutine [initcoulomb](#)

8.199.1 Function/Subroutine Documentation

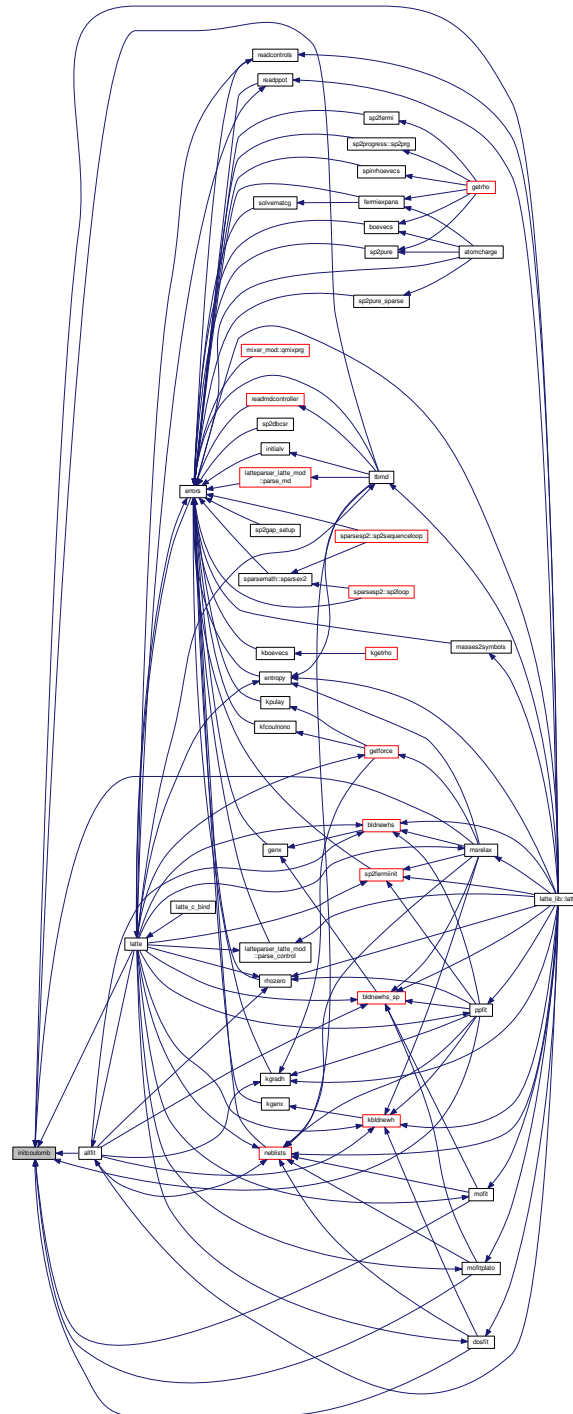
8.199.1.1 subroutine [initcoulomb](#) ()

Definition at line [23](#) of file [initcoulomb.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.200 initcoulomb.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE initcoulomb
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   REAL(LATTEPREC) :: A2XA3(3), A3XA1(3), A1XA2(3)
00032   REAL(LATTEPREC) :: DOT, P, TIMERRATIO, SQRTP
00033   IF (existerror) RETURN
00034
00035   keconst = 14.3996437701414*relperm
00036   tfact = 16.0/(5.0 * keconst)
00037
00038   IF (elecmeth.EQ. 1) THEN
00039
00040     !
00041     ! Get the coefficients for the cut-off tail for 1/R
00042     !
00043
00044     CALL coultailcoef
00045
00046     coulcut=abs(coulcut)
00047     coulcut2 = coulcut*coulcut
00048
00049   ELSE
00050
00051     twopi = two*pi
00052     pi2 = pi*pi
00053     sqrtpi = sqrt(pi)
00054     eightpi = eight*pi
00055
00056     ! First let's set up Ed's lattice_vecs
00057
00058     latticevecs = box
00059
00060     ! Ed's bit:
00061
00062     a2xa3(1) = latticevecs(2,2)*latticevecs(3,3) - &
00063               latticevecs(2,3)*latticevecs(3,2)
00064     a2xa3(2) = latticevecs(2,3)*latticevecs(3,1) - &
00065               latticevecs(2,1)*latticevecs(3,3)
00066     a2xa3(3) = latticevecs(2,1)*latticevecs(3,2) - &
00067               latticevecs(2,2)*latticevecs(3,1)
00068
00069     dot = latticevecs(1,1)*a2xa3(1) + latticevecs(1,2)*a2xa3(2) + &
00070          latticevecs(1,3)*a2xa3(3)
00071
00072     recipvecs(1,1) = twopi*a2xa3(1)/dot
00073     recipvecs(1,2) = twopi*a2xa3(2)/dot
00074     recipvecs(1,3) = twopi*a2xa3(3)/dot
00075
00076     a3xa1(1) = latticevecs(3,2)*latticevecs(1,3) - &
00077               latticevecs(3,3)*latticevecs(1,2)
00078     a3xa1(2) = latticevecs(3,3)*latticevecs(1,1) - &
00079               latticevecs(3,1)*latticevecs(1,3)
00080     a3xa1(3) = latticevecs(3,1)*latticevecs(1,2) - &
00081               latticevecs(3,2)*latticevecs(1,1)
00082
00083     dot = latticevecs(2,1)*a3xa1(1) + latticevecs(2,2)*a3xa1(2) + &
00084          latticevecs(2,3)*a3xa1(3)
00085
00086     recipvecs(2,1) = twopi*a3xa1(1)/dot
00087     recipvecs(2,2) = twopi*a3xa1(2)/dot
00088     recipvecs(2,3) = twopi*a3xa1(3)/dot
00089
00090     a1xa2(1) = latticevecs(1,2)*latticevecs(2,3) - &
00091               latticevecs(1,3)*latticevecs(2,2)
00092     a1xa2(2) = latticevecs(1,3)*latticevecs(2,1) - &
00093               latticevecs(1,1)*latticevecs(2,3)

```

```

00094      alxa2(3) = latticevecs(1,1)*latticevecs(2,2) - &
00095                latticevecs(1,2)*latticevecs(2,1)
00096
00097      dot = latticevecs(3,1)*alxa2(1) + latticevecs(3,2)*alxa2(2) + &
00098            latticevecs(3,3)*alxa2(3)
00099
00100      recipvecs(3,1) = twopi*alxa2(1)/dot
00101      recipvecs(3,2) = twopi*alxa2(2)/dot
00102      recipvecs(3,3) = twopi*alxa2(3)/dot
00103
00104      ! Calculate the cell volume
00105
00106      coulvol = dot
00107
00108      p = -log(coulacc)
00109      sqrtp = sqrt(p)
00110
00111      IF (coulcut .GT. zero) THEN
00112
00113          calpha = sqrtp/coulcut
00114          coulcut2 = coulcut*coulcut
00115          kcutoff = two*calpha*sqrtp
00116          kcutoff2 = kcutoff*kcutoff
00117          calpha2 = calpha*calpha
00118          fourcalpha2 = four*calpha2
00119
00120      ELSE
00121
00122          !
00123          ! Automatically determining the optimal real space
00124          ! cut-off if on input COUTCUT < 0. This is Sanville's code
00125          !
00126
00127          !   TIMERATIO = 50.0
00128          !   TIMERATIO = 1.00D0
00129          timeratio = 3.50d0
00130
00131          calpha = sqrtpi*((timeratio * REAL(NATS) / (coulvol*
coulvol))**(ONE/SIX))
00132          coulcut = sqrtp/calpha
00133
00134          !       PRINT*, "COUTCUT =", COUTCUT
00135          IF (coulcut .GT. five*ten) THEN
00136
00137              coulcut = five*ten
00138              calpha = sqrtp/coulcut
00139
00140          ENDIF
00141
00142          coulcut2 = coulcut*coulcut
00143          kcutoff = two*calpha*sqrtp
00144          kcutoff2 = kcutoff*kcutoff
00145          calpha2 = calpha*calpha
00146          fourcalpha2 = four*calpha2
00147
00148          ! Taking this bit from Coulomb Ewald so we don't have to
00149          ! recompute every time:
00150
00151      ENDIF
00152
00153      lmax = int(kcutoff / sqrt(recipvecs(1,1)*recipvecs(1,1) + &
00154            recipvecs(1,2)*recipvecs(1,2) + recipvecs(1,3)*
recipvecs(1,3)))
00155
00156      mmax = int(kcutoff / sqrt(recipvecs(2,1)*recipvecs(2,1) + &
00157            recipvecs(2,2)*recipvecs(2,2) + recipvecs(2,3)*
recipvecs(2,3)))
00158
00159      nmax = int(kcutoff / sqrt(recipvecs(3,1)*recipvecs(3,1) + &
00160            recipvecs(3,2)*recipvecs(3,2) + recipvecs(3,3)*
recipvecs(3,3)))
00161
00162
00163      ENDIF
00164
00165      CALL get_k_lists(recipvecs)
00166
00167  END SUBROUTINE initcoulomb

```

8.201 initcoulombklist.f90 File Reference

Functions/Subroutines

- subroutine [get_k_lists](#) (RECIP_VECTORS)
Constructs and list of vectors in the reciprocal space.

8.201.1 Function/Subroutine Documentation

8.201.1.1 subroutine [get_k_lists](#) (real(latteprec), dimension(3,3), intent(in) *RECIP_VECTORS*)

Constructs and list of vectors in the reciprocal space.

This routine constructs a list of reciprocal vectors needed to parallelize the reciprocal Ewald summation. These vectors are used as inputs for the COULOMBEWALD routine. Briefly, this will compute a set of vectors $K = \{\vec{k} \in \mathbb{R}^3 : k = l\vec{b}_1 + m\vec{b}_2 + n\vec{b}_3, \text{ with } 0 \leq l \leq l_{max}, 0 \leq m \leq m_{max} \text{ and } 0 \leq n \leq n_{max}\}$. In this case, $\vec{b}_1, \vec{b}_2, \vec{b}_3$ are the reciprocal vectors previously constructed in INITCOULOMB.

Parameters

<i>RECIP_VECTORS</i>	Reciprocal lattice vectors.
<i>K1_LIST</i>	(global output) list of Kx values.
<i>K2_LIST</i>	(global output) list of Ky values.
<i>K3_LIST</i>	(global output) list of Kz values.
<i>KSQ_LIST</i>	list of K2 (k squared) values.
<i>KCUTOFF</i>	(global input) Cutoff value for k.
<i>LMAX</i>	(global input) Number of point in the b1 direction.
<i>MMAX</i>	(global input) Number of point in the b2 direction.
<i>NMAX</i>	(global input) Number of point in the b3 direction.
<i>NK</i>	(global output) Number of total points in the reciprocal space.

Definition at line 44 of file [initcoulombklist.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00043 SUBROUTINE get_k_lists(RECIP_VECTORS)
00044
00045   USE constants_mod
00046   USE coulombarray
00047   USE myprecision
00048
00049   IMPLICIT NONE
00050   INTEGER                                :: I, L, M, N
00051   INTEGER                                :: LMIN, MMIN, NMIN
00052   REAL(LATTEPREC)                       :: L11, L12, L13, M21
00053   REAL(LATTEPREC)                       :: M22, M23
00054   REAL(LATTEPREC)                       :: K(3), K2
00055   REAL(LATTEPREC), INTENT(IN)           :: RECIP_VECTORS(3,3)
00056
00057   IF (existerror) RETURN
00058
00059   !Maximum value of vectors in the reciprocal space
00060   nk = (2*lmax+1)*(2*mmax+1)*(2*nmax+1)
00061
00062   IF (.NOT.ALLOCATED(k1_list)) ALLOCATE(k1_list(nk))
00063   IF (.NOT.ALLOCATED(k2_list)) ALLOCATE(k2_list(nk))
00064   IF (.NOT.ALLOCATED(k3_list)) ALLOCATE(k3_list(nk))
00065   IF (.NOT.ALLOCATED(ksq_list)) ALLOCATE(ksq_list(nk))
00066
00067   k1_list = 0.0
00068   k2_list = 0.0
00069   k3_list = 0.0
00070   ksq_list = 0.0
00071
00072   kcutoff2 = kcutoff * kcutoff
00073
00074   i = 1
00075   lmin = 0
00076
00077   DO l = lmin, lmax
00078
00079     IF (l == 0) THEN
00080       mmin = 0
00081     ELSE
00082       mmin = -mmax
00083     ENDIF
00084
00085     l11 = REAL(l, latteprec)*RECIP_VECTORS(1,1)
00086     l12 = REAL(l, latteprec)*RECIP_VECTORS(1,2)
00087     l13 = REAL(l, latteprec)*RECIP_VECTORS(1,3)
00088
00089     DO m = mmin, mmax
00090
00091       nmin = -nmax
00092
00093       IF (l .EQ. 0 .AND. m .EQ. 0) nmin = 1
00094
00095       m21 = l11 + REAL(m, latteprec)*RECIP_VECTORS(2,1)
00096       m22 = l12 + REAL(m, latteprec)*RECIP_VECTORS(2,2)
00097       m23 = l13 + REAL(m, latteprec)*RECIP_VECTORS(2,3)
00098
00099       DO n = nmin, nmax
00100
00101         k(1) = m21 + REAL(n, latteprec)*RECIP_VECTORS(3,1)
00102         k(2) = m22 + REAL(n, latteprec)*RECIP_VECTORS(3,2)
00103         k(3) = m23 + REAL(n, latteprec)*RECIP_VECTORS(3,3)
00104
00105         k2 = k(1)*k(1) + k(2)*k(2) + k(3)*k(3)
00106
00107         IF (k2 .LE. kcutoff2) THEN
00108
00109           k1_list(i) = k(1)
00110           k2_list(i) = k(2)
00111           k3_list(i) = k(3)
00112           ksq_list(i) = k2
00113           i = i + 1
00114

```

```
00115         END IF
00116
00117         END DO
00118
00119     END DO
00120
00121 END DO
00122
00123 nk = i-1
00124
00125 END SUBROUTINE get_k_lists
```

8.203 initialv.f90 File Reference

Functions/Subroutines

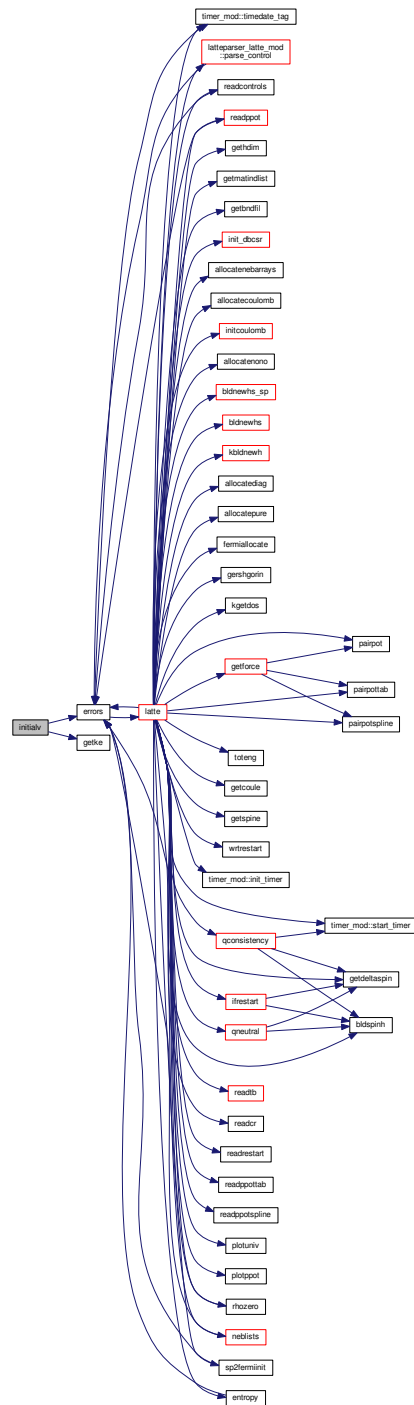
- subroutine [initialv](#)

8.203.1 Function/Subroutine Documentation

8.203.1.1 subroutine [initialv](#) ()

Definition at line [23](#) of file [initialv.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE initialv
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, J, K, N, CLOCK, COUNT
00032   REAL(LATTEPREC) :: RN(3)
00033   REAL(LATTEPREC) :: MYMASS, VELFACTOR
00034   REAL(LATTEPREC) :: CONV, TCURRENT, RESCALE
00035   REAL(LATTEPREC) :: MOMENTUM(3), STDDEV, BOLTZ
00036   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00037   IF (existerror) RETURN
00038
00039   momentum = zero
00040
00041   IF ( rndist .EQ. "UNIFORM" ) THEN
00042
00043     DO i = 1, nats
00044
00045       mymass = mass(elempointer(i))
00046
00047       velfactor = sqrt(one/mymass)
00048
00049       CALL random_number(rn)
00050
00051       v(1,i) = (two*rn(1) - one) * velfactor
00052       v(2,i) = (two*rn(2) - one) * velfactor
00053       v(3,i) = (two*rn(3) - one) * velfactor
00054
00055       momentum(1) = momentum(1) + v(1,i)*mymass
00056       momentum(2) = momentum(2) + v(2,i)*mymass
00057       momentum(3) = momentum(3) + v(3,i)*mymass
00058
00059     ENDDO
00060
00061   ELSEIF ( rndist .EQ. "GAUSSIAN" ) THEN
00062
00063     setth = 0
00064
00065     boltz = one/ke2t
00066
00067     DO i = 1, nats
00068
00069       mymass = mass(elempointer(i))
00070       stddev = sqrt(boltz*ttarget/(mymass*mvv2ke))
00071
00072       v(1,i) = gaussrn(zero, stddev)
00073       v(2,i) = gaussrn(zero, stddev)
00074       v(3,i) = gaussrn(zero, stddev)
00075
00076       momentum(1) = momentum(1) + v(1,i)*mymass
00077       momentum(2) = momentum(2) + v(2,i)*mymass
00078       momentum(3) = momentum(3) + v(3,i)*mymass
00079
00080     ENDDO
00081
00082   ELSE
00083
00084     CALL errors("initialv","Choose either UNIFORM or GAUSSIAN for the &
00085               & random number distribution")
00086
00087   ENDIF
00088
00089   momentum = momentum/summass
00090
00091   DO i = 1, nats
00092
00093     v(1,i) = v(1,i) - momentum(1)

```

```
00094      v(2,i) = v(2,i) - momentum(2)
00095      v(3,i) = v(3,i) - momentum(3)
00096
00097      ENDDO
00098
00099      CALL getke
00100
00101      rescale = sqrt(ttargt/temperature)
00102
00103      v = rescale * v
00104
00105      RETURN
00106
00107 END SUBROUTINE initialv
```

8.205 initrng.f90 File Reference

Functions/Subroutines

- subroutine [initrng](#)

8.205.1 Function/Subroutine Documentation

8.205.1.1 subroutine [initrng](#) ()

Definition at line 23 of file [initrng.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE initrng
00023
00024     USE constants_mod
00025
00026     #ifdef MPI_ON
00027         USE mpi
00028     #endif
00029
00030     IMPLICIT NONE
00031
00032     INTEGER :: I, N, COUNT, CLOCK
00033     INTEGER, ALLOCATABLE :: SEED(:)
00034     INTEGER :: MYID, IERR
00035     IF (existerror) RETURN
00036
00037     myid = 0
00038
00039     #ifdef MPI_ON
00040         CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00041     #endif
00042
00043
00044     CALL random_seed(SIZE = n)
00045
00046     ALLOCATE (seed(n))
00047
00048     CALL system_clock(count = clock)
00049
00050     ! Adding MPI rank to the seed should make each rank use a different seed...
00051
00052     IF (parrep .EQ. 0) myid = 0
00053
00054     DO i = 1, n
00055         seed(i) = clock + 37*(i - 1) + myid
00056     ENDDO
00057
00058     CALL random_seed(put = seed)
00059
00060     DEALLOCATE (seed)
00061
00062     RETURN
00063
00064 END SUBROUTINE initrng

```

8.207 initshockcomp.f90 File Reference

Functions/Subroutines

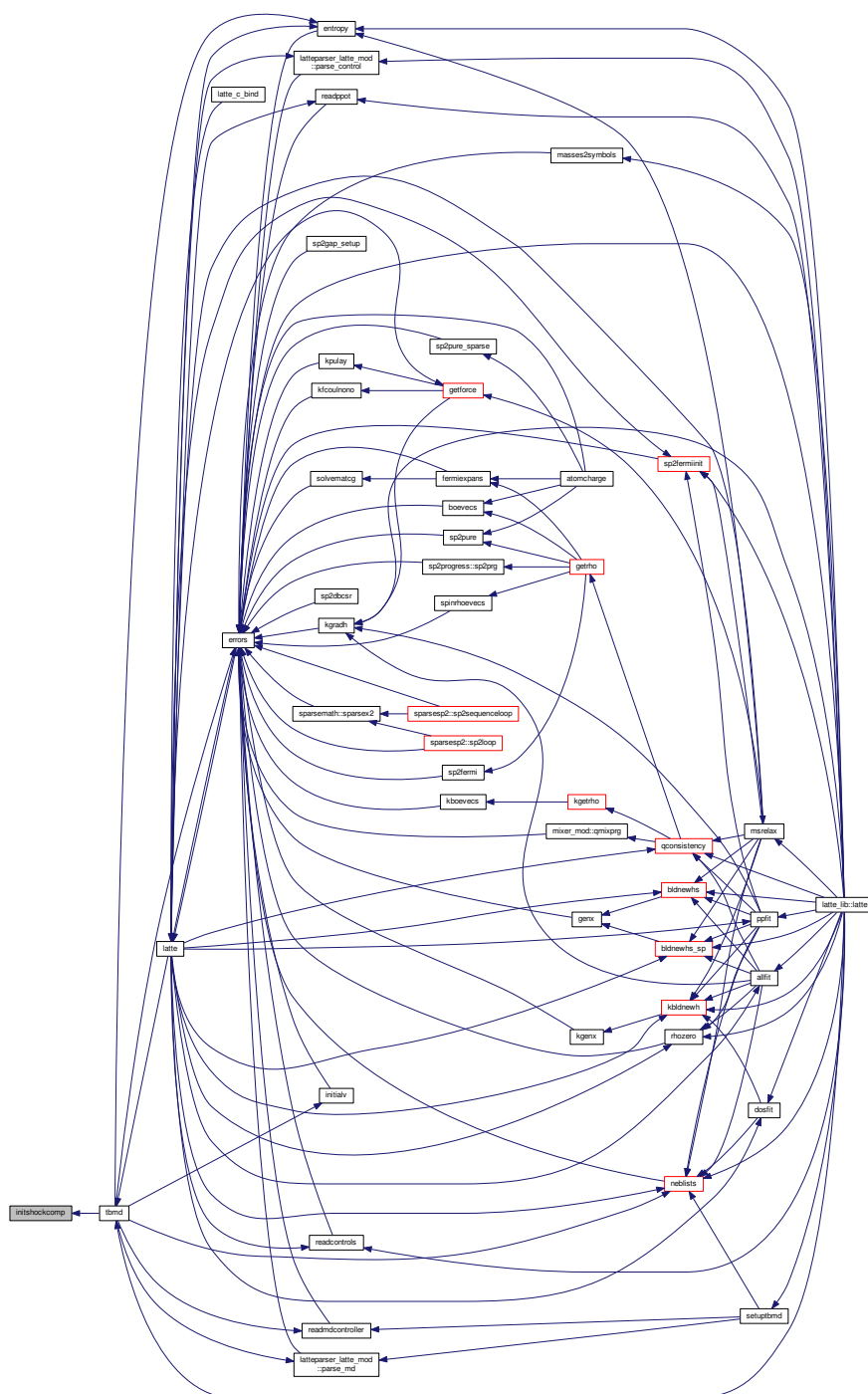
- subroutine [initshockcomp](#)

8.207.1 Function/Subroutine Documentation

8.207.1.1 subroutine initshockcomp ()

Definition at line 23 of file [initshockcomp.f90](#).

8.208 initshockcomp.f90



```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE initshockcomp
00023
00024   USE constants_mod
00025   USE mdarray
00026
00027   IMPLICIT NONE
00028   IF (existerror) RETURN
00029
00030   ! First convert units of Up and Us from m/s to A/fs
00031
00032   uparticle = uparticle*1.0d-5
00033   ushock = ushock*1.0d-5
00034
00035   ! If USHOCK is positive, then we'll use it. If it's negative then
00036   ! we'll compute it using the sound velocity (input) and the Universal
00037   ! liquid Hugoniot
00038
00039   IF (ushock .LT. 0.0) THEN
00040
00041     ! Converting sound velocity from m/s to A/fs
00042
00043     c0 = c0 * 1.0d-5
00044
00045     ushock = c0 * (1.37d0 - 0.37d0*exp( -two*uparticle / c0 )) &
00046             + 1.62d0*uparticle
00047
00048   ENDIF
00049
00050   ! PRINT*, "Up = ", UPARTICLE, "Us = ", USHOCK
00051   !
00052   ! The duration of the shock, i.e., the time taken for the
00053   ! shock front to traverse the simulation cell is t_dur = l_0/Us
00054   !
00055   ! So... we'll start the Hugoniotstat at timestep SHOCKSTART and
00056   ! turn it off INT(L_0/(U_s *dt)) time steps later
00057   !
00058
00059   ! If the box has 90 degree angles, then its length in the three
00060   ! directions is BOX(1,1), BOX(2,2), AND BOX(3,3)...
00061
00062   shockstop = shockstart + &
00063             int(box(shockdir,shockdir)/(ushock * dt))
00064   ! SHOCKSTOP = SHOCKSTART + &
00065   !             INT( (BOX(2,SHOCKDIR) - BOX(1,SHOCKDIR))/(USHOCK * DT))
00066
00067   ! PRINT*, "start, stop = ", SHOCKSTART, SHOCKSTOP
00068
00069 END SUBROUTINE initshockcomp
00070

```

8.209 kbldnewh.f90 File Reference

Functions/Subroutines

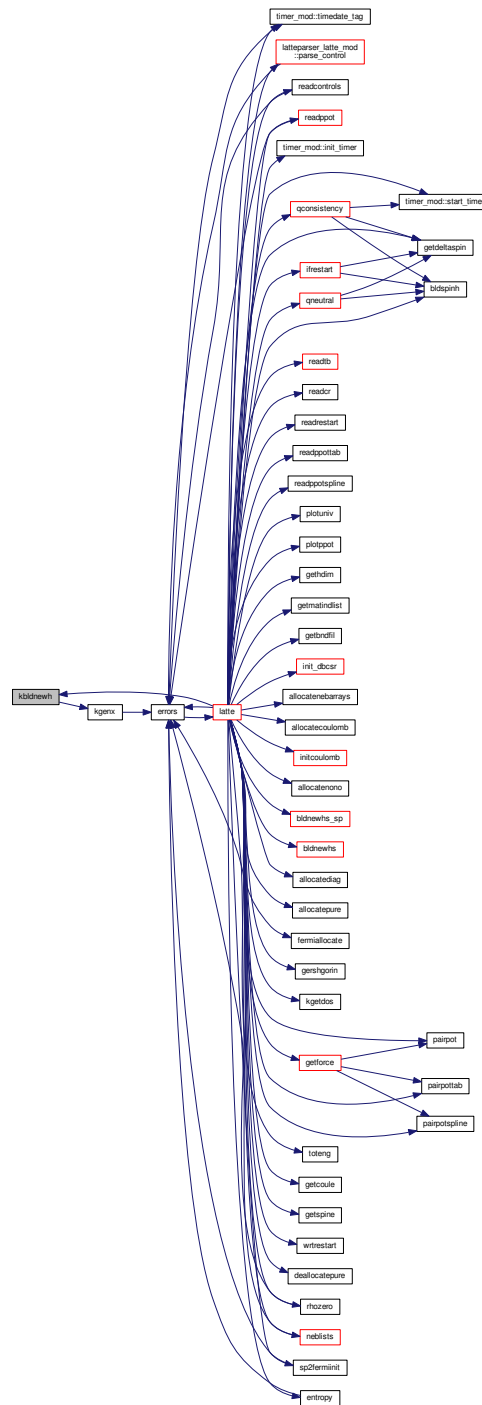
- subroutine [kbldnewh](#)

8.209.1 Function/Subroutine Documentation

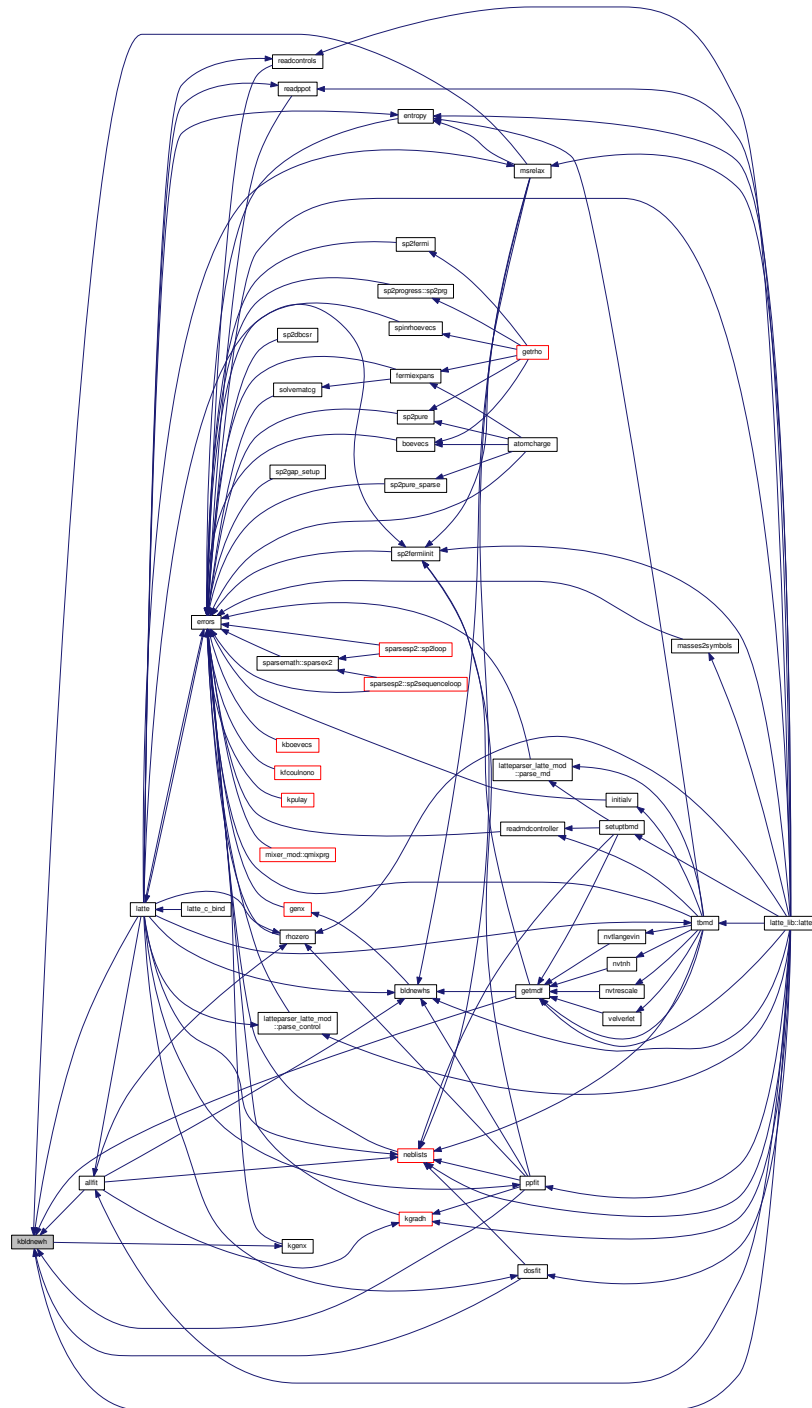
8.209.1.1 subroutine kbldnewh ()

Definition at line 23 of file [kbldnewh.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.210 kbldnewh.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kblidnewh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE xboarray
00028   USE nonoarray
00029   USE univarray
00030   USE kspacearray
00031   USE myprecision
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, J, NEWJ, K, L, II, JJ, KK, MM, MP, NN, SUBI
00036   INTEGER :: IBRA, IKET, LBRA, LKET, MBRA, MKET
00037   INTEGER :: INDEX, INDI, INDJ
00038   INTEGER :: SWITCH, PREVJ
00039   INTEGER :: PBCI, PBCJ, PBCK
00040   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00041   INTEGER :: NORBI, KCOUNT
00042   INTEGER :: KX, KY, KZ
00043   REAL(LATTEPREC) :: KX0, KY0, KZ0
00044   REAL(LATTEPREC) :: ALPHA, BETA, COSBETA, PHI, TMP, PERM
00045   REAL(LATTEPREC) :: RIJ(3), MAGR2, MAGR, MAGRP, RCUTTB
00046   REAL(LATTEPREC) :: MAXARRAY(20), MAXRCUT, MAXRCUT2
00047   REAL(LATTEPREC) :: ANGFACTOR
00048   REAL(LATTEPREC) :: KPOINT(3), KDOTL
00049   REAL(LATTEPREC) :: AMMBRA, WIGLBRAMBRA
00050   REAL(LATTEPREC) :: B1(3), B2(3), B3(3), A1A2XA3, K0(3)
00051   COMPLEX(LATTEPREC) :: BLOCH, KHTMP, KSTMP
00052   REAL(LATTEPREC), EXTERNAL :: UNIVSCALE, WIGNERD, SLMP, TLMMP, AM, BM
00053   IF (existerror) RETURN
00054
00055   hk = cmplx(zero)
00056
00057   ! Computing the reciprocal lattice vectors
00058
00059   b1(1) = box(2,2)*box(3,3) - box(3,2)*box(2,3)
00060   b1(2) = box(3,1)*box(2,3) - box(2,1)*box(3,3)
00061   b1(3) = box(2,1)*box(3,2) - box(3,1)*box(2,2)
00062
00063   a1a2xa3 = box(1,1)*b1(1) + box(1,2)*b1(2) + box(1,3)*b1(3)
00064
00065   ! B1 = (A2 X A3)/(A1.(A2 X A3))
00066
00067   b1 = b1/a1a2xa3
00068
00069   ! B2 = (A3 x A1)/(A1(A2 X A3))
00070
00071   b2(1) = (box(3,2)*box(1,3) - box(1,2)*box(3,3))/a1a2xa3
00072   b2(2) = (box(1,1)*box(3,3) - box(3,1)*box(1,3))/a1a2xa3
00073   b2(3) = (box(3,1)*box(1,2) - box(1,1)*box(3,2))/a1a2xa3
00074
00075   ! B3 = (A1 x A2)/(A1(A2 X A3))
00076
00077   b3(1) = (box(1,2)*box(2,3) - box(2,2)*box(1,3))/a1a2xa3
00078   b3(2) = (box(2,1)*box(1,3) - box(1,1)*box(2,3))/a1a2xa3
00079   b3(3) = (box(1,1)*box(2,2) - box(2,1)*box(1,2))/a1a2xa3
00080
00081   index = 0
00082
00083   ! Build diagonal elements
00084   DO i = 1, nats
00085
00086     SELECT CASE(basis(elempointer(i)))
00087
00088     CASE("s")
00089
00090       index = index + 1
00091       hk(index, index, 1) = cmplx(hes(elempointer(i)))
00092
00093     CASE("p")

```

```

00094
00095     DO subi = 1, 3
00096         index = index + 1
00097         hk(index,index, 1) = cmplx(hep(elempointer(i)))
00098     ENDDO
00099
00100 CASE("d")
00101
00102     DO subi = 1, 5
00103         index = index + 1
00104         hk(index,index, 1) = cmplx(hed(elempointer(i)))
00105     ENDDO
00106
00107 CASE("f")
00108
00109     DO subi = 1, 7
00110         index = index + 1
00111         hk(index,index, 1) = cmplx(hef(elempointer(i)))
00112     ENDDO
00113
00114 CASE("sp")
00115
00116     DO subi = 1, 4
00117
00118         index = index + 1
00119         IF (subi .EQ. 1) THEN
00120             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00121         ELSE
00122             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00123         ENDIF
00124
00125     ENDDO
00126
00127 CASE("sd")
00128
00129     DO subi = 1, 6
00130
00131         index = index + 1
00132         IF (subi .EQ. 1) THEN
00133             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00134         ELSE
00135             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00136         ENDIF
00137
00138     ENDDO
00139
00140 CASE("sf")
00141
00142     DO subi = 1, 8
00143
00144         index = index + 1
00145         IF (subi .EQ. 1) THEN
00146             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00147         ELSE
00148             hk(index,index, 1) = cmplx(hef(elempointer(i)))
00149         ENDIF
00150
00151     ENDDO
00152
00153 CASE("pd")
00154
00155     DO subi = 1, 8
00156
00157         index = index + 1
00158         IF (subi .LE. 3) THEN
00159             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00160         ELSE
00161             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00162         ENDIF
00163
00164     ENDDO
00165
00166 CASE("pf")
00167
00168     DO subi = 1, 10
00169
00170         index = index + 1
00171         IF (subi .LE. 3) THEN
00172             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00173         ELSE
00174             hk(index,index, 1) = cmplx(hef(elempointer(i)))
00175         ENDIF
00176
00177     ENDDO
00178
00179 CASE("df")
00180

```

```

00181      DO subi = 1, 12
00182
00183          index = index + 1
00184          IF (subi .LE. 5) THEN
00185              hk(index,index, 1) = cmplx(hed(elempointer(i)))
00186          ELSE
00187              hk(index,index, 1) = cmplx(hef(elempointer(i)))
00188          ENDIF
00189
00190      ENDDO
00191
00192      CASE("spd")
00193
00194      DO subi = 1, 9
00195
00196          index = index + 1
00197          IF (subi .EQ. 1) THEN
00198              hk(index,index, 1) = cmplx(hes(elempointer(i)))
00199          ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00200              hk(index,index, 1) = cmplx(hep(elempointer(i)))
00201          ELSE
00202              hk(index,index, 1) = cmplx(hed(elempointer(i)))
00203          ENDIF
00204
00205      ENDDO
00206
00207      CASE("spf")
00208
00209      DO subi = 1, 11
00210
00211          index = index + 1
00212          IF (subi .EQ. 1) THEN
00213              hk(index,index, 1) = cmplx(hes(elempointer(i)))
00214          ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00215              hk(index,index, 1) = cmplx(hep(elempointer(i)))
00216          ELSE
00217              hk(index,index, 1) = cmplx(hef(elempointer(i)))
00218          ENDIF
00219
00220      ENDDO
00221
00222      CASE("sdf")
00223
00224      DO subi = 1, 13
00225
00226          index = index + 1
00227          IF (subi .EQ. 1) THEN
00228              hk(index,index, 1) = cmplx(hes(elempointer(i)))
00229          ELSEIF (subi .GT. 1 .AND. subi .LE. 6) THEN
00230              hk(index,index, 1) = cmplx(hed(elempointer(i)))
00231          ELSE
00232              hk(index,index, 1) = cmplx(hef(elempointer(i)))
00233          ENDIF
00234
00235      ENDDO
00236
00237      CASE("pdf")
00238
00239      DO subi = 1, 15
00240
00241          index = index + 1
00242          IF (subi .LE. 3) THEN
00243              hk(index,index, 1) = cmplx(hep(elempointer(i)))
00244          ELSEIF (subi .GT. 3 .AND. subi .LE. 8) THEN
00245              hk(index,index, 1) = cmplx(hed(elempointer(i)))
00246          ELSE
00247              hk(index,index, 1) = cmplx(hef(elempointer(i)))
00248          ENDIF
00249
00250      ENDDO
00251
00252      CASE("spdf")
00253
00254      DO subi = 1, 16
00255
00256          index = index + 1
00257          IF (subi .EQ. 1) THEN
00258              hk(index, index, 1) = cmplx(hes(elempointer(i)))
00259          ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00260              hk(index, index, 1) = cmplx(hep(elempointer(i)))
00261          ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00262              hk(index, index, 1) = cmplx(hep(elempointer(i)))
00263          ELSEIF (subi .GT. 4 .AND. subi .LE. 9) THEN
00264              hk(index, index, 1) = cmplx(hed(elempointer(i)))
00265          ELSE
00266              hk(index, index, 1) = cmplx(hef(elempointer(i)))
00267

```

```

00268         ENDIF
00269
00270         ENDDO
00271
00272         END SELECT
00273
00274     ENDDO
00275
00276     DO i = 2, nktot
00277         DO j = 1, hdim
00278             hk(j,j,i) = hk(j,j,1)
00279         ENDDO
00280     ENDDO
00281
00282     ! We assign the diagonal elements in ADDQDEP
00283
00284
00285     IF (basistype .EQ. "NONORTHO") THEN
00286
00287         sk = cmplx(zero)
00288
00289         DO i = 1, nktot
00290             DO j = 1, hdim
00291                 sk(j,j,i) = cmplx(one,zero)
00292             ENDDO
00293         ENDDO
00294
00295     ENDIF
00296
00297     k0 = pi*(one - REAL(nkx))/(REAL(nkx))*b1 + &
00298         PI*(one - REAL(nky))/(REAL(nky))*B2 + &
00299         PI*(one - REAL(nkz))/(REAL(nkz))*B3 - PI*KSHIFT
00300
00301     !$OMP PARALLEL DO DEFAULT (NONE) &
00302     !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00303     !$OMP SHARED(CR, BOX, B1, B2, B3, NOINT, ATELE, ELE1, ELE2, HK, SK) &
00304     !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE, K0, NKX, NKY, NKZ) &
00305     !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00306     !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP, PHI, ALPHA, BETA, COSBETA) &
00307     !$OMP PRIVATE(LBRAINC, LBRA, MBRA, L, LKETINC, LKET, MKET) &
00308     !$OMP PRIVATE(BLOCH, KDOTL, KPOINT, KCOUNT, KHTMP, KSTMP)&
00309     !$OMP PRIVATE(RCUTTB, IBRA, IKET, AMMBRA, WIGLBRAMBRA, ANGFACTOR, MP)
00310     !!$OMP REDUCTION(+:HK)
00311
00312     DO i = 1, nats
00313
00314         ! Build the lists of orbitals on each atom
00315
00316         SELECT CASE(basis(elempointer(i)))
00317
00318         CASE("s")
00319             basisi(1) = 0
00320             basisi(2) = -1
00321         CASE("p")
00322             basisi(1) = 1
00323             basisi(2) = -1
00324         CASE("d")
00325             basisi(1) = 2
00326             basisi(2) = -1
00327         CASE("f")
00328             basisi(1) = 3
00329             basisi(2) = -1
00330         CASE("sp")
00331             basisi(1) = 0
00332             basisi(2) = 1
00333             basisi(3) = -1
00334         CASE("sd")
00335             basisi(1) = 0
00336             basisi(2) = 2
00337             basisi(3) = -1
00338         CASE("sf")
00339             basisi(1) = 0
00340             basisi(2) = 3
00341             basisi(3) = -1
00342         CASE("pd")
00343             basisi(1) = 1
00344             basisi(2) = 2
00345             basisi(3) = -1
00346         CASE("pf")
00347             basisi(1) = 1
00348             basisi(2) = 3
00349             basisi(3) = -1
00350         CASE("df")
00351             basisi(1) = 2
00352             basisi(2) = 3
00353             basisi(3) = -1
00354         CASE("spd")

```

```

00355      basisi(1) = 0
00356      basisi(2) = 1
00357      basisi(3) = 2
00358      basisi(4) = -1
00359      CASE("spf")
00360      basisi(1) = 0
00361      basisi(2) = 1
00362      basisi(3) = 3
00363      basisi(4) = -1
00364      CASE("sdf")
00365      basisi(1) = 0
00366      basisi(2) = 2
00367      basisi(3) = 3
00368      basisi(4) = -1
00369      CASE("pdf")
00370      basisi(1) = 1
00371      basisi(2) = 2
00372      basisi(3) = 3
00373      basisi(4) = -1
00374      CASE("spdf")
00375      basisi(1) = 0
00376      basisi(2) = 1
00377      basisi(3) = 2
00378      basisi(4) = 3
00379      basisi(5) = -1
00380      END SELECT
00381
00382      indi = matindlist(i)
00383
00384      ! open loop over neighbors J of atom I
00385      DO newj = 1, totnebtb(i)
00386
00387          j = nebtb(1, newj, i)
00388
00389          pbci = nebtb(2, newj, i)
00390          pbcj = nebtb(3, newj, i)
00391          pbck = nebtb(4, newj, i)
00392
00393          rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00394              REAL(pbck)*BOX(3,1) - CR(1,i)
00395
00396          rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00397              REAL(pbck)*BOX(3,2) - CR(2,i)
00398
00399          rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00400              REAL(pbck)*BOX(3,3) - CR(3,i)
00401
00402          magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00403
00404          rcuttb = zero
00405
00406          DO k = 1, noint
00407
00408              IF ( (atele(i) .EQ. ele1(k) .AND. &
00409                  atele(j) .EQ. ele2(k)) .OR. &
00410                  (atele(j) .EQ. ele1(k) .AND. &
00411                  atele(i) .EQ. ele2(k)) ) THEN
00412
00413                  IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00414
00415                  IF (basistype .EQ. "NONORTHO") THEN
00416                      IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00417                  ENDIF
00418
00419              ENDIF
00420
00421          ENDDO
00422
00423          IF (magr2 .LT. rcuttb*rcuttb) THEN
00424
00425              magr = sqrt(magr2)
00426
00427              SELECT CASE(basis(elempointer(j)))
00428              CASE("s")
00429                  basisj(1) = 0
00430                  basisj(2) = -1
00431              CASE("p")
00432                  basisj(1) = 1
00433                  basisj(2) = -1
00434              CASE("d")
00435                  basisj(1) = 2
00436                  basisj(2) = -1
00437              CASE("f")
00438                  basisj(1) = 3
00439                  basisj(2) = -1
00440              CASE("sp")
00441                  basisj(1) = 0

```

```

00442         basisj(2) = 1
00443         basisj(3) = -1
00444     CASE("sd")
00445         basisj(1) = 0
00446         basisj(2) = 2
00447         basisj(3) = -1
00448     CASE("sf")
00449         basisj(1) = 0
00450         basisj(2) = 3
00451         basisj(3) = -1
00452     CASE("pd")
00453         basisj(1) = 1
00454         basisj(2) = 2
00455         basisj(3) = -1
00456     CASE("pf")
00457         basisj(1) = 1
00458         basisj(2) = 3
00459         basisj(3) = -1
00460     CASE("df")
00461         basisj(1) = 2
00462         basisj(2) = 3
00463         basisj(3) = -1
00464     CASE("spd")
00465         basisj(1) = 0
00466         basisj(2) = 1
00467         basisj(3) = 2
00468         basisj(4) = -1
00469     CASE("spf")
00470         basisj(1) = 0
00471         basisj(2) = 1
00472         basisj(3) = 3
00473         basisj(4) = -1
00474     CASE("sdf")
00475         basisj(1) = 0
00476         basisj(2) = 2
00477         basisj(3) = 3
00478         basisj(4) = -1
00479     CASE("pdf")
00480         basisj(1) = 1
00481         basisj(2) = 2
00482         basisj(3) = 3
00483         basisj(4) = -1
00484     CASE("spdf")
00485         basisj(1) = 0
00486         basisj(2) = 1
00487         basisj(3) = 2
00488         basisj(4) = 3
00489         basisj(5) = -1
00490     END SELECT
00491
00492     indj = matindlist(j)
00493
00494     magrp = sqrt(rij(1) * rij(1) + rij(2) * rij(2))
00495
00496     ! transform to system in which z-axis is aligned with RIJ,
00497     IF (abs(rij(1)) .GT. 1.0e-12) THEN
00498
00499         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00500             phi = zero
00501         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00502             phi = two * pi
00503         ELSE
00504             phi = pi
00505         ENDIF
00506         alpha = atan(rij(2) / rij(1)) + phi
00507
00508     ELSEIF (abs(rij(2)) .GT. 1.0e-12) THEN
00509
00510         IF (rij(2) .GT. 1.0e-12) THEN
00511             alpha = pi / two
00512         ELSE
00513             alpha = three * pi / two
00514         ENDIF
00515
00516     ELSE
00517
00518         ! pathological case: beta=0 and alpha undefined, but
00519         ! this doesn't matter for matrix elements
00520
00521         alpha = zero
00522
00523     ENDIF
00524
00525     cosbeta = rij(3)/magr
00526     beta = acos( rij(3) / magr )
00527
00528     ! Build matrix elements using eqns (1)-(9) in PRB 72 165107

```



```

00529
00530      ! The loops over LBRA and LKET need to take into account
00531      ! the orbitals assigned to each atom, e.g., sd rather than
00532      ! spd...
00533
00534      ibra = indi + 1
00535
00536      lbrainc = 1
00537      DO WHILE (basisi(lbrainc) .NE. -1)
00538
00539          lbra = basisi(lbrainc)
00540          lbrainc = lbrainc + 1
00541
00542          DO mbra = -lbra, lbra
00543
00544              ! We can calculate these two outside the
00545              ! MKET loop...
00546
00547              ammbra = am(mbra, alpha)
00548              wiglbrambra = wignerd(lbra, abs(mbra), 0, cosbeta)
00549
00550              iket = indj + 1
00551
00552              lketinc = 1
00553              DO WHILE (basisj(lketinc) .NE. -1)
00554
00555                  lket = basisj(lketinc)
00556                  lketinc = lketinc + 1
00557
00558                  DO mket = -lket, lket
00559
00560                      ! This is the sigma bonds (mp = 0)
00561
00562                      ! Hamiltonian build
00563
00564                      ! Pre-compute the angular part so we can use it
00565                      ! again later if we're building the S matrix too
00566
00567                      angfactor = two * ammbra * &
00568                      am(mket, alpha) * &
00569                      wiglbrambra * &
00570                      wignerd(lket, abs(mket), 0, cosbeta)
00571
00572                      khtmp = cmplx(angfactor * &
00573                      univscale(i, j, lbra, lket, &
00574                      0, magr, "H"))
00575
00576                      IF (basistype .EQ. "NONORTHO") THEN
00577                          kstmp = cmplx(angfactor * &
00578                          univscale(i, j, lbra, lket, &
00579                          0, magr, "S"))
00580                      ENDIF
00581
00582                      kcount = 0
00583
00584                      !Loop over k-points
00585
00586                      DO kx = 1, nkx
00587
00588                          DO ky = 1, nky
00589
00590                              DO kz = 1, nkz
00591
00592                                  kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00593                                  REAL(ky-1)*B2/REAL(NKY) + &
00594                                  REAL(kz-1)*B3/REAL(nkz)) + k0
00595
00596                                  kcount = kcount+1
00597
00598                                  kdotl = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00599                                  kpoint(3)*rij(3)
00600
00601                                  bloch = exp(cmplx(zero,kdotl))
00602
00603                                  hk(ibra, iket, kcount) = &
00604                                  hk(ibra, iket, kcount) + &
00605                                  bloch*khtmp
00606
00607                                  IF (basistype .EQ. "NONORTHO") THEN
00608                                      sk(ibra, iket, kcount) = &
00609                                      sk(ibra, iket, kcount) + &
00610                                      bloch*kstmp
00611                                  ENDIF
00612                              ENDDO
00613                          ENDDO
00614                      ENDDO
00615                  ENDDO

```

```

00616
00617
00618      ! everything else
00619
00620      DO mp = 1, min(lbra, lket)
00621
00622          angfactor = &
00623              slmmp(lbra, mbra, mp, alpha, cosbeta)* &
00624              slmmp(lket, mket, mp, alpha, cosbeta)+ &
00625              tlmmp(lbra, mbra, mp, alpha, cosbeta)* &
00626              tlmmp(lket, mket, mp, alpha, cosbeta)
00627
00628          khtmp = cmplx(angfactor * &
00629              univscale(i, j, lbra, lket, &
00630                  mp, magr, "H"))
00631
00632          IF (basistype .EQ. "NONORTHO") THEN
00633              kstmp = cmplx(angfactor * &
00634                  univscale(i, j, lbra, lket, &
00635                      mp, magr, "S"))
00636          ENDIF
00637
00638          kcount = 0
00639
00640          DO kx = 1, nkx
00641              DO ky = 1, nky
00642                  DO kz = 1, nkz
00643
00644                      kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00645                          REAL(ky-1)*B2/REAL(NKY) + &
00646                          REAL(kz-1)*B3/REAL(nkz)) + k0
00647
00648                      kcount = kcount+1
00649
00650                      kdot1 = kpoint(1)*rij(1) + &
00651                          kpoint(2)*rij(2) + kpoint(3)*rij(3)
00652
00653                      bloch = exp(cmplx(zero,kdot1))
00654
00655                      hk(ibra, iket, kcount) = &
00656                          hk(ibra, iket, kcount) + &
00657                          bloch*khtmp
00658
00659                      IF (basistype .EQ. "NONORTHO") THEN
00660                          sk(ibra, iket, kcount) = &
00661                              sk(ibra, iket, kcount) + &
00662                              bloch*kstmp
00663                      ENDIF
00664
00665                  ENDDO
00666              ENDDO
00667          ENDDO
00668
00669          ENDDO
00670
00671          iket = iket + 1
00672
00673      ENDDO
00674
00675      ENDDO
00676
00677      ibra = ibra + 1
00678
00679      ENDDO
00680  ENDDO
00681  ENDDO
00682  ENDDO
00683  ENDDO
00684
00685  !$OMP END PARALLEL DO
00686
00687  ! Save the diagonal elements: it will help a lot when we add in the partial charges
00688
00689  IF (basistype .EQ. "ORTHO") THEN
00690
00691      DO i = 1, nktot
00692          DO j = 1, hdim
00693              hkdiag(j,i) = hk(j,j,i)
00694          ENDDO
00695      ENDDO
00696
00697  ELSEIF (basistype .EQ. "NONORTHO") THEN
00698
00699      ! keep a copy of the charge-independent Hamiltonian
00700
00701      hk0 = hk
00702

```

```

00703      CALL kgenx ! Get the factors for the Lowdin transform
00704
00705      ENDIF
00706
00707
00708      IF (debugon .EQ. 1 ) THEN
00709
00710          OPEN(unit=31, status="UNKNOWN", file="myS0_k_R.dat")
00711
00712          OPEN(unit=32, status="UNKNOWN", file="myS0_k_I.dat")
00713
00714          DO k = 1, nktot
00715              WRITE(31,*) k
00716              WRITE(32,*) k
00717              DO i = 1, hdim
00718                  WRITE(31,10) (REAL(SK(I,J,K)), J = 1, hdim)
00719                  WRITE(32,10) (aimag(sk(i,j,k)), j = 1, hdim)
00720              ENDDO
00721          ENDDO
00722
00723          CLOSE(31)
00724          CLOSE(32)
00725
00726 10    FORMAT(648f12.6)
00727
00728      ENDIF
00729
00730      RETURN
00731
00732 END SUBROUTINE kbldnewh

```

8.211 kbodirect.f90 File Reference

Functions/Subroutines

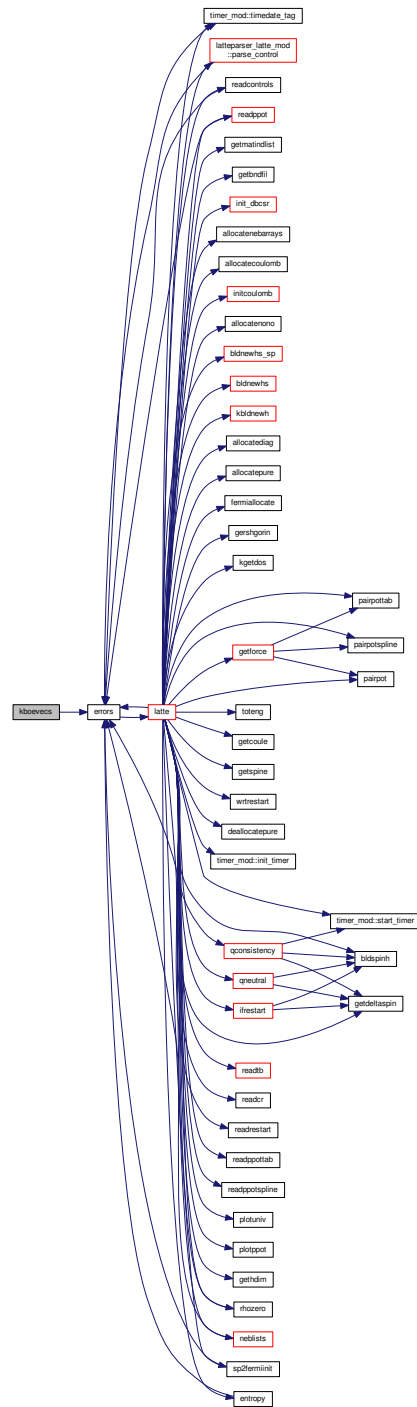
- subroutine [kboevcs](#)

8.211.1 Function/Subroutine Documentation

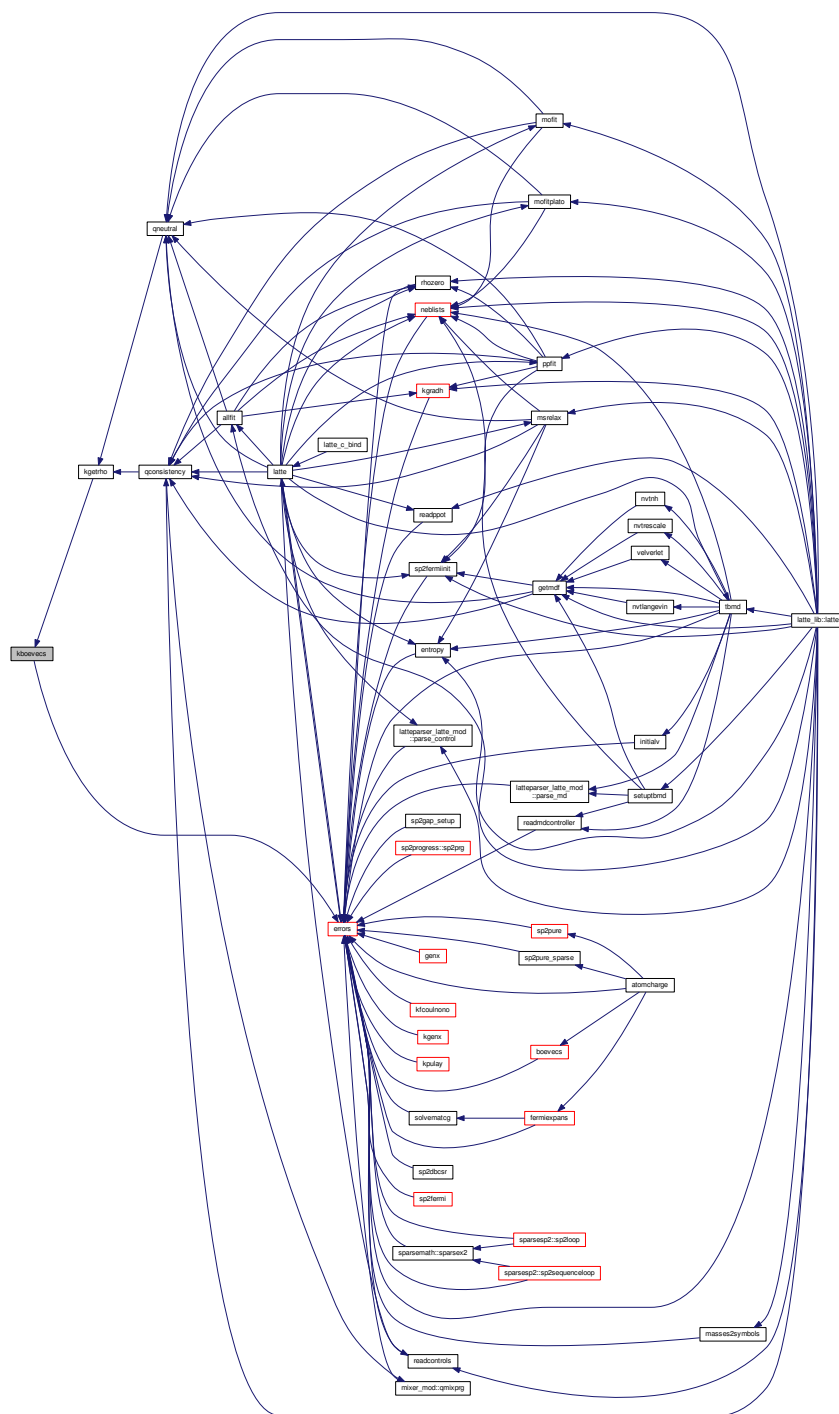
8.211.1.1 subroutine kboevcs ()

Definition at line 23 of file [kbodirect.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.212 kbodirect.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kboevects
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE myprecision
00028   USE kspacearray
00029   USE neblistarray
00030   USE mdarray
00031 #ifdef MPI_ON
00032   USE mpi
00033 #endif
00034
00035   IMPLICIT NONE
00036
00037   INTEGER :: I, J, K
00038   INTEGER :: ITER, BREAKLOOP, LOOPTARGET, COUNT, MYKPOINT
00039   INTEGER :: CPOC(1)
00040   REAL(LATTEPREC) :: OCCTARGET, OCC, FDIRAC, DFDIRAC
00041   REAL(LATTEPREC) :: OCCERROR, SHIFTC, FDIRACARG, EXPARG
00042   REAL(LATTEPREC), PARAMETER :: MAXSHIFT = one
00043   REAL(LATTEPREC) :: S, OCCLOGOCC_ELECTRONS, OCCLOGOCC_HOLES
00044   REAL(LATTEPREC) :: TRACE, EBAND
00045   COMPLEX(LATTEPREC) :: KEBAND, ZTRACE, ZFDIRAC, ZONE
00046 #ifdef MPI_ON
00047   INTEGER :: MYID, IERR, NUMPROCS, STATUS(mpi_status_size)
00048   COMPLEX(LATTEPREC), ALLOCATABLE :: TMPKBO(:, :)
00049   IF (existerror) RETURN
00050
00051   CALL mpi_comm_rank(mpi_comm_world, myid, ierr)
00052   CALL mpi_comm_size(mpi_comm_world, numprocs, ierr)
00053
00054   ALLOCATE(tmpkbo(hdim, hdim))
00055 #endif
00056
00057   zone = cmplx(one)
00058   kbo = cmplx(zero) ! Initialize the density matrix
00059
00060   occtarget = bndfil*REAL(hdim*nktot)
00061
00062   ! PRINT*, TOTNE, OCCTARGET
00063
00064   iter = 0
00065
00066   breakloop = 0
00067
00068   occerror = 1000000000.0
00069
00070   !
00071   ! The do-while loop uses a Newton-Raphson optimization of the chemical
00072   ! potential to obtain the correct occupation
00073   !
00074
00075   IF (kbt .GT. 0.001) THEN ! This bit is for a finite electronic temperature
00076
00077     DO WHILE (abs(occerror) .GT. breaktol .AND. iter .LT. 100)
00078
00079       iter = iter + 1
00080       occ = zero
00081       dfdirac = zero
00082
00083
00084       ! Loop over all k-points too
00085
00086       !$OMP PARALLEL DO DEFAULT(NONE) &
00087       !$OMP SHARED(NKTOT, HDIM, KEVALS, CHEMPOT, KBT) &
00088       !$OMP PRIVATE(K, I, FDIRACARG, EXPARG, FDIRAC) &
00089       !$OMP REDUCTION(+: OCC, DFDIRAC)
00090
00091       DO k = 1, nktot
00092         DO i = 1, hdim

```

```

00094         fdiracarg = (kevals(i, k) - chempot)/kbt
00095
00096         fdiracarg = max(fdiracarg, -exptol)
00097         fdiracarg = min(fdiracarg, exptol)
00098
00099         exparg = exp(fdiracarg)
00100         fdirac = one/(one + exparg)
00101         occ = occ + fdirac
00102         dfdirac = dfdirac + exparg*fdirac*fdirac
00103
00104         ENDDO
00105     ENDDO
00106
00107     !$OMP END PARALLEL DO
00108
00109     dfdirac = dfdirac/REAL(nktot)
00110
00111     dfdirac = dfdirac/kbt
00112
00113     occerror = (occtarget - occ)/REAL(nktot)
00114
00115     IF (abs(dfdirac) .LT. numlimit) dfdirac = sign(numlimit, dfdirac)
00116
00117     shiftcp = occerror/dfdirac
00118
00119     IF (abs(shiftcp) .GT. maxshift) shiftcp = sign(maxshift, shiftcp)
00120
00121     chempot = chempot + shiftcp
00122
00123     ENDDO
00124
00125     IF (iter .EQ. 100) THEN
00126         CALL errors("kbodirect", "Newton-Raphson scheme to find the Chemical potential does not
converge")
00127     ENDIF
00128
00129     ! Now we have the chemical potential we can build the density matrix
00130
00131     s = zero
00132
00133     IF (mdon .EQ. 0 .OR. &
00134         (mdon .EQ. 1 .AND. mod(entropyiter, wrtfreq) .EQ. 0)) THEN
00135
00136         !$OMP PARALLEL DO DEFAULT(NONE) &
00137         !$OMP SHARED(NKTOT, HDIM, KEVALS, CHEMPOT, KBT) &
00138         !$OMP PRIVATE(K, I, FDIRACARG, FDIRAC, OCCLOGOCC_HOLES, OCCLOGOCC_ELECTRONS) &
00139         !$OMP REDUCTION(+: S)
00140
00141         DO k = 1, nktot
00142             DO i = 1, hdim
00143
00144                 fdiracarg = (kevals(i, k) - chempot)/kbt
00145
00146                 fdiracarg = max(fdiracarg, -exptol)
00147                 fdiracarg = min(fdiracarg, exptol)
00148
00149                 fdirac = one/(one + exp(fdiracarg))
00150
00151                 occlogocc_electrons = fdirac * log(fdirac)
00152                 occlogocc_holes = (one - fdirac) * log(one - fdirac)
00153
00154                 s = s + two*(occlogocc_electrons + occlogocc_holes)
00155
00156             ENDDO
00157         ENDDO
00158
00159         !$OMP END PARALLEL DO
00160
00161         s = s/REAL(nktot)
00162
00163         ! Compute the gap only when we have to...
00164
00165         ! If we have an even number of electrons
00166
00167         !         IF (MOD(INT(TOTNE),2) .EQ. 0) THEN
00168         !             EGAP = EVALS(INT(OCCTARGET) + 1) - EVALS(INT(OCCTARGET))
00169         !         ELSE
00170         !             EGAP = ZERO
00171         !         ENDIF
00172
00173     ENDIF
00174
00175     ente = -kbt*s
00176
00177 #ifdef MPI_OFF
00178
00179     DO k = 1, nktot

```

```

00180         DO i = 1, hdim
00181             fdiracarg = (kevals(i,k) - chempot)/kbt
00182             fdiracarg = max(fdiracarg, -exptol)
00183             fdiracarg = min(fdiracarg, exptol)
00184             zfdirc = cmplx(one/(one + exp(fdiracarg)))
00185             CALL zgerc(hdim, hdim, zfdirc, kevecs(:,i,k), 1,
00186                     kevecs(:,i,k), 1, kbo(:, :, k), hdim)
00187         ENDDO
00188     ENDDO
00189 #elif defined(MPI_ON)
00190     DO k = 1, nktot
00191         IF (mod(k,numprocs) .EQ. myid) THEN
00192             DO i = 1, hdim
00193                 fdiracarg = (kevals(i,k) - chempot)/kbt
00194                 fdiracarg = max(fdiracarg, -exptol)
00195                 fdiracarg = min(fdiracarg, exptol)
00196                 zfdirc = cmplx(one/(one + exp(fdiracarg)))
00197                 CALL zgerc(hdim, hdim, zfdirc, kevecs(:,i,k), 1,
00198                         kevecs(:,i,k), 1, kbo(:, :, k), hdim)
00199             ENDDO
00200         ENDIF
00201     ENDDO
00202     ! CALL MPI_Barrier(MPI_COMM_WORLD, IERR)
00203     ! Collect KBO on rank 0 then broadcast
00204     IF (myid .NE. 0) THEN ! Try to collect on the master
00205         DO i = 1, nktot
00206             ! Check whether the rank owns the data...
00207             IF (mod(i,numprocs) .EQ. myid) THEN
00208                 ! If so, sent it to the others
00209                 CALL mpi_send(kbo(1,1,i), hdim*hdim, mpi_double_complex, &
00210                             0, i, mpi_comm_world, ierr)
00211             ENDIF
00212         ENDDO
00213     ELSE
00214         ! The master receives everything and puts them in the right place
00215         ! using the MPI_TAG
00216         DO i = 1, nktot - (nktot/numprocs)
00217             CALL mpi_recv(tmpkbo, hdim*hdim, mpi_double_complex, &
00218                         mpi_any_source, mpi_any_tag, mpi_comm_world, status, ierr)
00219             mykpoint = status(mpi_tag)
00220             ! PRINT*, MYID, MYKPOINT
00221             DO j = 1, hdim
00222                 DO k = 1, hdim
00223                     kbo(k,j,mykpoint) = tmpkbo(k,j)
00224                 ENDDO
00225             ENDDO
00226         ENDDO
00227     ENDIF
00228     CALL mpi_barrier(mpi_comm_world, ierr)
00229     CALL mpi_bcast(kbo, hdim*hdim*nktot, mpi_double_complex, 0, &
00230                 mpi_comm_world, ierr)
00231 #endif
00232
00233
00234
00235
00236
00237
00238
00239
00240
00241
00242
00243
00244
00245
00246
00247
00248
00249
00250
00251
00252
00253
00254
00255
00256
00257
00258
00259
00260
00261
00262
00263
00264

```



```

00265 ELSE ! This bit is for zero electronic temperature
00266
00267 IF (mod(int(totne),2) .NE. 0) THEN
00268 CALL errors("kbodirect","Odd number of electrons - run a &
00269 & spin-polarized calculation or use a finite electron temperature")
00270 ENDIF
00271 !
00272 !
00273 ! This definition of the chemical potential is a little arbitrary
00274 !
00275
00276 looptarget = int(occtarget)
00277
00278 ! Find the chemical potential - we need the looptarget'th lowest
00279 ! eigenvalue
00280
00281 count = 0
00282 DO i = 1, nktot
00283 DO j = 1, hdim
00284 count = count + 1
00285 cplist(count) = kevals(j,i)
00286 ENDDO
00287 ENDDO
00288
00289 DO i = 1, looptarget
00290 chempot = minval(cplist)
00291 cploc = minloc(cplist)
00292 cplist(cploc(1)) = 1.0d12 ! We do this so we don't get this eigenvalue again on the next loop
00293 ENDDO
00294
00295 egap = minval(cplist) - chempot
00296
00297 #ifdef MPI_OFF
00298
00299 count = 0
00300 DO k = 1, nktot
00301 DO i = 1, hdim
00302
00303 IF (kevals(i,k) .LE. chempot .AND. count .LT. looptarget) THEN
00304 count = count + 1
00305 CALL zgerc(hdim, hdim, zone, kevecs(:,i,k), 1, &
00306 kevecs(:,i,k), 1, kbo(:, :, k), hdim)
00307
00308 ENDIF
00309
00310 ENDDO
00311 ENDDO
00312
00313 #elif defined(MPI_ON)
00314
00315 DO k = 1, nktot
00316
00317 IF (mod(k,numprocs) .EQ. myid) THEN
00318
00319 DO i = 1, hdim
00320
00321 IF (kevals(i,k) .LE. chempot) THEN
00322
00323 CALL zgerc(hdim, hdim, zone, kevecs(:,i,k), 1, &
00324 kevecs(:,i,k), 1, kbo(:, :, k), hdim)
00325
00326 ENDIF
00327 ENDDO
00328 ENDIF
00329 ENDDO
00330
00331 ! CALL MPI_Barrier(MPI_COMM_WORLD, IERR)
00332
00333 ! Collect KBO on rank 0 then broadcast
00334
00335 IF (myid .NE. 0) THEN ! Try to collect on the master
00336
00337 DO i = 1, nktot
00338
00339 ! Check whether the rank owns the data...
00340
00341 IF (mod(i,numprocs) .EQ. myid) THEN
00342
00343 ! If so, sent it to the others
00344
00345 CALL mpi_send(kbo(1,1,i), hdim*hdim, mpi_double_complex, &
00346 0, i, mpi_comm_world, ierr)
00347
00348 ENDIF
00349
00350 ENDDO
00351

```

```

00352      ELSE
00353
00354          ! The master receives everything and puts them in the right place
00355          ! using the MPI_TAG
00356
00357          DO i = 1, nktot - (nktot/numprocs)
00358
00359              CALL mpi_recv(tmpkbo, hdim*hdim, mpi_double_complex, &
00360                  mpi_any_source, mpi_any_tag, mpi_comm_world, status, ierr)
00361
00362              mykpoint = status(mpi_tag)
00363              ! PRINT*, MYID, MYKPOINT
00364              DO j = 1, hdim
00365                  DO k = 1, hdim
00366                      kbo(k,j,mykpoint) = tmpkbo(k,j)
00367                  ENDDO
00368              ENDDO
00369
00370          ENDDO
00371
00372      ENDIF
00373
00374      CALL mpi_barrier(mpi_comm_world, ierr)
00375
00376      CALL mpi_bcast(kbo, hdim*hdim*nktot, mpi_double_complex, 0, &
00377          mpi_comm_world, ierr)
00378
00379 #endif
00380
00381
00382      ENDIF
00383
00384      ! PRINT*, COUNT, OCCTARGET
00385
00386      kbo = kbo*cmlpx(two)
00387
00388      IF (debugon .EQ. 1) THEN
00389
00390          OPEN(unit=31, status="UNKNOWN", file="mykBO.dat")
00391
00392          DO k = 1, nktot
00393              WRITE(31,*) k
00394              DO i = 1, hdim
00395                  WRITE(31,12) (kbo(i,j,k), j = 1, hdim)
00396              ENDDO
00397          ENDDO
00398
00399          CLOSE(31)
00400
00401      ENDIF
00402
00403 12 FORMAT(100f8.3)
00404
00405 #ifdef MPI_ON
00406      DEALLOCATE(tmpkbo)
00407 #endif
00408
00409      RETURN
00410
00411 END SUBROUTINE kbovecs

```

8.213 kdeorthomyrho.f90 File Reference

Functions/Subroutines

- subroutine [kdeorthomyrho](#)

8.213.1 Function/Subroutine Documentation

8.213.1.1 subroutine [kdeorthomyrho](#) ()

Definition at line 23 of file [kdeorthomyrho.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kdeorthomyrho
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE nonoarray
00027   USE spinarray
00028   USE kspacearray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I
00034   COMPLEX (LATTEPREC), ALLOCATABLE :: KTMP(:, :)
00035   COMPLEX (LATTEPREC), PARAMETER :: ALPHA = cmplx(one, zero), beta=cmplx(
zero, zero)
00036   IF (existerror) RETURN
00037
00038   !
00039   ! RHO = X ORTHORHO X^dag
00040   !
00041
00042   ALLOCATE (ktmp (hdim, hdim))
00043
00044   DO i = 1, nktot
00045
00046     CALL zgemm('N','N', hdim, hdim, hdim, alpha, kxmat(:, :, i),
hdim, &
00047             kbo(:, :, i), hdim, beta, ktmp, hdim)
00048     CALL zgemm('N','C', hdim, hdim, hdim, alpha, ktmp, hdim, &
00049             kxmat(:, :, i), hdim, beta, kbo(:, :, i), hdim)
00050
00051   ENDDO
00052
00053   DEALLOCATE (ktmp)
00054
00055   RETURN
00056
00057 END SUBROUTINE kdeorthomyrho
00058

```

8.215 kdiagmyh.f90 File Reference

Functions/Subroutines

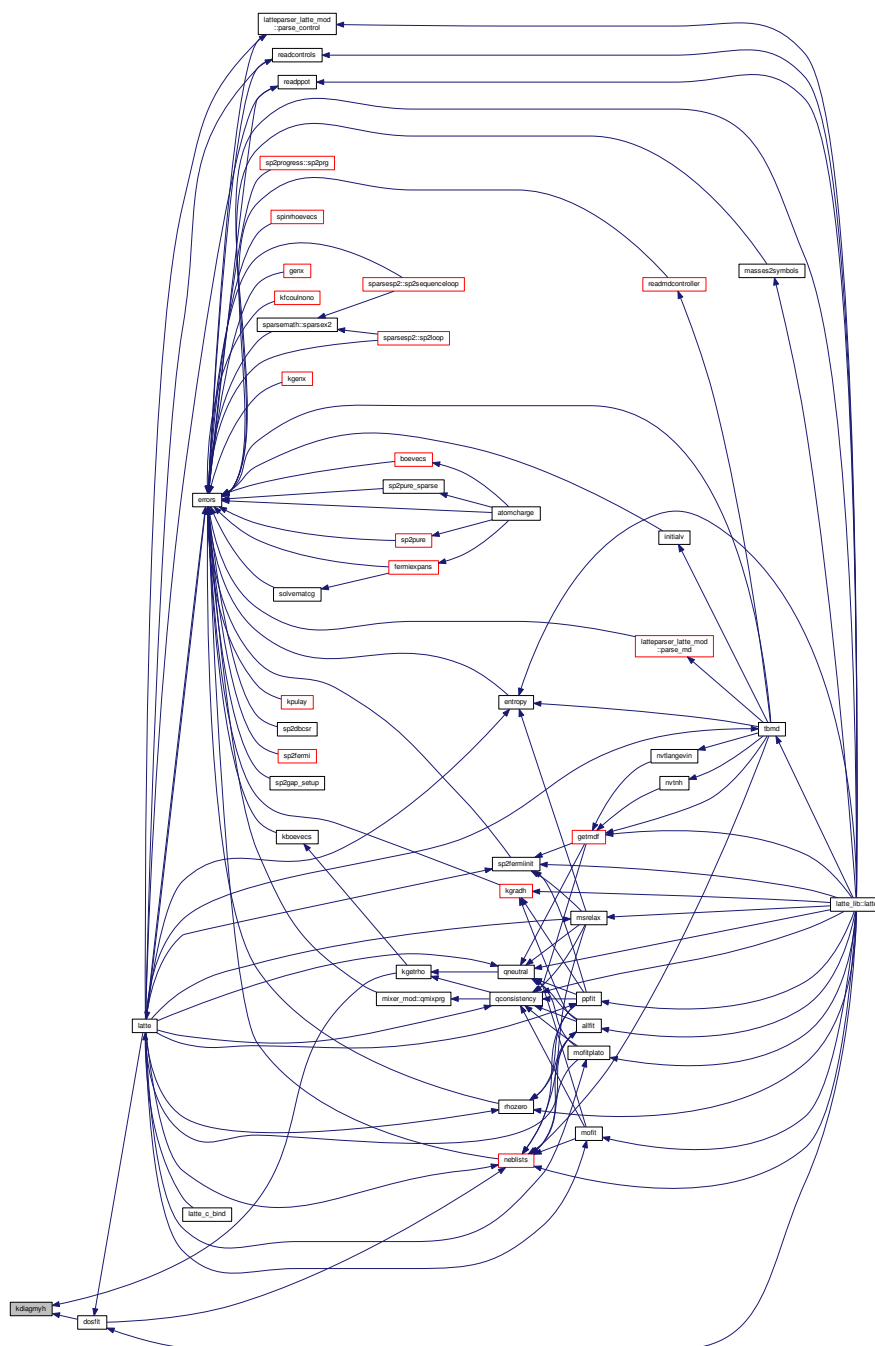
- subroutine [kdiagmyh](#)

8.215.1 Function/Subroutine Documentation

8.215.1.1 subroutine [kdiagmyh](#) ()

Definition at line 23 of file [kdiagmyh.f90](#).

Here is the caller graph for this function:



8.216 kdiagmyh.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National     !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,       !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE. If software is modified to produce derivative works, such
```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kdiagmyh
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE diagarray
00027     USE spinarray
00028     USE nonoarray
00029     USE kspacearray
00030     USE myprecision
00031 #ifdef MPI_ON
00032     USE mpi
00033 #endif
00034
00035     IMPLICIT NONE
00036
00037     INTEGER :: INFO
00038     INTEGER :: I, J, K
00039 #ifdef MPI_ON
00040     INTEGER :: MYID, IERR, NUMPROCS, MYKPOINT, STATUS(mpi_status_size)
00041     REAL(LATTEPREC), ALLOCATABLE :: TMPVALS(:)
00042     COMPLEX(LATTEPREC), ALLOCATABLE :: TMPVECS(:, :)
00043     IF (existerror) RETURN
00044
00045     ALLOCATE(tmpvecs(hdim, hdim), tmpvals(hdim))
00046
00047     CALL mpi_comm_rank(mpi_comm_world, myid, ierr)
00048     CALL mpi_comm_size(mpi_comm_world, numprocs, ierr)
00049
00050 #endif
00051
00052
00053     ! Now we're doing k-space. We have to get the eigenvalues and
00054     ! eigenvectors of all NKTOT of our H matrices
00055
00056     IF (basistype .EQ. "ORTHO") THEN
00057         kevecs = hk
00058     ELSEIF (basistype .EQ. "NONORTHO") THEN
00059         kevecs = korthoh
00060     ENDIF
00061
00062 #ifdef MPI_OFF
00063
00064     DO i = 1, nktot
00065
00066         CALL zheev('V', 'U', hdim, kevecs(:, :, i), hdim, kevals(:, i), &
00067             diag_zwork, diag_lzwork, diag_rwork, info)
00068
00069     ENDDO
00070
00071 #elif defined(MPI_ON)
00072
00073     DO i = 1, nktot
00074
00075         IF (mod(i, numprocs) .EQ. myid) THEN
00076
00077             CALL zheev('V', 'U', hdim, kevecs(:, :, i), hdim, kevals(:, i), &
00078                 diag_zwork, diag_lzwork, diag_rwork, info)
00079
00080             ENDIF
00081
00082     ENDDO
00083
00084     ! CALL MPI_Barrier(MPI_COMM_WORLD, IERR)
00085
00086     ! Loop through all the k-points. Every processor is going to
00087     ! send all the sub-matrices it computed to all the others (coz
00088     ! they don't have them yet)
00089     ! We will label the sends with the k-point ID in MPI_TAG
00090
00091     IF (myid .NE. 0) THEN ! Try to collect on the master
00092
00093         DO i = 1, nktot
00094
00095             ! Check whether the rank owns the data...

```

```

00097
00098      IF (mod(i,numprocs) .EQ. myid) THEN
00099
00100          ! If so, sent it to the others
00101
00102
00103          CALL mpi_send(kevals(1,i), hdim, mpi_double_precision, &
00104                        0, i, mpi_comm_world, ierr)
00105
00106      ENDIF
00107
00108  ENDDO
00109
00110  ELSE
00111
00112      DO i = 1, nktot - (nktot/numprocs)
00113
00114          CALL mpi_recv(tmpvals, hdim, mpi_double_precision, &
00115                        mpi_any_source, mpi_any_tag, mpi_comm_world, status, ierr)
00116
00117          mykpoint = status(mpi_tag)
00118          !      PRINT*, MYID, MYKPOINT
00119          DO k = 1, hdim
00120              kevals(k,mykpoint) = tmpvals(k)
00121          ENDDO
00122      ENDDO
00123
00124  ENDIF
00125
00126  CALL mpi_barrier(mpi_comm_world, ierr)
00127
00128  CALL mpi_bcast(kevals, hdim*nktot, mpi_double_precision, 0, &
00129                mpi_comm_world, ierr)
00130
00131  IF (myid .NE. 0) THEN ! Try to collect on the master
00132
00133      DO i = 1, nktot
00134
00135          ! Check whether the rank owns the data...
00136
00137          IF (mod(i,numprocs) .EQ. myid) THEN
00138
00139              ! If so, sent it to the myid = 0
00140
00141
00142              CALL mpi_send(kevecs(1,1,i), hdim*hdim, mpi_double_complex, &
00143                            0, i, mpi_comm_world, ierr)
00144
00145          ENDIF
00146      ENDDO
00147
00148  ELSE
00149
00150      ! myid = 0 receives all the data
00151
00152      DO i = 1, nktot - (nktot/numprocs)
00153
00154          CALL mpi_recv(tmpvecs, hdim*hdim, mpi_double_complex, &
00155                        mpi_any_source, mpi_any_tag, mpi_comm_world, status, ierr)
00156
00157          mykpoint = status(mpi_tag)
00158          !      PRINT*, MYID, MYKPOINT
00159          DO j = 1, hdim
00160              DO k = 1, hdim
00161                  kevecs(k,j,mykpoint) = tmpvecs(k,j)
00162              ENDDO
00163          ENDDO
00164      ENDDO
00165
00166  ENDDO
00167
00168  ENDIF
00169
00170  CALL mpi_barrier(mpi_comm_world, ierr)
00171
00172  CALL mpi_bcast(kevecs, hdim*hdim*nktot, mpi_double_complex, 0, &
00173                mpi_comm_world, ierr)
00174
00175  DEALLOCATE(tmpvecs, tmpvals)
00176
00177  #endif
00178
00179  RETURN
00180
00181  END SUBROUTINE kdiagmyh

```

8.217 kernelparser_mod.f90 File Reference

Modules

- module [kernelparser_mod](#)
Some general parsing functions.

Functions/Subroutines

- subroutine, public [kernelparser_mod::parsing_kernel](#) (KEYVECTOR_CHAR, VALVECTOR_CHAR, KEYVECTOR_INT, VALVECTOR_INT, KEYVECTOR_RE, VALVECTOR_RE, KEYVECTOR_LOG, VALVECTOR_LOG, FILENAME, STARTSTOP)
The general parsing function. It is used to vectorize a set of "keywords" "value" pairs as included in a general input file.

8.218 kernelparser_mod.f90

```

00001
00007 MODULE kernelparser_mod
00008
00009     USE openfiles_mod
00010     USE constants_mod, ONLY: verbose
00011     ! USE PARALLEL_MOD
00012
00013     IMPLICIT NONE
00014
00015     PRIVATE
00016
00017     INTEGER, PARAMETER :: dp = kind(1.0d0)
00018
00019     PUBLIC :: parsing_kernel
00020
00021 CONTAINS
00022
00031     SUBROUTINE parsing_kernel (KEYVECTOR_CHAR, VALVECTOR_CHAR &
00032     , keyvector_int, valvector_int, keyvector_re, valvector_re, &
00033     keyvector_log, valvector_log, filename, startstop)
00034     IMPLICIT NONE
00035     CHARACTER(1), ALLOCATABLE :: tempc(:)
00036     CHARACTER(100), ALLOCATABLE :: vect(:, :)
00037     CHARACTER(50) :: keyvector_char(:), keyvector_int(:), keyvector_log(:),
00038     keyvector_re(:)
00039     CHARACTER(100) :: valvector_char(:)
00040     CHARACTER(LEN=*) :: FILENAME
00041     CHARACTER(LEN=*) , INTENT(IN), OPTIONAL :: STARTSTOP(2)
00042     CHARACTER(LEN=100) :: TEMPCFLEX
00043     INTEGER :: i, io_control, ios, j
00044     INTEGER :: k, l, lenc, nkey_char
00045     INTEGER :: nkey_int, nkey_log, nkey_re, readmaxi
00046     INTEGER :: readmaxj, readmini, valvector_int(:)
00047     INTEGER :: startatj, totalwords
00048     LOGICAL :: start, stopl, valid, valvector_log(:), stopparsing,
00049     defaultnone
00050     LOGICAL, ALLOCATABLE :: checkmissing_char(:), checkmissing_int(:), checkmissing_log(
00051     :), checkmissing_re(:)
00052     REAL(dp) :: valvector_re(:)
00053
00054     readmaxi = 5 ; readmaxj = 1000
00055     ALLOCATE (vect (readmaxi, readmaxj))
00056     nkey_char = SIZE (keyvector_char, dim=1)
00057     nkey_re = SIZE (keyvector_re, dim=1)
00058     nkey_int = SIZE (keyvector_int, dim=1)
00059     nkey_log = SIZE (keyvector_log, dim=1)
00060
00061     CALL open_file_to_read (io_control, filename)
00062
00063     ALLOCATE (checkmissing_char (nkey_char), checkmissing_re (nkey_re), &
00064     checkmissing_int (nkey_int), checkmissing_log (nkey_log))
00065
00066     !Initialize the checkmissing flags and the vect arrays
00067     checkmissing_char = .false.

```



```

00065     checkmissing_re = .false.
00066     checkmissing_int = .false.
00067     checkmissing_log = .false.
00068     stopparsing = .false.
00069     defaultnone = .false.
00070     vect = '          '
00071
00072     DO i=1,readmaxi !Here we read all the input into vect
00073         READ(io_control,*,iostat=ios) (vect(i,j),j=1,readmaxj)
00074     END DO
00075
00076     CLOSE(io_control)
00077
00078     !Look up for floating hashes (#)
00079     totalwords = 0
00080     DO i=1,readmaxi
00081         DO k=1,readmaxj
00082             IF (adjustl(trim(vect(i,k))).NE."")totalwords = totalwords + 1
00083             IF (adjustl(trim(vect(i,k))).EQ."#") THEN
00084                 WRITE(*,*) " "
00085                 WRITE(*,*) "ERROR in the the input file ..."
00086                 WRITE(*,*) " "
00087                 WRITE(*,*) "In the LATTE parsing routine everything is a comment by default unless theres an =
sign"
00088                 WRITE(*,*) "next to a word, in which case, it will be recognized as a keyword."
00089                 WRITE(*,*) "This parser does not accept floating hashes (#). This is done in order to make
sure"
00090                 WRITE(*,*) "that a specific keyword is commented"
00091                 WRITE(*,*) " "
00092                 WRITE(*,*) "If you have a commented keyword make sure there is a # symbol right next to it"
00093                 WRITE(*,*) " "
00094                 WRITE(*,*) "    The following commented keyword is correct: "
00095                 WRITE(*,*) "                #Keyword= 1 "
00096                 WRITE(*,*) " "
00097                 WRITE(*,*) "    The following commented keyword is NOT correct: "
00098                 WRITE(*,*) "                # Keyword= 1 "
00099                 WRITE(*,*) " "
00100                 stop
00101             ENDIF
00102             IF (adjustl(trim(vect(i,k))).EQ."STOP{ }") stopparsing = .true.
00103             IF (adjustl(trim(vect(i,k))).EQ."DEFAULTNONE") defaultnone = .true.
00104         ENDDO
00105     ENDDO
00106
00107     IF (totalwords > readmaxi*readmaxj - 100) THEN
00108         WRITE(*,*) " "; WRITE(*,*) "Stopping ... Maximum allowed (keys + values + comments) words close to the
limit "
00109         WRITE(*,*) "Increase the readmaxj variable in the parsing_kernel subroutine or reduce the comments in
the input"
00110         stop
00111     ENDIF
00112
00113     !Look up for boundaries
00114     readmini=1
00115     start=.false.
00116     IF (PRESENT(startstop)) THEN
00117         DO i=1,readmaxi
00118             DO k=1,readmaxj
00119                 IF (trim(vect(i,k)).EQ.trim(startstop(1))) THEN
00120                     readmini=i
00121                     startatj=k
00122                     start=.true.
00123                 ENDIF
00124                 IF (start.AND.trim(vect(i,k)).EQ.trim(startstop(2))) THEN
00125                     readmaxi=i
00126                 ENDIF
00127             ENDDO
00128         ENDDO
00129     ENDIF
00130
00131     ! Look for invalid characters if startstop is present
00132     IF (start) THEN
00133         start=.false.
00134         stopl=.false.
00135         DO i=readmini,readmaxi
00136             DO k=1,readmaxj
00137                 IF (trim(vect(i,k)).EQ.trim(startstop(1))) start=.true.
00138                 valid = .false.
00139                 IF (start) THEN
00140                     IF (vect(i,k).NE.'          ' ) THEN
00141                         DO j=1,nkey_char
00142                             IF (trim(vect(i,k)).EQ.trim(keyvector_char(j))) THEN
00143                                 valid = .true.
00144                             ENDIF
00145                         ENDDO
00146                         DO j=1,nkey_int
00147                             IF (trim(vect(i,k)).EQ.trim(keyvector_int(j))) THEN

```

```

00148             valid = .true.
00149             ENDDIF
00150         ENDDO
00151         DO j=1,nkey_re
00152             IF(trim(vect(i,k)).EQ.trim(keyvector_re(j))) THEN
00153                 valid = .true.
00154             ENDDIF
00155         ENDDO
00156         DO j=1,nkey_log
00157             IF(trim(vect(i,k)).EQ.trim(keyvector_log(j))) THEN
00158                 valid = .true.
00159             ENDDIF
00160         ENDDO
00161         IF(trim(vect(i,k)).EQ.trim(startstop(2))) THEN
00162             stopl=.true.
00163         ENDDIF
00164         IF(.NOT.valid.AND..NOT.stopl)CALL check_valid(vect(i,k))
00165         ENDDIF
00166     ENDDIF
00167 ENDDO
00168 ENDDO
00169 ENDDIF
00170
00171 stopl = .false.
00172 DO i=readmini,readmaxi !We search for the character keys
00173     IF(stopl)EXIT
00174     DO k=1,readmaxj
00175         IF(stopl)EXIT
00176         IF(vect(i,k).NE.' ' ) THEN
00177             IF(start) THEN !If we have a start key:
00178                 IF(readmaxj*(i-1)+k .GE.readmaxj*(readmini-1)+startatj) THEN !If the position is beyond the
start key:
00179                     IF(trim(vect(i,k)).NE.' ' ) THEN !If we don't have a stop key:
00180                         DO j=1,nkey_char
00181                             IF(adjustl(trim(vect(i,k))).EQ.adjustl(trim(keyvector_char(j)))) THEN
00182                                 valvector_char(j)=adjustl(trim(vect(i,k+1)))
00183                                 checkmissing_char(j) = .true.
00184                             ENDDIF
00185                         END DO
00186                     ELSE
00187                         stopl = .true.
00188                     ENDDIF
00189                 ENDDIF
00190             ELSE !If we don't have a start key:
00191                 DO j=1,nkey_char
00192                     IF(trim(vect(i,k)).EQ.trim(keyvector_char(j))) THEN
00193                         valvector_char(j)=trim(vect(i,k+1))
00194                         checkmissing_char(j) = .true.
00195                     ENDDIF
00196                 END DO
00197             ENDDIF
00198         ELSE
00199             EXIT
00200         ENDDIF
00201     ENDDO
00202 ENDDO
00203
00204 stopl = .false.
00205 DO i=readmini,readmaxi !We search for the integer keys
00206     IF(stopl)EXIT
00207     DO k=1,readmaxj
00208         IF(stopl)EXIT
00209         IF(vect(i,k).NE.' ' ) THEN
00210             IF(start) THEN
00211                 IF(readmaxj*(i-1)+k .GE.readmaxj*(readmini-1)+startatj) THEN
00212                     IF(adjustl(trim(vect(i,k))).NE.' ' ) THEN
00213                         DO j=1,nkey_int
00214                             IF(trim(vect(i,k)).EQ.trim(keyvector_int(j))) THEN
00215                                 READ(vect(i,k+1),*)valvector_int(j)
00216                                 checkmissing_int(j) = .true.
00217                             ENDDIF
00218                         ENDDO
00219                     ELSE
00220                         stopl = .true.
00221                     ENDDIF
00222                 ENDDIF
00223             ELSE
00224                 DO j=1,nkey_int
00225                     IF(trim(vect(i,k)).EQ.trim(keyvector_int(j))) THEN
00226                         READ(vect(i,k+1),*)valvector_int(j)
00227                         checkmissing_int(j) = .true.
00228                     ENDDIF
00229                 ENDDO
00230             ENDDIF
00231         ELSE
00232             EXIT
00233         ENDDIF

```

```

00234      ENDDO
00235      ENDDO
00236
00237      stop1 = .false.
00238      DO i=readmini,readmaxi !We search for the real keys
00239          IF(stop1)EXIT
00240          DO k=1,readmaxj
00241              IF(stop1)EXIT
00242              IF(vect(i,k).NE.' ' ) THEN
00243                  IF(start)THEN
00244                      IF(readmaxj*(i-1)+k .GE.readmaxj*(readmini-1)+startatj) THEN
00245                          IF(trim(vect(i,k)).NE.' ') THEN
00246                              DO j=1,nkey_re
00247                                  IF(trim(vect(i,k)).EQ.trim(keyvector_re(j))) THEN
00248                                      READ(vect(i,k+1),*)valvector_re(j)
00249                                      checkmissing_re(j) = .true.
00250                                  ENDIF
00251                              ENDDO
00252                          ELSE
00253                              stop1 = .true.
00254                          ENDIF
00255                      ENDIF
00256                  ELSE
00257                      DO j=1,nkey_re
00258                          IF(trim(vect(i,k)).EQ.trim(keyvector_re(j))) THEN
00259                              READ(vect(i,k+1),*)valvector_re(j)
00260                              checkmissing_re(j) = .true.
00261                          ENDIF
00262                      ENDDO
00263                  ENDIF
00264              ELSE
00265                  EXIT
00266              ENDIF
00267          ENDDO
00268      ENDDO
00269
00270      stop1 = .false.
00271      DO i=1,readmaxi !We search for the logical keys
00272          IF(stop1)EXIT
00273          DO k=1,readmaxj
00274              IF(stop1)EXIT
00275              IF(vect(i,k).NE.' ' ) THEN
00276                  IF(start)THEN
00277                      IF(readmaxj*(i-1)+k .GE.readmaxj*(readmini-1)+startatj) THEN
00278                          IF(trim(vect(i,k)).NE.' ') THEN
00279                              DO j=1,nkey_log
00280                                  IF(trim(vect(i,k)).EQ.trim(keyvector_log(j))) THEN
00281                                      READ(vect(i,k+1),*)valvector_log(j)
00282                                      checkmissing_log(j) = .true.
00283                                  END IF
00284                              END DO
00285                          ELSE
00286                              stop1 = .true.
00287                          ENDIF
00288                      ENDIF
00289                  ELSE
00290                      DO j=1,nkey_log
00291                          IF(trim(vect(i,k)).EQ.trim(keyvector_log(j))) THEN
00292                              READ(vect(i,k+1),*)valvector_log(j)
00293                              checkmissing_log(j) = .true.
00294                          ENDIF
00295                      ENDDO
00296                  ENDIF
00297              ELSE
00298                  EXIT
00299              ENDIF
00300          ENDDO
00301      ENDDO
00302
00303      !Check for missing keywords
00304      WRITE(*,*)' '
00305      DO i = 1,nkey_char
00306          IF(defaultnone .EQV..true.)THEN
00307              IF(checkmissing_char(i).NEQV..true..AND.trim(keyvector_char(i)).NE."DUMMY=") THEN
00308                  WRITE(*,*)'ERROR: variable ',trim(keyvector_char(i)),&
00309                      ' is missing. Set this variable or remove the DEFAULTNONE keyword from the input file...'
00310                  WRITE(*,*)'Default value is:',valvector_char(i)
00311                  stop
00312              ENDIF
00313          ENDIF
00314          IF((checkmissing_char(i).NEQV..true.) WRITE(*,*)'# WARNING: variable ',trim(keyvector_char(i)),&
00315              ' is missing. I will use a default value instead ...'
00316      ENDDO
00317      DO i = 1,nkey_int
00318          IF(defaultnone .EQV..true.)THEN
00319              IF(checkmissing_int(i).NEQV..true..AND.trim(keyvector_int(i)).NE."DUMMY=") THEN
00320                  WRITE(*,*)'ERROR: variable ',trim(keyvector_int(i)),&

```

```

00321         ' is missing. Set this variable or remove the DEFAULTNONE keyword from the input file...'
00322         WRITE(*,*)'Default value is:',valvector_int(i)
00323         stop
00324     ENDIF
00325 ENDIF
00326 IF((checkmissing_int(i).NEQV..true.)) WRITE(*,*)'# WARNING: variable ',trim(keyvector_int(i)),&
00327     ' is missing. I will use a default value instead ...'
00328 ENDDO
00329 DO i = 1,nkey_re
00330     IF(defaultnone .EQV..true.)THEN
00331         IF(checkmissing_re(i).NEQV..true..AND.trim(keyvector_re(i)).NE."DUMMY=")THEN
00332             WRITE(*,*)'ERROR: variable ',trim(keyvector_re(i)),&
00333                 ' is missing. Set this variable or remove the DEFAULTNONE keyword from the input file...'
00334             WRITE(*,*)'Default value is:',valvector_re(i)
00335             stop
00336         ENDIF
00337     ENDIF
00338     IF((checkmissing_re(i).NEQV..true.)) WRITE(*,*)'# WARNING: variable ',trim(keyvector_re(i)),&
00339         ' is missing. I will use a default value instead ...'
00340 ENDDO
00341 DO i = 1,nkey_log
00342     IF(defaultnone .EQV..true.)THEN
00343         IF(checkmissing_log(i).NEQV..true..AND.trim(keyvector_log(i)).NE."DUMMY=")THEN
00344             WRITE(*,*)'ERROR: variable ',trim(keyvector_log(i)),&
00345                 ' is missing. Set this variable or remove the DEFAULTNONE keyword from the input file...'
00346             WRITE(*,*)'Default value is:',valvector_log(i)
00347             stop
00348         ENDIF
00349     ENDIF
00350     IF((checkmissing_log(i).NEQV..true.)) WRITE(*,*)'# WARNING: variable ',trim(keyvector_log(i)),&
00351         ' is missing. I will use a default value instead ...'
00352 ENDDO
00353 WRITE(*,*)' '
00354
00355 DEALLOCATE(checkmissing_char,checkmissing_re, checkmissing_int, checkmissing_log)
00356
00357 ! Only rank 0 prints parameters if compiled with MPI
00358 WRITE(*,*)' '
00359 !     if (printRank() .eq. 1) then
00360
00361 WRITE(*,*)"##### PARAMETERS USED FOR THIS RUN #####"
00362 IF(start)WRITE(*,*)"# ",startstop(1)
00363 DO j=1,nkey_int
00364     WRITE(*,*)"# ",trim(keyvector_int(j)),valvector_int(j)
00365 ENDDO
00366
00367 DO j=1,nkey_re
00368     WRITE(*,*)"# ",trim(keyvector_re(j)),valvector_re(j)
00369 ENDDO
00370
00371 DO j=1,nkey_char
00372     WRITE(*,*)"# ",trim(keyvector_char(j)),valvector_char(j)
00373 ENDDO
00374
00375 DO j=1,nkey_log
00376     WRITE(*,*)"# ",trim(keyvector_log(j)),valvector_log(j)
00377 ENDDO
00378 IF(start)WRITE(*,*)"# ",startstop(2)
00379
00380 WRITE(*,*)' '
00381
00382 !     endif
00383
00384 IF(stopparsing)THEN
00385     WRITE(*,*)" " ; WRITE(*,*)"STOP key found. Stop parsing ... "; WRITE(*,*)" "
00386     stop
00387 ENDIF
00388
00389 DEALLOCATE(vect)
00390
00391 END SUBROUTINE parsing_kernel
00392
00393 SUBROUTINE check_valid(INVALIDC)
00394     IMPLICIT NONE
00395     CHARACTER(1), ALLOCATABLE :: TEMPC(:)
00396     CHARACTER(LEN=*), INTENT(IN) :: INVALIDC
00400     CHARACTER(LEN=100) :: TEMPCFLEX
00401     INTEGER :: L, LENC
00402
00403     lenc=len(adjustl(trim(invalidc)))
00404     IF(.NOT.ALLOCATED(tempc))ALLOCATE(tempc(lenc))
00405     DO l = 1,len(adjustl(trim(invalidc)))
00406         tempcflex = adjustl(trim(invalidc))
00407         tempc(l) = tempcflex(l:l)
00408         IF(tempc(l).EQ."="AND.tempc(l).NE."#")THEN
00409             WRITE(*,*)"Input ERROR: ",adjustl(trim(invalidc))," is not a valid keyword"
00410             stop

```

```
00411      ENDIF
00412      ENDDO
00413
00414      END SUBROUTINE check_valid
00415
00416 END MODULE kernelparser_mod
```

8.219 kfcoulnono.f90 File Reference

Functions/Subroutines

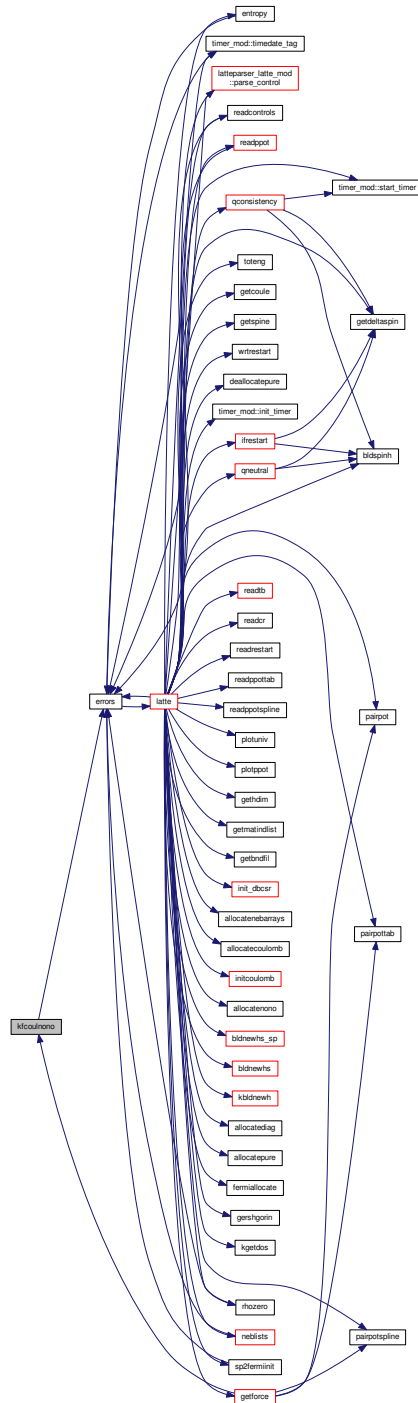
- subroutine [kfcoulnono](#)

8.219.1 Function/Subroutine Documentation

8.219.1.1 subroutine [kfcoulnono](#) ()

Definition at line 23 of file [kfcoulnono.f90](#).

Here is the call graph for this function:




```

00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kfcoulnono
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE coulombarray
00029   USE neblistarray
00030   USE spinarray
00031   USE virialarray
00032   USE kspacearray
00033   USE myprecision
00034
00035   IMPLICIT NONE
00036
00037   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00038   INTEGER :: LBRA, MBRA, LKET, MKET
00039   INTEGER :: PREVJ, NEWJ
00040   INTEGER :: PBCI, PBCJ, PBCK
00041   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00042   INTEGER :: KX, KY, KZ, KCOUNT
00043   REAL(LATTEPREC) :: ALPHA, BETA, PHI, COSBETA
00044   REAL(LATTEPREC) :: RIJ(3), DC(3)
00045   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2
00046   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00047   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTB
00048   REAL(LATTEPREC) :: KPOINT(3), KX0, KY0, KZ0, KDOTL
00049   REAL(LATTEPREC) :: B1(3), B2(3), B3(3), MAG1, MAG2, MAG3, A1A2XA3, K0(3)
00050   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00051   COMPLEX(LATTEPREC) :: FTMP(3), RHO, CONJGBLOCH
00052   LOGICAL PATH
00053   IF (existerror) RETURN
00054
00055   ! These were allocated elsewhere. We'll use them to accumulate the complex forces
00056
00057   IF (spinon .EQ. 1) THEN
00058     CALL errors("kfcoulnono", "Non-ortho k-space and spin polarization not yet implemented")
00059   ENDIF
00060
00061   kf = cmplx(zero)
00062   virbondk = cmplx(zero)
00063
00064   ! Computing the reciprocal lattice vectors
00065
00066   b1(1) = box(2,2)*box(3,3) - box(3,2)*box(2,3)
00067   b1(2) = box(3,1)*box(2,3) - box(2,1)*box(3,3)
00068   b1(3) = box(2,1)*box(3,2) - box(3,1)*box(2,2)
00069
00070   a1a2xa3 = box(1,1)*b1(1) + box(1,2)*b1(2) + box(1,3)*b1(3)
00071
00072   ! B1 = 2*PI*(A2 X A3)/(A1.(A2 X A3))
00073
00074   b1 = b1/a1a2xa3
00075
00076   ! B2 = 2*PI*(A3 X A1)/(A1(A2 X A3))
00077
00078   b2(1) = (box(3,2)*box(1,3) - box(1,2)*box(3,3))/a1a2xa3
00079   b2(2) = (box(1,1)*box(3,3) - box(3,1)*box(1,3))/a1a2xa3
00080   b2(3) = (box(3,1)*box(1,2) - box(1,1)*box(3,2))/a1a2xa3
00081
00082   ! B3 = 2*PI*(A1 X A2)/(A1(A2 X A3))
00083
00084   b3(1) = (box(1,2)*box(2,3) - box(2,2)*box(1,3))/a1a2xa3
00085   b3(2) = (box(2,1)*box(1,3) - box(1,1)*box(2,3))/a1a2xa3
00086   b3(3) = (box(1,1)*box(2,2) - box(2,1)*box(1,2))/a1a2xa3
00087
00088   k0 = pi*(one - REAL(nkx))/(REAL(nkx))*b1 + &
00089       pi*(one - REAL(nky))/(REAL(nky))*b2 + &
00090       pi*(one - REAL(nkz))/(REAL(nkz))*b3 - PI*KSHIFT
00091
00092
00093   !$OMP PARALLEL DO DEFAULT (NONE) &
00094   !$OMP SHARED(NATS, BASIS, ELEMPINTER, TOTNEBTB, NEBTB) &
00095   !$OMP SHARED(CR, BOX, KBO, RHOU, RHODOWN, SPINON, NOINT, ATELE, ELE1, ELE2) &
00096   !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00097   !$OMP SHARED(HUBBARDU, DELTAQ, COULOMBV) &
00098   !$OMP SHARED(K0, B1, B2, B3, NKX, NKY, NKZ, KF) &
00099   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00100   !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00101   !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &

```



```

00102      !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB, CONJGBLOCH, KDOTL) &
00103      !$OMP PRIVATE(KPOINT, KCOUNT) &
00104      !$OMP REDUCTION(+: VIRBONDK)
00105
00106
00107      DO i = 1, nats
00108
00109          ! Build list of orbitals on atom I
00110          SELECT CASE(basis(elempointer(i)))
00111
00112              CASE("s")
00113                  basisi(1) = 0
00114                  basisi(2) = -1
00115              CASE("p")
00116                  basisi(1) = 1
00117                  basisi(2) = -1
00118              CASE("d")
00119                  basisi(1) = 2
00120                  basisi(2) = -1
00121              CASE("f")
00122                  basisi(1) = 3
00123                  basisi(2) = -1
00124              CASE("sp")
00125                  basisi(1) = 0
00126                  basisi(2) = 1
00127                  basisi(3) = -1
00128              CASE("sd")
00129                  basisi(1) = 0
00130                  basisi(2) = 2
00131                  basisi(3) = -1
00132              CASE("sf")
00133                  basisi(1) = 0
00134                  basisi(2) = 3
00135                  basisi(3) = -1
00136              CASE("pd")
00137                  basisi(1) = 1
00138                  basisi(2) = 2
00139                  basisi(3) = -1
00140              CASE("pf")
00141                  basisi(1) = 1
00142                  basisi(2) = 3
00143                  basisi(3) = -1
00144              CASE("df")
00145                  basisi(1) = 2
00146                  basisi(2) = 3
00147                  basisi(3) = -1
00148              CASE("spd")
00149                  basisi(1) = 0
00150                  basisi(2) = 1
00151                  basisi(3) = 2
00152                  basisi(4) = -1
00153              CASE("spf")
00154                  basisi(1) = 0
00155                  basisi(2) = 1
00156                  basisi(3) = 3
00157                  basisi(4) = -1
00158              CASE("sdf")
00159                  basisi(1) = 0
00160                  basisi(2) = 2
00161                  basisi(3) = 3
00162                  basisi(4) = -1
00163              CASE("pdf")
00164                  basisi(1) = 1
00165                  basisi(2) = 2
00166                  basisi(3) = 3
00167                  basisi(4) = -1
00168              CASE("spdf")
00169                  basisi(1) = 0
00170                  basisi(2) = 1
00171                  basisi(3) = 2
00172                  basisi(4) = 3
00173                  basisi(5) = -1
00174          END SELECT
00175
00176          indi = matindlist(i)
00177
00178          DO newj = 1, totnebtb(i)
00179
00180              j = nebtb(1, newj, i)
00181              pbci = nebtb(2, newj, i)
00182              pbcj = nebtb(3, newj, i)
00183              pbck = nebtb(4, newj, i)
00184
00185              rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00186                  REAL(pbck)*BOX(3,1) - CR(1,i)
00187
00188              rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &

```

```

00189         REAL(pbcck)*BOX(3,2) - CR(2,i)
00190
00191     rij(3) = cr(3,j) + REAL(pbcic)*BOX(1,3) + REAL(pbcjc)*BOX(2,3) + &
00192         REAL(pbcck)*BOX(3,3) - CR(3,i)
00193
00194     magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00195
00196     rcuttb = zero
00197     DO k = 1, noint
00198
00199         IF ( (atele(i) .EQ. ele1(k) .AND. &
00200             atele(j) .EQ. ele2(k)) .OR. &
00201             (atele(j) .EQ. ele1(k) .AND. &
00202             atele(i) .EQ. ele2(k)) ) THEN
00203
00204             IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00205
00206             IF (basistype .EQ. "NONORTHO") THEN
00207                 IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00208             ENDIF
00209
00210         ENDIF
00211
00212     ENDDO
00213
00214     IF (magr2 .LT. rcuttb*rcuttb) THEN
00215
00216         magr = sqrt(magr2)
00217
00218         ! Build list of orbitals on atom J
00219
00220         SELECT CASE(basis(elempointer(j)))
00221         CASE("s")
00222             basisj(1) = 0
00223             basisj(2) = -1
00224         CASE("p")
00225             basisj(1) = 1
00226             basisj(2) = -1
00227         CASE("d")
00228             basisj(1) = 2
00229             basisj(2) = -1
00230         CASE("f")
00231             basisj(1) = 3
00232             basisj(2) = -1
00233         CASE("sp")
00234             basisj(1) = 0
00235             basisj(2) = 1
00236             basisj(3) = -1
00237         CASE("sd")
00238             basisj(1) = 0
00239             basisj(2) = 2
00240             basisj(3) = -1
00241         CASE("sf")
00242             basisj(1) = 0
00243             basisj(2) = 3
00244             basisj(3) = -1
00245         CASE("pd")
00246             basisj(1) = 1
00247             basisj(2) = 2
00248             basisj(3) = -1
00249         CASE("pf")
00250             basisj(1) = 1
00251             basisj(2) = 3
00252             basisj(3) = -1
00253         CASE("df")
00254             basisj(1) = 2
00255             basisj(2) = 3
00256             basisj(3) = -1
00257         CASE("spd")
00258             basisj(1) = 0
00259             basisj(2) = 1
00260             basisj(3) = 2
00261             basisj(4) = -1
00262         CASE("spf")
00263             basisj(1) = 0
00264             basisj(2) = 1
00265             basisj(3) = 3
00266             basisj(4) = -1
00267         CASE("sdf")
00268             basisj(1) = 0
00269             basisj(2) = 2
00270             basisj(3) = 3
00271             basisj(4) = -1
00272         CASE("pdf")
00273             basisj(1) = 1
00274             basisj(2) = 2
00275             basisj(3) = 3

```

```

00276         basisj(4) = -1
00277     CASE("spdf")
00278         basisj(1) = 0
00279         basisj(2) = 1
00280         basisj(3) = 2
00281         basisj(4) = 3
00282         basisj(5) = -1
00283     END SELECT
00284
00285     indj = matindlist(j)
00286
00287     magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00288     magrp = sqrt(magrp2)
00289
00290     ! transform to system in which z-axis is aligned with RIJ
00291
00292     path = .false.
00293     IF (abs(rij(1)) .GT. 1e-12) THEN
00294         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00295             phi = zero
00296         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00297             phi = two * pi
00298         ELSE
00299             phi = pi
00300         ENDIF
00301         alpha = atan(rij(2) / rij(1)) + phi
00302     ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00303         IF (rij(2) .GT. 1e-12) THEN
00304             alpha = pi / two
00305         ELSE
00306             alpha = three * pi / two
00307         ENDIF
00308     ELSE
00309         ! pathological case: pole in alpha at beta=0
00310         path = .true.
00311     ENDIF
00312
00313     cosbeta = rij(3)/magr
00314     beta = acos(rij(3) / magr)
00315
00316     dc = rij/magr
00317
00318     ! build forces using PRB 72 165107 eq. (12) - the sign of the
00319     ! dfda contribution seems to be wrong, but gives the right
00320     ! answer(?)
00321
00322     ftmp = zero
00323     k = indi
00324
00325     lbrainc = 1
00326     DO WHILE (basisi(lbrainc) .NE. -1)
00327
00328         lbra = basisi(lbrainc)
00329         lbrainc = lbrainc + 1
00330
00331         DO mbra = -lbra, lbra
00332
00333             k = k + 1
00334             l = indj
00335
00336             lketinc = 1
00337             DO WHILE (basisj(lketinc) .NE. -1)
00338
00339                 lket = basisj(lketinc)
00340                 lketinc = lketinc + 1
00341
00342                 DO mket = -lket, lket
00343
00344                     l = l + 1
00345
00346                     !
00347                     ! SELECT CASE (SPINON)
00348                     ! CASE(0)
00349                     !     RHO = BO(L, K)
00350                     ! CASE(1)
00351                     !     RHO = RHOU(L, K) + RHODOWN(L, K)
00352                     ! END SELECT
00353
00354                     IF (.NOT. path) THEN
00355                         ! Unroll loops and pre-compute
00356
00357                         mydfda = dfda(i, j, lbra, lket, mbra, &
00358                             mket, magr, alpha, cosbeta, "S")
00359
00360                         mydfdb = dfdb(i, j, lbra, lket, mbra, &
00361                             mket, magr, alpha, cosbeta, "S")
00362

```

```

00363      mydfdr = dfdr(i, j, lbra, lket, mbra, &
00364      mket, magr, alpha, cosbeta, "S")
00365
00366      kcount = 0
00367
00368      DO kx = 1, nkx
00369
00370          DO ky = 1, nky
00371
00372              DO kz = 1, nkz
00373
00374                  kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00375                  REAL(ky-1)*B2/REAL(NKY) + &
00376                  REAL(kz-1)*B3/REAL(nkz)) + k0
00377
00378                  kcount = kcount+1
00379
00380                  kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00381                  kpoint(3)*rij(3)
00382
00383                  conjgbloch = exp(cmplx(zero,-kdot1))
00384
00385                  rho = kbo(k,l,kcount)*conjgbloch
00386
00387                  !
00388                  ! d/d_alpha
00389                  !
00390
00391                  ftmp(1) = ftmp(1) + rho * &
00392                  (-rij(2) / magrp2 * mydfda)
00393
00394                  ftmp(2) = ftmp(2) + rho * &
00395                  (rij(1)/ magrp2 * mydfda)
00396
00397                  !
00398                  ! d/d_beta
00399                  !
00400
00401                  ftmp(1) = ftmp(1) + rho * &
00402                  (((rij(3) * rij(1)) / &
00403                  magr2)) / magrp) * mydfdb)
00404
00405                  ftmp(2) = ftmp(2) + rho * &
00406                  (((rij(3) * rij(2)) / &
00407                  magr2)) / magrp) * mydfdb)
00408
00409                  ftmp(3) = ftmp(3) - rho * &
00410                  ((one - ((rij(3) * rij(3)) / &
00411                  magr2)) / magrp) * mydfdb)
00412
00413                  !
00414                  ! d/dR
00415                  !
00416
00417                  ftmp(1) = ftmp(1) - rho * dc(1) * &
00418                  mydfdr
00419
00420                  ftmp(2) = ftmp(2) - rho * dc(2) * &
00421                  mydfdr
00422
00423                  ftmp(3) = ftmp(3) - rho * dc(3) * &
00424                  mydfdr
00425
00426                      ENDDO
00427              ENDDO
00428          ENDDO
00429      ELSE
00430
00431          ! pathological configuration in which beta=0
00432          ! or pi => alpha undefined
00433
00434          ! fixed: MJC 12/17/13
00435
00436          mydfdb = dfdb(i, j, lbra, lket, &
00437          mbra, mket, magr, zero, cosbeta, "S") / magr
00438
00439          mydfdb = dfdb(i, j, lbra, lket, &
00440          mbra, mket, magr, pi/two, cosbeta, "S") / magr
00441
00442          mydfdr = dfdr(i, j, lbra, lket, mbra, &
00443          mket, magr, zero, cosbeta, "S")
00444
00445          kcount = 0
00446
00447      DO kx = 1, nkx
00448          DO ky = 1, nky

```

```

00450                                DO kz = 1, nkz
00451
00452                                kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00453                                    REAL(ky-1)*B2/REAL(NKY) + &
00454                                    REAL(kz-1)*B3/REAL(nkz)) + k0
00455
00456                                kcount = kcount+1
00457
00458                                kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00459                                    kpoint(3)*rij(3)
00460
00461                                conjgbloch = exp(cmplx(zero,-kdot1))
00462
00463                                rho = kbo(k,l,kcount)*conjgbloch
00464
00465                                ftmp(1) = ftmp(1) - rho * cosbeta * mydfdb
00466                                ftmp(2) = ftmp(2) - rho * cosbeta * mydfdb
00467                                ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00468
00469                                ENDDO
00470                                ENDDO
00471                                ENDDO
00472
00473                                ENDIF
00474
00475                                ENDDO
00476                                ENDDO
00477                                ENDDO
00478                                ENDDO
00479
00480                                ftmp = ftmp * ( hubbardu(elempointer(j))*deltaq(j) +
coulombv(j) &
00481                                +hubbardu(elempointer(i))*deltaq(i) +
coulombv(i))
00482
00483
00484                                kf(1,i) = kf(1,i) + ftmp(1)
00485                                kf(2,i) = kf(2,i) + ftmp(2)
00486                                kf(3,i) = kf(3,i) + ftmp(3)
00487
00488                                ! with the factor of 2...
00489
00490                                virbondk(1) = virbondk(1) + rij(1)*ftmp(1)
00491                                virbondk(2) = virbondk(2) + rij(2)*ftmp(2)
00492                                virbondk(3) = virbondk(3) + rij(3)*ftmp(3)
00493                                virbondk(4) = virbondk(4) + rij(1)*ftmp(2)
00494                                virbondk(5) = virbondk(5) + rij(2)*ftmp(3)
00495                                virbondk(6) = virbondk(6) + rij(3)*ftmp(1)
00496
00497                                ENDF
00498                                ENDDO
00499
00500
00501                                ENDDO
00502
00503                                !$OMP END PARALLEL DO
00504
00505                                virbondk = virbondk/two
00506
00507                                ! PRINT*, KF(1,1), KF(2,1), KF(3,1)
00508
00509                                fscoul = REAL(kf)/REAL(nktot)
00510                                virscoul = REAL(virbondk)/REAL(nktot)
00511
00512                                ! DO I = 1, NATS
00513                                !     WRITE(6,10) I, FSCOUL(1,I), FSCOUL(2,I), FSCOUL(3,I)
00514                                !     ENDDO
00515
00516                                !10 FORMAT(I4, 3F12.6)
00517
00518                                RETURN
00519
00520                                END SUBROUTINE kfcoulnono

```

8.221 kgenX.f90 File Reference

Functions/Subroutines

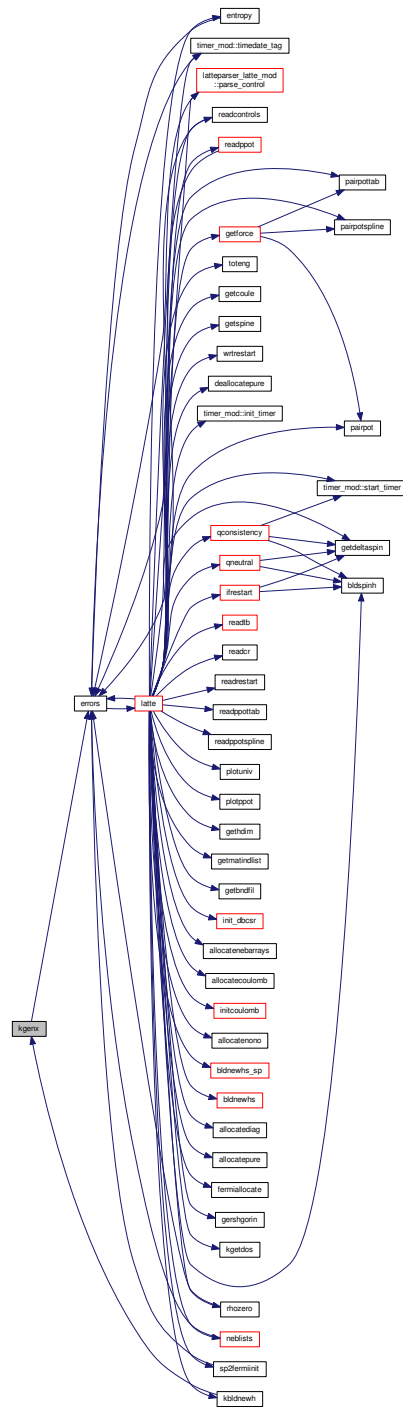
- subroutine [kgenx](#)

8.221.1 Function/Subroutine Documentation

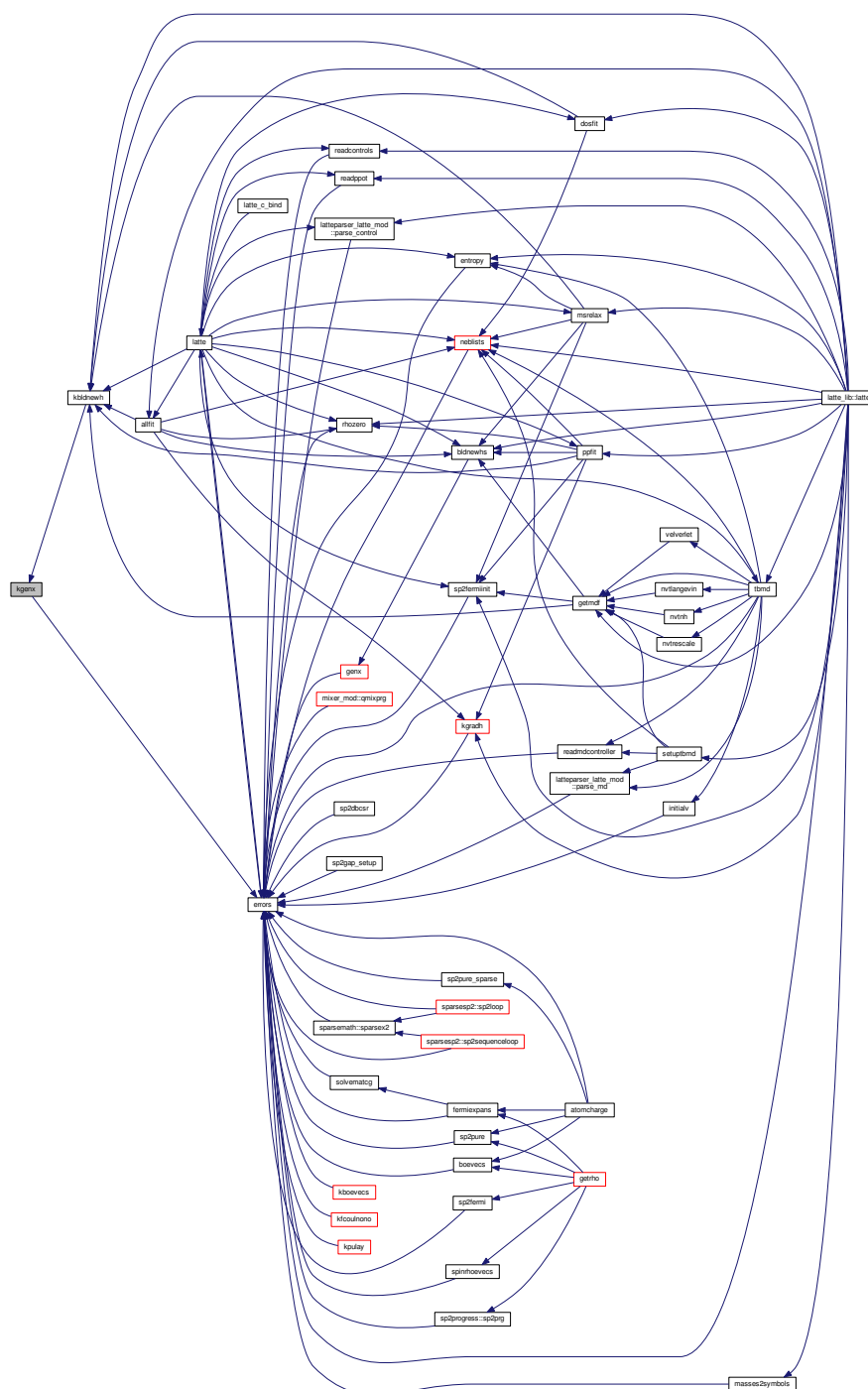
8.221.1.1 subroutine kgenx ()

Definition at line 23 of file [kgenX.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.222 kgenX.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kgenx
00023
00024   USE constants_mod
00025   USE nonoarray
00026   USE kspacearray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, J, K, INFO, II, LWORK
00032   REAL(LATTEPREC) :: INVSQRT
00033   REAL(LATTEPREC), ALLOCATABLE :: EVAL(:), RWORK(:)
00034   COMPLEX(LATTEPREC), ALLOCATABLE :: UK(:, :), KTMPMAT(:, :), WORK(:)
00035   COMPLEX(LATTEPREC) :: ALPHA, BETA
00036   IF (existerror) RETURN
00037
00038   lwork = 2*hdim - 1
00039
00040   ALLOCATE(uk(hdim, hdim), eval(hdim), ktmpmat(hdim, hdim))
00041   ALLOCATE(work(lwork), rwork(3*hdim - 2))
00042   !
00043   ! X = U s^-1/2 U^dag
00044   !
00045
00046   ! Eigenvectors overwrite S (S = U)
00047
00048   DO ii = 1, nktot
00049
00050     uk(:, :) = sk(:, :, ii)
00051
00052     CALL zheev('V', 'U', hdim, uk, hdim, eval, work, lwork, rwork, info)
00053
00054     IF (eval(1) .LT. zero) THEN
00055       CALL errors("kgenX", "Eigenvalue of complex S matrix < 0: STOP!")
00056     ENDIF
00057
00058     DO i = 1, hdim
00059
00060       invsqrt = one/sqrt(eval(i))
00061
00062       DO j = 1, hdim
00063         ktmpmat(j,i) = uk(j,i) * invsqrt
00064       ENDDO
00065
00066     ENDDO
00067
00068     alpha = cmplx(one, zero)
00069     beta = cmplx(zero, zero)
00070
00071     CALL zgemm('N', 'C', hdim, hdim, hdim, alpha, ktmpmat, hdim, uk, &
00072               hdim, beta, kxmat(:, :, ii), hdim)
00073
00074   ENDDO
00075
00076   DEALLOCATE(uk, eval, ktmpmat, work, rwork)
00077
00078   RETURN
00079
00080 END SUBROUTINE kgenx
00081

```

8.223 kgetdos.f90 File Reference

Functions/Subroutines

- subroutine [kgetdos](#)

8.224 kgetdos.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kgetdos
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE myprecision
00028   USE kspacearray
00029   USE mdarray
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, COUNT
00034   REAL(LATTEPREC), PARAMETER :: EWINDOW = 0.05, eta = 0.05
00035   REAL(LATTEPREC) :: INTDOS, ENERGY, NUME, DOSMIN, DOSMAX
00036   INTEGER :: NDOSBINS, BINID, NUMORB
00037   REAL(LATTEPREC), ALLOCATABLE :: DOS(:)
00038   COMPLEX(LATTEPREC) :: CMPARG
00039   IF (existerror) RETURN
00040
00041   dosmax = maxval(kevals) + 10.0*ewindow
00042   dosmin = minval(kevals) - 10.0*ewindow
00043
00044   ndosbins = int((dosmax - dosmin)/ewindow)
00045
00046   ALLOCATE(dos(ndosbins))
00047
00048   dos = zero
00049
00050   DO i = 1, ndosbins
00051
00052     energy = dosmin + REAL(i-1)*EWINDOW
00053
00054     DO k = 1, nktot
00055       DO j = 1, hdim
00056
00057         cmparg = one/(energy - kevals(j,k) + cmplx(zero,eta))
00058
00059         dos(i) = dos(i) - (one/pi)*aimag(cmparg)
00060
00061       ENDDO
00062     ENDDO
00063
00064   ENDDO
00065
00066   dos = dos/REAL(nktot)
00067
00068   ! To normalize, lets integrate the dos
00069
00070   intdos = zero
00071   count = 0
00072   DO i = 1, ndosbins
00073
00074     energy = dosmin + REAL(i-1)*EWINDOW
00075
00076     IF (energy .LE. chempot) THEN
00077       count = count + 1
00078       IF (mod(i,2) .EQ. 0) THEN
00079         intdos = intdos + four*dos(i)
00080       ELSE
00081         intdos = intdos + two*dos(i)
00082       ENDIF
00083     ENDIF
00084

```

```

00085  ENDDO
00086
00087  intdos = intdos - dos(1) - dos(count)
00088
00089  intdos = intdos*ewindow/three
00090
00091  ! Let's figure out what the integrated DOS should be
00092  ! In VASP the integral of the DOS up to the chemical
00093  ! potential = the total number of electrons. Let's do the
00094  ! same.
00095
00096  nume = zero
00097  DO i = 1, nats
00098      nume = nume + atocc(elempointer(i))
00099  ENDDO
00100
00101  dos = dos*nume/intdos
00102
00103
00104  OPEN(unit=50, status="UNKNOWN", file="DoS.dat")
00105
00106  DO i = 1, ndosbins
00107      WRITE(50, 10) dosmin + REAL(i-1)*EWINDOW - CHEMPOT, DOS(i)
00108  ENDDO
00109
00110  10 FORMAT(2f12.6)
00111
00112  CLOSE(50)
00113
00114  DEALLOCATE(dos)
00115
00116  RETURN
00117
00118  END SUBROUTINE kgetdos
00119
00120
00121

```

8.225 kgetrho.f90 File Reference

Functions/Subroutines

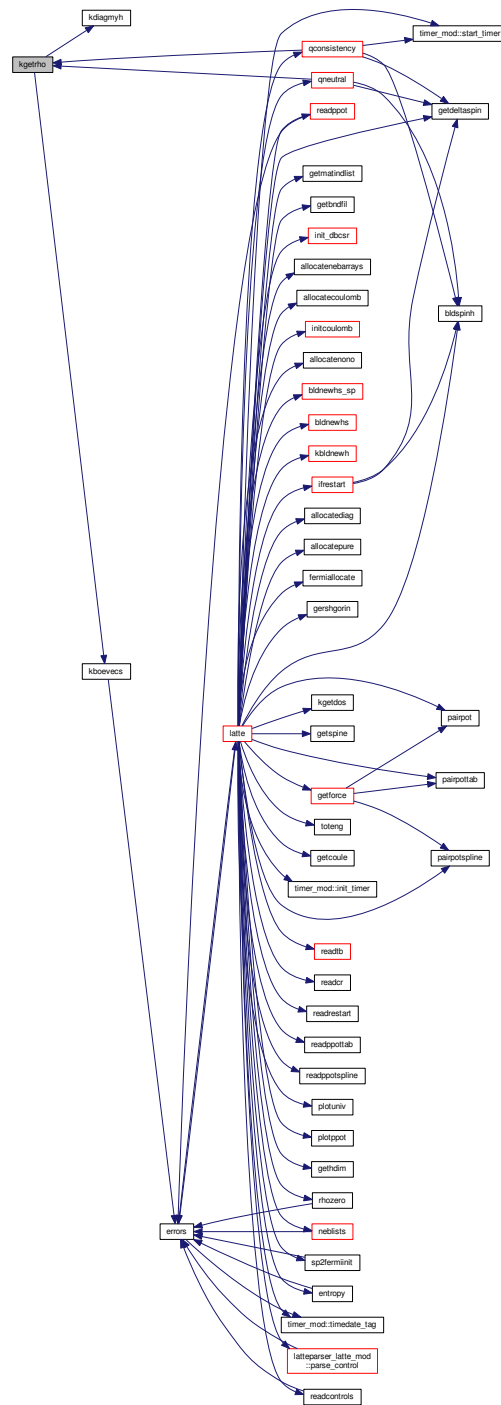
- subroutine [kgetrho](#)

8.225.1 Function/Subroutine Documentation

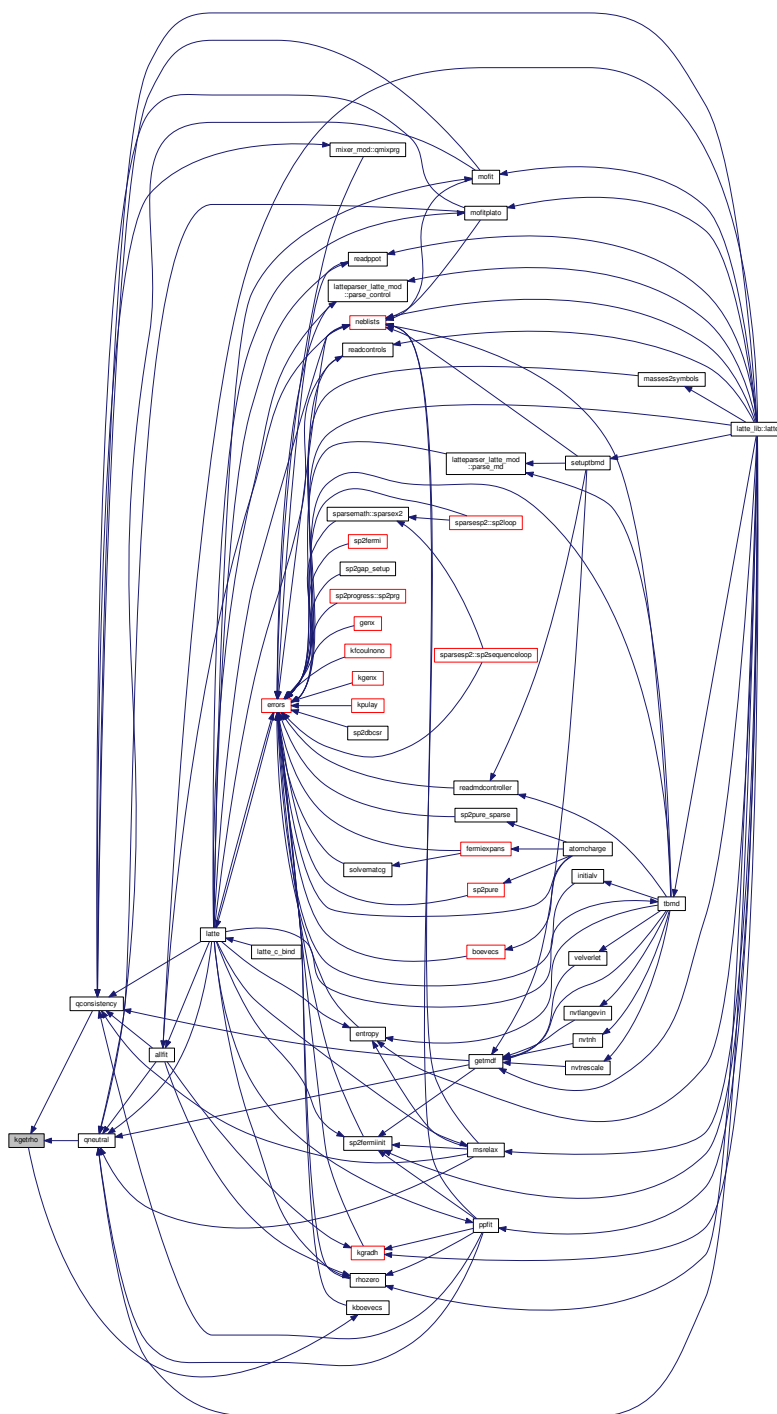
8.225.1.1 subroutine kgetrho ()

Definition at line 23 of file [kgetrho.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.226 kgetrho.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kgetrho
00023
00024   USE constants_mod
00025   USE myprecision
00026   USE kspacearray
00027   USE diagarray
00028   ! USE MPI
00029
00030   IMPLICIT NONE
00031   IF (existerror) RETURN
00032
00033   ! INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00034
00035   ! INTEGER :: I, J, MYID, IERR
00036   ! Lets first build our NKTOT Hamiltonians (we'll need all of them
00037   ! in order to get the density matrix via SP2....
00038
00039   ! CALL MPI_COMM_RANK(MPI_COMM_WORLD, MYID, IERR)
00040
00041   ! CALL SYSTEM_CLOCK(START_CLOCK, CLOCK_RATE, CLOCK_MAX)
00042
00043   CALL kdiagmyh
00044
00045   ! CALL SYSTEM_CLOCK(STOP_CLOCK, CLOCK_RATE, CLOCK_MAX)
00046
00047   ! PRINT*, "# Diag time (s) = ", &
00048   !     REAL(STOP_CLOCK - START_CLOCK)/REAL(CLOCK_RATE)
00049
00050   ! DO I = 1, NKTOT
00051   !     PRINT*, I, MYID, KEVALS(1,I)
00052   ! ENDDO
00053
00054   CALL kbovecs
00055
00056
00057
00058   ! DO I = 1, NKTOT
00059   !     PRINT*, I, MYID, KBO(1,1,I), KBO(HDIM, HDIM, I)
00060   ! ENDDO
00061
00062   RETURN
00063
00064 END SUBROUTINE kgetrho

```

8.227 kgradH.f90 File Reference

Functions/Subroutines

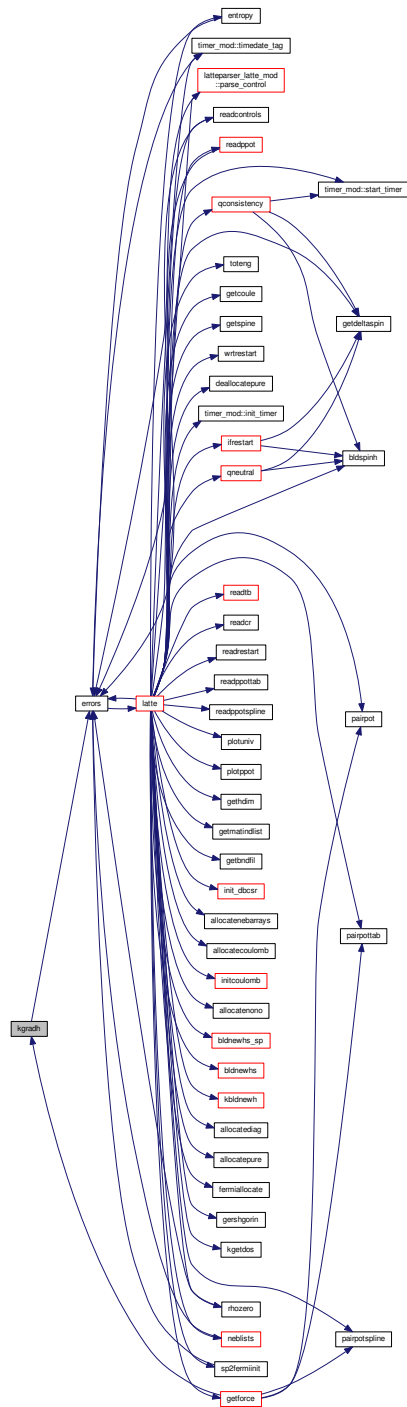
- subroutine [kgradh](#)

8.227.1 Function/Subroutine Documentation

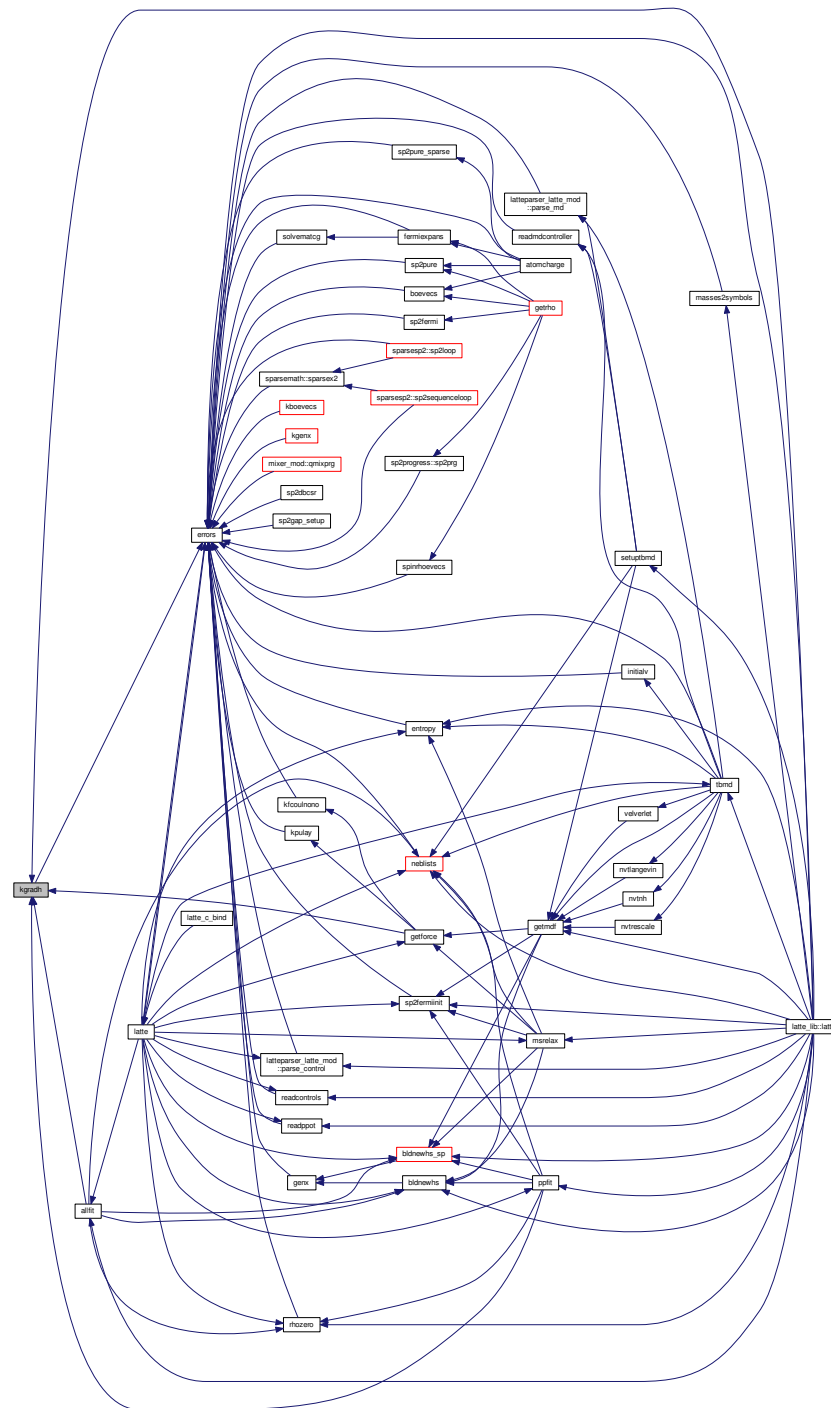
8.227.1.1 subroutine kgradh ()

Definition at line 23 of file [kgradH.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.228 kgradH.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kgradh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE univarray
00028   USE spinarray
00029   USE virialarray
00030   USE kspacearray
00031   USE myprecision
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00036   INTEGER :: LBRA, MBRA, LKET, MKET
00037   INTEGER :: PREVJ, NEWJ
00038   INTEGER :: PBCI, PBCJ, PBCK
00039   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00040   INTEGER :: KX, KY, KZ, KCOUNT
00041   REAL(LATTEPREC) :: ALPHA, BETA, PHI, COSBETA
00042   REAL(LATTEPREC) :: RIJ(3), DC(3)
00043   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2
00044   REAL(LATTEPREC) :: MAXARRAY(20), MAXRCUT, MAXRCUT2
00045   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00046   REAL(LATTEPREC) :: KPOINT(3), KX0, KY0, KZ0, KDOTL
00047   REAL(LATTEPREC) :: B1(3), B2(3), B3(3), MAG1, MAG2, MAG3, A1A2XA3, K0(3)
00048   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00049   COMPLEX(LATTEPREC) :: FTMP(3), RHO, CONJGBLOCH
00050   LOGICAL PATH
00051   IF (existerror) RETURN
00052
00053   kf = cmplx(zero)
00054   virbondk = cmplx(zero)
00055
00056   ! Computing the reciprocal lattice vectors
00057
00058   b1(1) = box(2,2)*box(3,3) - box(3,2)*box(2,3)
00059   b1(2) = box(3,1)*box(2,3) - box(2,1)*box(3,3)
00060   b1(3) = box(2,1)*box(3,2) - box(3,1)*box(2,2)
00061
00062   a1a2xa3 = box(1,1)*b1(1) + box(1,2)*b1(2) + box(1,3)*b1(3)
00063
00064   ! B1 = 2*PI*(A2 X A3)/(A1.(A2 X A3))
00065
00066   b1 = b1/a1a2xa3
00067
00068   ! B2 = 2*PI*(A3 x A1)/(A1(A2 X A3))
00069
00070   b2(1) = (box(3,2)*box(1,3) - box(1,2)*box(3,3))/a1a2xa3
00071   b2(2) = (box(1,1)*box(3,3) - box(3,1)*box(1,3))/a1a2xa3
00072   b2(3) = (box(3,1)*box(1,2) - box(1,1)*box(3,2))/a1a2xa3
00073
00074   ! B3 = 2*PI*(A1 x A2)/(A1(A2 X A3))
00075
00076   b3(1) = (box(1,2)*box(2,3) - box(2,2)*box(1,3))/a1a2xa3
00077   b3(2) = (box(2,1)*box(1,3) - box(1,1)*box(2,3))/a1a2xa3
00078   b3(3) = (box(1,1)*box(2,2) - box(2,1)*box(1,2))/a1a2xa3
00079
00080   k0 = pi*(one - REAL(nkx))/(REAL(nkx))*b1 + &
00081        pi*(one - REAL(nky))/(REAL(nky))*b2 + &
00082        pi*(one - REAL(nkz))/(REAL(nkz))*b3 - PI*KSHIFT
00083
00084
00085   !$OMP PARALLEL DO DEFAULT (NONE) &
00086   !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00087   !$OMP SHARED(CR, BOX, KBO, SPINON, NOINT, ATELE, ELE1, ELE2) &
00088   !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00089   !$OMP SHARED(K0, B1, B2, B3, NKX, NKY, NKZ, KF) &
00090   !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00091   !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00092   !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00093   !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, CONJGBLOCH, KDOTL, RCUTTB) &

```

```

00094      !$OMP PRIVATE(KPOINT, KCOUNT) &
00095      !$OMP REDUCTION(+: VIRBONDK)
00096
00097      DO i = 1, nats
00098
00099          ! Build list of orbitals on atom I
00100          SELECT CASE(basis(elempointer(i)))
00101
00102              CASE("s")
00103                  basisi(1) = 0
00104                  basisi(2) = -1
00105              CASE("p")
00106                  basisi(1) = 1
00107                  basisi(2) = -1
00108              CASE("d")
00109                  basisi(1) = 2
00110                  basisi(2) = -1
00111              CASE("f")
00112                  basisi(1) = 3
00113                  basisi(2) = -1
00114              CASE("sp")
00115                  basisi(1) = 0
00116                  basisi(2) = 1
00117                  basisi(3) = -1
00118              CASE("sd")
00119                  basisi(1) = 0
00120                  basisi(2) = 2
00121                  basisi(3) = -1
00122              CASE("sf")
00123                  basisi(1) = 0
00124                  basisi(2) = 3
00125                  basisi(3) = -1
00126              CASE("pd")
00127                  basisi(1) = 1
00128                  basisi(2) = 2
00129                  basisi(3) = -1
00130              CASE("pf")
00131                  basisi(1) = 1
00132                  basisi(2) = 3
00133                  basisi(3) = -1
00134              CASE("df")
00135                  basisi(1) = 2
00136                  basisi(2) = 3
00137                  basisi(3) = -1
00138              CASE("spd")
00139                  basisi(1) = 0
00140                  basisi(2) = 1
00141                  basisi(3) = 2
00142                  basisi(4) = -1
00143              CASE("spf")
00144                  basisi(1) = 0
00145                  basisi(2) = 1
00146                  basisi(3) = 3
00147                  basisi(4) = -1
00148              CASE("sdf")
00149                  basisi(1) = 0
00150                  basisi(2) = 2
00151                  basisi(3) = 3
00152                  basisi(4) = -1
00153              CASE("pdf")
00154                  basisi(1) = 1
00155                  basisi(2) = 2
00156                  basisi(3) = 3
00157                  basisi(4) = -1
00158              CASE("spdf")
00159                  basisi(1) = 0
00160                  basisi(2) = 1
00161                  basisi(3) = 2
00162                  basisi(4) = 3
00163                  basisi(5) = -1
00164          END SELECT
00165
00166          ! find the right place in the array
00167
00168          indi = matindlist(i)
00169
00170          DO newj = 1, totnebtb(i)
00171
00172              j = nebtb(1, newj, i)
00173              pbci = nebtb(2, newj, i)
00174              pbcj = nebtb(3, newj, i)
00175              pbck = nebtb(4, newj, i)
00176
00177              rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00178                  REAL(pbck)*BOX(3,1) - CR(1,i)
00179
00180              rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &

```

```

00181      REAL(pbck)*BOX(3,2) - CR(2,i)
00182
00183      rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00184      REAL(pbck)*BOX(3,3) - CR(3,i)
00185
00186      magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00187
00188      ! Building the forces is expensive - use the cut-off
00189      rcuttb = zero
00190      DO k = 1, noint
00191
00192          IF ( (atele(i) .EQ. ele1(k) .AND. &
00193              atele(j) .EQ. ele2(k)) .OR. &
00194              (atele(j) .EQ. ele1(k) .AND. &
00195              atele(i) .EQ. ele2(k) )) THEN
00196
00197              IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00198
00199              IF (basistype .EQ. "NONORTHO") THEN
00200                  IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00201              ENDIF
00202
00203          ENDIF
00204
00205      ENDDO
00206
00207      IF (magr2 .LT. rcuttb*rcuttb) THEN
00208
00209          magr = sqrt(magr2)
00210          ! IF (MAGR .LT. 2.5) PRINT*, "Short bond"
00211
00212          ! Build list of orbitals on atom J
00213
00214          SELECT CASE(basis(elempointer(j)))
00215          CASE("s")
00216              basisj(1) = 0
00217              basisj(2) = -1
00218          CASE("p")
00219              basisj(1) = 1
00220              basisj(2) = -1
00221          CASE("d")
00222              basisj(1) = 2
00223              basisj(2) = -1
00224          CASE("f")
00225              basisj(1) = 3
00226              basisj(2) = -1
00227          CASE("sp")
00228              basisj(1) = 0
00229              basisj(2) = 1
00230              basisj(3) = -1
00231          CASE("sd")
00232              basisj(1) = 0
00233              basisj(2) = 2
00234              basisj(3) = -1
00235          CASE("sf")
00236              basisj(1) = 0
00237              basisj(2) = 3
00238              basisj(3) = -1
00239          CASE("pd")
00240              basisj(1) = 1
00241              basisj(2) = 2
00242              basisj(3) = -1
00243          CASE("pf")
00244              basisj(1) = 1
00245              basisj(2) = 3
00246              basisj(3) = -1
00247          CASE("df")
00248              basisj(1) = 2
00249              basisj(2) = 3
00250              basisj(3) = -1
00251          CASE("spd")
00252              basisj(1) = 0
00253              basisj(2) = 1
00254              basisj(3) = 2
00255              basisj(4) = -1
00256          CASE("spf")
00257              basisj(1) = 0
00258              basisj(2) = 1
00259              basisj(3) = 3
00260              basisj(4) = -1
00261          CASE("sdf")
00262              basisj(1) = 0
00263              basisj(2) = 2
00264              basisj(3) = 3
00265              basisj(4) = -1
00266          CASE("pdf")
00267              basisj(1) = 1

```

```

00268         basisj(2) = 2
00269         basisj(3) = 3
00270         basisj(4) = -1
00271     CASE("spdf")
00272         basisj(1) = 0
00273         basisj(2) = 1
00274         basisj(3) = 2
00275         basisj(4) = 3
00276         basisj(5) = -1
00277     END SELECT
00278
00279     indj = matindlist(j)
00280
00281     magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00282     magrp = sqrt(magrp2)
00283
00284     ! transform to system in which z-axis is aligned with RIJ
00285
00286     path = .false.
00287     IF (abs(rij(1)) .GT. 1e-12) THEN
00288         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00289             phi = zero
00290         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00291             phi = two * pi
00292         ELSE
00293             phi = pi
00294         ENDIF
00295         alpha = atan(rij(2) / rij(1)) + phi
00296     ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00297         IF (rij(2) .GT. 1e-12) THEN
00298             alpha = pi / two
00299         ELSE
00300             alpha = three * pi / two
00301         ENDIF
00302     ELSE
00303         ! pathological case: pole in alpha at beta=0
00304         path = .true.
00305     ENDIF
00306
00307     cosbeta = rij(3) / magr
00308     beta = acos( cosbeta )
00309
00310     !          PRINT*, ALPHA, BETA
00311
00312     dc = rij/magr
00313
00314     ! build forces using PRB 72 165107 eq. (12) - the sign of the
00315     ! dfda contribution seems to be wrong, but gives the right
00316     ! answer(?)
00317
00318     ftmp = cmplx(zero)
00319     k = indi
00320
00321     lbrainc = 1
00322     DO WHILE (basisi(lbrainc) .NE. -1)
00323
00324         lbra = basisi(lbrainc)
00325         lbrainc = lbrainc + 1
00326
00327         DO mbra = -lbra, lbra
00328
00329             k = k + 1
00330             l = indj
00331
00332             lketinc = 1
00333             DO WHILE (basisj(lketinc) .NE. -1)
00334
00335                 lket = basisj(lketinc)
00336                 lketinc = lketinc + 1
00337
00338                 DO mket = -lket, lket
00339
00340                     l = l + 1
00341
00342                     IF (.NOT. path) THEN
00343
00344                         ! Unroll loops and pre-compute
00345
00346                         mydfda = dfda(i, j, lbra, lket, mbra, &
00347                             mket, magr, alpha, cosbeta, "H")
00348
00349                         mydfdb = dfdb(i, j, lbra, lket, mbra, &
00350                             mket, magr, alpha, cosbeta, "H")
00351
00352                         mydfdr = dfdr(i, j, lbra, lket, mbra, &
00353                             mket, magr, alpha, cosbeta, "H")
00354

```

```

00355      !
00356      ! d/d_alpha
00357      !
00358
00359      kcount = 0
00360
00361      DO kx = 1, nkx
00362
00363          DO ky = 1, nky
00364
00365              DO kz = 1, nkz
00366
00367                  kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00368                      REAL(ky-1)*B2/REAL(NKY) + &
00369                      REAL(kz-1)*B3/REAL(nkz)) + k0
00370
00371                  kcount = kcount+1
00372
00373                  kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00374                      kpoint(3)*rij(3)
00375
00376                  conjgbloch = exp(cmplx(zero,-kdot1))
00377
00378                  SELECT CASE(spinon)
00379                  CASE(0)
00380                      rho = kbo(k,1, kcount)
00381                      !
00382                      RHO = CONJG(KBO(L,K,KCOUNT))
00383                  CASE(1)
00384                      CALL errors("kgradH","KSPACE AND SPIN POLARIZATION NOT
IMPLEMENTED")
00385
00386                      !
00387                      RHO = RHOUP(L, K) + RHODOWN(L, K)
00388                  END SELECT
00389
00390                  ftmp(1) = ftmp(1) + rho * &
00391                      (-rij(2) / magrp2 * mydfda)*conjgbloch
00392
00393                  ftmp(2) = ftmp(2) + rho * &
00394                      (rij(1)/ magrp2 * mydfda)*conjgbloch
00395
00396                  !
00397                  ! d/d_beta
00398                  !
00399
00400                  ftmp(1) = ftmp(1) + rho * &
00401                      (((rij(3) * rij(1)) / &
00402                      magr2)) / magrp) * mydfdb)*conjgbloch
00403
00404                  ftmp(2) = ftmp(2) + rho * &
00405                      (((rij(3) * rij(2)) / &
00406                      magr2)) / magrp) * mydfdb)*conjgbloch
00407
00408                  ftmp(3) = ftmp(3) - rho * &
00409                      ((one - (rij(3) * rij(3)) / &
00410                      magr2)) / magrp) * mydfdb)*conjgbloch
00411
00412                  !
00413                  ! d/dR
00414                  !
00415
00416                  ftmp(1) = ftmp(1) - rho * dc(1) * &
00417                      mydfdr*conjgbloch
00418
00419                  ftmp(2) = ftmp(2) - rho * dc(2) * &
00420                      mydfdr*conjgbloch
00421
00422                  ftmp(3) = ftmp(3) - rho * dc(3) * &
00423                      mydfdr*conjgbloch
00424
00425                      ENDDO
00426                  ENDDO
00427              ENDDO
00428
00429          ELSE
00430
00431              ! pathological configuration in which beta=0
00432              ! or pi => alpha undefined
00433
00434              ! Bug fixed: MJC 12/17/13
00435
00436
00437
00438              mydfdb = dfdb(i, j, lbra, lket, mbra, &
00439                  mket, magr, zero, cosbeta, "H") / magr
00440

```

```

00441      mydfdb = dfdb(i, j, lbra, lket, &
00442      mbra, mket, magr, pi/two, cosbeta, "H") / magr
00443
00444      mydfdr = dfdr(i, j, lbra, lket, mbra, &
00445      mket, magr, zero, cosbeta, "H")
00446
00447
00448      kcount = 0
00449      DO kx = 1, nkx
00450
00451          DO ky = 1, nky
00452
00453              DO kz = 1, nkz
00454
00455                  kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00456                  REAL(ky-1)*B2/REAL(NKY) + &
00457                  REAL(kz-1)*B3/REAL(nkz)) + k0
00458
00459                  kcount = kcount+1
00460
00461                  kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00462                  kpoint(3)*rij(3)
00463
00464                  conjgbloch = exp(cmplx(zero,-kdot1))
00465
00466                  SELECT CASE(spinon)
00467                  CASE(0)
00468                      rho = kbo(k,l, kcount)
00469                      !
00470                      RHO = CONJG(KBO(L,K,KCOUNT))
00471                  CASE(1)
00472                      CALL errors("kgradH","KSPACE AND SPIN POLARIZATION NOT
IMPLEMENTED")
00473                      !
00474                      RHO = RHOU(L, K) + RHODOWN(L, K)
00475                  END SELECT
00476
00477                  ftmp(1) = ftmp(1) - rho * cosbeta * mydfdb*conjgbloch
00478                  ftmp(2) = ftmp(2) - rho * cosbeta * mydfdb*conjgbloch
00479                  ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr*conjgbloch
00480
00481              ENDDO
00482          ENDDO
00483      ENDDO
00484  ENDDO
00485  ENDDO
00486  ENDDO
00487  ENDDO
00488  ENDDO
00489
00490      kf(1,i) = kf(1,i) + ftmp(1)
00491      kf(2,i) = kf(2,i) + ftmp(2)
00492      kf(3,i) = kf(3,i) + ftmp(3)
00493
00494      virbondk(1) = virbondk(1) + rij(1) * ftmp(1)
00495      virbondk(2) = virbondk(2) + rij(2) * ftmp(2)
00496      virbondk(3) = virbondk(3) + rij(3) * ftmp(3)
00497      virbondk(4) = virbondk(4) + rij(1) * ftmp(2)
00498      virbondk(5) = virbondk(5) + rij(2) * ftmp(3)
00499      virbondk(6) = virbondk(6) + rij(3) * ftmp(1)
00500
00501  ENDDO
00502  ENDDO
00503  ENDDO
00504  ENDDO
00505  ENDDO
00506  !$OMP END PARALLEL DO
00507
00508      ! PRINT *, KF(1,1)/REAL(NKTOT)
00509      f = REAL(kf)/REAL(nktot)
00510      virbond = REAL(virbondk)/REAL(nktot)
00511
00512      RETURN
00513
00514  END SUBROUTINE kgradh

```

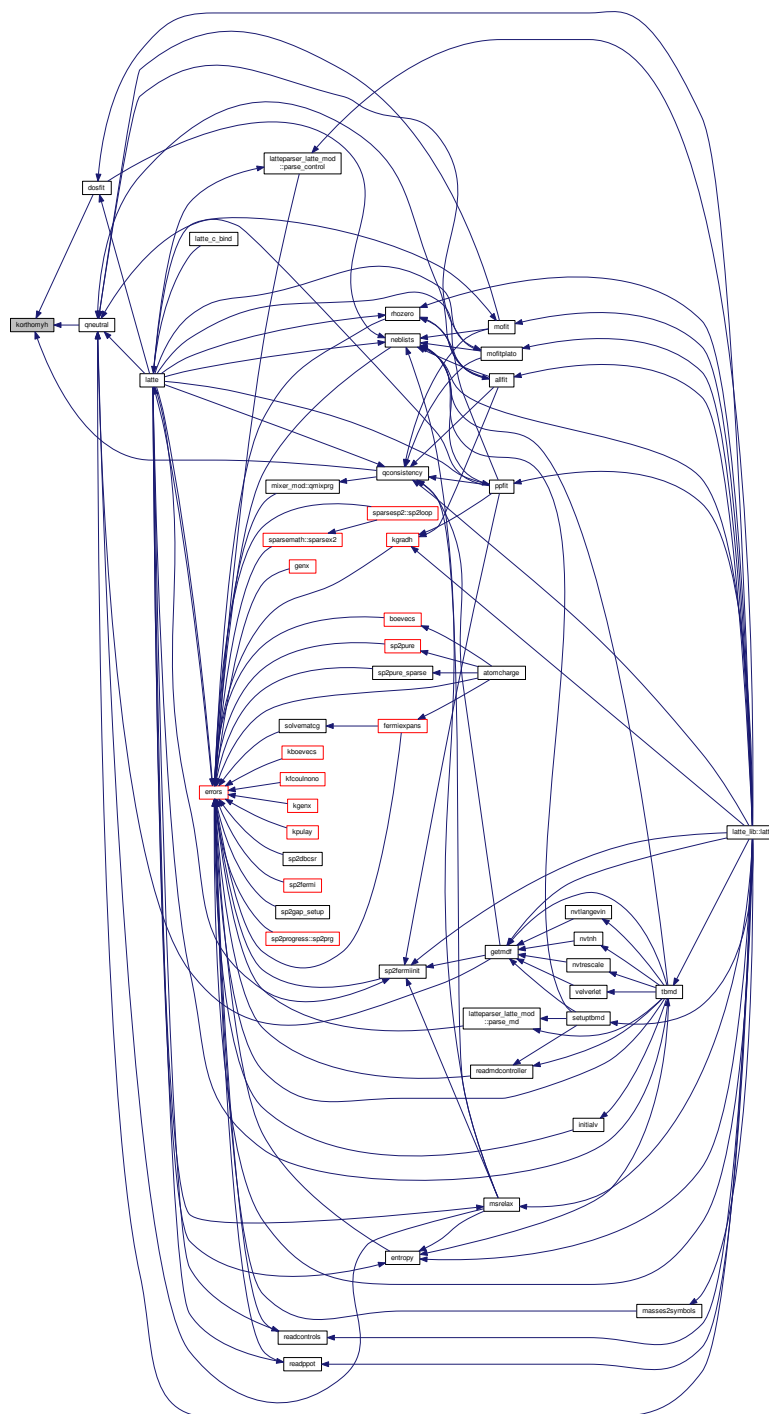
8.229 korthomyH.f90 File Reference

Functions/Subroutines

- subroutine [korthomyh](#)

8.229.1.1 subroutine korthomyh ()

Here is the caller graph for this function:



8.230 korthomyH.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE korthomyh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE nonoarray
00027   USE kspacearray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER I, J, II
00033   COMPLEX(LATTEPREC), PARAMETER :: ALPHA = cmplx(one, zero), beta=cmplx(
zero, zero)
00034   COMPLEX(LATTEPREC), ALLOCATABLE :: KTMP(:, :)
00035   IF (existerror) RETURN
00036
00037   ALLOCATE (ktmp (hdim, hdim))
00038
00039   !
00040   ! ORTHOH = X^dag H X
00041   !
00042
00043   DO ii = 1, nktot
00044
00045       CALL zgemm('C', 'N', hdim, hdim, hdim, alpha, kxmat(:, :, ii), &
00046               hdim, hk(:, :, ii), hdim, beta, ktmp, hdim)
00047       CALL zgemm('N', 'N', hdim, hdim, hdim, alpha, ktmp, hdim, &
00048               kxmat(:, :, ii), hdim, beta, korthoh(:, :, ii), hdim)
00049
00050   ENDDO
00051
00052
00053   RETURN
00054
00055 END SUBROUTINE korthomyh
00056
00057

```

8.231 kpulay.f90 File Reference

Functions/Subroutines

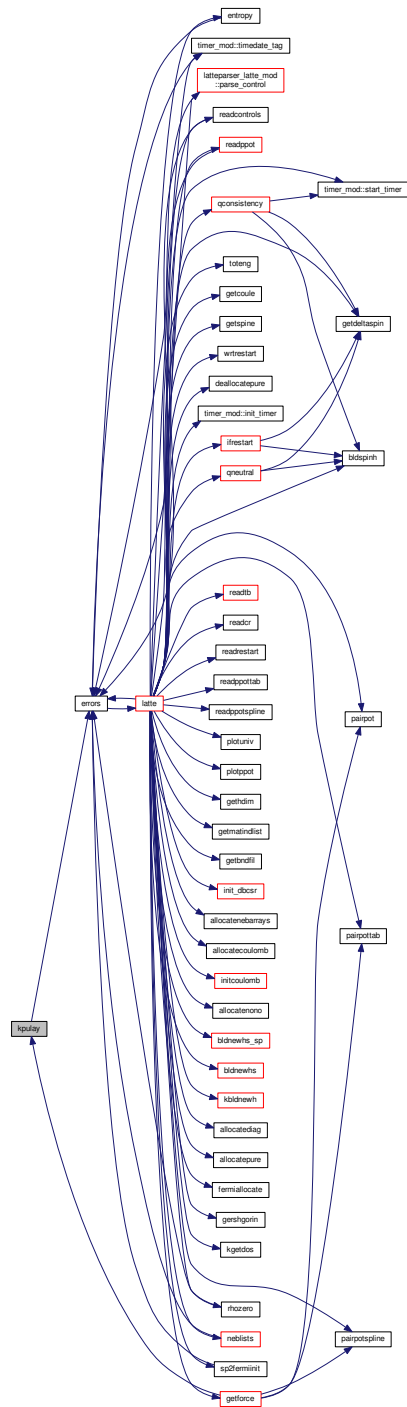
- subroutine [kpulay](#)

8.231.1 Function/Subroutine Documentation

8.231.1.1 subroutine kpulay ()

Definition at line 23 of file [kpulay.f90](#).

Here is the call graph for this function:




```

00012 !
00013 ! Additionally, this program is free software; you can redistribute it
00014 ! and/or modify it under the terms of the GNU General Public License as
00015 ! published by the Free Software Foundation; version 2.0 of the License.
00016 ! Accordingly, this program is distributed in the hope that it will be
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
00019 ! Public License for more details.
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kpulay
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE neblistarray
00029   USE spinarray
00030   USE virialarray
00031   USE kspacearray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00037   INTEGER :: LBRA, MBRA, LKET, MKET
00038   INTEGER :: PREVJ, NEWJ
00039   INTEGER :: PBCI, PBCJ, PBCK
00040   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00041   INTEGER :: KX, KY, KZ, KCOUNT
00042   REAL(LATTEPREC) :: ALPHA, BETA, PHI, COSBETA
00043   REAL(LATTEPREC) :: RIJ(3), DC(3)
00044   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2
00045   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00046   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00047   REAL(LATTEPREC) :: KPOINT(3), KX0, KY0, KZ0, KDOTL
00048   REAL(LATTEPREC) :: B1(3), B2(3), B3(3), MAG1, MAG2, MAG3, A1A2XA3, K0(3)
00049   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00050   COMPLEX(LATTEPREC) :: FTMP(3), RHO, CONJGBLOCH
00051   COMPLEX(LATTEPREC), ALLOCATABLE :: KX2HRHO(:, :, :), KTMP(:, :)
00052   COMPLEX(LATTEPREC), PARAMETER :: ZONE=cmplx(one), zzero=cmplx(zero), zhalf=cmplx(
half)
00053
00054   LOGICAL PATH
00055   IF (existerror) RETURN
00056
00057   ! These were allocated elsewhere. We'll use them to accumulate the complex forces
00058
00059   IF (spinon .EQ. 1) THEN
00060     CALL errors("kpulay", "Non-ortho k-space and spin polarization not yet implemented")
00061   ENDIF
00062
00063   kf = cmplx(zero)
00064   virbondk = cmplx(zero)
00065
00066   ALLOCATE(kx2hrho(hdim, hdim, nktot), ktmp(hdim, hdim))
00067
00068   ! Computing the reciprocal lattice vectors
00069
00070   b1(1) = box(2,2)*box(3,3) - box(3,2)*box(2,3)
00071   b1(2) = box(3,1)*box(2,3) - box(2,1)*box(3,3)
00072   b1(3) = box(2,1)*box(3,2) - box(3,1)*box(2,2)
00073
00074   a1a2xa3 = box(1,1)*b1(1) + box(1,2)*b1(2) + box(1,3)*b1(3)
00075
00076   ! B1 = 2*PI*(A2 X A3)/(A1.(A2 X A3))
00077
00078   b1 = b1/a1a2xa3
00079
00080   ! B2 = 2*PI*(A3 x A1)/(A1(A2 X A3))
00081
00082   b2(1) = (box(3,2)*box(1,3) - box(1,2)*box(3,3))/a1a2xa3
00083   b2(2) = (box(1,1)*box(3,3) - box(3,1)*box(1,3))/a1a2xa3
00084   b2(3) = (box(3,1)*box(1,2) - box(1,1)*box(3,2))/a1a2xa3
00085
00086   ! B3 = 2*PI*(A1 x A2)/(A1(A2 X A3))
00087
00088   b3(1) = (box(1,2)*box(2,3) - box(2,2)*box(1,3))/a1a2xa3
00089   b3(2) = (box(2,1)*box(1,3) - box(1,1)*box(2,3))/a1a2xa3
00090   b3(3) = (box(1,1)*box(2,2) - box(2,1)*box(1,2))/a1a2xa3
00091
00092   k0 = pi*(one - REAL(nkx))/(REAL(nkx))*b1 + &
00093       pi*(one - REAL(nky))/(REAL(nky))*b2 + &
00094       pi*(one - REAL(nkz))/(REAL(nkz))*b3 - PI*KSHIFT
00095
00096   ! We first have to make the matrix S^-1 H rho = X^2 H rho
00097

```

```

00098     IF (kbt .GT. 0.000001) THEN
00099
00100         ! Finite temperature
00101
00102         DO k = 1, nktot
00103
00104             CALL zgemm('N', 'N', hdim, hdim, hdim, zone, &
00105                 kxmat(:, :, k), hdim, kxmat(:, :, k), hdim, zzero, kx2hrho(:, :, k),
00106                 hdim)
00107
00108             CALL zgemm('N', 'N', hdim, hdim, hdim, zone, &
00109                 kx2hrho(:, :, k), hdim, hk(:, :, k), hdim, zzero, ktmp, hdim)
00110
00111             ! (S^-1 * H)*RHO
00112
00113             CALL zgemm('N', 'N', hdim, hdim, hdim, zone, &
00114                 ktmp, hdim, kbo(:, :, k), hdim, zzero, kx2hrho(:, :, k),
00115                 hdim)
00116
00117             ENDDO
00118
00119         ELSE
00120
00121             ! Te = 0 : Fp = 2Tr[rho H rho dS/dR]
00122
00123             ! Be careful - we're working with bo = 2rho, so we need
00124             ! the factor of 1/2...
00125
00126             DO k = 1, nktot
00127
00128                 CALL zgemm('N', 'N', hdim, hdim, hdim, zone, &
00129                     kbo(:, :, k), hdim, hk(:, :, k), hdim, zzero, ktmp, hdim)
00130
00131                 CALL zgemm('N', 'N', hdim, hdim, hdim, zhalf, &
00132                     ktmp, hdim, kbo(:, :, k), hdim, zzero, kx2hrho(:, :, k),
00133                     hdim)
00134
00135                 ENDDO
00136
00137             ENDIF
00138
00139             !$OMP PARALLEL DO DEFAULT (NONE) &
00140             !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00141             !$OMP SHARED(CR, BOX, KX2HRHO, NOINT, ATELE, ELE1, ELE2) &
00142             !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00143             !$OMP SHARED(K0, B1, B2, B3, NKX, NKY, NKZ, KF) &
00144             !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00145             !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00146             !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00147             !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB, CONJGBLOCH, KDOTL) &
00148             !$OMP PRIVATE(KPOINT, KCOUNT) &
00149             !$OMP REDUCTION(+: VIRBONDK)
00150
00151             DO i = 1, nats
00152
00153                 ! Build list of orbitals on atom I
00154                 SELECT CASE(basis(elempointer(i)))
00155
00156                     CASE("s")
00157                         basisi(1) = 0
00158                         basisi(2) = -1
00159
00160                     CASE("p")
00161                         basisi(1) = 1
00162                         basisi(2) = -1
00163
00164                     CASE("d")
00165                         basisi(1) = 2
00166                         basisi(2) = -1
00167
00168                     CASE("f")
00169                         basisi(1) = 3
00170                         basisi(2) = -1
00171
00172                     CASE("sp")
00173                         basisi(1) = 0
00174                         basisi(2) = 1
00175                         basisi(3) = -1
00176
00177                     CASE("sd")
00178                         basisi(1) = 0
00179                         basisi(2) = 2
00180                         basisi(3) = -1
00181
00182                     CASE("sf")
00183                         basisi(1) = 0
00184                         basisi(2) = 3
00185                         basisi(3) = -1
00186
00187                     CASE("pd")
00188                         basisi(1) = 1
00189                         basisi(2) = 2
00190                         basisi(3) = -1

```

```

00182      CASE("pf")
00183          basisi(1) = 1
00184          basisi(2) = 3
00185          basisi(3) = -1
00186      CASE("df")
00187          basisi(1) = 2
00188          basisi(2) = 3
00189          basisi(3) = -1
00190      CASE("spd")
00191          basisi(1) = 0
00192          basisi(2) = 1
00193          basisi(3) = 2
00194          basisi(4) = -1
00195      CASE("spf")
00196          basisi(1) = 0
00197          basisi(2) = 1
00198          basisi(3) = 3
00199          basisi(4) = -1
00200      CASE("sdf")
00201          basisi(1) = 0
00202          basisi(2) = 2
00203          basisi(3) = 3
00204          basisi(4) = -1
00205      CASE("pdf")
00206          basisi(1) = 1
00207          basisi(2) = 2
00208          basisi(3) = 3
00209          basisi(4) = -1
00210      CASE("spdf")
00211          basisi(1) = 0
00212          basisi(2) = 1
00213          basisi(3) = 2
00214          basisi(4) = 3
00215          basisi(5) = -1
00216      END SELECT
00217
00218      indi = matindlist(i)
00219
00220      DO newj = 1, totnebtb(i)
00221
00222          j = nebtb(1, newj, i)
00223          pbci = nebtb(2, newj, i)
00224          pbcj = nebtb(3, newj, i)
00225          pbck = nebtb(4, newj, i)
00226
00227          rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00228              REAL(pbck)*BOX(3,1) - CR(1,i)
00229
00230          rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00231              REAL(pbck)*BOX(3,2) - CR(2,i)
00232
00233          rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00234              REAL(pbck)*BOX(3,3) - CR(3,i)
00235
00236          magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00237
00238          ! Building the forces is expensive - use the cut-off
00239
00240          rcuttb = zero
00241          DO k = 1, noint
00242
00243              IF ( (atele(i) .EQ. ele1(k) .AND. &
00244                  atele(j) .EQ. ele2(k)) .OR. &
00245                  (atele(j) .EQ. ele1(k) .AND. &
00246                  atele(i) .EQ. ele2(k) ) ) THEN
00247
00248                  IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00249
00250              ENDIF
00251
00252          ENDDO
00253
00254          IF (magr2 .LT. rcuttb*rcuttb) THEN
00255
00256              magr = sqrt(magr2)
00257
00258              ! Build list of orbitals on atom J
00259
00260              SELECT CASE(basis(elempointer(j)))
00261              CASE("s")
00262                  basisj(1) = 0
00263                  basisj(2) = -1
00264              CASE("p")
00265                  basisj(1) = 1
00266                  basisj(2) = -1
00267              CASE("d")
00268                  basisj(1) = 2

```

```

00269         basisj(2) = -1
00270     CASE("f")
00271         basisj(1) = 3
00272         basisj(2) = -1
00273     CASE("sp")
00274         basisj(1) = 0
00275         basisj(2) = 1
00276         basisj(3) = -1
00277     CASE("sd")
00278         basisj(1) = 0
00279         basisj(2) = 2
00280         basisj(3) = -1
00281     CASE("sf")
00282         basisj(1) = 0
00283         basisj(2) = 3
00284         basisj(3) = -1
00285     CASE("pd")
00286         basisj(1) = 1
00287         basisj(2) = 2
00288         basisj(3) = -1
00289     CASE("pf")
00290         basisj(1) = 1
00291         basisj(2) = 3
00292         basisj(3) = -1
00293     CASE("df")
00294         basisj(1) = 2
00295         basisj(2) = 3
00296         basisj(3) = -1
00297     CASE("spd")
00298         basisj(1) = 0
00299         basisj(2) = 1
00300         basisj(3) = 2
00301         basisj(4) = -1
00302     CASE("spf")
00303         basisj(1) = 0
00304         basisj(2) = 1
00305         basisj(3) = 3
00306         basisj(4) = -1
00307     CASE("sdf")
00308         basisj(1) = 0
00309         basisj(2) = 2
00310         basisj(3) = 3
00311         basisj(4) = -1
00312     CASE("pdf")
00313         basisj(1) = 1
00314         basisj(2) = 2
00315         basisj(3) = 3
00316         basisj(4) = -1
00317     CASE("spdf")
00318         basisj(1) = 0
00319         basisj(2) = 1
00320         basisj(3) = 2
00321         basisj(4) = 3
00322         basisj(5) = -1
00323     END SELECT
00324
00325     indj = matindlist(j)
00326
00327     magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00328     magrp = sqrt(magrp2)
00329
00330
00331     ! transform to system in which z-axis is aligned with RIJ
00332
00333     path = .false.
00334     IF (abs(rij(1)) .GT. 1e-12) THEN
00335         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00336             phi = zero
00337         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00338             phi = two * pi
00339         ELSE
00340             phi = pi
00341         ENDIF
00342         alpha = atan(rij(2) / rij(1)) + phi
00343     ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00344         IF (rij(2) .GT. 1e-12) THEN
00345             alpha = pi / two
00346         ELSE
00347             alpha = three * pi / two
00348         ENDIF
00349     ELSE
00350         ! pathological case: pole in alpha at beta=0
00351         path = .true.
00352     ENDIF
00353
00354     cosbeta = rij(3)/magr
00355     beta = acos(rij(3) / magr)

```

```

00356
00357      dc = rij/magr
00358
00359      ! build forces using PRB 72 165107 eq. (12) - the sign of the
00360      ! dfda contribution seems to be wrong, but gives the right
00361      ! answer(?)
00362
00363      ftmp = zero
00364      k = indi
00365
00366      lbrainc = 1
00367      DO WHILE (basisi(lbrainc) .NE. -1)
00368
00369          lbra = basisi(lbrainc)
00370          lbrainc = lbrainc + 1
00371
00372          DO mbra = -lbra, lbra
00373
00374              k = k + 1
00375              l = indj
00376
00377              lketinc = 1
00378              DO WHILE (basisj(lketinc) .NE. -1)
00379
00380                  lket = basisj(lketinc)
00381                  lketinc = lketinc + 1
00382
00383                  DO mket = -lket, lket
00384
00385                      l = l + 1
00386
00387                      !RHO = X2HRHO(L, K)
00388
00389                      IF (.NOT. path) THEN
00390
00391                          ! Unroll loops and pre-compute
00392
00393                          mydfda = dfda(i, j, lbra, lket, mbra, &
00394                              mket, magr, alpha, cosbeta, "S")
00395
00396                          mydfdb = dfdb(i, j, lbra, lket, mbra, &
00397                              mket, magr, alpha, cosbeta, "S")
00398
00399                          mydfdr = dfdr(i, j, lbra, lket, mbra, &
00400                              mket, magr, alpha, cosbeta, "S")
00401
00402
00403                          kcount = 0
00404
00405                          DO kx = 1, nkx
00406
00407                              DO ky = 1, nky
00408
00409                                  DO kz = 1, nkz
00410
00411                                      kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00412                                          REAL(ky-1)*B2/REAL(NKY) + &
00413                                          REAL(kz-1)*B3/REAL(nkz)) + k0
00414
00415                                      kcount = kcount+1
00416
00417                                      kdotl = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00418                                          kpoint(3)*rij(3)
00419
00420                                      conjgbloch = exp(cmplx(zero,-kdotl))
00421
00422                                      rho = kx2hrho(k,l,kcount)*conjgbloch
00423
00424                                      !
00425                                      ! d/d_alpha
00426                                      !
00427
00428                                      ftmp(1) = ftmp(1) + rho * &
00429                                          (-rij(2) / magrp2 * mydfda)
00430
00431                                      ftmp(2) = ftmp(2) + rho * &
00432                                          (rij(1)/ magrp2 * mydfda)
00433
00434                                      !
00435                                      ! d/d_beta
00436                                      !
00437
00438                                      ftmp(1) = ftmp(1) + rho * &
00439                                          (((rij(3) * rij(1)) / &
00440                                          magr2)) / magrp) * mydfdb)
00441
00442                                      ftmp(2) = ftmp(2) + rho * &

```

```

00443          (((rij(3) * rij(2)) / &
00444          magr2)) / magrp) * mydfdb)
00445
00446          ftmp(3) = ftmp(3) - rho * &
00447          ((one - (rij(3) * rij(3)) / &
00448          magr2)) / magrp) * mydfdb)
00449
00450          !
00451          ! d/dR
00452          !
00453
00454          ftmp(1) = ftmp(1) - rho * dc(1) * &
00455          mydfdr
00456
00457          ftmp(2) = ftmp(2) - rho * dc(2) * &
00458          mydfdr
00459
00460          ftmp(3) = ftmp(3) - rho * dc(3) * &
00461          mydfdr
00462
00463          ENDDO
00464          ENDDO
00465          ENDDO
00466
00467          ELSE
00468
00469          ! pathological configuration in which beta=0
00470          ! or pi => alpha undefined
00471
00472          ! fixed: MJC 12/17/13
00473
00474          mydfdb = dfdb(i, j, lbra, lket, &
00475          mbra, mket, magr, zero, cosbeta, "S") / magr
00476
00477
00478          mydfdb = dfdb(i, j, lbra, lket, &
00479          mbra, mket, magr, pi/two, cosbeta, "S") / magr
00480
00481          mydfdr = dfdr(i, j, lbra, lket, mbra, &
00482          mket, magr, zero, cosbeta, "S")
00483
00484          kcount = 0
00485
00486          DO kx = 1, nkx
00487
00488          DO ky = 1, nky
00489
00490          DO kz = 1, nkz
00491
00492          kpoint = two*pi*(REAL(kx-1)*B1/REAL(NKX) + &
00493          REAL(ky-1)*B2/REAL(NKY) + &
00494          REAL(kz-1)*B3/REAL(nkz)) + k0
00495
00496          kcount = kcount+1
00497
00498          kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00499          kpoint(3)*rij(3)
00500
00501          conjgbloch = exp(cmplx(zero,-kdot1))
00502
00503          rho = kx2hrho(k,l,kcount)*conjgbloch
00504
00505          ftmp(1) = ftmp(1) - rho * cosbeta * mydfdb
00506          ftmp(2) = ftmp(2) - rho * cosbeta * mydfdb
00507          ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00508
00509          ENDDO
00510          ENDDO
00511          ENDDO
00512
00513          ENDIF
00514
00515          ENDDO
00516          ENDDO
00517          ENDDO
00518          ENDDO
00519
00520          kf(1,i) = kf(1,i) + ftmp(1)
00521          kf(2,i) = kf(2,i) + ftmp(2)
00522          kf(3,i) = kf(3,i) + ftmp(3)
00523
00524          virbondk(1) = virbondk(1) + rij(1) * ftmp(1)
00525          virbondk(2) = virbondk(2) + rij(2) * ftmp(2)
00526          virbondk(3) = virbondk(3) + rij(3) * ftmp(3)
00527          virbondk(4) = virbondk(4) + rij(1) * ftmp(2)
00528          virbondk(5) = virbondk(5) + rij(2) * ftmp(3)
00529          virbondk(6) = virbondk(6) + rij(3) * ftmp(1)

```



```

00530
00531         ENDIF
00532
00533         ENDDO
00534
00535     ENDDO
00536
00537     !$OMP END PARALLEL DO
00538
00539     DEALLOCATE(kx2hrho, ktmp)
00540
00541     ! PRINT*, KF(1,1), KF(2,1), KF(3,1)
00542
00543     ! DO I= 1, NATS
00544     !     WRITE(6,10) I, FPUL(1,I), FPUL(2,I), FPUL(3,I)
00545     ! ENDDO
00546
00547     !10 FORMAT(I4, 3F12.6)
00548
00549     fpul = REAL(kf)/REAL(nktot)
00550     virpul = REAL(virbondk)/REAL(nktot)
00551
00552     ! print*, VIRPUL(1)
00553     RETURN
00554
00555 END SUBROUTINE kpulay

```

8.233 kspacearray.f90 File Reference

Modules

- module [kspacearray](#)

Variables

- complex(latteprec), dimension(:,:), allocatable [kspacearray::hk](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::hk0](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::hkdiag](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::kbo](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::kf](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::korthoh](#)
- real(latteprec), dimension(3) [kspacearray::kshift](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::kxmat](#)
- integer [kspacearray::nktot](#)
- integer [kspacearray::nkx](#)
- integer [kspacearray::nky](#)
- integer [kspacearray::nkz](#)
- complex(latteprec), dimension(:,:), allocatable [kspacearray::sk](#)
- complex(latteprec), dimension(6) [kspacearray::virbondk](#)
- complex(latteprec), dimension(:), allocatable [kspacearray::zhjj](#)

8.234 kspacearray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE kspacearray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER :: nkx, nky, nkz, nktot
00030   REAL(LATTEPREC) :: kshift(3)
00031   COMPLEX(LATTEPREC) :: virbondk(6)
00032   COMPLEX(LATTEPREC), ALLOCATABLE :: hk(:, :, :), hkdiag(:, :), kbo(:, :, :)
00033   COMPLEX(LATTEPREC), ALLOCATABLE :: kf(:, :), sk(:, :, :), kxmat(:, :, :)
00034   COMPLEX(LATTEPREC), ALLOCATABLE :: korthoh(:, :, :), zhjj(:)
00035   COMPLEX(LATTEPREC), ALLOCATABLE :: hk0(:, :, :)
00036
00037 END MODULE kspacearray

```

8.235 latte.f90 File Reference

Functions/Subroutines

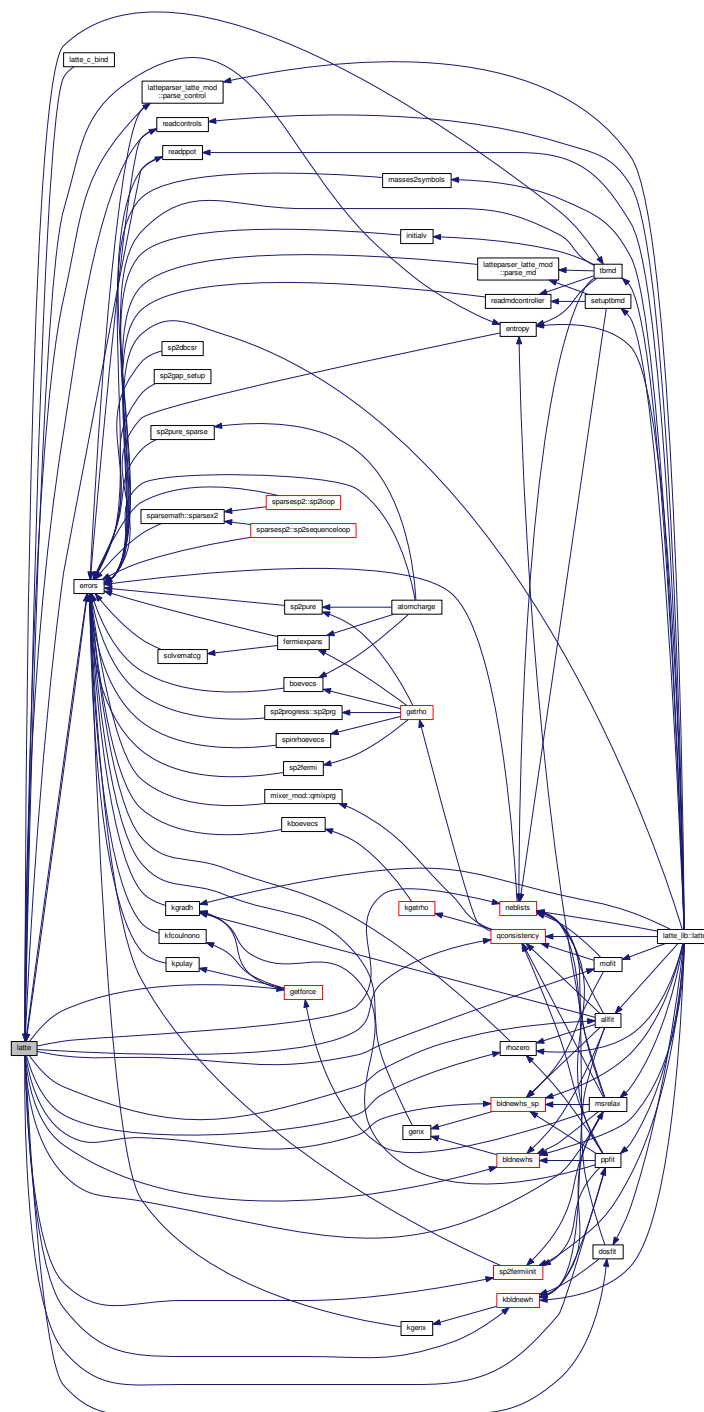
- program [latte](#)

8.235.1 Function/Subroutine Documentation

8.235.1.1 program latte ()

Definition at line 22 of file [latte.f90](#).

Here is the caller graph for this function:



8.236 latte.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 PROGRAM latte
00023
00024   USE constants_mod
00025   USE timer_mod
00026   USE setuparray
00027   USE ppotarray
00028   USE purearray
00029   USE coulombarray
00030   USE spinarray
00031   USE sparsearray
00032   USE myprecision
00033   USE virialarray
00034   USE diagarray
00035   USE kspacearray
00036   USE latteparser_latte_mod
00037 #ifdef MPI_ON
00038   USE mpi
00039 #endif
00040 #ifdef DBCSR_ON
00041   USE dbcsr_var_mod
00042 #endif
00043
00044 #ifdef PROGRESSION
00045   USE prg_system_mod ! FROM PROGRESS
00046   USE prg_pulaymixer_mod
00047   USE mixer_mod
00048   USE bml
00049 #endif
00050
00051
00052   IMPLICIT NONE
00053
00054   INTEGER :: I, J
00055   INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00056   INTEGER :: MYID = 0
00057   REAL :: TARRAY(2), RESULT, SYSTDIA, SYSTPURE
00058   CHARACTER(LEN=50) :: FLNM
00059
00060 #ifdef MPI_ON
00061   INTEGER :: IERR, STATUS(mpi_status_size), NUMPROCS
00062
00063   CALL mpi_init( ierr )
00064   CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00065   CALL mpi_comm_size( mpi_comm_world, numprocs, ierr )
00066 #endif
00067
00068   numscf = 0
00069   chempot = zero
00070
00071   ! Start timers
00072   tx = init_timer()
00073   tx = start_timer(latte_timer)
00074   IF (verbose >= 1) CALL timedate_tag("# LATTE started at : ")
00075
00076   INQUIRE( file="latte.in", exist=latteinexists )
00077   IF (latteinexists) THEN
00078     IF (.NOT. libinit) CALL parse_control("latte.in")
00079
00080 #ifdef PROGRESSION
00081     CALL prg_parse_mixer(mx,"latte.in")
00082 #endif
00083
00084   ELSE
00085     CALL readcontrols
00086   ENDIF
00087
00088   CALL readtb
00089
00090   IF (restart .EQ. 0) THEN
00091     CALL readcr
00092   ELSE
00093     CALL readrestart

```

```

00094     ENDIF
00095
00096     IF (ppoton .EQ. 1) CALL readppot
00097     IF (ppoton .EQ. 2) CALL readppottab
00098     IF (ppoton .EQ. 3) CALL readppotspline
00099
00100     IF (debugon .EQ. 1) THEN
00101         CALL plotuniv
00102         IF (ppoton .EQ. 1) CALL plotppot
00103     ENDIF
00104
00105     CALL gethdim
00106
00107     CALL getmatindlist
00108
00109     CALL rhozero
00110
00111     CALL getbndfil()
00112
00113     #ifdef GPUON
00114
00115         CALL initialize( ngpu )
00116
00117     #endif
00118
00119     #ifdef DBCSR_ON
00120
00121
00122         IF (control .EQ. 2 .AND. sparseon .EQ. 1) CALL init_dbcsr
00123
00124     #endif
00125
00126     IF (kbt .LT. 0.0000001 .OR. control .EQ. 2) ente = zero
00127
00128     IF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 0 &
00129         .AND. ppfiton .EQ. 0 .AND. allfiton .EQ. 0) THEN
00130
00131         !
00132         ! Start the timers
00133         !
00134
00135         CALL system_clock(start_clock, clock_rate, clock_max)
00136         CALL dtime(tarray, result)
00137
00138         ! Set up neighbor lists for building the H and pair potentials
00139
00140         CALL allocatenebararrays
00141
00142         IF (electro .EQ. 1) THEN
00143
00144             CALL allocatecoulomb
00145
00146             CALL initcoulomb
00147
00148         ENDIF
00149
00150         IF (basistype .EQ. "NONORTHO") CALL allocatenono
00151
00152         CALL neblists(0)
00153
00154         ! Build the charge independent H matrix
00155
00156         IF (kon .EQ. 0) THEN
00157
00158             IF (sponly .EQ. 0) THEN
00159                 CALL bldnewhs_sp
00160             ELSE
00161                 CALL bldnewhs
00162             ENDIF
00163
00164         ELSE
00165
00166             CALL kbldnewh
00167
00168         ENDIF
00169
00170         !
00171         ! If we're starting from a restart file, we need to modify H such
00172         ! that it agrees with the density matrix elements read from file
00173         !
00174
00175         IF (restart .EQ. 1) CALL ifrestart
00176
00177
00178         !
00179         ! See whether we need spin-dependence too
00180         !

```

```

00181
00182     IF (spinon .EQ. 1) THEN
00183         CALL getdeltaspin
00184         CALL bldspinh
00185     ENDIF
00186
00187     IF (control .EQ. 1) THEN
00188         CALL allocatediag
00189     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00190         CALL allocatpure
00191     ELSEIF (control .EQ. 3) THEN
00192         CALL fermiallocate
00193     ENDIF
00194
00195     IF (control .EQ. 5) THEN
00196
00197         CALL gershgorin
00198         CALL sp2fermiinit
00199
00200     ENDIF
00201
00202     IF (electro .EQ. 0) CALL qneutral(0,1) ! Local charge neutrality
00203
00204     IF (electro .EQ. 1) CALL qconsistency(0,1) ! Self-consistent charges
00205
00206     ! We have to build our NKTOT complex H matrices and compute the
00207     ! self consistent density matrix
00208
00209     ! Tr[rho dH/dR], Pulay force, and Tr[rho H] need to de-orthogonalized rho
00210
00211     IF (kon .EQ. 1) CALL kgetdos
00212
00213     IF (debugon .EQ. 1 .AND. spinon .EQ. 0 .AND. kon .EQ. 0) THEN
00214
00215         print*, "Caution - you're writing to file the density matrix!"
00216
00217         OPEN(unit=31, status="UNKNOWN", file="myrho.dat")
00218
00219         DO i = 1, hdim
00220             WRITE(31,10) (bo(i,j), j = 1, hdim)
00221         ENDDO
00222
00223         CLOSE(31)
00224
00225 10      FORMAT(100g18.8)
00226
00227     ENDIF
00228
00229     IF (compforce .EQ. 1) CALL getforce
00230
00231     erep = zero
00232     IF (ppoton .EQ. 1) THEN
00233         CALL pairpot
00234     ENDIF
00235
00236     IF (ppoton .EQ. 2) THEN
00237         CALL pairpottab
00238     ENDIF
00239
00240     IF (ppoton .EQ. 3) THEN
00241         CALL pairpotspline
00242     ENDIF
00243
00244     CALL toteng
00245
00246     ecoul = zero
00247     IF (electro .EQ. 1) CALL getcoule
00248
00249     espin = zero
00250     IF (spinon .EQ. 1) CALL getspine
00251
00252     IF (control .NE. 1 .AND. control .NE. 2 .AND. kbt .GT. 0.000001 ) THEN
00253
00254     ! We get the entropy automatically when using diagonalization.
00255     ! This is only required when employing the recursive expansion
00256     ! of the Fermi-operator at finite electronic temperature
00257
00258         CALL entropy
00259
00260     ENDIF
00261
00262     CALL wrtrestart(0)
00263
00264     IF (control .EQ. 1) THEN
00265         ! CALL DEALLOCATEDIAG
00266     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00267         CALL deallocatpure

```

```

00268      ELSEIF (control .EQ. 3) THEN
00269          CALL fermideallocate
00270      ENDIF
00271
00272      !
00273      ! Stop the clocks
00274      !
00275
00276      tx = stop_timer(latte_timer)
00277      CALL dtime(tarray, result)
00278      CALL system_clock(stop_clock, clock_rate, clock_max)
00279
00280      CALL getpressure
00281
00282      !      WRITE(6,*) "Force ", FPP(1,1), FPP(2,1), FPP(3,1)
00283      !      PRINT*, "PCHECK ", (1.0/3.0)*(VIRBOND(1)+VIRBOND(2) + VIRBOND(3)), &
00284      !          (1.0/3.0)*(VIRCOUL(1)+VIRCOUL(2) + VIRCOUL(3)), &
00285      !          (1.0/3.0)*(VIRPAIR(1)+VIRPAIR(2) + VIRPAIR(3)), &
00286      !          (1.0/3.0)*(VIRPUL(1)+VIRPUL(2) + VIRPUL(3)), &
00287      !          (1.0/3.0)*(VIRSCOUL(1)+VIRSCOUL(2) + VIRSCOUL(3))
00288
00289      #ifdef DBCSR_ON
00290
00291          IF (control .EQ. 2 .AND. sparseon .EQ. 1 .AND. mynode .EQ. 0) THEN
00292
00293      #endif
00294
00295          IF (myid .EQ. 0) THEN
00296              CALL fittingoutput(0) ! This has to come first (MJC)
00297              CALL summary
00298
00299              !      IF (SPINON .EQ. 0) CALL NORMS
00300
00301              print*, "# System time = ", tarray(1)
00302              print*, "# Wall time = ", float(stop_clock - start_clock)/float(clock_rate)
00303              print*, "# Wall time per SCF =", &
00304                  float(stop_clock - start_clock)/(float(clock_rate)*float(numscf))
00305              !      PRINT*, HDIM, FLOAT(STOP_CLOCK - START_CLOCK)/FLOAT(CLOCK_RATE)
00306              tx = timer_results()
00307              print*, "# NUMSCF = ", numscf
00308
00309          ENDIF
00310      #ifdef DBCSR_ON
00311
00312          ENDIF
00313
00314      #endif
00315
00316
00317      !      CALL ASSESSOCC
00318
00319      IF (electro .EQ. 1) CALL deallocatecoulomb
00320
00321      IF (basistype .EQ. "NONORTHO") CALL deallocateenono
00322
00323      CALL deallocatearrays
00324
00325      ELSEIF (mdon .EQ. 1 .AND. relaxme .EQ. 0) THEN
00326
00327          IF (basistype .EQ. "NONORTHO") CALL allocateenono
00328
00329          IF (xboon .EQ. 1) CALL allocatexbo
00330
00331          IF (electro .EQ. 1) THEN
00332              CALL allocatecoulomb
00333              CALL initcoulomb
00334          ENDIF
00335
00336          ! Start the timers
00337
00338          CALL system_clock(start_clock, clock_rate, clock_max)
00339          CALL dtime(tarray, result)
00340
00341          !
00342          ! Call TBMD
00343          !
00344
00345          CALL tbmd
00346
00347
00348      #ifdef MPI_ON
00349          IF (parrep .EQ. 1) CALL mpi_barrier (mpi_comm_world, ierr )
00350      #endif
00351
00352
00353      ! Stop the timers
00354

```

```

00355     CALL dtime(tarray, result)
00356     CALL system_clock(stop_clock, clock_rate, clock_max)
00357
00358     CALL summary
00359
00360     IF (pbcon .EQ. 0) CLOSE(23)
00361
00362     IF (basistype .EQ. "NONORTHO") CALL deallocate_nono
00363
00364     IF (xboon .EQ. 1) CALL deallocate_xbo
00365
00366     IF (electro .EQ. 1) CALL deallocate_coulomb
00367
00368     !      SYSTPURE = TARRAY(1)
00369     !      WRITE(6, '( "# System time for MD run = ", F12.2, " s")') SYSTPURE
00370     WRITE(6, '( "# Wall time for MD run = ", F12.2, " s")') &
00371         float(stop_clock - start_clock)/float(clock_rate)
00372
00373     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 1) THEN
00374
00375         CALL msrelax
00376
00377     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 1) THEN
00378
00379         CALL system_clock(start_clock, clock_rate, clock_max)
00380
00381         CALL dosfit
00382
00383         CALL system_clock(stop_clock, clock_rate, clock_max)
00384
00385         WRITE(6, '( "# Wall time = ", F12.2, " s")') &
00386             float(stop_clock - start_clock)/float(clock_rate)
00387
00388     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 2) THEN
00389
00390
00391         CALL mofit
00392
00393     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 3) THEN
00394
00395         CALL mofitplato
00396
00397     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. ppfiton .EQ. 1) THEN
00398
00399         CALL ppfit
00400
00401     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. allfiton .EQ. 1) THEN
00402
00403         CALL allfit
00404
00405     ELSE
00406
00407         CALL errors("latte", "You can't have RELAXME = 1 and MDON = 1")
00408
00409     ENDIF
00410
00411 #ifdef GPUON
00412
00413     CALL shutdown()
00414
00415 #endif
00416
00417
00418     CALL deallocate_all
00419
00420 #ifdef DBCSR_ON
00421
00422     !ends mpi
00423
00424     IF (control .EQ. 2 .AND. sparseon .EQ. 1) CALL shutdown_dbcsr
00425
00426 #endif
00427
00428     ! Done with timers
00429     tx = stop_timer(latte_timer)
00430     tx = timer_results()
00431     tx = shutdown_timer()
00432
00433 #ifdef MPI_ON
00434     CALL mpi_finalize(ierr)
00435 #endif
00436
00437 END PROGRAM latte

```


8.237 latte_c_bind.f90 File Reference

Functions/Subroutines

- integer(c_int) function [latte_c_abiversion](#) ()
Function for Latte-C++ interfacing.
- subroutine [latte_c_bind](#) (FLAGS, NATS, COORDS, TYPES, NTYPES, MASSES, XLO, XHI, XY, XZ, YZ, FORCES, MAXITER, VENERG, VEL, DT, VIRIAL_INOUT, NEWSYSTEM, EXISTERROR)
Subroutine for Latte-C++ interfacing.

8.237.1 Function/Subroutine Documentation

8.237.1.1 integer(c_int) function latte_c_abiversion ()

Function for Latte-C++ interfacing.

Returns

ABIVERSION integer representing the date of the last change to the C/C++ interface (e.g. 20180221)

This function will be used prior to calling the LATTE library to allow the calling code to ensure the linked library version is compatible.

Definition at line 96 of file [latte_c_bind.f90](#).

8.237.1.2 subroutine [latte_c_bind](#) (integer(c_int), dimension(5) *FLAGS*, integer(c_int) *NATS*, real(c_double), dimension(3,nats) *COORDS*, integer(c_int), dimension(nats) *TYPES*, integer(c_int) *NTYPES*, real(c_double), dimension(ntypes) *MASSES*, real(c_double), dimension(3) *XLO*, real(c_double), dimension(3) *XHI*, real(c_double) *XY*, real(c_double) *XZ*, real(c_double) *YZ*, real(c_double), dimension(3, nats), intent(inout) *FORCES*, integer(c_int) *MAXITER*, real(c_double) *VENERG*, real(c_double), dimension(3, nats), intent(inout) *VEL*, real(c_double) *DT*, real(c_double), dimension(6), intent(inout) *VIRIAL_INOUT*, integer(c_int), intent(inout) *NEWSYSTEM*, logical(c_bool) *EXISTERROR*)

Subroutine for Latte-C++ interfacing.

Parameters

<i>FLAGS</i>	Different control flags that can be passed to LATTE (not in use yet)
<i>NATS</i>	Number of atoms
<i>COORDS</i>	Coordinates. Example: y-coordinate of atom 1 = COORDS(2,1)
<i>TYPES</i>	An index for all the different atoms in the system.
<i>NTYPES</i>	Number of different elements in the system
<i>MASSES</i>	Element masses for every different element of the system.
<i>XLO</i>	Lowest dimensions of the box
<i>XHI</i>	Highest dimensions of the box
<i>XY</i>	Tilt factor.
<i>XZ</i>	Tilt factor.
<i>YZ</i>	Tilt factor. The lattice vectors are constructed as: a = (xhi-xlo,0,0); b = (xy,yhi-ylo,0); c = (xz,yz,zhi-zlo).
<i>FORCES</i>	Forces for every atom as output.

Parameters

<i>MAXITER</i>	Latte MAXITER keyword. If MAXITER = -1, only the Forces are computed. If MAXITER = 0, MAXITER is read from latte.in file. IF MAXITER > 0, MAXITER is passed through the library call.
<i>VENERG</i>	This is the potential Energy that is given back from latte to the hosting code.
<i>VEL</i>	Velocities passed to latte.
<i>DT</i>	integration step passed to latte.
<i>DT</i>	integration step passed to latte.
<i>VIRIAL_INOUT</i>	Components of the second virial coefficient
<i>NEWSYSTEM</i>	Tells LATTE if a new system is passed.
<i>EXISTERROR</i>	Returns an error flag (.true.) to the hosting code.

This routine will be used load call [latte_lib](#) from a C/C++ program:

Note: To get the mass of atom 3 we do:

```
MASS (TYPES (3))
```

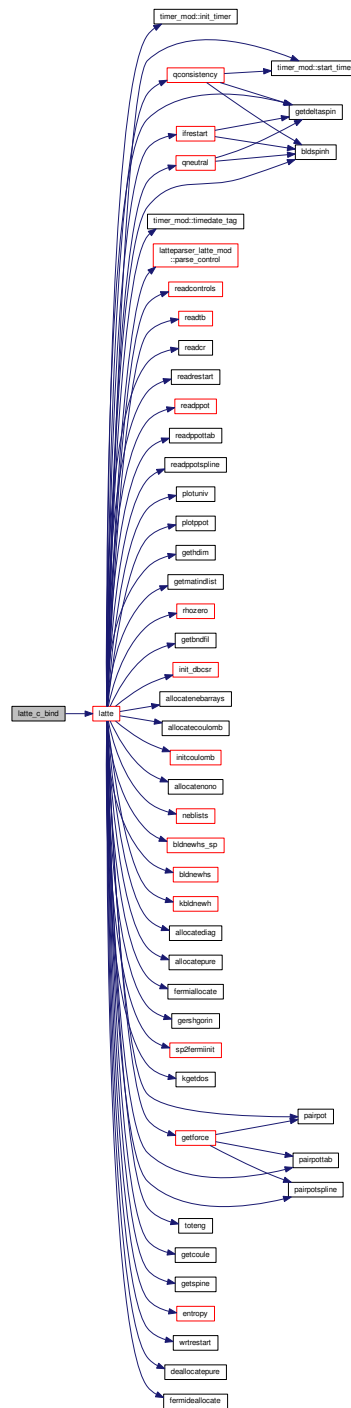
Note: To get the lattice vectors as formatted in LATTE we do:

```
BOX (1,1) = XHI (1) - XLO (1); ...
```

Note: All units are LATTE units by default. See https://github.com/losalamos/LATTE/blob/master/Manual/LATTE_manual.pdf

Definition at line 65 of file [latte_c_bind.f90](#).

Here is the call graph for this function:



8.238 latte_c_bind.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00062 SUBROUTINE latte_c_bind (FLAGS, NATS, COORDS, TYPES, NTYPES, MASSES, XLO &
00063     , xhi, xy, xz, yz, forces, maxiter, venerg, &
00064     vel, dt, virial_inout, newsystem, existerror) bind(c, name="latte")
00065
00066 USE iso_c_binding, ONLY: c_char, c_null_char, c_double, c_int, c_bool
00067 USE latte_lib
00068
00069 IMPLICIT NONE
00070
00071 INTEGER(C_INT)           :: NATS, NTYPES, MAXITER
00072 INTEGER(C_INT)           :: TYPES(nats), FLAGS(5)
00073 REAL(C_DOUBLE)           :: COORDS(3,nats), MASSES(ntypes), XHI(3)
00074 REAL(C_DOUBLE)           :: XLO(3), EKIN, VENERG, DT
00075 REAL(C_DOUBLE)           :: XY, XZ, YZ
00076 REAL(C_DOUBLE), INTENT(INOUT) :: FORCES(3, nats), VEL(3, nats)
00077 REAL(C_DOUBLE), INTENT(INOUT) :: VIRIAL_INOUT(6)
00078 LOGICAL(C_BOOL)          :: EXISTERROR
00079 INTEGER(C_INT), INTENT(INOUT) :: NEWSYSTEM
00080
00081 CALL latte(ntypes, types, coords, masses, xlo, xhi, xy, xz, yz, forces, &
00082     maxiter, venerg, vel, dt, virial_inout, newsystem, existerror)
00083
00084 RETURN
00085
00086 END SUBROUTINE latte_c_bind
00087
00095 INTEGER(C_INT) FUNCTION latte_c_abiversion() BIND (C, NAME="latte_abiversion")
00096 USE iso_c_binding, ONLY: c_int
00097 USE latte_lib, ONLY: latte_abiversion
00098 IMPLICIT NONE
00099
00100 latte_c_abiversion = latte_abiversion
00101 RETURN
00102
00103 END FUNCTION latte_c_abiversion

```

8.239 latte_lib.f90 File Reference

Modules

- module [latte_lib](#)

Functions/Subroutines

- subroutine, public [latte_lib::latte](#) (NTYPES, TYPES, CR_IN, MASSES_IN, XLO, XHI, XY, XZ, YZ, FTOT_OUT, MAXITER_IN, VENERG, VEL_IN, DT_IN, VIRIAL_INOUT, NEWSYSTEM, EXISTERROR_INOUT)

Variables

- integer, parameter, public [latte_lib::latte_abiversion](#) = 20180207

8.240 latte_lib.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE latte_lib
00023
00024   USE constants_mod
00025   USE timer_mod
00026   USE setuparray
00027   USE ppotarray
00028   USE purearray
00029   USE coulombarray
00030   USE spinarray
00031   USE sparsearray
00032   USE mdarray
00033   USE myprecision
00034   USE virialarray
00035   USE diagarray
00036   USE kspacearray
00037   USE latteparser_latte_mod
00038   USE neblistarray
00039   USE nonoarray
00040   USE constraints_mod
00041
00042   #ifdef PROGRESSON
00043     USE prg_system_mod ! FROM PROGRESS
00044     USE prg_pulaymixer_mod
00045     USE prg_extras_mod
00046     USE mixer_mod
00047     USE bml
00048   #endif
00049
00050   #ifdef MPI_ON
00051     USE mpi
00052   #endif
00053   #ifdef DBCSR_ON
00054     USE dbcsr_var_mod
00055   #endif
00056
00057   IMPLICIT NONE
00058
00059   PRIVATE
00060
00061   ! Defines the version of the binary interface.
00062   ! Adjust if non-backward compatible changes are made to the interface.
00063   ! Use in codes using the library interface to make certain a compatible
00064   ! version of the LATTE library is used.
00065   INTEGER, PARAMETER :: latte_abiversion = 20180207
00066
00067   PUBLIC :: latte, latte_abiversion
00068
00069 CONTAINS
00070
00071   !! Latte subroutine
00072   !! \param FLAGS Different control flags that can be passed to LATTE (not in use yet)
00073   !! \param NATS Number of atoms
00074   !! \param COORDS Coordinates. Example: y-coordinate of atom 1 = COORDS(2,1)
00075   !! \param TYPES An index for all the different atoms in the system.
00076   !! \param NTYPES Number of different elements in the system
00077   !! \param MASSES Element masses for every different element of the system.
00078   !! \param XLO Lowest dimensions of the box
00079   !! \param XHI Highest dimensions of the box
00080   !! \param XY Tilt factor.
00081   !! \param XZ Tilt factor.
00082   !! \param YZ Tilt factor. The lattice vectors are constructed as:
00083   !! a = (xhi-xlo,0,0); b = (xy,yhi-ylo,0); c = (xz,yz,zhi-zlo).
00084   !! \param FORCES Forces for every atom as output.

```

```

00085  !! \param MAXITER Latte MAXITER keyword. If MAXITER = -1, only the Forces are computed.
00086  !!      If MAXITER = 0, MAXITER is read from latte.in file.
00087  !!      IF MAXITER > 0, MAXITER is passed through the library call.
00088  !! \param VENERG This is the potential Energy that is given back from latte to the hosting code.
00089  !! \param VEL Velocities passed to latte.
00090  !! \param DT integration step passed to latte.
00091  !! \param VIRIAL_INOUT Components of the second virial coefficient.
00092  !! \param NEWSYSTEM Tells LATTE if a new system is passed.
00093  !! \param EXISTERROR_INOUT Returns an error flag (.true.) to the hosting code.
00094  !!
00095  !! \brief This routine will be used load call latte_lib from a C/C++ program:
00096  !!
00097  !! \brief Note: To get the mass of atom 3 we do:
00098  !! \verbatim
00099  !!      MASS(TYPES(3))
00100  !! \endverbatim
00101  !!
00102  !! \brief Note: To get the lattice vectors as formatted in LATTE we do:
00103  !! \verbatim
00104  !!      BOX(1,1) = XHI(1) - XLO(1); ...
00105  !! \endverbatim
00106  !!
00107  !! \brief Note: All units are LATTE units by default. See
00108  !! https://github.com/losalamos/LATTE/blob/master/Manual/LATTE\_manual.pdf
00109  !!
00110  SUBROUTINE latte(NTYPES, TYPES, CR_IN, MASSES_IN, XLO, XHI, XY, XZ, YZ, FTOT_OUT, &
00111  maxiter_in, venerg, vel_in, dt_in, virial_inout, newsystem, existerror_inout)
00112  USE constants_mod, ONLY: existerror
00113
00114  IMPLICIT NONE
00115
00116  INTEGER :: I, J
00117  INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00118  INTEGER :: MYID = 0
00119  REAL :: TARRAY(2), RESULT, SYSTDIA, SYSTPURE
00120  REAL(LATTEPREC) :: DBOX
00121  CHARACTER(LEN=50) :: FLNM
00122
00123  REAL(LATTEPREC), INTENT(IN) :: CR_IN(:, :), VEL_IN(:, :), MASSES_IN(:, :), XLO(3), XHI(3)
00124  REAL(LATTEPREC), INTENT(IN) :: DT_IN, XY, XZ, YZ
00125  REAL(LATTEPREC), INTENT(OUT) :: FTOT_OUT(:, :), VENERG
00126  REAL(LATTEPREC), INTENT(OUT) :: VIRIAL_INOUT(6)
00127  INTEGER, INTENT(IN) :: NTYPES, TYPES(:), MAXITER_IN
00128  LOGICAL(1), INTENT(INOUT) :: EXISTERROR_INOUT
00129  REAL(LATTEPREC) :: MLSI, LUMO, HOMO
00130  INTEGER, INTENT(INOUT) :: NEWSYSTEM
00131
00132  #ifdef PROGRESSON
00133  TYPE(system_type) :: SY
00134  #endif
00135
00136  #ifdef MPI_ON
00137  INTEGER :: IERR, STATUS(mpi_status_size), NUMPROCS
00138  CALL mpi_init( ierr )
00139  CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00140  CALL mpi_comm_size( mpi_comm_world, numprocs, ierr )
00141  #endif
00142
00143  existerror = .false. !We assume we start the lib call without errors
00144
00145  IF(.NOT. libinit .OR. newsystem == 1) THEN
00146
00147  CALL deallocateall()
00148
00149  librun = .true.
00150
00151  libcalls = 0 ; maxiter = -10
00152
00153  OPEN(unit=6, file="log.latte", form="formatted")
00154
00155  IF(verbose >= 1) THEN
00156  WRITE(*,*) "# The log file for latte_lib"
00157  WRITE(*,*) ""
00158  CALL timedate_tag("LATTE started at : ")
00159  ENDIF
00160
00161  INQUIRE( file="animate/.", exist=latteinexists)
00162  IF (.NOT. latteinexists) CALL system("mkdir animate")
00163
00164  numscf = 0
00165  chempot = zero
00166
00167  ! Start timers
00168  tx = init_timer()
00169  tx = start_timer(latte_timer)
00170

```

```

00171      INQUIRE( file="latte.in", exist=latteinexists )
00172
00173      IF (latteinexists) THEN
00174          CALL parse_control("latte.in")
00175
00176      #ifdef PROGRESSION
00177          CALL prg_parse_mixer(mx,"latte.in")
00178      #endif
00179
00180      ELSE
00181          CALL readcontrols
00182      ENDIF
00183
00184      CALL readtb
00185
00186      IF (restart .EQ. 0) THEN
00187
00188          box = 0.0d0
00189          box(1,1) = xhi(1) - xlo(1)
00190          box(2,1) = xy
00191          box(2,2) = xhi(2) - xlo(2)
00192          box(3,1) = xz
00193          box(3,2) = yz
00194          box(3,3) = xhi(3) - xlo(3)
00195
00196          IF(verbose >= 1) THEN
00197              WRITE(*,*) "Lattice vectors:"
00198              WRITE(*,*) "a=",box(1,1),box(1,2),box(1,3)
00199              WRITE(*,*) "b=",box(2,1),box(2,2),box(2,3)
00200              WRITE(*,*) "c=",box(3,1),box(3,2),box(3,3)
00201              WRITE(*,*) ""
00202          ENDIF
00203
00204          box_old = box
00205
00206          nats = SIZE(cr_in,dim=2)
00207
00208          IF (.NOT.ALLOCATED(cr)) ALLOCATE(cr(3,nats))
00209          cr = cr_in
00210
00211          IF(verbose >= 1)WRITE(*,*) "Converting masses to symbols ..."
00212          IF(.NOT. ALLOCATED(atele)) ALLOCATE(atele(nats))
00213          CALL masses2symbols(types,ntypes,masses_in,nats,atele)
00214
00215          !Forces, charges and element pointers are allocated in reader
00216          CALL readcr
00217
00218          CALL flush(6)
00219
00220      ELSE
00221
00222          IF(verbose >= 1)WRITE(*,*) "Restarting calculation from file ..."
00223          CALL readrestart
00224
00225      ENDIF
00226
00227      IF (verbose >= 1) WRITE(*,*) "Reading ppots from file (if PPOTON >= 1) ..."
00228      IF (ppoton .EQ. 1) CALL readppot
00229      IF (ppoton .EQ. 2) CALL readppottab
00230      IF (ppoton .EQ. 3) CALL readppotspline
00231
00232      IF (debugon .EQ. 1) THEN
00233          CALL plotuniv
00234          IF (ppoton .EQ. 1) CALL plotppot
00235      ENDIF
00236
00237      CALL gethdim
00238
00239      CALL getmatindlist
00240
00241      IF (verbose >= 1) WRITE(*,*) "Getting rho0 ..."
00242      CALL rhozero
00243
00244      CALL getbndfil()
00245
00246      CALL flush(6)
00247
00248      #ifdef GPUON
00249          CALL initialize( ngpu )
00250      #endif
00251
00252      #ifdef DBCSR_ON
00253
00254          IF (control .EQ. 2 .AND. sparseon .EQ. 1) CALL init_dbcsr
00255
00256      #endif
00257

```

```

00258 #endif
00259
00260     IF (kbt .LT. 0.0000001 .OR. control .EQ. 2) ente = zero
00261
00262     IF (.NOT. ALLOCATED(v)) ALLOCATE(v(3,nats))
00263     IF(verbose >= 1)WRITE(*,*)"End of INITIALIZATION"
00264
00265     ELSE
00266
00267         box = 0.0d0
00268         box(1,1) = xhi(1) - xlo(1)
00269         box(2,1) = xy
00270         box(2,2) = xhi(2) - xlo(2)
00271         box(3,1) = xz
00272         box(3,2) = yz
00273         box(3,3) = xhi(3) - xlo(3)
00274
00275         IF(verbose >= 1)THEN
00276             WRITE(*,*)"Lattice vectors:"
00277             WRITE(*,*)"a=",box(1,1),box(1,2),box(1,3)
00278             WRITE(*,*)"b=",box(2,1),box(2,2),box(2,3)
00279             WRITE(*,*)"c=",box(3,1),box(3,2),box(3,3)
00280             WRITE(*,*)" "
00281         ENDIF
00282
00283         libcalls = libcalls + 1
00284
00285         nats = SIZE(cr_in,dim=2)
00286
00287         IF(.NOT.ALLOCATED(cr)) ALLOCATE(cr(3,nats))
00288         cr = cr_in
00289
00290         CALL flush(6)
00291
00292     ENDIF
00293     !End of initialization
00294
00295
00296     IF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 0 &
00297         .AND. ppfiton .EQ. 0 .AND. allfiton .EQ. 0) THEN
00298
00299
00300         ! IF (.NOT. LIBINIT) THEN
00301         !
00302         ! Start the timers
00303         !
00304
00305
00306         CALL system_clock(start_clock, clock_rate, clock_max)
00307         CALL dtimer(tarray, result)
00308
00309
00310
00311         ! Set up neighbor lists for building the H and pair potentials
00312
00313         CALL allocatenebarrays
00314
00315
00316         IF (electro .EQ. 1) THEN
00317
00318             CALL allocatcoulomb
00319
00320             CALL initcoulomb
00321
00322         ENDIF
00323
00324
00325         IF (basistype .EQ. "NONORTHO") CALL allocatenono
00326
00327
00328
00329         CALL neblists(0)
00330
00331
00332         ! Build the charge independent H matrix
00333
00334         IF (kon .EQ. 0) THEN
00335
00336             IF (sponly .EQ. 0) THEN
00337                 CALL bldnewhs_sp
00338             ELSE
00339                 CALL bldnewhs
00340             ENDIF
00341
00342         ELSE
00343
00344             CALL kbldnewh

```



```

00345
00346     ENDIF
00347
00348     !
00349     ! If we're starting from a restart file, we need to modify H such
00350     ! that it agrees with the density matrix elements read from file
00351     !
00352
00353     IF (restart .EQ. 1) CALL ifrestart
00354
00355     !
00356     ! See whether we need spin-dependence too
00357     !
00358
00359     IF (spinon .EQ. 1) THEN
00360         CALL getdeltaspin
00361         CALL bldspinh
00362     ENDIF
00363
00364     IF (control .EQ. 1) THEN
00365         CALL allocatediag
00366     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00367         CALL allocatepure
00368     ELSEIF (control .EQ. 3) THEN
00369         CALL fermiallocate
00370     ENDIF
00371
00372     ! ELSE
00373     !     CALL ERRORS("latte_lib", "Attempting to perform multiple single point calculations. &
00374     !         & MDON .EQ. 0 .AND. RELAXME .EQ. 0 can be done &
00375     !         &for only one geometry (A single call to the library). Please reduce &
00376     !         &the number of steps (md of relaxation) at the host code")
00377     !     EXISTERROR_INOUT = EXISTERROR
00378     !     RETURN
00379     ! ENDIF
00380
00381     IF (control .EQ. 5) THEN
00382
00383         CALL gershgorin
00384         CALL sp2fermiinit
00385
00386     ENDIF
00387
00388     IF (electro .EQ. 0) CALL qneutral(0,1) ! Local charge neutrality
00389
00390     IF (electro .EQ. 1) CALL qconsistency(0,1) ! Self-consistent charges
00391
00392     ! We have to build our NKTOT complex H matrices and compute the
00393     ! self consistent density matrix
00394
00395     ! Tr[rho dH/dR], Pulay force, and Tr[rho H] need to de-orthogonalized rho
00396
00397     IF (kon .EQ. 1) CALL kgetdos
00398
00399     ! OPEN(UNIT=31, STATUS="UNKNOWN", FILE="myrho.dat")
00400
00401     ! DO I = 1, HDIM
00402     !     DO J = 1, HDIM
00403
00404         !         IF (ABS(BO(J,I)) .GT. 1.0D-5) WRITE(31,99) I, J
00405
00406     !     ENDDO
00407     ! ENDDO
00408
00409     !99 FORMAT(2I9)
00410     !CLOSE(31)
00411
00412     IF (debugon .EQ. 1 .AND. spinon .EQ. 0 .AND. kon .EQ. 0) THEN
00413         print*, "Caution - you're writing to file the density matrix!"
00414
00415         OPEN(unit=31, status="UNKNOWN", file="myrho.dat")
00416
00417         DO i = 1, hdim
00418             WRITE(31,10) (bo(i,j), j = 1, hdim)
00419         ENDDO
00420
00421         CLOSE(31)
00422
00423         10     FORMAT(100g18.8)
00424
00425     ENDIF
00426
00427     ftot = zero
00428
00429
00430     IF (compforce .EQ. 1) THEN
00431

```

```

00432         IF (kon .EQ. 0) THEN
00433
00434             IF (sponly .EQ. 0) THEN
00435                 CALL gradhsp
00436             ELSE
00437                 CALL gradh
00438             ENDIF
00439
00440         ELSE
00441             CALL kgradh
00442         ENDIF
00443
00444         ftot = two * f
00445
00446     ENDIF
00447
00448     erep = zero
00449     IF (ppoton .EQ. 1) THEN
00450         CALL pairpot
00451         ftot = ftot + fpp
00452     ENDIF
00453
00454     IF (ppoton .EQ. 2) THEN
00455         CALL pairpottab
00456         ftot = ftot + fpp
00457     ENDIF
00458
00459     IF (ppoton .EQ. 3) THEN
00460         CALL pairpotspline
00461         ftot = ftot + fpp
00462     ENDIF
00463
00464     IF (electro .EQ. 1) ftot = ftot + fcoul
00465
00466     IF (basistype .EQ. "NONORTHO") THEN
00467
00468         IF (sponly .EQ. 0) THEN
00469             ! s/sp orbitals only so we can use the analytic code
00470             CALL fcoulnono_sp
00471             CALL pulay_sp
00472             IF (spinon .EQ. 1) CALL fspinnono_sp
00473         ELSE
00474             ! Otherwise use the complex but general expansions Josh
00475             ! Coe implemented
00476             CALL fcoulnono
00477             CALL pulay
00478             IF (spinon .EQ. 1) CALL fspinnono
00479         ENDIF
00480
00481         ftot = ftot - two*fpul + fscoul
00482
00483         IF (spinon .EQ. 1) ftot = ftot + fsspin
00484
00485     ENDIF
00486
00487     CALL toteng
00488
00489     ecoul = zero
00490     IF (electro .EQ. 1) CALL getcoule
00491
00492     espin = zero
00493     IF (spinon .EQ. 1) CALL getspine
00494
00495     IF (control .NE. 1 .AND. control .NE. 2 .AND. kbt .GT. 0.000001 ) THEN
00496
00497         ! We get the entropy automatically when using diagonalization.
00498         ! This is only required when employing the recursive expansion
00499         ! of the Fermi-operator at finite electronic temperature
00500
00501         CALL entropy
00502
00503     ENDIF
00504
00505     CALL wrtrestart(0)
00506
00507     IF (control .EQ. 1) THEN
00508         ! CALL DEALLOCATEDIAG
00509     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00510         CALL deallocatepure
00511     ELSEIF (control .EQ. 3) THEN
00512         CALL fermideallocate
00513     ENDIF
00514
00515     !
00516     ! Stop the clocks
00517     !
00518

```

```

00519         tx = stop_timer(latte_timer)
00520         CALL dtimer(tarray, result)
00521         CALL system_clock(stop_clock, clock_rate, clock_max)
00522
00523         CALL getpressure
00524
00525         ! WRITE(*,*) "Force", FPP(1,1), FPP(2,1), FPP(3,1)
00526         ! PRINT*, "PCHECK ", (1.0/3.0)*(VIRBOND(1)+VIRBOND(2) + VIRBOND(3)), &
00527         !       (1.0/3.0)*(VIRCOUL(1)+VIRCOUL(2) + VIRCOUL(3)), &
00528         !       (1.0/3.0)*(VIRPAIR(1)+VIRPAIR(2) + VIRPAIR(3)), &
00529         !       (1.0/3.0)*(VIRPUL(1)+VIRPUL(2) + VIRPUL(3)), &
00530         !       (1.0/3.0)*(VIRSCOUL(1)+VIRSCOUL(2) + VIRSCOUL(3))
00531
00532 #ifdef DBCSR_ON
00533
00534         IF (control .EQ. 2 .AND. sparseon .EQ. 1 .AND. mynode .EQ. 0) THEN
00535
00536 #endif
00537
00538         IF (myid .EQ. 0) THEN
00539             CALL fittingoutput(0)
00540             CALL summary
00541
00542             ! IF (SPINON .EQ. 0) CALL NORMS
00543
00544             print*, "# System time = ", tarray(1)
00545             print*, "# Wall time = ", float(stop_clock - start_clock)/float(clock_rate)
00546             print*, "# Wall time per SCF =", &
00547                   float(stop_clock - start_clock)/(float(clock_rate)*float(numscf))
00548             ! PRINT*, HDIM, FLOAT(STOP_CLOCK - START_CLOCK)/FLOAT(CLOCK_RATE)
00549             tx = timer_results()
00550             print*, "# NUMSCF = ", numscf
00551
00552         ENDIF
00553 #ifdef DBCSR_ON
00554
00555         ENDIF
00556
00557 #endif
00558
00559         ! CALL ASSESSOCC
00560
00561         IF (electro .EQ. 1) CALL deallocatecoulomb
00562
00563         IF (basistype .EQ. "NONORTHO") CALL deallocatenono
00564
00565         CALL deallocatenebarrays
00566
00567         CALL deallocateall
00568
00569         libinit = .false.
00570
00571         RETURN
00572
00573     ELSEIF (mdon .EQ. 1 .AND. relaxme .EQ. 0 .AND. maxiter_in < 0 ) THEN
00574
00575         IF(verbose >= 1)WRITE(*,*)"Insie MDON= 1 and RELAXME= 0 ..."
00576
00577         dt = dt_in ! Get the integration step from the hosting code.
00578
00579         v = vel_in/1000.0d0 !Convert from Ang/ps to Ang/fs
00580
00581         !Control for implicit geometry optimization.
00582         !This will need to be replaced by a proper flag.
00583         IF (dt_in == 0) THEN
00584             IF (verbose >= 1) WRITE(*,*)"NOTE: DT = 0 => FULLQCONV = 1"
00585             IF (verbose >= 1) WRITE(*,*)"NOTE: DT = 0 => MDMIX = QMIX"
00586             fullqconv = 1
00587             mdmix = qmix
00588             CALL flush(6)
00589         ENDIF
00590
00591         IF (libcalls == 0) THEN
00592
00593             IF (verbose >= 1)WRITE(*,*)"Allocating nonorthogonal arrays ..."
00594             IF (basistype .EQ. "NONORTHO") CALL allocatenono
00595
00596             IF (verbose >= 1)WRITE(*,*)"Allocating XLBO arrays ..."
00597             IF (xboon .EQ. 1) CALL allocatexbo
00598
00599             IF (verbose >= 1)WRITE(*,*)"Allocating COULOMB arrays ..."
00600             IF (electro .EQ. 1) THEN
00601                 CALL allocatcoulomb
00602                 CALL initcoulomb
00603             ENDIF
00604
00605             ! Start the timers

```

```

00606         IF (verbose >= 1)WRITE(*,*)"Starting timers ..."
00607         CALL system_clock(start_clock, clock_rate, clock_max)
00608         CALL dtimetime(tarray, result)
00609
00610         IF (verbose >= 1)WRITE(*,*)"Setting up TBMD ..."
00611         CALL setup_tmbd(newsystem)
00612
00613         CALL flush(6)
00614
00615     ELSEIF (libcalls > 0 .AND. restartlib == 0) THEN
00616
00617         dbox = sqrt((box(1,1)-box_old(1,1))**2 + &
00618             & (box(2,2)-box_old(2,2))**2 + (box(3,3)-box_old(3,3))**2)
00619
00620         ! Reinitializing Coulombic contribution if Kpoints are used
00621         IF ((electro .EQ. 1 .AND. elecmeth .EQ. 0) .OR. dbox .GT. 0.0d0) THEN
00622             CALL initcoulomb
00623         ENDIF
00624
00625         IF (mod(libcalls, udneigh) .EQ. 0) THEN
00626             !If box is changing
00627             IF (dbox .GT. 0.0d0) THEN
00628                 CALL neblists(0)
00629                 CALL initcoulomb !If the box is changing we need to recompute the kspace list
00630             ELSE
00631                 CALL neblists(1)
00632             ENDIF
00633         ENDIF
00634
00635         box_old = box
00636
00637     ENDIF
00638
00639     IF (qiter .NE. 0) THEN
00640         ecoul = zero
00641         IF (electro .EQ. 1) CALL getcoule
00642     ENDIF
00643
00644     IF(verbose >= 1) WRITE(*,*)"LIBCALLS",libcalls
00645
00646     IF(verbose >= 1) mlsi = time_mls()
00647     IF(libcalls > 0) CALL getmdf(1, libcalls)
00648     IF(verbose >= 1) WRITE(*,*)"Time for GETMDF =", time_mls()-msli
00649
00650     CALL toteng
00651
00652     ! For the 0 SCF MD the coulomb energy is calculated in GETMDF
00653
00654     IF (ppoton .EQ. 1) THEN
00655         CALL pairpot
00656     ELSEIF (ppoton .EQ. 2) THEN
00657         CALL pairpottab
00658     ELSEIF (ppoton .EQ. 3) THEN
00659         CALL pairpotspline
00660     ENDIF
00661
00662     IF (qiter .NE. 0) THEN
00663         ecoul = zero
00664         IF (electro .EQ. 1) CALL getcoule
00665     ENDIF
00666
00667     espin = zero
00668     IF (spinon .EQ. 1) CALL getspine
00669
00670     IF (control .NE. 1 .AND. control .NE. 2 .AND. kbt .GT. 0.000001) THEN
00671
00672         ! Only required when using the recursive expansion of the Fermi operator
00673
00674         ! 2/26/13
00675         ! The entropy is now calculated when we get the density
00676         ! matrix in the spin polarized case with diagonalization,
00677         ! as it should be...
00678
00679         CALL entropy
00680
00681     ENDIF
00682
00683     IF(verbose >= 1) WRITE(*,*)"Energy Components (TRRH0H, EREP, ENTE, ECOUL)",trrhoh, erep, ente, ecoul
00684
00685     IF (freeze .EQ. 1) CALL freeze_atoms(ftot,v)
00686
00687     IF(maxval(ftot_out) .NE. 0.0d0)THEN
00688         IF(verbose >= 1) WRITE(*,*)"Adding force components and energies from application code ..."
00689         IF(verbose >= 1) WRITE(*,*)"APPCODE,LATTE",venerg,trrhoh + erep - ente - ecoul + espin
00690         venerg = trrhoh + erep - ente - ecoul + espin
00691         ftot_out = ftot_out + ftot
00692     ELSE

```

```

00693         venerg = trrhoh + erep - ente - ecoul + espin
00694         ftot_out = ftot
00695     ENDIF
00696
00697
00698     ! Get the second virial coefficient to pass it to the application program
00699     IF (electro .EQ. 0) vircoul = zero
00700     virial = virbond + virpair + vircoul
00701
00702     IF (spinon .EQ. 1) virial = virial + virsspin
00703
00704     IF (basistype .EQ. "NONORTHO") THEN
00705         virial = virial - virpul + virscoul
00706     ENDIF
00707
00708     !         CALL GETPRESSURE
00709
00710     virial_inout = -virial
00711
00712     libinit = .true.
00713     newsystem = 0 !Setting newsystem back to 0.
00714
00715 #ifdef PROGRESSON
00716     IF (mod(libcalls,wrtfreq) == 0) THEN
00717         IF (verbose >= 1) WRITE(*,*) "Writing trajectory into trajectory.pdb ..."
00718         sy%NATS = nats
00719         IF (.NOT. ALLOCATED(sy%COORDINATE)) ALLOCATE(sy%COORDINATE(3,nats))
00720         sy%COORDINATE = cr
00721         sy%SYMBOL = atele
00722         sy%LATTICE_VECTOR = box
00723         sy%NET_CHARGE = deltaq
00724         CALL prg_write_trajectory(sy,libcalls,wrtfreq,dt_in,"trajectory","pdb")
00725         CALL prg_write_trajectory(sy,libcalls,wrtfreq,dt_in,"trajectory","xyz")
00726     ENDIF
00727 #else
00728     IF (mod(libcalls,wrtfreq) == 0) THEN
00729         IF (verbose >= 1) WRITE(*,*) "Writing trajectory into trajectory.xyz ..."
00730         IF (libcalls .EQ. 0) THEN
00731             OPEN(unit=20,file="trajectory.xyz",status='unknown')
00732         ELSE
00733             OPEN(unit=20,file="trajectory.xyz",access='append',status='old')
00734         ENDIF
00735         !Extended xyz file.
00736         WRITE(20,*) nats
00737         WRITE(20,*) 'Lattice=',box(1,1),box(1,2),box(1,3),&
00738             &box(2,1),box(2,2),box(2,3),box(3,1),box(3,2),box(3,3),' ','&
00739             &"Properties=species:S:1:pos:R:3:vel:R:3:for:R:3:cha:R:1 Time=",libcalls*dt_in
00740         DO i=1,nats
00741             WRITE(20,*) atele(i),cr(1,i),cr(2,i),cr(3,i),v(1,i),v(2,i),
00742                 v(3,i),&
00743                 &ftot(1,i),ftot(2,i),ftot(3,i),-deltaq(i)
00744         ENDDO
00745         CLOSE(20)
00746     ENDIF
00747 #endif
00748
00749     IF (verbose >= 1 .AND. control == 1) THEN
00750         IF (spinon == 0) THEN
00751             homo = evals(floor(bndfil*float(hdim)))
00752             lumo = evals(floor(bndfil*float(hdim))+1)
00753             WRITE(*,*) "HOMO=",homo, "LUMO=",lumo
00754             WRITE(*,*) "EGAP=",lumo - homo
00755         ELSE
00756             homo = max(downevals(floor(bndfil*float(hdim))),upevals(floor(bndfil*float(
00757             hdim))))
00758             lumo = min(downevals(floor(bndfil*float(hdim))+1),upevals(floor(bndfil*float(
00759             hdim))+1))
00760             WRITE(*,*) "HOMO=",homo, "LUMO=",lumo
00761             WRITE(*,*) "EGAP=",lumo - homo
00762         ENDIF
00763     ENDIF
00764
00765     IF (mod(libcalls, rsfreq) .EQ. 0) THEN
00766         CALL wrtrestartlib(libcalls)
00767     ENDIF
00768
00769     IF (restartlib == 1 .AND. libcalls == 0) THEN
00770         CALL readrestartlib(libcalls)
00771     ENDIF
00772
00773 #ifdef PROGRESSON
00774     ! IF (VERBOSE >= 1) THEN
00775     !     CALL GETDIPOLE(DIPOLEMAG,DIPOLEVEECOUT=DIPOLEVEECOUT)
00776     !     WRITE(*,*) "Dipole Magnitude = DIPOLEMAG"
00777     !     WRITE(*,*) "Dipole Vector:"
00778     !     WRITE(*,*) DIPOLEVEECOUT(1),DIPOLEVEECOUT(2),DIPOLEVEECOUT(3)
00779     ! ENDIF

```

```

00777 #endif
00778
00779     CALL flush(6) !To force writing to file at every call
00780
00781     existerror_inout = existerror
00782
00783     RETURN
00784
00785     ELSEIF (mdon .EQ. 1 .AND. relaxme .EQ. 0 .AND. maxiter_in >= 0) THEN
00786
00787         IF (basistype .EQ. "NONORTHO") CALL allocatenono
00788
00789         IF (xboon .EQ. 1) CALL allocatexbo
00790
00791         IF (electro .EQ. 1) THEN
00792             CALL allocatcoulomb
00793             CALL initcoulomb
00794         ENDIF
00795
00796         ! Start the timers
00797
00798         CALL system_clock(start_clock, clock_rate, clock_max)
00799         CALL dtimer(tarray, result)
00800
00801         !
00802         ! Call TBMD
00803         !
00804
00805         CALL tbmd
00806
00807 #ifdef MPI_ON
00808     IF (parrep .EQ. 1) CALL mpi_barrier (mpi_comm_world, ierr )
00809 #endif
00810
00811     ! Stop the timers
00812
00813     CALL dtimer(tarray, result)
00814     CALL system_clock(stop_clock, clock_rate, clock_max)
00815
00816     CALL summary
00817
00818     IF (pbcon .EQ. 0) CLOSE(23)
00819
00820     IF (basistype .EQ. "NONORTHO") CALL deallocatenono
00821
00822     IF (xboon .EQ. 1) CALL deallocatexbo
00823
00824     IF (electro .EQ. 1) CALL deallocatcoulomb
00825
00826     !     SYSTPURE = TARRAY(1)
00827     !     WRITE(6, '( "# System time for MD run = ", F12.2, " s") ') SYSTPURE
00828     WRITE(6, '( "# Wall time for MD run = ", F12.2, " s") ') &
00829         float(stop_clock - start_clock)/float(clock_rate)
00830
00831
00832     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 1) THEN
00833
00834         CALL errors("latte_lib","This option was not tested for the &
00835             &library version of LATTE: RELAXME= 1")
00836         RETURN
00837
00838         CALL msrelax
00839
00840     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 1) THEN
00841
00842         CALL errors("latte_lib","This option was not tested for the &
00843             &library version of LATTE: DOSFITON= 1")
00844         RETURN
00845
00846         CALL system_clock(start_clock, clock_rate, clock_max)
00847
00848         CALL dosfit
00849
00850         CALL system_clock(stop_clock, clock_rate, clock_max)
00851
00852         WRITE(6, '( "# Wall time = ", F12.2, " s") ') &
00853             float(stop_clock - start_clock)/float(clock_rate)
00854
00855     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 2) THEN
00856
00857         CALL errors("latte_lib","This option was not tested for the &
00858             &library version of LATTE: DOSFITON= 3")
00859         RETURN
00860
00861         CALL mofit
00862
00863     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. dosfiton .EQ. 3) THEN

```

```

00864
00865     CALL errors("latte_lib","This option was not tested for the &
00866               &library version of LATTE: DOSFITON= 3")
00867     RETURN
00868
00869     CALL mofitplato
00870
00871     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. ppfiton .EQ. 1) THEN
00872
00873     CALL errors("latte_lib","This option was not tested for the &
00874               &library version of LATTE: PPFITON= 1")
00875     RETURN
00876
00877     CALL ppfit
00878
00879     ELSEIF (mdon .EQ. 0 .AND. relaxme .EQ. 0 .AND. allfiton .EQ. 1) THEN
00880
00881     CALL errors("latte_lib","This option was not tested for the &
00882               &library version of LATTE: ALLFITON= 1")
00883     RETURN
00884
00885     CALL allfit
00886
00887     ELSE
00888
00889     CALL errors("latte_lib","You can't have RELAXME = 1 and MDON = 1")
00890
00891     ENDIF
00892
00893 #ifdef GPUON
00894
00895     CALL shutdown()
00896
00897 #endif
00898
00899
00900     CALL deallocateall
00901
00902 #ifdef DBCSR_ON
00903
00904     !ends mpi
00905
00906     IF (control .EQ. 2 .AND. sparseon .EQ. 1) CALL shutdown_dbcsr
00907
00908 #endif
00909
00910     ! Done with timers
00911     tx = shutdown_timer()
00912
00913 #ifdef MPI_ON
00914     CALL mpi_finalize( ierr )
00915 #endif
00916
00917     libinit = .true.
00918     newsystem = 0 !Setting newsystem back to 0.
00919     existerror_inout = existerror
00920
00921 END SUBROUTINE latte
00922
00923 END MODULE latte_lib

```

8.241 latteparser_latte_mod.f90 File Reference

Data Types

- type [latteparser_latte_mod::latte_type](#)
General latte input variables type.

Modules

- module [latteparser_latte_mod](#)
LATTE parser.

Functions/Subroutines

- subroutine, public [latteparser_latte_mod::parse_control](#) (FILENAME)
The parser for Latte General input variables.
- subroutine, public [latteparser_latte_mod::parse_kmesh](#) (FILENAME)
The parser for K Mesh input variables.
- subroutine, public [latteparser_latte_mod::parse_md](#) (FILENAME)
The parser for Latte General input variables.

Variables

- type(latte_type), public [latteparser_latte_mod::lt](#)

8.242 latteparser_latte_mod.f90

```

00001
00012 MODULE latteparser_latte_mod
00013
00014     USE constants_mod
00015     USE setuparray
00016     USE ppotarray
00017     USE neblistarray
00018     USE coulombarray
00019     USE sparsearray
00020     USE relaxcommon
00021     USE mdarray
00022     USE kspacearray
00023
00024     USE openfiles_mod
00025     USE kernelparser_mod
00026
00027 #ifdef PROGRESSION
00028     USE bml
00029 #endif
00030
00031     IMPLICIT NONE
00032
00033     PRIVATE
00034
00035     INTEGER, PARAMETER :: dp = latteprec
00036
00037     PUBLIC :: parse_control, parse_md, parse_kmesh
00038
00039 #ifdef PROGRESSION
00040
00042     TYPE, PUBLIC :: latte_type
00043
00045         CHARACTER(20) :: jobname
00046
00048         INTEGER :: verbose
00049
00051         REAL(DP) :: threshold
00052
00054         INTEGER :: mdim
00055
00057         CHARACTER(20) :: bml_type
00058
00060         CHARACTER(20) :: bml_dmode
00061
00063         REAL(DP) :: coul_acc
00064
00066         REAL(DP) :: pulaycoeff
00067
00069         REAL(DP) :: mixcoeff
00070
00072         INTEGER :: mpulay
00073
00075         INTEGER :: maxscf
00076
00078         REAL(DP) :: scftol
00079
00081         CHARACTER(20) :: zmat
00082

```



```

00084      CHARACTER(20) :: method
00085
00087      REAL(DP) :: timeratio
00088
00090      INTEGER :: mdsteps
00091
00093      REAL(DP) :: timestep
00094
00096      CHARACTER(100) :: parampath
00097
00099      CHARACTER(100) :: coordsfile
00100
00102      INTEGER :: nlisteach
00103
00105      LOGICAL :: restart
00106
00108      REAL(DP) :: efermi
00109
00110
00111      END TYPE latte_type
00112
00113      TYPE(latte_type), PUBLIC :: lt
00114
00115  #endif
00116
00117  CONTAINS
00118
00121      SUBROUTINE parse_control(FILENAME)
00122
00123          USE fermicommon
00124
00125          IMPLICIT NONE
00126          INTEGER, PARAMETER :: NKEY_CHAR = 6, nkey_int = 53, nkey_re = 21, nkey_log = 1
00127          CHARACTER(LEN=*) :: FILENAME
00128
00129          !Library of keywords with the respective defaults.
00130          CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00131              'JOBNAME=', 'BASISTYPE=', 'SP2CONV=', 'RELAXTYPE=', 'PARAMPATH=', 'COORDSFILE=']
00132          CHARACTER(LEN=100) :: VALVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00133              'MyJob', 'NONORTHO', 'REL', 'SD', './TBparam', './bl/inputblock.dat']
00134
00135          CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_INT(nkey_int) = [character(len=50) :: &
00136              'XCONTROL=', 'DEBUGON=', 'FERMIM=', 'CGORLIB=', 'NORECS=', 'ENTROPYKIND=', &
00137              'PPOTON=', 'VDWON=', 'SPINON=', 'ELECTRO=', 'ELECMETH=', 'MAXSCF=', & !12
00138              'MINSP2ITER=', 'FULLQCONV=', 'QITER=', 'ORDERNMOL=', 'SPARSEON=', 'THRESHOLDON=', & !18
00139              'FILLINSTOP=', 'BLKSZ=', 'MSPARSE=', 'LCNON=', 'LCNITER=', 'RELAX=', 'MAXITER=', & !25
00140              'MDON=', 'PBCON=', 'RESTART=', 'CHARGE=', 'XBO=', 'XBODISON=', 'XBODISORDER=', 'NGPU=', & !33
00141              'KON=', 'COMPFORCE=', 'DOSFIT=', 'INTS2FIT=', 'NFITSTEP=', 'QFIT=', & !39
00142              'PPFITON=', 'ALLFITON=', 'PPSTEP=', 'BISTEP=', 'PP2FIT=', 'BINT2FIT=', 'PPNMOL=', & !46
00143              'PPNGEOM=', 'PARREP=', 'VERBOSE=', 'MIXER=', 'RESTARTLIB=', 'FREEZE=', 'xControl=']
00144          INTEGER :: VALVECTOR_INT(nkey_int) = (/ &
00145              1,0,6,1,1,1, &
00146              1,0,0,1,0,250, &
00147              22,0,1,0,0,1, &
00148              100,4,3000,0,4,0,100, &
00149              1,1,0,0,1,1,5,2, &
00150              0,1,0,1,5000,0,&
00151              0,0,500,500,2,6,10,&
00152              200,0,1,0,0,0,-1 /)
00153
00154          CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_RE(nkey_re) = [character(len=50) :: &
00155              'CGTOL=', 'KBT=', 'SPINTOL=', 'ELEC_ETOL=', 'ELEC_QTOL=', 'COULACC=', 'COULCUT=', 'COULR1=', & !8
00156              'BREAKTOL=', 'QMIX=', 'SPINMIX=', 'MDMIX=', 'NUMTHRESH=', 'CHTOL=', 'SKIN=', & !15
00157              'RLXFTOL=', 'BETA=', 'MCSIGMA=', 'PPBETA=', 'PPSIGMA=', 'ER='] !21
00158          REAL(DP) :: VALVECTOR_RE(nkey_re) = (/&
00159              1.0e-6,0.0,1.0e-4,0.001,1.0e-8,1.0e-6,-500.0, 500.0,&
00160              1.0e-6,0.25,0.25,0.25,1.0e-6,0.01,1.0,&
00161              1.0e-7,1000.0,0.2,1000.0,0.01,1.0/)
00162
00163          CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_LOG(nkey_log) = [character(len=100) :: &
00164              'LIBINIT=']
00165          LOGICAL :: VALVECTOR_LOG(nkey_log) = (/&
00166              .false./)
00167
00168          !Start and stop characters
00169          CHARACTER(LEN=50), PARAMETER :: STARTSTOP(2) = [character(len=50) :: &
00170              'CONTROL{' , '}' ]
00171
00172          CALL parsing_kernel(keyvector_char,valvector_char&
00173              ,keyvector_int,valvector_int,keyvector_re,valvector_re,&
00174              keyvector_log,valvector_log,trim(filename),startstop)
00175
00176
00177          job = valvector_char(1)
00178
00179          ! CONTROL determines how the density matrix is going to be
00180          ! calculated: 1 = diagonalization, 2 = SP2 purification,

```

```

00181      ! 3 = recursive expansion of the Fermi operator, 4 = SP2T,
00182      ! 5 = SP2 Fermi
00183      !
00184
00185      control = valvector_int(1)
00186      IF (valvector_int(53) > 0) THEN !Someone is using xControl=
00187          CALL errors("latteparser_latte_mod","xControl= is not longer in use. Please use XCONTROL=
instead.")
00188      ENDIF
00189
00190
00191      !
00192      ! BASISTYPE can equal "ORTHO" OR "NONORTHO",
00193      !
00194
00195      basistype = valvector_char(2)
00196
00197      IF (basistype .NE. "ORTHO" .AND. basistype .NE. "NONORTHO") THEN
00198          CALL errors("latteparser_latte_mod","Error defining basis type (ortho/nonortho)")
00199      ENDIF
00200
00201      debugon = valvector_int(2)
00202
00203
00204      !
00205      ! Read the order of the recursion in the expansion of the Fermi
00206      ! operator, M.
00207      !
00208
00209      fermim = valvector_int(3)
00210
00211      ! If we're using the expansion of the Fermi operator, we can
00212      ! use a LAPACK routine or Niklasson's conjugate gradient method to
00213      ! solve AX = B. CGORLIB: 0 = LAPACK, 1 = conjugate gradient
00214      ! CGTOL = the user-supplied tolerance for the CG solution of AX = B
00215
00216      cgorlib = valvector_int(4); cgtol = valvector_re(1)
00217
00218      cgtol2 = cgtol*cgtol
00219
00220      ! Electronic temperature, in eV
00221
00222      kbt = valvector_re(2)
00223
00224      !
00225      ! Read the number of recursions for the truncated, finite
00226      ! temperature SP2 algorithm
00227      !
00228
00229      norecs = valvector_int(5)
00230
00231      !
00232      ! What kind of entropy are we going to use in a finite Te calculation
00233      !
00234      ! ENTROPYKIND = 0 : none
00235      ! ENTROPYKIND = 1 : exact for Fermi-Dirac occupation
00236      ! ENTROPYKIND = 2 : Different form of exact expression that may be useful
00237      ! when using CONTROL = 5
00238      ! ENTROPYKIND = 3 : 4th order expansion of exact form (no diag)
00239      ! ENTROPYKIND = 4 : 8th order expansion of exact form (no diag)
00240      !
00241
00242      entropykind = valvector_int(6)
00243
00244      !
00245      ! Do we want long-range C/R^6 tails?
00246      !
00247      ! PPOTON = 1: Turn on pairwise interaction
00248      ! PPOTON = 0: Turn it off (useful for fitting)
00249      !
00250      ! VDWON = 0: No C/R^6 tails
00251      ! VDWON = 1: Use tails
00252      !
00253
00254      ppoton = valvector_int(7); vdwon = valvector_int(8)
00255
00256
00257      !
00258      ! Are we doing a spin-polarized calculation?
00259      ! SPINON = 1 = yes
00260      ! SPINON = 0 = no
00261
00262      spinon = valvector_int(9); spintol = valvector_re(3)
00263
00264      !
00265      ! Controls for electrostatics:
00266      !

```

```

00267      ! ELECTRO: 0 = LCN is applied, 1 = charge dependent TB on
00268      ! ELECMETH: 0 = Ewald summation, 1 = All real space
00269      ! ELEC_ETOL: Tolerance on energy when determining charges (not implemented)
00270      ! ELEC_QTOL: Tolerance on charges during self-consistent calc
00271      !
00272
00273      electro = valvector_int(10); elecmeth = valvector_int(11)
00274      elec_etol = valvector_re(4); elec_qtol = valvector_re(5)
00275
00276      !
00277      ! COULACC: Accuracy for the Ewald method (1.0e-4 works)
00278      ! COULCUT: If we're using the Ewald method, this is the cut-off for the
00279      ! real space part. If we're doing it all in real space, this is the radial
00280      ! cut-off for the sum.
00281      ! COULR1: If we're doing it in real space, the cut-off tail on 1/R is
00282      ! applied here at terminated at COULCUT.
00283      !
00284
00285      coulacc = valvector_re(6); coulcut = valvector_re(7); coulrl = valvector_re(8)
00286
00287      !
00288      ! MAXSCF: Maximum number of SCF cycles
00289      !
00290
00291      maxscf = valvector_int(12)
00292
00293      !
00294      ! BREAKTOL: Tolerance for breaking SP2 loops
00295      ! MINSP2ITER: Minimum number of iterations during SP2 purification
00296      !
00297
00298      breaktol = valvector_re(9); minsp2iter = valvector_int(13);
00299      sp2conv = valvector_char(3)
00300
00301      !
00302      ! FULLQCONV: 0 = We'll run QITER SCF cycles during MD, = 1, we'll run
00303      ! SCF cycles until we've reached ELEC_QTOL. Only important for MD
00304      ! QITER: Number of SCF cycles we're going to run at each MD time step
00305      !
00306
00307      fullqconv = valvector_int(14); qiter = valvector_int(15)
00308
00309      !
00310      ! QMIX AND SPINMIX are the coefficients for the linear mixing of
00311      ! new and old charge and spin densities, respectively, during SCF cycles
00312      !
00313
00314      qmix = valvector_re(10); spinmix = valvector_re(11); mdmix = valvector_re(12)
00315
00316      !
00317      ! ORDERNMOL: Turn on molecule-ID-based density matrix blocking
00318      !
00319
00320      ordernmol = valvector_int(16)
00321
00322      !
00323      ! SPARSEON: 0 = all dense matrix stuff, 1 = use CSR format and
00324      ! Gustavson's algorithm for matrix-matrix multiplication
00325      ! THRESHOLDON: 0 = do not throw away elements; 1 = throw away elements
00326      ! NUMTHRESH: If THRESHOLDON = 1 throw away element whose absolute value is
00327      ! smaller than NUMTHRESH
00328      ! FILLINSTOP: Number of purification cycles beyond which we stop allowing
00329      ! for further fill-in
00330      !
00331
00332      sparseon = valvector_int(17); thresholdon = valvector_int(18);
00333      numthresh = valvector_re(13)
00334
00335      #ifdef PROGRESSON
00336
00337      IF(sparseon == 0) THEN
00338        !t%BML_TYPE = bml_matrix_dense
00339      ELSEIF(sparseon == 1) THEN
00340        !t%BML_TYPE = bml_matrix_ellpack
00341      ELSE
00342        CALL errors("latteparser_latte_mod", "SPARSEON > 1 yet not implemented")
00343      ENDIF
00344
00345      IF(thresholdon == 0) THEN
00346        !t%THRESHOLD = 0.0_dp
00347      ELSEIF(thresholdon == 1) THEN
00348        !t%THRESHOLD = numthresh
00349      ELSE
00350        CALL errors("latteparser_latte_mod", "THRESHOLDON > 1 yet not implemented")
00351      ENDIF
00352
00353      #endif

```

```

00352
00353     fillinstop = valvector_int(19); blkksz = valvector_int(20)
00354
00355     !
00356     ! MSPARSE: value for M when SPARSEON = 1, used by sp2 sparse algorithm
00357     !           0 = value for M is not known, defaults to N
00358     !
00359
00360     msparse = valvector_int(21)
00361
00362     #ifdef PROGRESSION
00363
00364     IF(msparse == 0) THEN
00365         ! %MDIM = -1 !Defaults to N
00366     ELSEIF(msparse > 0) THEN
00367         ! %MDIM = msparse
00368     ELSE
00369         CALL errors("latteparser_latte_mod","MSPARSE cannot be negative")
00370     ENDIF
00371
00372     #endif
00373
00374     !
00375     ! LCNON: 0 = during charge neutral MD simulations we'll run LCNITER SCF
00376     ! cycles at each time step, 1 = we'll run SCF cycles until CHTOL is reached
00377     ! LCNITER: Number of SCF cycles to achieve LCN at each MD time step
00378     ! CHTOL: Tolerance on atomic charges (Mulliken) before LCN is declared
00379     !
00380
00381     lcnon = valvector_int(22); lcniter = valvector_int(23); chtol = valvector_re(14)
00382
00383     !
00384     ! Read the SKIN for the neighbor list (Angstrom)
00385     !
00386
00387     skin = valvector_re(15)
00388
00389     !
00390     ! RELAXME: 0 = Don't run relaxation, 1 = relax geometry
00391     ! RELTYPE: SD = steepest descent, CG = conjugate gradient
00392     ! MXRLX: Maximum number of steps in the geometry optimization
00393     ! RLXFTOT: Run optimization until all forces are less than RLXFTOL
00394     !
00395
00396     relaxme = valvector_int(24); reltype = valvector_char(4) ;
00397     mxrlx = valvector_int(25)
00398     rlxfitol = valvector_re(16)
00399
00400     !
00401     ! MDON: 0 = Molecular dynamics off, 1 = Molecular dynamics on
00402     ! (MD is controlled using the file MDcontroller)
00403     !
00404
00405     mdon = valvector_int(26)
00406
00407     !
00408     ! PBCON: 1 = full periodic boundary conditions, 0 = gas phase: no pbc and
00409     ! electrostatics done all in real space
00410     !
00411
00412     pbcon = valvector_int(27)
00413
00414     restart = valvector_int(28)
00415
00416     ! Add or remove electrons. 2+ -> charge = +2 since TOTNE = TOTNE - CHARGE
00417
00418     charge = valvector_int(29)
00419
00420     !
00421     ! XBOON: 0 = Niklasson's extended Lagrangian Born-Oppenheimer MD off,
00422     ! 1 = on.
00423     !
00424
00425     xboon = valvector_int(30)
00426
00427     !
00428     ! XBODISON: We have the option of turning on damping for the XBO
00429     ! to remedy the accumulation of noise. 0 = off, 1 = on.
00430     !
00431
00432     xbodison = valvector_int(31)
00433
00434     !
00435     ! XBODISORDER: = Order of the damping function (1 - 9)
00436     !
00437
00438     xbodisorder = valvector_int(32)

```

```

00438
00439     fiton = 0
00440
00441     !
00442     ! Read in the number of GPUs per node
00443     !
00444
00445     ngpu = valvector_int(33)
00446
00447     ! Are we doing k-space?
00448
00449     kon = valvector_int(34)
00450
00451     ! Do we want to calculate forces too (not always necessary when fitting)
00452
00453     compforce = valvector_int(35)
00454
00455     ! Turn on the simulated annealing subroutine to fit DOS
00456
00457
00458     dosfiton = valvector_int(36); int2fit = valvector_int(37)
00459     mcbeta = valvector_re(17); nfitstep = valvector_int(38); qfit = valvector_int(39)
00460     mcsigma = valvector_re(18)
00461
00462     ppfiton = valvector_int(40)
00463
00464     allfiton = valvector_int(41)
00465
00466     ppnfitstep = valvector_int(42); binfitstep = valvector_int(43)
00467     pp2fit = valvector_int(44); bint2fit = valvector_int(45)
00468
00469     ppbeta = valvector_re(19); ppsigma = valvector_re(20)
00470     ppnmol = valvector_int(46); ppngeom = valvector_int(47)
00471
00472     parrep = valvector_int(48)
00473
00474     ! Verbosity level to control general output
00475
00476     verbose = valvector_int(49)
00477
00478     ! If Pulay Mixer
00479
00480     mixer = valvector_int(50)
00481
00482     ! Restart option for latte lib.
00483
00484     mixer = valvector_int(51)
00485
00486     ! Dielectric constant
00487
00488     relperm = valvector_re(21)
00489
00490     ! Coordinates and parameter paths
00491
00492     parampath = valvector_char(5)
00493     coordsfile = valvector_char(6)
00494
00495     ! If latte_lib has to be restarted
00496
00497     restartlib = valvector_int(51)
00498
00499     ! Freeze atoms
00500
00501     freeze = valvector_int(52)
00502
00503     IF (electro == 1 .AND. elecmeth == 1 .AND. pbcon == 1) THEN
00504         CALL errors("latteparser_latte_mod","If CONTROL{ELECTRO= 1 ELECMEH= 1} &
00505             &then CONTROL{PBCON= 0}")
00506     END IF
00507
00508 END SUBROUTINE parse_control
00509
00510
00513 SUBROUTINE parse_md(FILENAME)
00514
00515     IMPLICIT NONE
00516     INTEGER, PARAMETER :: NKEY_CHAR = 3, nkey_int = 18, nkey_re = 10, nkey_log = 1
00517     CHARACTER(LEN=*) :: FILENAME
00518
00519     !Library of keywords with the respective defaults.
00520     CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00521         'RNDIST=', 'SEEDINIT=', 'NPTTYPE=']
00522     CHARACTER(LEN=100) :: VALVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00523         'GAUSSIAN', 'UNIFORM', 'ISO']
00524
00525     CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_INT(nkey_int) = [character(len=50) :: &
00526         'MAXITER=', 'UDNEIGH=', 'DUMPFREQ=', 'RSFREQ=', 'WRTFREQ=', 'TOINITTEMP5=', 'THERMPER=', & !7

```

```

00527      'THERMRUN=', 'NVTON=', 'NPTON=', 'AVEPER=', 'SEED=', 'SHOCKON=', &
00528      'SHOCKSTART=', 'SHOCKDIR=', 'MDADAPT=', 'GETHUG=', 'RSLEVEL=' ]
00529  INTEGER :: VALVECTOR_INT(nkey_int) = (/ &
00530      5000,1,250,500,25,1,500, &
00531      50000,0,0,1000,54,0, &
00532      100000,1,0,0,0/)
00533
00534  CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_RE(nkey_re) = [character(len=50) :: &
00535      'DT=', 'TEMPERATURE=', 'FRICTION=', 'PTARGET=', 'UPARTICLE=', 'USHOCK=', 'C0=', 'E0=', &
00536      'V0=', 'P0=']
00537  REAL(DP) :: VALVECTOR_RE(nkey_re) = (/&
00538      0.25,300.00,1000.0,0.0,500.0,-4590.0,1300.0,-795.725,&
00539      896.984864,0.083149/)
00540
00541  CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_LOG(nkey_log) = [character(len=100) :: &
00542      'DUMMY=']
00543  LOGICAL :: VALVECTOR_LOG(nkey_log) = (/&
00544      .false./)
00545
00546  !Start and stop characters
00547  CHARACTER(LEN=50), PARAMETER :: STARTSTOP(2) = [character(len=50) :: &
00548      'MDCONTROL{', ' '}]
00549
00550  CALL parsing_kernel(keyvector_char,valvector_char&
00551      ,keyvector_int,valvector_int,keyvector_re,valvector_re,&
00552      keyvector_log,valvector_log,trim(filename),startstop)
00553
00554
00555  !
00556  ! MAXITER = run this many MD time steps
00557  !
00558
00559  maxiter = valvector_int(1)
00560
00561  !
00562  ! UDNEIGH = update the neighbor lists every UDNEIGH time steps
00563  !
00564
00565  udneigh = valvector_int(2)
00566  !   write(*,*)"UDNEIGH",UDNEIGH
00567  !
00568  ! DT = size of the time step in fs
00569  !
00570
00571  dt = valvector_re(1)
00572  !   write(*,*)"DT",DT
00573  !
00574  ! TTARGET = temperature in K were initialize and aim for during NVT MD
00575  ! RNDIST = Type of distribution of random numbers used to initialize T
00576  !         = GAUSSIAN or UNIFORM
00577  ! SEEDINIT = Type of seed used in the generation of random numbers
00578  !         = RANDOM - seed changes every time
00579  !         = DEFAULT - use the same, default seed every time
00580  !
00581
00582  ttargt = valvector_re(2); rndist = valvector_char(1); seedinit = valvector_char(2
)
00583  !   write(*,*)"TTARGET,RNDIST,SEEDINIT",TTARGET,RNDIST,SEEDINIT
00584  !
00585  ! DUMPFREQ: Write a dump file every DUMPFREQ time steps
00586  !
00587
00588  dumpfreq = valvector_int(3)
00589  !   write(*,*)"DUMPFREQ",DUMPFREQ
00590  !
00591  ! RSFREQ: Write a restart file every RSFREQ time steps
00592  !
00593
00594  rsfreq = valvector_int(4)
00595  !   write(*,*)"RSFREQ",RSFREQ
00596  !
00597  ! WRTFREQ: Output energy and temperature every WRTFREQ time steps
00598  !
00599
00600  wrtfreq = valvector_int(5)
00601  !   write(*,*)"WRTFREQ",WRTFREQ
00602  !
00603  ! TOINITTEMP: Whether or not we are going to initialize velocities
00604  ! using a random number generator (sometimes during a restart we
00605  ! may not want to reinitialize the temperature
00606  !
00607
00608  toinittemp = valvector_int(6)
00609  !   write(*,*)"TOINITTEMP",TOINITTEMP
00610  !
00611  ! THERMPER: If we're running NVT, rescale velocities every THERMPER
00612  ! time steps.

```

```

00613      !
00614
00615      thermper = valvector_int(7)
00616      !   write(*,*) "THERMPER", THERMPER
00617      !
00618      ! THERMRUN: Thermalize over this many time steps when NVT is on
00619      !
00620
00621      thermrun = valvector_int(8)
00622      !   write(*,*) "THERMRUN", THERMRUN
00623      !
00624      ! NVTON: 0 = running NVE MD, 1 = running NVT MD
00625      ! AVEPER: Average the temperature over AVEPER time steps when determining
00626      ! how to rescale velocities
00627      !
00628
00629      nvton = valvector_int(9); npton = valvector_int(10)
00630      !   write(*,*) "NVTON,NPTON", NVTON, NPTON
00631      aveper = valvector_int(11); friction = valvector_re(3); seedth = valvector_int(12)
00632      !   write(*,*) "AVEPER, FRICTION, SEEDTH", AVEPER, FRICTION, SEEDTH
00633
00634
00635      IF (nvton.EQ. 1 .AND. npton.EQ. 1) THEN
00636          CALL errors("latteparser_latte_mod", "You can't have NVTON = 1 and NPTON = 1")
00637      ENDIF
00638
00639      ! PTARGET = Target pressure (in GPa) when running NPT
00640      ! NPCTYPE = ISO or ANISO
00641
00642      ptarget = valvector_re(4); npctype = valvector_char(3)
00643      !   write(*,*) "PTARGET, NPCTYPE", PTARGET, NPCTYPE
00644
00645      !
00646      ! The following are for the Hugoniosat
00647      !
00648
00649      ! On (1) or off (0)?
00650
00651      shockon = valvector_int(13)
00652      !   write(*,*) "SHOCKON", SHOCKON
00653      !
00654      ! SHOCKSTART = the MD iteration where we will start to compress
00655      ! the iteration when we stop depends on the size of the block and Us
00656      !
00657
00658      shockstart = valvector_int(14)
00659      !   write(*,*) "SHOCKSTART", SHOCKSTART
00660      !
00661      ! SHOCKDIR is the cartesian direction (1 = X, 2 = Y, 3 = Z),
00662      ! parallel to which we're going to compress uniaxially
00663      !
00664
00665      shockdir = valvector_int(15)
00666      !   write(*,*) "SHOCKDIR", SHOCKDIR
00667      !
00668      ! And finally, the particle and shock velocities
00669      ! IN UNITS OF METRES PER SECOND
00670      !
00671
00672      uparticle = valvector_re(5); ushock = valvector_re(6); c0 = valvector_re(7)
00673      !   write(*,*) "UPARTICLE, USHOCK, C0", UPARTICLE, USHOCK, C0
00674      ! Adapt SCF on the fly?
00675
00676      mdadapt = valvector_int(16)
00677      !   write(*,*) "MDADAPT", MDADAPT
00678      ! Calculating Hugoniot points?
00679
00680      gethug = valvector_int(17)
00681
00682      rslevel = valvector_int(18)
00683
00684      !   write(*,*) "GETHUG", GETHUG
00685      e0 = valvector_re(8); v0 = valvector_re(9); p0 = valvector_re(10)
00686      !   write(*,*) "E0, V0, P0", E0, V0, P0
00687
00688      END SUBROUTINE parse_md
00689
00690
00691      SUBROUTINE parse_kmesh(FILENAME)
00692
00693      IMPLICIT NONE
00694      INTEGER, PARAMETER :: NKEY_CHAR = 1, nkey_int = 3, nkey_re = 3, nkey_log = 1
00695      CHARACTER(LEN=*) :: FILENAME
00696
00697      !Library of keywords with the respective defaults.
00698      CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00699          'DUMMY=' ]
00700
00701

```

```

00702     CHARACTER(LEN=100) :: VALVECTOR_CHAR(nkey_char) = [character(len=100) :: &
00703         'DUMMY']
00704
00705     CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_INT(nkey_int) = [character(len=50) :: &
00706         'NKX=', 'NKY=', 'NKZ=']
00707     INTEGER :: VALVECTOR_INT(nkey_int) = (/ &
00708         1,1,1/)
00709
00710     CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_RE(nkey_re) = [character(len=50) :: &
00711         'KSHIFTX=', 'KSHIFTY=', 'KSHIFTZ=']
00712     REAL(DP) :: VALVECTOR_RE(nkey_re) = (/&
00713         0.0,0.0,0.0/)
00714
00715     CHARACTER(LEN=50), PARAMETER :: KEYVECTOR_LOG(nkey_log) = [character(len=100) :: &
00716         'DUMMY=']
00717     LOGICAL :: VALVECTOR_LOG(nkey_log) = (/&
00718         .false./)
00719
00720     !Start and stop characters
00721     CHARACTER(LEN=50), PARAMETER :: STARTSTOP(2) = [character(len=50) :: &
00722         'KMESH{' , ' '}']
00723
00724     CALL parsing_kernel(keyvector_char, valvector_char&
00725         ,keyvector_int, valvector_int, keyvector_re, valvector_re, &
00726         keyvector_log, valvector_log, trim(filename), startstop)
00727
00728     nkx= valvector_int(1); nky= valvector_int(2); nkz=valvector_int(3)
00729     kshift(1)= valvector_re(1); kshift(2)= valvector_re(2); kshift(3)= valvector_re(3)
00730
00731
00732     END SUBROUTINE parse_kmesh
00733
00734
00735     END MODULE latteparser_latte_mod

```

8.243 masses2symbols.f90 File Reference

Functions/Subroutines

- subroutine [masses2symbols](#) (TYPES, NTYPES, MASSES_IN, NATSIN, SYMBOLS)

Subroutine to get the Symbols out of the masses of the elements.

8.243.1 Function/Subroutine Documentation

8.243.1.1 subroutine `masses2symbols` (integer, dimension(ntypes), intent(in) *TYPES*, integer, intent(in) *NTYPES*, real(latteprec), dimension(ntypes), intent(in) *MASSSES_IN*, integer, intent(in) *NATSIN*, character(len=2), dimension(natsin), intent(inout) *SYMBOLS*)

Subroutine to get the Symbols out of the masses of the elements.

Parameters

<i>TYPES</i>	atom type index.
<i>NTYPES</i>	Number of types.
<i>MASSSES_IN</i>	Masses for every type.
<i>NATSIN</i>	Number of total atoms.
<i>SYMBOLS</i>	Symbols for every atom.

Read ptable

Definition at line 30 of file [masses2symbols.f90](#).

Here is the caller graph for this function:



8.244 masses2symbols.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00029 SUBROUTINE masses2symbols (TYPES,NTYPES,MASSSES_IN,NATSIN,SYMBOLS)
00030
00031   USE setuparray
00032   USE myprecision
00033   USE constants_mod
00034
00035   IMPLICIT NONE
00036
00037   INTEGER, INTENT(IN) :: NTYPES
00038   INTEGER, INTENT(IN) :: TYPES(ntypes)
00039   INTEGER, INTENT(IN) :: NATSIN
00040   REAL(LATTEPREC), INTENT(IN) :: MASSSES_IN(ntypes)
00041   CHARACTER(LEN=2), INTENT(INOUT) :: SYMBOLS(natsin)
00042   INTEGER :: NZ = 103
00043   INTEGER, PARAMETER :: DP = latteprec
00044   CHARACTER(2), ALLOCATABLE :: ELEMENT_SYMBOL(:), TYPE_SYMBOLS(:)
00045   CHARACTER(20) :: DUMMYC1, DUMMYC2, DUMMYC3, DUMMYC4
00046   REAL(DP), ALLOCATABLE :: ELEMENT_MASS(:)
00047   REAL(DP) :: DUMMYR(20)
00048   CHARACTER(20) :: DUMMYC(20)
00049   REAL(DP) :: DUMMYR1, DUMMYR2
00050   INTEGER :: I, J
00051
00054   OPEN(unit=14,status="OLD", file=trim(parampath)//"/electrons.dat")
00055
00056   ALLOCATE(element_symbol(nz))
00057   ALLOCATE(element_mass(nz))
00058
00059   READ(14,*) dummyc1, nz
00060   READ(14,*) (dummyc(j),j=1,13)
00061   DO i = 1,nz
00062     READ(14,*) element_symbol(i), dummyc1, (dummyr(j),j=1,5), element_mass(i), (dummyr(j),j=6,10)
00063   ENDDO
00064
00065   CLOSE(14)
00066
00067   ALLOCATE(type_symbols(ntypes))
00068   type_symbols=""
00069
00070   DO i=1,ntypes
00071     DO j=1,nz

```

```

00072         IF (abs(masses_in(i) - element_mass(j)) < 0.01) THEN
00073             type_symbols(i) = element_symbol(j)
00074             IF (existerror) RETURN
00075             EXIT
00076         ENDIF
00077     ENDDO
00078     IF (type_symbols(i) == "") THEN
00079         WRITE(*,*) "ERROR: Mass of element", i, "cannot be identified."
00080         CALL errors("masses2symbols", "The mass of an element in the &
00081             & coordinates file cannot be identified. Please verify that &
00082             & masses in electrons.dat coincide with the masses in the input file")
00083     ENDIF
00084 ENDDO
00085
00086 DO i=1,natsin
00087     symbols(i) = type_symbols(types(i))
00088 ENDDO
00089
00090 DEALLOCATE(type_symbols)
00091 DEALLOCATE(element_symbol)
00092 DEALLOCATE(element_mass)
00093
00094 END SUBROUTINE masses2symbols

```

8.245 matrixio.f90 File Reference

Modules

- module [matrixio](#)

Functions/Subroutines

- subroutine [matrixio::writedmatrix](#) (HSIZE, DARRAY)
- subroutine [matrixio::writehmatrix](#) (HSIZE, MSIZE, HARRAY, NITER, PVEC)
- subroutine [matrixio::writemtx](#) (ITER, HSIZE, II, JJ, VAL)

8.246 matrixio.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE matrixio
00023
00024     IMPLICIT NONE
00025
00026     CONTAINS
00027
00028     !
00029     ! writehmatrix - Write out hamiltonian matrix to a file.
00030     !
00031     SUBROUTINE writehmatrix(HSIZE, MSIZE, HARRAY, NITER, PVEC)

```

```

00032
00033     USE myprecision
00034
00035     IMPLICIT NONE
00036     INTEGER, INTENT(IN) :: HSIZE, MSIZE, NITER
00037     INTEGER, INTENT(IN) :: PVEC(:)
00038     INTEGER :: I, J
00039     REAL(LATTEPREC), INTENT(IN) :: HARRAY(:, :)
00040     CHARACTER(LEN=100) :: FLNM
00041
00042     WRITE(flnm, '("hmatrix.in.dat")')
00043
00044     OPEN (unit = 10, status="UNKNOWN", file=flnm)
00045
00046     WRITE(10,*) hsize, msize
00047     WRITE(10,*) niter, (pvec(i), i = 1,niter)
00048     DO i = 1,hsize
00049         WRITE(10,*) (harray(j,i), j = 1,hsize)
00050         !!WRITE(10,10) (HARRAY(J,I), J = 1,HSIZE)
00051     ENDDO
00052     !!10 FORMAT(100(E15.5,3X))
00053
00054     CLOSE(10)
00055
00056 END SUBROUTINE writehmatrix
00057
00058 !
00059 ! writedmatrix - Write out density matrix to a file.
00060 !
00061 SUBROUTINE writedmatrix(HSIZE, DARRAY)
00062
00063     USE myprecision
00064     USE setuparray
00065
00066     IMPLICIT NONE
00067     INTEGER, INTENT(IN) :: HSIZE
00068     INTEGER :: I, J
00069     REAL(LATTEPREC), INTENT(IN) :: DARRAY(:, :)
00070     CHARACTER(LEN=100) :: FLNM
00071
00072     WRITE(flnm, '("dmatrix.out.dat")')
00073
00074     OPEN (unit = 10, status="UNKNOWN", file=flnm)
00075     WRITE(10,*) hsize
00076     DO i = 1,hsize
00077         WRITE(10,10) (darray(j,i), j = 1,hsize)
00078     ENDDO
00079 10 FORMAT(100(e15.5,3x))
00080     CLOSE(10)
00081
00082 END SUBROUTINE writedmatrix
00083
00084 SUBROUTINE writemtx(ITER, HSIZE, II, JJ, VAL)
00085
00086     USE myprecision
00087
00088     IMPLICIT NONE
00089     INTEGER, INTENT(IN) :: HSIZE, ITER
00090     INTEGER, INTENT(IN) :: II(:), JJ(:, :)
00091     INTEGER :: I, J, MSUM
00092     REAL(LATTEPREC), INTENT(IN) :: VAL(:, :)
00093     CHARACTER(LEN=100) :: FLNM
00094     CHARACTER(LEN=20) :: FFMT
00095
00096     ffmt = ' (A9,I2.2,A4)'
00097
00098     WRITE(flnm, ffmt) "spmatrix_", iter, ".mtx"
00099
00100     OPEN (unit = 10, status="UNKNOWN", file=flnm)
00101     WRITE(10,*) "%%MatrixMarket sparse coordinate real general"
00102     msum = 0
00103     DO i = 1, hsize
00104         msum = msum + ii(i)
00105     ENDDO
00106     WRITE(10,*) hsize, hsize, msum
00107     DO i = 1,hsize
00108         DO j = 1,ii(i)
00109             WRITE(10,*) i, jj(j,i), val(j,i)
00110         ENDDO
00111     ENDDO
00112     !10 FORMAT(100(E15.5,3X))
00113     CLOSE(10)
00114
00115 END SUBROUTINE writemtx
00116
00117 END MODULE matrixio

```

8.247 mdarray.f90 File Reference

Modules

- module [mdarray](#)

Variables

- integer [mdarray::aveper](#)
- real(latteprec) [mdarray::avet](#)
- real(latteprec) [mdarray::c0](#)
- real(latteprec) [mdarray::consmot](#)
- integer [mdarray::contiter](#)
- real(latteprec) [mdarray::cumdt](#)
- real(latteprec) [mdarray::dgamma](#)
- real(latteprec) [mdarray::dt](#)
- real(latteprec) [mdarray::dtzero](#)
- integer [mdarray::dumpfreq](#)
- real(latteprec) [mdarray::e0](#)
- real(latteprec), dimension(:), allocatable [mdarray::ehist](#)
- integer [mdarray::entropyiter](#)
- real(latteprec), dimension(:, :), allocatable [mdarray::franprev](#)
- real(latteprec) [mdarray::friction](#)
- real(latteprec) [mdarray::gamma](#)
- integer [mdarray::gethug](#)
- real(latteprec) [mdarray::hg](#)
- integer [mdarray::iset](#)
- real(latteprec), dimension(:), allocatable [mdarray::mass](#)
- integer [mdarray::maxiter](#)
- integer [mdarray::npton](#)
- character(len=10) [mdarray::nptype](#)
- integer [mdarray::nvton](#)
- real(latteprec) [mdarray::p0](#)
- real(latteprec), dimension(:), allocatable [mdarray::phist](#)
- real(latteprec), dimension(:), allocatable [mdarray::phistx](#)
- real(latteprec), dimension(:), allocatable [mdarray::phisty](#)
- real(latteprec), dimension(:), allocatable [mdarray::phistz](#)
- real(latteprec) [mdarray::ptarget](#)
- character(len=10) [mdarray::rndist](#)
- integer [mdarray::rsfreq](#)
- character(len=10) [mdarray::seedinit](#)
- integer [mdarray::seedth](#)
- integer [mdarray::setth](#)
- integer [mdarray::shockdir](#)
- integer [mdarray::shockon](#)
- integer [mdarray::shockstart](#)
- integer [mdarray::shockstop](#)
- real(latteprec) [mdarray::temperature](#)
- integer [mdarray::thermper](#)
- integer [mdarray::thermrn](#)
- real(latteprec), dimension(:), allocatable [mdarray::thist](#)
- integer [mdarray::toinittemp](#)
- real(latteprec) [mdarray::ttarget](#)

- `real(latteprec) mdarray::tzero`
- `real(latteprec) mdarray::uparticle`
- `real(latteprec) mdarray::ushock`
- `real(latteprec), dimension(:, :), allocatable mdarray::v`
- `real(latteprec) mdarray::v0`
- `real(latteprec), dimension(:), allocatable mdarray::vhist`
- `integer mdarray::wrtfreq`

8.248 mdarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE mdarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER :: iset
00030   REAL(LATTEPREC), ALLOCATABLE :: v(:, :), mass(:)
00031   REAL(LATTEPREC), ALLOCATABLE :: thist(:), phist(:), ehist(:),
00032   vhist(:)
00033   REAL(LATTEPREC), ALLOCATABLE :: phistx(:), phisty(:), phistz(:)
00034   REAL(LATTEPREC) :: dt, ttarget, ptarget, temperature,
00035   dtzero, tzero
00036   REAL(LATTEPREC) :: avet, hg, p0, v0, e0
00037   INTEGER :: contiter, nvton, npton, aveper, toinittemp,
00038   gethug
00039   INTEGER :: maxiter, dumpfreq, rsfreq, wrtfreq,
00040   thermper, thermrun
00041   INTEGER :: entropyiter
00042   CHARACTER(LEN=10) :: rndist, seedinit, npttype
00043   ! These are for the Hugonostat
00044   INTEGER :: shockon, shockstart, shockstop, shockdir
00045   REAL(LATTEPREC) :: uparticle, ushock, c0
00046   !!$ Thermostating
00047   INTEGER :: seedth
00048   INTEGER :: setth
00049   REAL(LATTEPREC) :: friction
00050   !!$ These are for Langevin dynamics
00051
00052   REAL(LATTEPREC), ALLOCATABLE :: franprev(:, :)
00053
00054   !!$ These are for Andersen thermostat
00055
00056   REAL(LATTEPREC) :: cumdt
00057   ! REAL(LATTEPREC) :: CUMDT, TAU
00058   ! INTEGER :: QITERAND, QITERIN
00059
00060   !!$ These are for Nose thermostat
00061
00062   REAL(LATTEPREC) :: gamma, dgamma
00063   REAL(LATTEPREC) :: consmot
00064
00065 END MODULE mdarray

```

8.249 mixer_mod.f90 File Reference

Modules

- module `mixer_mod`
To implement mixing schemes from the progress library.

Functions/Subroutines

- subroutine, public `mixer_mod::qmixprg` (PITER)

Variables

- `real(latteprec)`, `dimension(:, :)`, allocatable, public `mixer_mod::dqin`
- `real(latteprec)`, `dimension(:, :)`, allocatable, public `mixer_mod::dqout`
- logical, public `mixer_mod::mixinit` = .FALSE.
- `type(mx_type)`, public `mixer_mod::mx`
- `real(latteprec)`, public `mixer_mod::scferror`

8.250 mixer_mod.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00024 MODULE mixer_mod
00025
00026 #ifdef PROGRESSON
00027
00028   USE myprecision
00029   USE coulombarray
00030   USE setuparray
00031   USE prg_pulaymixer_mod
00032
00033   PRIVATE
00034
00035   PUBLIC :: qmixprg
00036
00037   !For mixing scheme
00038   LOGICAL, PUBLIC :: mixinit = .false.
00039   REAL(LATTEPREC), ALLOCATABLE, PUBLIC :: dqin(:, :), dqout(:, :)
00040   REAL(LATTEPREC), PUBLIC :: scferror
00041   TYPE(mx_type), PUBLIC :: mx
00042
00043 CONTAINS
00044
00045   SUBROUTINE qmixprg(PITER)
00046     IMPLICIT NONE
00047     INTEGER, INTENT(IN) :: PITER
00048

```

```

00049      IF (mx%MIXERTYPE == "Linear") THEN
00050          CALL prg_linear mixer (deltaq, olddeltaqs, scferror,
mx%MIXCOEFF, mx%VERBOSE)
00051      ELSEIF (mx%MIXERTYPE == "Pulay") THEN
00052          CALL prg_qmixer (deltaq, olddeltaqs, dqin, dqout,
scferror, piter, mx%MIXCOEFF, mx%MPULAY, mx%VERBOSE)
00053      ELSE
00054          CALL errors ("mixer_mod:qmixprg", "Mixing scheme not implemented. &
00055                      & Check MixerType keyword in the input file")
00056      ENDIF
00057
00058  END SUBROUTINE qmixprg
00059
00060 #endif
00061
00062 END MODULE mixer_mod

```

8.251 mofit.f90 File Reference

Functions/Subroutines

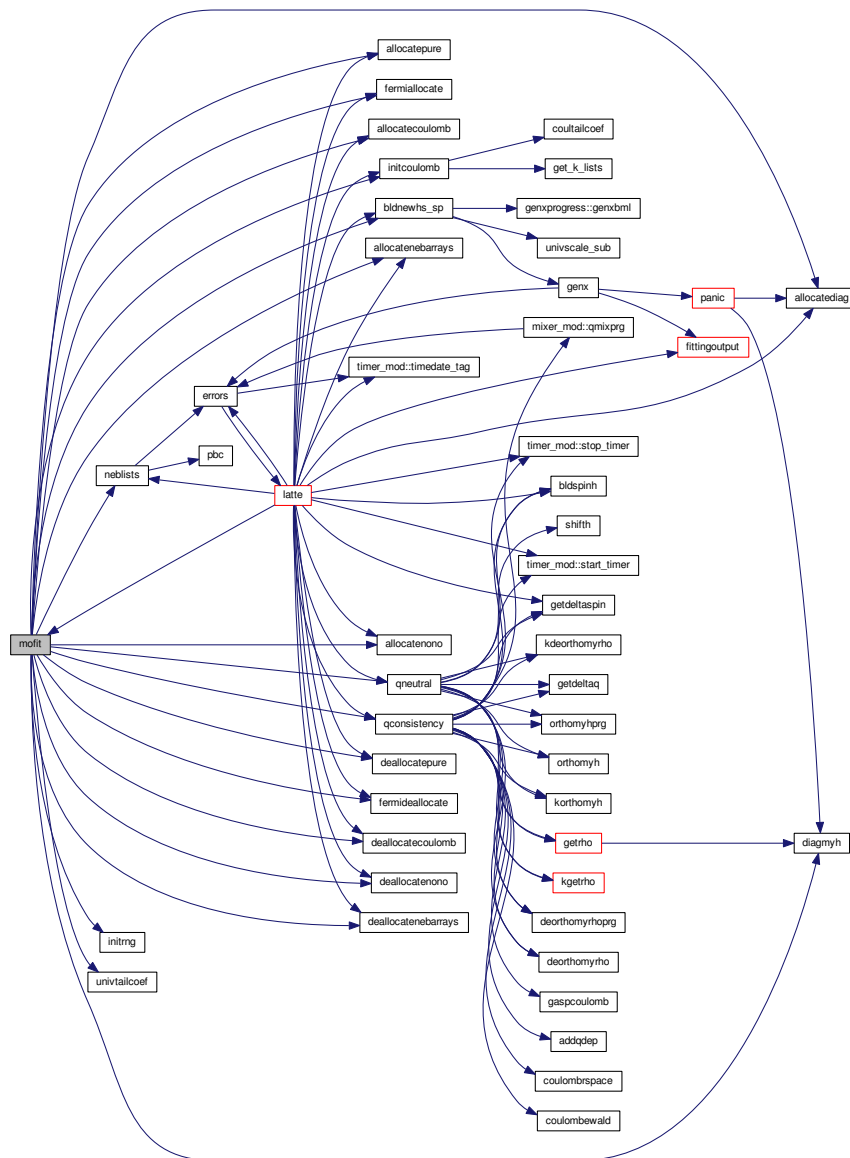
- subroutine [mofit](#)

8.251.1 Function/Subroutine Documentation

8.251.1.1 subroutine mofit ()

Definition at line 23 of file [mofit.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE mofit
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE kspacearray
00028   USE univarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, JUNK, ACC, II, COUNT
00034   INTEGER :: NSTEP, PICK, CPLOC(1), LOOPTARGET, AB
00035   REAL(LATTEPREC) :: EMIN, EMAX, MERIT, OLDMERIT, RN, ESTEP, EF
00036   REAL(LATTEPREC), ALLOCATABLE :: DFTMO(:)
00037   REAL(LATTEPREC), ALLOCATABLE :: TBDOS(:), TBOLD(:), TBORIG(:)
00038   REAL(LATTEPREC), ALLOCATABLE :: TBSCLOLD(:), TBSCLOLIG(:)
00039   REAL(LATTEPREC), ALLOCATABLE :: TBBEST(:), TBBESTSCL(:)
00040   REAL(LATTEPREC) :: ENERGY, INTDOS, NUME, JUNKNUM, MINERR
00041   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00042   IF (existerror) RETURN
00043
00044   ! Read in the DOS from a VASP calculation
00045
00046   OPEN(unit=20, status="OLD", file="M02fit.dat")
00047
00048   ALLOCATE(dftmo(hdim))
00049   ALLOCATE(tbold(int2fit), tborig(int2fit))
00050   ALLOCATE(tbsclold(int2fit), tbsclorig(int2fit))
00051   ALLOCATE(tbbest(int2fit), tbbestscl(int2fit))
00052
00053   DO i = 1, hdim
00054
00055     READ(20,*) j, dftmo(i), junknum
00056
00057   ENDDO
00058
00059   CLOSE(20)
00060
00061   ! Initialize stuff
00062
00063
00064   IF (control .EQ. 1) THEN
00065     CALL allocatediag
00066   ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00067     CALL allocatepure
00068   ELSEIF (control .EQ. 3) THEN
00069     CALL fermiallocate
00070   ENDIF
00071
00072   IF (electro .EQ. 1) THEN
00073     CALL allocatecoulomb
00074     CALL initcoulomb
00075   ENDIF
00076
00077   !
00078   ! Allocate stuff for building the neighbor lists
00079   !
00080
00081   CALL allocatenebarrays
00082
00083   CALL neblists(0)
00084
00085   IF (basistype .EQ. "NONORTHO") CALL allocatenono
00086
00087   ! Compute DOS from LATTE and compare
00088
00089   CALL initrng
00090
00091   acc = 0
00092
00093   DO i = 1, int2fit

```

```

00094      tborig(i) = bond(1,i)
00095      tbsclorig(i) = bond(2,i)
00096      tbbest(i) = bond(1,i)
00097      tbbestscl(i) = bond(2,i)
00098      ENDDO
00099
00100
00101      DO ii = 1, nfitstep
00102
00103          ! Change set of TB parameters
00104
00105          DO i = 1, int2fit
00106
00107              tbold(i) = bond(1,i)
00108              tbsclold(i) = bond(2,i)
00109          ENDDO
00110
00111          ! Pick one to change
00112
00113
00114          IF (ii .GT. 1) THEN
00115
00116              CALL random_number(rn)
00117
00118              pick = int(rn*REAL(int2fit)) + 1
00119
00120              CALL random_number(rn)
00121
00122              !          AB = INT(RN*2.0D0) + 1
00123              ab = 1
00124              CALL random_number(rn)
00125
00126              ! Change by +/- 20%
00127
00128              bond(ab,pick) = bond(ab,pick) * gaussrn(1.0, mcsigma)
00129
00130
00131              IF (abs(bond(1,pick)) .LT. 0.01) &
00132                  bond(1,pick) = sign(0.01d0, tborig(pick))
00133
00134              !          IF (BOND(1,PICK)/TBORIG(PICK) .LT. 0.5D0) THEN
00135              !              BOND(1,PICK) = 0.5D0*TBORIG(PICK)
00136              !          ELSEIF (BOND(1,PICK)/TBORIG(PICK) .GT. 1.5D0) THEN
00137              !              BOND(1,PICK) = 1.5D0*TBORIG(PICK)
00138              !          ENDIF
00139
00140              !          IF (BOND(1,4) .GT. BOND(1,3)) BOND(1,3) = BOND(1,4)
00141
00142              !          IF (ABS(BOND(1,3)/BOND(1,4)) .LT. 2.0D0) THEN
00143              !              BOND(1,3) = 2.0D0*BOND(1,4)
00144              !          ENDIF
00145
00146
00147
00148              !          IF (BOND(2,PICK)/TBSCLOLIG(PICK) .LT. 0.5D0) THEN
00149              !              BOND(2,PICK) = 0.5D0*TBSCLOLIG(PICK)
00150              !          ELSEIF (BOND(1,PICK)/TBORIG(PICK) .GT. 1.5D0) THEN
00151              !              BOND(2,PICK) = 1.5D0*TBSCLOLIG(PICK)
00152              !          ENDIF
00153
00154
00155              CALL univtailcoef(bond(:,pick))
00156
00157          ENDF
00158
00159          ! Re-build cut-off tails
00160
00161          ! Build H
00162
00163          CALL bldnewhs_sp
00164
00165          IF (qfit .EQ. 0) THEN
00166
00167              CALL diagmyh
00168
00169          ELSE
00170
00171              IF (electro .EQ. 0) THEN
00172                  CALL qneutral(0, 1) ! Local charge neutrality
00173              ELSE
00174                  CALL qconsistency(0,1) ! Self-consistent charges
00175              ENDF
00176
00177          ENDF
00178
00179          ! Compute errors
00180

```

```

00181      merit = zero
00182
00183      DO i = 1, hdim
00184
00185          merit = merit + (evals(i) - dftmo(i))*(evals(i) - dftmo(i))
00186
00187      ENDDO
00188
00189      IF (ii .EQ. 1) oldmerit = merit
00190      IF (ii .EQ. 1) minerr = merit
00191
00192      IF (merit .LT. oldmerit) THEN
00193
00194          acc = acc + 1
00195          oldmerit = merit
00196
00197          WRITE(*,11) ii, acc, merit, oldmerit, &
00198              (bond(1,j), bond(2,j), j = 1, int2fit)
00199 11      FORMAT(2i9, 2f12.6, 50f12.6)
00200
00201          IF (merit .LT. minerr) THEN
00202              minerr = merit
00203              DO i = 1, int2fit
00204                  tbbest(i) = bond(1,i)
00205                  tbbestscl(i) = bond(2,i)
00206              ENDDO
00207          ENDIF
00208
00209      ELSE
00210
00211          CALL random_number(rn)
00212
00213          IF ( exp( -(merit - oldmerit)*mcbeta) .GT. rn ) THEN
00214
00215              acc = acc + 1
00216              oldmerit = merit
00217
00218              WRITE(*,11) ii, acc, merit, oldmerit, &
00219                  (bond(1,j), bond(2,j), j = 1, int2fit)
00220
00221          ELSE
00222
00223              DO i = 1, int2fit
00224
00225                  bond(1,i) = tbold(i)
00226                  bond(2,i) = tbsclold(i)
00227
00228              ENDDO
00229
00230          ENDIF
00231
00232      ENDIF
00233
00234  ENDIF
00235
00236      !      WRITE(*,11) II, ACC, MERIT, OLDMERIT, &
00237      !          (BOND(1,J), BOND(2,J), J = 1, INT2FIT)
00238      !      11 FORMAT(2I9, 2F12.6, 50F12.6)
00239
00240  ENDDO
00241
00242  print*, "MINERR = ", minerr
00243
00244  DO i = 1, int2fit
00245      bond(1,i) = tbbest(i)
00246      bond(2,i) = tbbestscl(i)
00247  ENDDO
00248
00249
00250  DO i = 1, int2fit
00251      CALL univtailcoef(bond(:,i))
00252  ENDDO
00253
00254
00255  CALL bldnewhs_sp
00256
00257  IF (qfit .EQ. 0) THEN
00258
00259      CALL diagmyh
00260
00261  ELSE
00262
00263      IF (electro .EQ. 0) THEN
00264          CALL qneutral(0, 1) ! Local charge neutrality
00265      ELSE
00266          CALL qconsistency(0,1) ! Self-consistent charges
00267      ENDIF

```

```

00268
00269
00270   ENDIF
00271
00272
00273   OPEN(unit=20, status="UNKNOWN", file="checkmofit.dat")
00274
00275   DO i = 1, hdim
00276       WRITE(20,10) evals(i), dftmo(i)
00277
00278   ENDDO
00279
00280   DO i = 1, int2fit
00281       print*, "# ", bond(1,i)
00282       WRITE(20,*) "# ", bond(1,i)
00283
00284   ENDDO
00285
00286   WRITE(20,20) (bond(1,i), i = 1, int2fit)
00287
00288   CLOSE(20)
00289
00290 10 FORMAT(f12.3, 2g18.9)
00291 20 FORMAT(100g18.9)
00292
00293   DEALLOCATE(dftmo, tbsclorig, tbsclold, tbold, tborig, tbbest, tbbestsc1)
00294
00295   IF (control .EQ. 1) THEN
00296       ! CALL DEALLOCATEDIAG
00297   ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00298       CALL deallocatepure
00299   ELSEIF (control .EQ. 3) THEN
00300       CALL fermideallocate
00301   ENDIF
00302
00303   IF (electro .EQ. 1) CALL deallocatecoulomb
00304
00305   IF (basistype .EQ. "NONORTHO") CALL deallocatenono
00306
00307   CALL deallocatearrays
00308
00309   RETURN
00310
00311   END SUBROUTINE mofit

```

8.253 mofit_plato.f90 File Reference

Functions/Subroutines

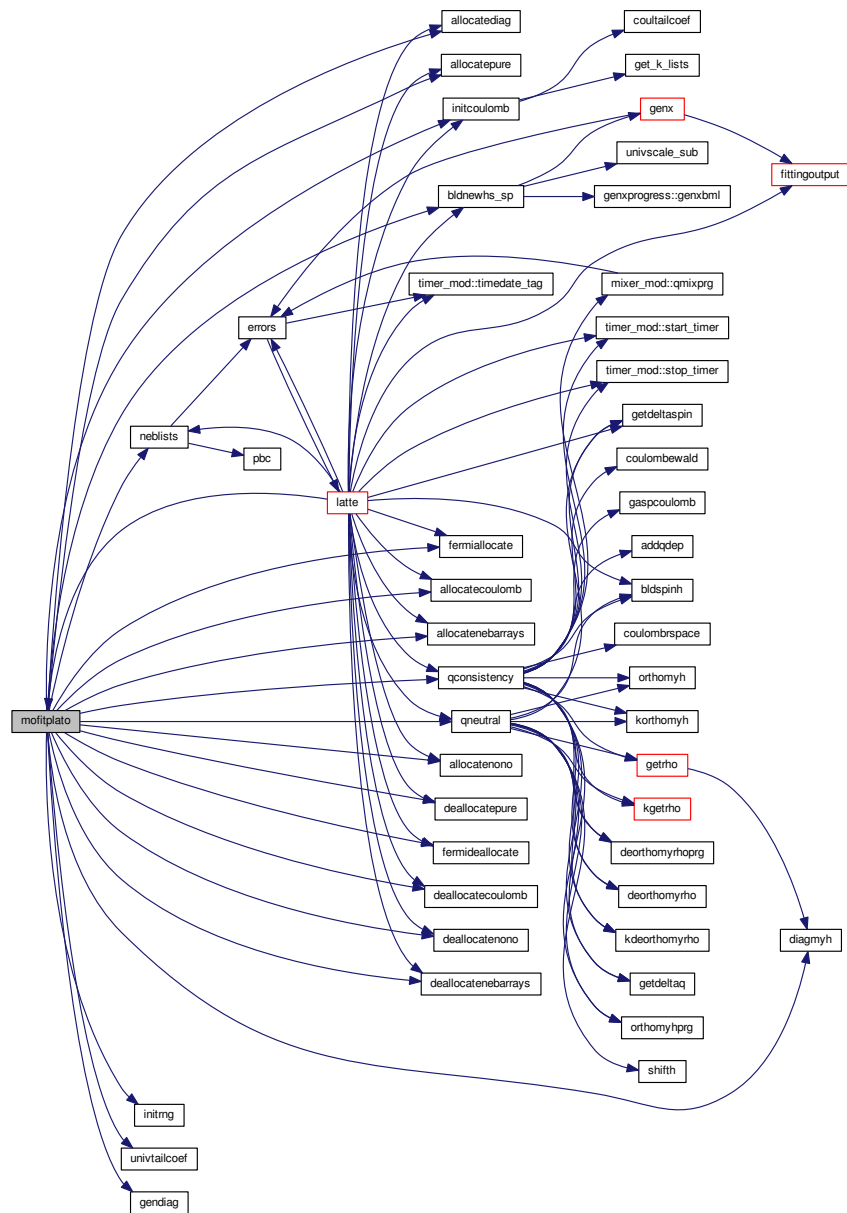
- subroutine [mofitplato](#)

8.253.1 Function/Subroutine Documentation

8.253.1.1 subroutine mofitplato ()

Definition at line 23 of file [mofit_plato.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE mofitplato
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE kspacearray
00028   USE univarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K, JUNK, ACC, II, COUNT
00034   INTEGER :: NSTEP, PICK, CPLOC(1), LOOPTARGET, AB
00035   REAL(LATTEPREC) :: EMIN, EMAX, MERIT, OLDMERIT, RN, ESTEP, EF
00036   REAL(LATTEPREC), ALLOCATABLE :: DFTMO(:), DFTEVEC(:, :)
00037   REAL(LATTEPREC), ALLOCATABLE :: TBDOS(:), TBOLD(:), TBORIG(:)
00038   REAL(LATTEPREC), ALLOCATABLE :: TBSCLOLD(:), TBSCLOLORIG(:)
00039   REAL(LATTEPREC), ALLOCATABLE :: TBBEST(:), TBBESTSCL(:)
00040   REAL(LATTEPREC) :: ENERGY, INTDOS, NUME, JUNKNUM, MINERR, NEWERR
00041   REAL(LATTEPREC) :: DOTMAX, DOT, MYNORM, DOTSUM
00042   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00043   IF (existerror) RETURN
00044
00045   ! Read in the DOS from a VASP calculation
00046
00047   OPEN(unit=20, status="OLD", file="MO2fit_plato.dat")
00048
00049   ALLOCATE(dftmo(hdim), dftevec(hdim,hdim))
00050   ALLOCATE(tbold(int2fit), tborig(int2fit))
00051   ALLOCATE(tbsclold(int2fit), tbsclorig(int2fit))
00052   ALLOCATE(tbbest(int2fit), tbbestscl(int2fit))
00053
00054   DO i = 1, hdim
00055
00056     READ(20,*) dftmo(i)
00057
00058
00059     ! READ(20,*) J, DFTMO(I), JUNKNUM
00060
00061   ENDDO
00062
00063   DO i = 1, hdim
00064     READ(20,*) (dftevec(j,i), j = 1, hdim)
00065   ENDDO
00066
00067   ! Normalize
00068
00069   DO i = 1, hdim
00070
00071     mynorm = 0.00d0
00072
00073     DO j = 1, hdim
00074       mynorm = mynorm + dftevec(j,i)*dftevec(j,i)
00075     ENDDO
00076
00077     mynorm = sqrt(mynorm)
00078     DO j = 1, hdim
00079       dftevec(j,i) = dftevec(j,i)/mynorm
00080     ENDDO
00081   ENDDO
00082
00083
00084
00085   ! DO I = 1, HDIM
00086   !   WRITE(6,'(100F12.6)') (DFTEVEC(J,I), J = 1, HDIM)
00087   ! ENDDO
00088
00089
00090   CLOSE(20)
00091
00092   ! Initialize stuff
00093

```

```

00094
00095 IF (control .EQ. 1) THEN
00096   CALL allocatediag
00097 ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00098   CALL allocatpure
00099 ELSEIF (control .EQ. 3) THEN
00100   CALL fermiallocate
00101 ENDIF
00102
00103 IF (electro .EQ. 1) THEN
00104   CALL allocatecoulomb
00105   CALL initcoulomb
00106 ENDIF
00107
00108 !
00109 ! Allocate stuff for building the neighbor lists
00110 !
00111
00112 CALL allocatenebararrays
00113
00114 CALL neblists(0)
00115
00116 IF (basistype .EQ. "NONORTHO") CALL allocatenono
00117
00118 ! Compute DOS from LATTE and compare
00119
00120 CALL initrng
00121
00122 acc = 0
00123
00124 DO i = 1, int2fit
00125   tborig(i) = bond(1,i)
00126   tbsclorig(i) = bond(2,i)
00127   tbbest(i) = bond(1,i)
00128   tbbestscl(i) = bond(2,i)
00129 ENDDO
00130
00131
00132 DO ii = 1, nfitstep
00133
00134   ! Change set of TB parameters
00135
00136   DO i = 1, int2fit
00137
00138     tbold(i) = bond(1,i)
00139     tbsclold(i) = bond(2,i)
00140   ENDDO
00141
00142   ! Pick one to change
00143
00144
00145   IF (ii .GT. 1) THEN
00146
00147     CALL random_number(rn)
00148
00149     pick = int(rn*REAL(int2fit)) + 1
00150
00151     CALL random_number(rn)
00152
00153     ! AB = INT(RN*2.0D0) + 1
00154     ab = 1
00155     CALL random_number(rn)
00156
00157     ! Change by +/- 20%
00158
00159     bond(ab,pick) = bond(ab,pick) * gaussrn(one, mcsigma)
00160
00161
00162     IF (abs(bond(1,pick)) .LT. 0.01) &
00163       bond(1,pick) = sign(0.01d0, tborig(pick))
00164
00165     CALL univtailcoef(bond(:,pick))
00166
00167   ENDIF
00168
00169   ! Re-build cut-off tails
00170
00171   ! Build H
00172
00173   CALL bldnewhs_sp
00174
00175   IF (qfit .EQ. 0) THEN
00176
00177     ! IF (BASISTYPE .EQ. "NONORTHO") CALL ORTHOMYH
00178
00179     ! CALL DIAGMYH
00180

```

```

00181         IF (basistype .EQ. "ORTHO") THEN
00182             CALL diagmyh
00183         ELSE
00184             CALL gendiag
00185         ENDIF
00186
00187     ELSE
00188
00189         IF (electro .EQ. 0) THEN
00190             CALL qneutral(0, 1) ! Local charge neutrality
00191         ELSE
00192             CALL qconsistency(0,1) ! Self-consistent charges
00193         ENDIF
00194
00195         IF (basistype .EQ. "NONORTHO") CALL gendiag
00196     ENDIF
00197
00198     ! Normalize
00199
00200     ! Normalize
00201
00202
00203
00204
00205     DO i = 1, hdim
00206
00207         mynorm = 0.00d0
00208
00209         DO j = 1, hdim
00210             mynorm = mynorm + evecs(j,i)*evecs(j,i)
00211         ENDDO
00212
00213         mynorm = sqrt(mynorm)
00214         DO j = 1, hdim
00215             evecs(j,i) = evecs(j,i)/mynorm
00216         ENDDO
00217     ENDDO
00218
00219
00220
00221
00222     ! Compute errors - this time match like to like using dot products of eigenvectos
00223
00224     merit = zero
00225
00226     DO i = 1, hdim
00227
00228         dotmax = -1000.0d0
00229
00230         DO j = 1, hdim
00231
00232             dot = 0.0d0
00233             DO k = 1, hdim
00234
00235                 dot = dot + abs(evecs(k,i)*dftvec(k,j))
00236                 !DOT = DOT + EVECS(K,I)*DFTEVEC(K,J)
00237             ENDDO
00238
00239             ! print*, DOT
00240             !IF (DOT .GT. DOTMAX) THEN
00241                 IF (dot .GT. dotmax .AND. evals(i)*dftmo(j) .GT. zero) THEN ! We want the one with the
max dot product
00242                     dotmax = dot
00243
00244                     ! NEWERR = (EVALS(I) - DFTMO(J))*(EVALS(I) - DFTMO(J))
00245                     newerr = abs((evals(i) - dftmo(j))/dftmo(j))
00246
00247                     ! If (I .GT. 5) NEWERR = NEWERR*0.0D0
00248
00249                 ENDIF
00250                 ! PRINT*, DOTMAX
00251
00252             ENDDO
00253
00254             ! PRINT*, DOTMAX
00255
00256             ! NEWERR = ABS(EVALS(I)-DFTMO(I))
00257             !NEWERR = (EVALS(I)-DFTMO(I))*(EVALS(I)-DFTMO(I))
00258
00259             newerr = abs((evals(i) - dftmo(i))/dftmo(i))
00260
00261             ! IF (I .GT. HDIM/2) NEWERR = NEWERR*0.01D0
00262             ! IF (I .GT. 1) NEWERR = ZERO
00263
00264             merit = merit + newerr
00265

```

```

00266      ENDDO
00267
00268      dotsum = 0.0
00269      DO i = 1, hdim
00270          dot = 0.0d0
00271
00272          DO j = 1, hdim
00273              dot = dot + abs(evecs(j,i)*dftevec(j,i))
00274          ENDDO
00275
00276          !PRINT*, I, DOT
00277
00278          dotsum = dotsum + dot
00279
00280      ENDDO
00281
00282      !PRINT*, DOTSUM
00283
00284      ! Penalize not having the states in the correct order
00285
00286      !      MERIT = MERIT*(REAL(HDIM) - DOTSUM)*(REAL(HDIM) - DOTSUM)
00287
00288
00289      !      PRINT*, II, MERIT
00290      !      DO I = 1, HDIM
00291
00292          !          MERIT = MERIT + (EVALS(I) - DFTMO(I))*(EVALS(I) - DFTMO(I))
00293
00294      !      ENDDO
00295
00296      IF (ii .EQ. 1) oldmerit = merit
00297      IF (ii .EQ. 1) minerr = 1.0d9
00298
00299      IF (merit .LT. oldmerit) THEN
00300
00301          acc = acc + 1
00302          oldmerit = merit
00303
00304          WRITE(*,11) ii, acc, merit, oldmerit, &
00305              (bond(1,j), bond(2,j), j = 1, int2fit)
00306 11      FORMAT(2i9, 2f12.6, 50f12.6)
00307
00308          IF (merit .LT. minerr) THEN
00309              minerr = merit
00310              DO i = 1, int2fit
00311                  tbbest(i) = bond(1,i)
00312                  tbbestscl(i) = bond(2,i)
00313              ENDDO
00314          ENDIF
00315
00316      ELSE
00317
00318          CALL random_number(rn)
00319
00320          IF ( exp(-(merit - oldmerit)*mcbeta) .GT. rn ) THEN
00321
00322              acc = acc + 1
00323              oldmerit = merit
00324
00325              WRITE(*,11) ii, acc, merit, oldmerit, &
00326                  (bond(1,j), bond(2,j), j = 1, int2fit)
00327
00328          ELSE
00329
00330              DO i = 1, int2fit
00331
00332                  bond(1,i) = tbold(i)
00333                  bond(2,i) = tbsclold(i)
00334
00335              ENDDO
00336
00337          ENDIF
00338
00339      ENDIF
00340
00341  ENDIF
00342
00343      !      WRITE(*,11) II, ACC, MERIT, OLDMERIT, &
00344      !          (BOND(1,J), BOND(2,J), J = 1, INT2FIT)
00345      !      11 FORMAT(2I9, 2F12.6, 50F12.6)
00346
00347      ENDDO
00348
00349      ! DOTSUM = 0.0
00350      ! DO I = 1, HDIM
00351      !      DOT = 0.0D0
00352

```

```

00353      !      DO J = 1, HDIM
00354      !          DOT = DOT + ABS(EVECS(J,I)*DFTEVEC(J,I))
00355      !      ENDDO
00356
00357      !      PRINT*, I, DOT
00358
00359      !      DOTSUM = DOTSUM + DOT
00360
00361      !  ENDDO
00362
00363      !  PRINT*, DOTSUM
00364
00365
00366      print*, "MINERR = ", minerr
00367
00368      DO i = 1, int2fit
00369          bond(1,i) = tbbest(i)
00370          bond(2,i) = tbbestscl(i)
00371      ENDDO
00372
00373
00374      DO i = 1, int2fit
00375          CALL univtailcoef(bond(:,i))
00376      ENDDO
00377
00378      CALL bldnews_sp
00379
00380      IF (qfit .EQ. 0) THEN
00381
00382          !          IF (BASISTYPE .EQ. "NONORTHO") CALL ORTHOMYH
00383
00384          !          CALL DIAGMYH
00385
00386          IF (basistype .EQ. "ORTHO") THEN
00387              CALL diagmyh
00388          ELSE
00389              CALL gendiag
00390          ENDIF
00391
00392
00393          !CALL DIAGMYH
00394
00395      ELSE
00396
00397          IF (electro .EQ. 0) THEN
00398              CALL qneutral(0, 1) ! Local charge neutrality
00399          ELSE
00400              CALL qconsistency(0,1) ! Self-consistent charges
00401          ENDIF
00402
00403          IF (basistype .EQ. "NONORTHO") CALL gendiag
00404
00405      ENDIF
00406
00407
00408      OPEN(unit=20, status="UNKNOWN", file="checkmofit.dat")
00409
00410      DO i = 1, hdim
00411
00412          WRITE(20,10) evals(i), dftmo(i)
00413
00414      ENDDO
00415
00416      DO i = 1, int2fit
00417
00418          print*, "# ", bond(1,i)
00419          WRITE(20,*) "# ", bond(1,i)
00420
00421      ENDDO
00422
00423      WRITE(20,20) (bond(1,i), i = 1, int2fit), minerr
00424
00425      CLOSE(20)
00426
00427 10 FORMAT(f12.3, 2g18.9)
00428 20 FORMAT(100g18.9)
00429
00430      DEALLOCATE(dftmo, tbsclorig, tbsclold, tbold, tborig, tbbest, tbbestscl)
00431
00432      IF (control .EQ. 1) THEN
00433          !      CALL DEALLOCATEDIAG
00434      ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00435          CALL deallocatepure
00436      ELSEIF (control .EQ. 3) THEN
00437          CALL fermideallocate
00438      ENDIF
00439

```

```
00440  IF (electro .EQ. 1) CALL deallocatecoulomb
00441
00442  IF (basistype .EQ. "NONORTHO") CALL deallocatenono
00443
00444  CALL deallocatebarrays
00445
00446
00447  RETURN
00448
00449  END SUBROUTINE mofitplato
```

8.255 msrelax.f90 File Reference

Functions/Subroutines

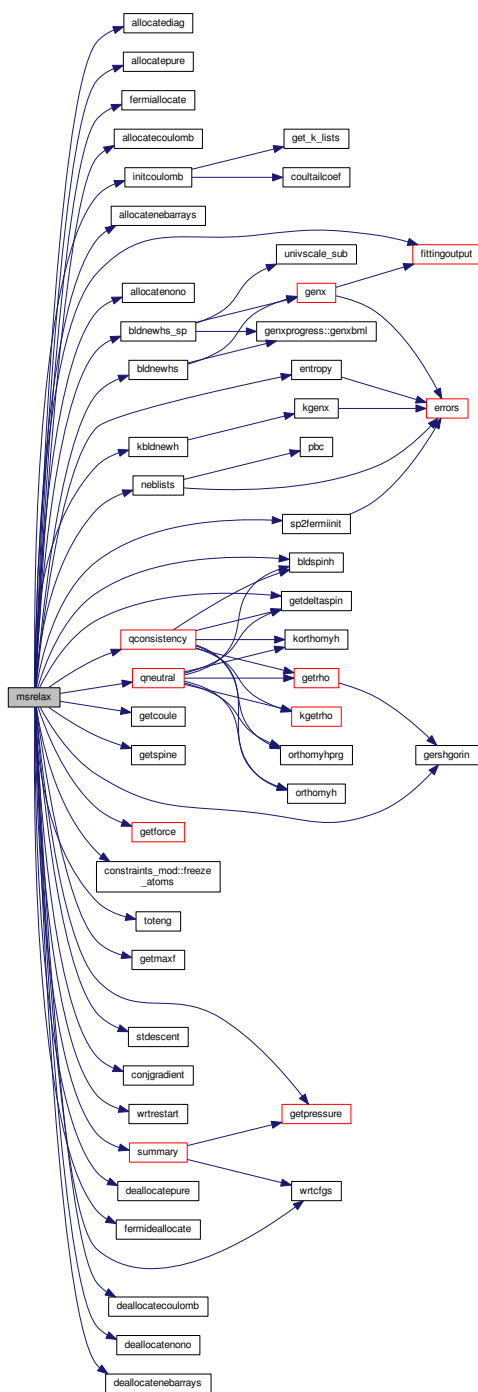
- subroutine [msrelax](#)

8.255.1 Function/Subroutine Documentation

8.255.1.1 subroutine [msrelax](#) ()

Definition at line 23 of file [msrelax.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE msrelax
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE neblistarray
00028   USE coulombarray
00029   USE virialarray
00030   USE relaxcommon
00031   USE myprecision
00032   USE constraints_mod
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: ITER, I, BREAKLOOP
00037   REAL(LATTEPREC) :: MAXF, DELTAF, PREVE, DELTAENERGY, PHI
00038   REAL(LATTEPREC) :: PSCALE
00039   REAL(LATTEPREC), PARAMETER :: MAXPSCALE = 0.01
00040   IF (existerror) RETURN
00041
00042   ! REAL(LATTEPREC) :: PIE = 3.141592654
00043
00044   iter = 0
00045   maxf = 1000000000000.0
00046   ente = zero
00047
00048
00049   OPEN(unit=20, status="UNKNOWN", file="monitorrelax.xyz")
00050
00051   WRITE(20, 10) nats
00052   WRITE(20, ' ("Molecular statics relaxation")')
00053   DO i = 1, nats
00054     WRITE(20,11) atele(i), cr(1,i), cr(2,i), cr(3,i)
00055   ENDDO
00056
00057 10 FORMAT(i6)
00058 11 FORMAT(a2, 1x, 3f24.9)
00059
00060   IF (control .EQ. 1) THEN
00061     CALL allocatediag
00062   ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00063     CALL allocatepure
00064   ELSEIF (control .EQ. 3) THEN
00065     CALL fermiallocate
00066   ENDIF
00067
00068   IF (electro .EQ. 1) THEN
00069     CALL allocatcoulomb
00070     CALL initcoulomb
00071   ENDIF
00072
00073   !
00074   ! Allocate stuff for building the neighbor lists
00075   !
00076
00077   CALL allocatenebararrays
00078
00079   CALL neblists(0)
00080
00081   IF (basistype .EQ. "NONORTHO") CALL allocatenono
00082
00083   ! CALL BLDNEWHS(0)
00084
00085   IF (kon .EQ. 0) THEN
00086
00087     IF (sponly .EQ. 0) THEN
00088       CALL bldnewhs_sp
00089     ELSE
00090       CALL bldnewhs
00091     ENDIF
00092
00093   ELSE

```

```

00094
00095     CALL kbldnewh
00096
00097 ENDIF
00098
00099
00100 IF (spinon .EQ. 1) THEN
00101     CALL getdeltaspin
00102     CALL bldspinh
00103 ENDIF
00104
00105 IF (control .EQ. 5) THEN
00106     CALL gershgorin
00107     CALL sp2fermiinit
00108 ENDIF
00109
00110 IF ( reltype .EQ. "CG" ) ALLOCATE(olddf(3,nats), d1(3,nats),
00111     oldd(3,nats))
00112
00113 WRITE(6,'("# Iteration           Max. Force           Total Energy           Pressure")')
00114
00115 breakloop = 0
00116
00117 IF (reltype .EQ. "SD" .OR. reltype .EQ. "CG") THEN
00118     DO WHILE ( breakloop .EQ. 0)
00119
00120
00121         iter = iter + 1
00122
00123         IF (electro .EQ. 0) THEN
00124             CALL qneutral(0, 1) ! Local charge neutrality
00125         ELSE
00126             CALL qconsistency(0,1) ! Self-consistent charges
00127         ENDIF
00128
00129         IF (electro .EQ. 1) CALL getcoule
00130
00131         espin = zero
00132         IF (spinon .EQ. 1) CALL getspine
00133
00134         IF (control .NE. 1 .AND. control .NE. 2 &
00135             .AND. kbt .GT. 0.000001 ) CALL entropy
00136
00137         CALL getforce ! Get all the forces
00138
00139         IF (freeze .EQ. 1) CALL freeze_atoms(ftot)
00140
00141         preve = tote
00142
00143         CALL toteng
00144
00145         IF (electro .EQ. 0) THEN
00146
00147             tote = trrhoh + erep - ente
00148
00149         ELSEIF (electro .EQ. 1) THEN
00150
00151             tote = trrhoh + erep - ecoul - ente
00152
00153         ENDIF
00154
00155         IF (spinon .EQ. 1) tote = tote + espin
00156
00157         prevf = maxf
00158         CALL getmaxf(maxf)
00159
00160         CALL getpressure
00161
00162
00163         WRITE(6,20) iter, maxf, tote, pressure
00164
00165         FLUSH(6)
00166
00167 20    FORMAT(1x,i6,10x,g14.6,1x,f20.10,1x,f6.2,1x,f12.6)
00168
00169         deltaenergy = tote - preve
00170         deltaf = maxf - prevf
00171
00172         IF ( reltype .EQ. "SD" ) THEN
00173             CALL stdescent(iter, deltaenergy, deltaf)
00174         ELSEIF ( reltype .EQ. "CG" ) THEN
00175             CALL conjgradient(iter, deltaenergy)
00176         ENDIF
00177
00178         IF (abs(maxf) .LE. rlxftol .OR. iter .GT. mxrlx ) breakloop = 1
00179

```

```

00180
00181      !
00182      ! After moving atoms, apply PBCs again
00183      !
00184
00185      IF (mod(iter,25) .EQ. 0) CALL neblists(1)
00186
00187
00188      WRITE(20, 10) nats
00189      WRITE(20, '("Molecular statics relaxation")')
00190      DO i = 1, nats
00191          WRITE(20,11) atele(i), cr(1,i), cr(2,i), cr(3,i)
00192      ENDDO
00193
00194      IF (kon .EQ. 0) THEN
00195
00196          IF (sponly .EQ. 0) THEN
00197              CALL bldnewhs_sp
00198          ELSE
00199              CALL bldnewhs
00200          ENDIF
00201
00202      ELSE
00203
00204          CALL kbldnewh
00205
00206      ENDIF
00207
00208      ENDDO
00209
00210      ELSEIF (reltype .EQ. "VL") THEN
00211
00212          DO WHILE ( breakloop .EQ. 0)
00213
00214              iter = 0
00215
00216              DO WHILE (abs(maxf) .GT. rlxftol .AND. iter .LT. mxrlx)
00217
00218                  iter = iter + 1
00219
00220                  IF (electro .EQ. 0) THEN
00221                      CALL qneutral(0,1)
00222                  ELSE
00223                      CALL qconsistency(0, 1)
00224                  ENDIF
00225
00226                  IF (electro .EQ. 1) CALL getcoule
00227
00228                  espin = zero
00229                  IF (spinon .EQ. 1) CALL getspine
00230
00231                  IF ( control .NE. 1 .AND. control .NE. 2 &
00232                      .AND. kbt .GT. 0.000001 ) CALL entropy
00233
00234                  CALL getforce
00235
00236                  IF (freeze .EQ. 1) CALL freeze_atoms(ftot)
00237
00238                  CALL toteng
00239
00240                  preve = tote
00241
00242                  IF (electro .EQ. 0) THEN
00243
00244                      tote = trrhoh + erep - ente
00245
00246                  ELSEIF (electro .EQ. 1) THEN
00247
00248                      tote = trrhoh + erep - ecoul - ente
00249
00250                  ENDIF
00251
00252                  IF (spinon .EQ. 1) tote = tote + espin
00253
00254                  prevf = maxf
00255                  CALL getmaxf(maxf)
00256
00257                  CALL getpressure
00258
00259                  WRITE(6,21) iter, maxf, tote, pressure, boxdims(1)*
00260                      boxdims(2)*boxdims(3)
00261
00262                  21      FORMAT(1x,i6,10x,g14.6,1x,f20.10,1x,f6.2,1x,f12.6,1x,f12.6 )
00263
00264
00265

```

```

00266         deltaenergy = tote - preve
00267         deltaf = maxf - prevf
00268
00269         CALL stdescent(iter, deltaenergy, deltaf)
00270
00271         IF (mod(iter,25) .EQ. 0) THEN
00272             CALL neblists(1)
00273         ENDIF
00274
00275         WRITE(20, 10) nats
00276         WRITE(20, ' ("Molecular statics relaxation)")')
00277         DO i = 1, nats
00278             WRITE(20,11) atele(i), cr(1,i), cr(2,i), cr(3,i)
00279         ENDDO
00280
00281         IF (kon .EQ. 0) THEN
00282
00283             IF (sponly .EQ. 0) THEN
00284                 CALL bldnewhs_sp
00285             ELSE
00286                 CALL bldnewhs
00287             ENDIF
00288
00289         ELSE
00290
00291             CALL kbldnewh
00292
00293         ENDIF
00294
00295     ENDDO
00296
00297     CALL getpressure
00298
00299     pscale = min(abs(pressure), maxpscale)
00300
00301     pscale = sign(pscale, pressure)
00302
00303     cr = cr * (one + pscale)
00304
00305     box = box * (one + pscale)
00306
00307     boxdims = boxdims * (one + pscale)
00308
00309     CALL neblists(1)
00310
00311     IF (kon .EQ. 0) THEN
00312
00313         IF (sponly .EQ. 0) THEN
00314             CALL bldnewhs_sp
00315         ELSE
00316             CALL bldnewhs
00317         ENDIF
00318
00319     ELSE
00320
00321         CALL kbldnewh
00322
00323     ENDIF
00324
00325     IF (abs(pressure) .LT. 0.01) breakloop = 1
00326
00327 ENDDO
00328
00329 ENDIF
00330
00331 CLOSE(20)
00332
00333 CALL wrtrestart(iter)
00334
00335 CALL wrtcfgs(-999)
00336
00337 IF (control .EQ. 1) THEN
00338     ! CALL DEALLOCATEDIAG
00339 ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00340     CALL deallocatepure
00341 ELSEIF (control .EQ. 3) THEN
00342     CALL fermideallocate
00343 ENDIF
00344
00345 CALL summary
00346 CALL fittingoutput(0)
00347
00348 IF (electro .EQ. 1) CALL deallocatecoulomb
00349
00350 IF (basistype .EQ. "NONORTHO") CALL deallocateonono
00351
00352 CALL deallocatebarrays

```

```

00353
00354   IF ( reltype .EQ. "CG" ) DEALLOCATE( oldf, oldd, dl )
00355
00356
00357   RETURN
00358
00359 END SUBROUTINE msrelax

```

8.257 myprecision.f90 File Reference

Modules

- module [myprecision](#)

Variables

- real(latteprec), parameter [myprecision::eight](#) = 8.0
- real(latteprec), parameter [myprecision::eleven](#) = 11.0
- real(latteprec), parameter [myprecision::fifteen](#) = 15.0
- real(latteprec), parameter [myprecision::five](#) = 5.0
- real(latteprec), parameter [myprecision::fortyeight](#) = 48.0
- real(latteprec), parameter [myprecision::four](#) = 4.0
- real(latteprec), parameter [myprecision::fourteen](#) = 14.0
- real(latteprec), parameter [myprecision::half](#) = 0.5
- complex(latteprec), parameter [myprecision::im](#) = (ZERO, ONE)
- integer, parameter [myprecision::latteprec](#) = 8
- real(latteprec), parameter [myprecision::minusone](#) = -1.0
- real(latteprec), parameter [myprecision::nine](#) = 9.0
- real(latteprec), parameter [myprecision::one](#) = 1.0
- real(latteprec), parameter [myprecision::quarter](#) = 0.25
- complex(latteprec), parameter [myprecision::re](#) = (ONE, ZERO)
- real(latteprec), parameter [myprecision::seven](#) = 7.0
- real(latteprec), parameter [myprecision::six](#) = 6.0
- real(latteprec), parameter [myprecision::sixteen](#) = 16.0
- real(latteprec), parameter [myprecision::sqrt2](#) = SQRT(2.0D0)
- real(latteprec), parameter [myprecision::ten](#) = 10.0
- real(latteprec), parameter [myprecision::third](#) = 1.0D0/3.0D0
- real(latteprec), parameter [myprecision::thousand](#) = 1000.0
- real(latteprec), parameter [myprecision::three](#) = 3.0
- real(latteprec), parameter [myprecision::threequart](#) = 0.75
- real(latteprec), parameter [myprecision::twelve](#) = 12.0
- real(latteprec), parameter [myprecision::twenty](#) = 20.0
- real(latteprec), parameter [myprecision::twentyfour](#) = 24.0
- real(latteprec), parameter [myprecision::twentysix](#) = 26.0
- real(latteprec), parameter [myprecision::two](#) = 2.0
- real(latteprec), parameter [myprecision::zero](#) = 0.0

8.258 myprecision.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE myprecision
00023
00024     IMPLICIT NONE
00025     SAVE
00026
00027     !
00028     ! The precision for the calculation (double = 8, real = 4)
00029     ! Selected during the preprocessing of myprecision.F
00030     !
00031
00032     #ifndef DOUBLEPREC
00033     INTEGER, PARAMETER :: latteprec = 8
00034     #elif defined(SINGLEPREC)
00035     INTEGER, PARAMETER :: latteprec = 4
00036     #endif
00037
00038     !
00039     ! In the case of single precision, we assume (uh uh!) that the compiler
00040     ! will round our doubles down to singles for us...
00041     !
00042
00043     REAL(LATTEPREC), PARAMETER :: zero = 0.0, one = 1.0, two = 2.0
00044     REAL(LATTEPREC), PARAMETER :: three = 3.0, four = 4.0, five = 5.0
00045     REAL(LATTEPREC), PARAMETER :: six = 6.0, seven = 7.0, eight = 8.0
00046     REAL(LATTEPREC), PARAMETER :: nine = 9.0, ten = 10.0, eleven = 11.0
00047     REAL(LATTEPREC), PARAMETER :: twelve = 12.0, fourteen = 14.0,
    fifteen = 15.0
00048     REAL(LATTEPREC), PARAMETER :: sixteen = 16.0, twenty = 20.0
00049     REAL(LATTEPREC), PARAMETER :: twentyfour = 24.0, twentysix = 26.0
00050     REAL(LATTEPREC), PARAMETER :: fortyeight = 48.0
00051     REAL(LATTEPREC), PARAMETER :: half = 0.5, third = 1.0d0/3.0d0
00052     REAL(LATTEPREC), PARAMETER :: quarter = 0.25, threequart = 0.75
00053     REAL(LATTEPREC), PARAMETER :: minusone = -1.0, thousand = 1000.0
00054     REAL(LATTEPREC), PARAMETER :: sqrt2 = sqrt(2.0d0)
00055     COMPLEX(LATTEPREC), PARAMETER :: re = (one, zero), im = (zero,
    one)
00056
00057 END MODULE myprecision

```

8.259 neblast_cell.f90 File Reference

Functions/Subroutines

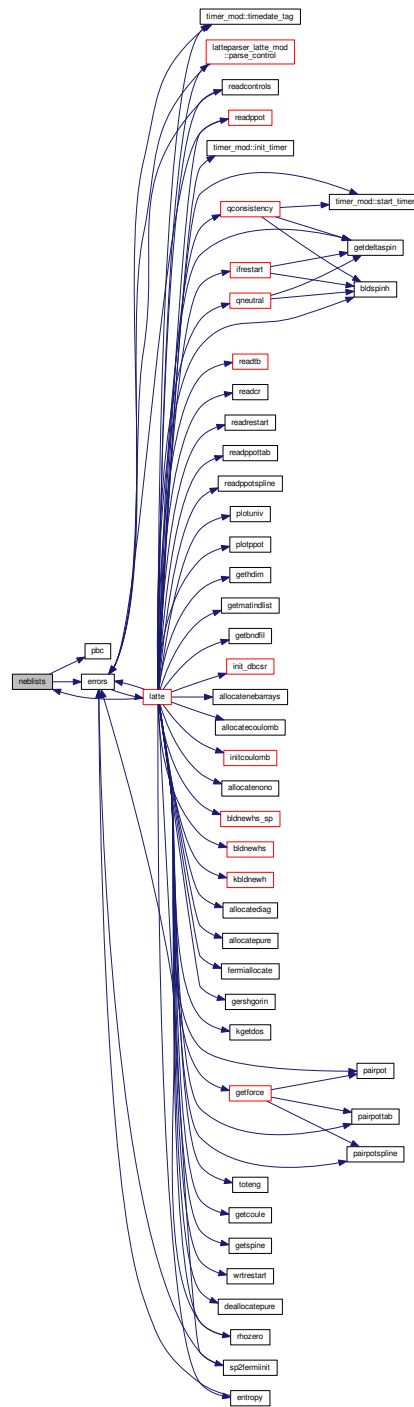
- subroutine [neblists](#) (AMIALLO)

8.259.1 Function/Subroutine Documentation

8.259.1.1 subroutine neblists (integer AMIALLO)

Definition at line 23 of file [neblast_cell.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE neblis(AMIALLO)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE ppotarray
00028   USE neblisarray
00029   USE coulombarray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, L, M
00035   INTEGER :: II, JJ, KK
00036   INTEGER :: AMIALLO
00037   INTEGER :: X RANGE, Y RANGE, Z RANGE
00038   INTEGER :: MAXNEBTB, MAXNEBPP, MAXNEBCOUL
00039   INTEGER :: NX, NY, NZ, XCELL, YCELL, ZCELL, NUMCELLS
00040   INTEGER :: PBCI, PBCJ, PBCK, ATOMJ, CELLD, NEB
00041   INTEGER, ALLOCATABLE :: TOTPERCELL(:), CELLATOMID(:, :, :)
00042   REAL(LATTEPREC) :: X, Y, Z, R2
00043   REAL(LATTEPREC) :: MINCRX, MINCRY, MINCRZ
00044   REAL(LATTEPREC), PARAMETER :: MINR = 0.01
00045   REAL(LATTEPREC), SAVE :: MAXCUTTB, MAXCUTPP, MAXCUTCUL
00046
00047   IF (pbcon .EQ. 1) CALL pbc
00048
00049   ! Ensure all coordinates are > 0.0 when building the list.
00050   ! We will shift them back when we're done
00051
00052   mincrx = minval(cr(1,:))
00053   mincry = minval(cr(2,:))
00054   mincrz = minval(cr(3,:))
00055
00056   DO i = 1, nats
00057     cr(1,i) = cr(1,i) - mincrx
00058     cr(2,i) = cr(2,i) - mincry
00059     cr(3,i) = cr(3,i) - mincrz
00060   ENDDO
00061
00062   ! First pass: set up cut-offs
00063
00064   print*, "1"
00065   IF (amiallo .EQ. 0) THEN
00066     maxcuttb = zero
00067     maxcutpp = zero
00068
00069     !$OMP PARALLEL DO DEFAULT(NONE) &
00070     !$OMP SHARED(NATS, NOINT, ATELE, ELE1, ELE2, BOND, OVERL) &
00071     !$OMP SHARED(BASISTYPE, PPOTON, NOPPS, POTCOEF, PPELE1, PPELE2) &
00072     !$OMP PRIVATE(I, J, K) &
00073     !$OMP REDUCTION(MAX: MAXCUTTB, MAXCUTPP)
00074
00075     DO i = 1, nats
00076       DO j = i, nats
00077         DO k = 1, noint
00078           IF ( (atele(i) .EQ. ele1(k) .AND. &
00079                atele(j) .EQ. ele2(k)) .OR. &
00080                (atele(j) .EQ. ele1(k) .AND. &
00081                 atele(i) .EQ. ele2(k)) ) THEN
00082             maxcuttb = max(bond(8,k), maxcuttb)
00083             IF (basistype .EQ. "NONORTHO") &
00084               maxcuttb = max(overl(8,k), maxcuttb)
00085           ENDIF
00086         ENDDO
00087       ENDDO
00088     ENDDO
00089   ENDIF
00090
00091   ENDDO
00092
00093

```

```

00094         IF (ppoton .EQ. 1) THEN
00095
00096             DO k = 1, nopps
00097
00098                 IF ( (atele(i) .EQ. ppele1(k) .AND. &
00099                     atele(j) .EQ. ppele2(k)) .OR. &
00100                     (atele(j) .EQ. ppele1(k) .AND. &
00101                     atele(i) .EQ. ppele2(k)) ) THEN
00102
00103                     maxcutpp = max(potcoef(10,k), maxcutpp)
00104
00105                 ENDIF
00106             ENDDO
00107         ENDDO
00108     ENDIF
00109 ENDIF
00110 ENDDO
00111 ENDDO
00112
00113 !$OMP END PARALLEL DO
00114
00115 print*, "2"
00116 maxcuttb = maxcuttb + skin
00117 maxcutpp = maxcutpp + skin
00118 maxcutcoul = coulcut + skin
00119
00120 ! We will figure out good values for these array dimensions later
00121
00122 maxdimtb = 0
00123 maxdimpp = 0
00124 maxdimcoul = 0
00125 ELSE
00126
00127     !
00128     ! Allow the arrays to grow. It's pointless to let them shrink.
00129     !
00130
00131     IF (maxdimtb .GT. prevdimtb) THEN
00132         DEALLOCATE(nebtb)
00133         ALLOCATE(nebtb( 4, maxdimtb, nats ))
00134     ENDIF
00135
00136     IF (ppoton .EQ. 1) THEN
00137         IF (maxdimpp .GT. prevdimpp) THEN
00138             DEALLOCATE(nebpp)
00139             ALLOCATE(nebpp( 4, maxdimpp, nats ))
00140         ENDIF
00141     ENDIF
00142
00143     IF (electro .EQ. 1) THEN
00144         IF (maxdimcoul .GT. prevdimcoul) THEN
00145             DEALLOCATE(nebcoul)
00146             ALLOCATE(nebcoul( 4, maxdimcoul, nats))
00147         ENDIF
00148     ENDIF
00149
00150 ENDIF
00151
00152 print*, "3"
00153 ! TB
00154
00155 IF (pbcon .EQ. 1) THEN
00156
00157     xrange = int(maxcuttb/boxdims(1)) + 1
00158     yrange = int(maxcuttb/boxdims(2)) + 1
00159     zrange = int(maxcuttb/boxdims(3)) + 1
00160
00161     nx = int(boxdims(1)*(one + REAL(xrange))/maxcuttb) + 1
00162     ny = int(boxdims(2)*(one + REAL(yrange))/maxcuttb) + 1
00163     nz = int(boxdims(3)*(one + REAL(zrange))/maxcuttb) + 1
00164
00165     numcells = (nx+1) * (ny+1) * (nz+1)
00166
00167 ELSE
00168
00169     xrange = 0
00170     yrange = 0
00171     zrange = 0
00172
00173     nx = int(maxval(cr(1,:))/maxcuttb) + 1
00174     ny = int(maxval(cr(2,:))/maxcuttb) + 1
00175     nz = int(maxval(cr(3,:))/maxcuttb) + 1
00176
00177     numcells = nx * ny * nz
00178
00179 ENDIF
00180

```

```

00181  ALLOCATE(totpercell(numcells), cellatomid(4, 200, numcells))
00182
00183  ! Put all the atoms in their cell
00184
00185  totpercell = 0
00186
00187  print*, "4"
00188
00189  IF (pbcon.EQ. 1) THEN
00190
00191      DO ii = -xrange, xrange
00192          DO jj = -yrange, yrange
00193              DO kk = -zrange, zrange
00194
00195                  DO i = 1, nats
00196
00197                      IF (cr(1,i) + REAL(ii)*BOXDIMS(1) .LT. ZERO .AND. &
00198                          CR(1,i) + REAL(ii)*BOXDIMS(1) .GT. -MAXCUTTB) then
00199                          xcell = 0
00200                      ELSEIF (cr(1,i) + REAL(ii)*BOXDIMS(1) .GE. ZERO) then
00201                          xcell = int((cr(1,i) + REAL(ii)*BOXDIMS(1))/maxcuttb) + 1
00202                      ELSE
00203                          xcell = -1
00204                      ENDIF
00205
00206                      IF (cr(2,i) + REAL(jj)*BOXDIMS(2) .LT. ZERO .AND. &
00207                          CR(2,i) + REAL(jj)*BOXDIMS(2) .GT. -MAXCUTTB) then
00208                          ycell = 0
00209                      ELSEIF (cr(2,i) + REAL(jj)*BOXDIMS(2) .GE. ZERO) then
00210                          ycell = int((cr(2,i) + REAL(jj)*BOXDIMS(2))/maxcuttb) + 1
00211                      ELSE
00212                          ycell = -1
00213                      ENDIF
00214
00215                      IF (cr(3,i) + REAL(kk)*BOXDIMS(3) .LT. ZERO .AND. &
00216                          CR(3,i) + REAL(kk)*BOXDIMS(3) .GT. -MAXCUTTB) then
00217                          zcell = 0
00218                      ELSEIF (cr(3,i) + REAL(kk)*BOXDIMS(3) .GE. ZERO) then
00219                          zcell = int((cr(3,i) + REAL(kk)*BOXDIMS(3))/maxcuttb) + 1
00220                      ELSE
00221                          zcell = -1
00222                      ENDIF
00223
00224                      IF (xcell .GE. 0 .AND. ycell .GE. 0 .AND. zcell .GE. 0 &
00225                          .AND. xcell .LE. nx .AND. ycell .LE. ny .AND. &
00226                          zcell .LE. nz) THEN
00227
00228                          cellid = xcell + (nx+1)*(ycell-1) + (nx+1)*(ny+1)*(zcell-1)
00229                          cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00230
00231                          totpercell(cellid) = totpercell(cellid) + 1
00232                          cellatomid(1, totpercell(cellid), cellid) = i
00233                          cellatomid(2, totpercell(cellid), cellid) = ii
00234                          cellatomid(3, totpercell(cellid), cellid) = jj
00235                          cellatomid(4, totpercell(cellid), cellid) = kk
00236
00237                      ENDIF
00238
00239                  ENDDO
00240              ENDDO
00241          ENDDO
00242      ENDDO
00243  ELSE
00244
00245      DO i = 1, nats
00246
00247          xcell = int(cr(1,i)/maxcuttb) + 1
00248          ycell = int(cr(2,i)/maxcuttb) + 1
00249          zcell = int(cr(3,i)/maxcuttb) + 1
00250
00251          cellid = xcell + nx*(ycell-1) + ny*nx*(zcell-1)
00252
00253          totpercell(cellid) = totpercell(cellid) + 1
00254          cellatomid(1, totpercell(cellid), cellid) = i
00255          cellatomid(2, totpercell(cellid), cellid) = 0
00256          cellatomid(3, totpercell(cellid), cellid) = 0
00257          cellatomid(4, totpercell(cellid), cellid) = 0
00258
00259      ENDDO
00260  ENDIF
00261
00262  print*, "5"
00263
00264  ! On the first pass through we can use the cell list

```

```

00268      ! to figure out how to dimension the neighbor list arrays
00269
00270      IF (amiallo .EQ. 0) THEN
00271
00272          IF (pbcon .EQ. 1) THEN
00273
00274              !$OMP PARALLEL DO DEFAULT(NONE) &
00275              !$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
00276              !$OMP SHARED(NX, NY, NZ, MAXCUTTB) &
00277              !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00278              !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2, NEB) &
00279              !$OMP REDUCTION(MAX: MAXDIMTB)
00280
00281              DO i = 1, nats
00282
00283                  neb = 0
00284
00285                  ! Find its cell
00286
00287                  xcell = int(cr(1,i)/maxcuttb) + 1
00288                  ycell = int(cr(2,i)/maxcuttb) + 1
00289                  zcell = int(cr(3,i)/maxcuttb) + 1
00290
00291                  ! Loop over all neighboring cells including its own cell
00292
00293                  DO kk = zcell - 1, zcell + 1
00294                      DO jj = ycell - 1, ycell + 1
00295                          DO ii = xcell - 1, xcell + 1
00296
00297                              cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
00298                              cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00299
00300                              DO j = 1, totpercell(cellid)
00301
00302                                  atomj = cellatomid(1, j, cellid)
00303                                  pbci = cellatomid(2, j, cellid)
00304                                  pbcj = cellatomid(3, j, cellid)
00305                                  pbck = cellatomid(4, j, cellid)
00306
00307                                  x = cr(1,atomj) + REAL(pbci)*BOXDIMS(1) - CR(1,i)
00308                                  y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)
00309                                  z = cr(3,atomj) + REAL(pbck)*BOXDIMS(3) - CR(3,i)
00310
00311                                  r2 = x*x + y*y + z*z
00312
00313                                  IF (r2 .LT. maxcuttb*maxcuttb .AND. &
00314                                      r2 .GT. minr) neb = neb + 1
00315
00316                                  ENDDO
00317
00318                              ENDDO
00319                          ENDDO
00320                      ENDDO
00321
00322                      maxdimtb = max(neb, maxdimtb)
00323
00324                  ENDDO
00325
00326              !$OMP END PARALLEL DO
00327
00328          ELSE ! NO PBC
00329
00330              !$OMP PARALLEL DO DEFAULT(NONE) &
00331              !$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
00332              !$OMP SHARED(NX, NY, NZ, MAXCUTTB) &
00333              !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00334              !$OMP PRIVATE(X, Y, Z, R2, NEB) &
00335              !$OMP REDUCTION(MAX: MAXDIMTB)
00336
00337              DO i = 1, nats
00338
00339                  neb = 0
00340
00341                  xcell = int(cr(1,i)/maxcuttb) + 1
00342                  ycell = int(cr(2,i)/maxcuttb) + 1
00343                  zcell = int(cr(3,i)/maxcuttb) + 1
00344
00345                  DO kk = zcell - 1, zcell + 1
00346                      DO jj = ycell - 1, ycell + 1
00347                          DO ii = xcell - 1, xcell + 1
00348
00349                              IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
00350                                  .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
00351                                  kk .LE. nz) THEN
00352
00353                                  cellid = ii + nx*(jj-1) + ny*nx*(kk-1)

```

```

00355
00356      DO j = 1, totpercell(cellid)
00357
00358          atomj = cellatomid(1, j, cellid)
00359
00360          x = cr(1,atomj) - cr(1,i)
00361          y = cr(2,atomj) - cr(2,i)
00362          z = cr(3,atomj) - cr(3,i)
00363
00364          r2 = x*x + y*y + z*z
00365
00366          IF (r2 .LT. maxcuttb*maxcuttb .AND. &
00367              r2 .GT. minr) neb = neb + 1
00368
00369      ENDDO
00370  ENDDIF
00371  ENDDO
00372  ENDDO
00373  ENDDO
00374
00375      maxdimtb = max(neb, maxdimtb)
00376
00377  ENDDO
00378
00379  !$OMP END PARALLEL DO
00380
00381  ENDDIF
00382
00383      maxdimtb = int(1.5*REAL(maxdimtb))
00384
00385      ALLOCATE( nebtb(4, maxdimtb, nats) )
00386
00387  ENDDIF
00388  print*, "6"
00389
00390
00391  ! Start with atom 1
00392
00393  totnebtb = 0
00394
00395  IF (pbcon .EQ. 1) THEN
00396
00397      !$OMP PARALLEL DO DEFAULT(NONE) &
00398      !$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTTB, TOTPERCELL, CELLATOMID) &
00399      !$OMP SHARED(NEBTB, TOTNEBTB, NX, NY, NZ, MAXDIMTB, PBCON) &
00400      !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00401      !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2)
00402
00403      DO i = 1, nats
00404
00405          ! Find its cell
00406
00407          xcell = int(cr(1,i)/maxcuttb) + 1
00408          ycell = int(cr(2,i)/maxcuttb) + 1
00409          zcell = int(cr(3,i)/maxcuttb) + 1
00410
00411          ! PRINT*, I, XCELL, YCELL, ZCELL
00412
00413          ! Loop over all neighboring cells including its own cell
00414
00415          DO kk = zcell - 1, zcell + 1
00416              DO jj = ycell - 1, ycell + 1
00417                  DO ii = xcell - 1, xcell + 1
00418
00419                      cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
00420                      cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00421
00422                      DO j = 1, totpercell(cellid)
00423
00424                          atomj = cellatomid(1, j, cellid)
00425                          pbcj = cellatomid(2, j, cellid)
00426                          pbcj = cellatomid(3, j, cellid)
00427                          pbck = cellatomid(4, j, cellid)
00428
00429                          x = cr(1,atomj) + REAL(pbcj)*BOXDIMS(1) - CR(1,i)
00430                          y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)
00431                          z = cr(3,atomj) + REAL(pbcj)*BOXDIMS(3) - CR(3,i)
00432
00433                          r2 = x*x + y*y + z*z
00434
00435                          IF (r2 .LT. maxcuttb*maxcuttb .AND. r2 .GT. minr) THEN
00436
00437                              totnebtb(i) = totnebtb(i) + 1
00438
00439                              IF (totnebtb(i) .GT. maxdimtb) THEN
00440                                  CALL errors("neblast_cell","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
(TB) ")

```

```

00441                                ENDIF
00442
00443                                nebtb( 1, totnebtb(i), i ) = atomj
00444                                nebtb( 2, totnebtb(i), i ) = pbcj
00445                                nebtb( 3, totnebtb(i), i ) = pbcj
00446                                nebtb( 4, totnebtb(i), i ) = pbck
00447
00448                                ENDIF
00449
00450                                ENDDO
00451
00452                                ENDDO
00453                                ENDDO
00454                                ENDDO
00455                                ENDDO
00456                                !$OMP END PARALLEL DO
00457
00458                                ELSE ! NO PBC
00459
00460                                !$OMP PARALLEL DO DEFAULT(NONE) &
00461                                !$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTTB, TOTPERCELL, CELLATOMID) &
00462                                !$OMP SHARED(NEBTB, TOTNEBTB, NX, NY, NZ, MAXDIMTB) &
00463                                !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00464                                !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2)
00465
00466                                DO i = 1, nats
00467
00468                                    xcell = int(cr(1,i)/maxcuttb) + 1
00469                                    ycell = int(cr(2,i)/maxcuttb) + 1
00470                                    zcell = int(cr(3,i)/maxcuttb) + 1
00471
00472                                    DO kk = zcell - 1, zcell + 1
00473                                        DO jj = ycell - 1, ycell + 1
00474                                            DO ii = xcell - 1, xcell + 1
00475
00476                                                IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
00477                                                    .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
00478                                                    kk .LE. nz) THEN
00479
00480                                                    cellid = ii + nx*(jj-1) + ny*nx*(kk-1)
00481
00482                                                    DO j = 1, totpercell(cellid)
00483
00484                                                        atomj = cellatomid(1, j, cellid)
00485
00486                                                        x = cr(1,atomj) - cr(1,i)
00487                                                        y = cr(2,atomj) - cr(2,i)
00488                                                        z = cr(3,atomj) - cr(3,i)
00489
00490                                                        r2 = x*x + y*y + z*z
00491
00492                                                        IF (r2 .LT. maxcuttb*maxcuttb .AND. r2 .GT. minr) THEN
00493
00494                                                            totnebtb(i) = totnebtb(i) + 1
00495
00496                                                            IF (totnebtb(i) .GT. maxdimtb) THEN
00497                                                                CALL errors("neblast_cell", "NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
00498                                                                (TB) ")
00499
00500                                                                ENDF
00501
00502                                                                nebtb( 1, totnebtb(i), i ) = atomj
00503                                                                nebtb( 2, totnebtb(i), i ) = 0
00504                                                                nebtb( 3, totnebtb(i), i ) = 0
00505                                                                nebtb( 4, totnebtb(i), i ) = 0
00506
00507                                                                ENDF
00508
00509                                                                ENDDO
00510                                                                ENDF
00511                                                                ENDDO
00512                                                                ENDDO
00513                                                                ENDDO
00514                                                                ENDDO
00515
00516                                                                !$OMP END PARALLEL DO
00517
00518                                ENDF
00519                                print*, "7"
00520                                ! print*, "here"
00521                                DEALLOCATE(totpercell, cellatomid)
00522
00523                                ! Now the list for the pair potential
00524
00525                                IF (ppoton .EQ. 1) THEN
00526

```

```

00527     IF (pbcon .EQ. 1) THEN
00528
00529         xrange = int(maxcutpp/boxdims(1)) + 1
00530         yrange = int(maxcutpp/boxdims(2)) + 1
00531         zrange = int(maxcutpp/boxdims(3)) + 1
00532
00533         nx = int(boxdims(1)*(one + REAL(xrange))/maxcutpp) + 1
00534         ny = int(boxdims(2)*(one + REAL(yrange))/maxcutpp) + 1
00535         nz = int(boxdims(3)*(one + REAL(zrange))/maxcutpp) + 1
00536
00537         numcells = (nx+1) * (ny+1) * (nz+1)
00538
00539     ELSE
00540
00541         xrange = 0
00542         yrange = 0
00543         zrange = 0
00544
00545         nx = int(maxval(cr(1,:))/maxcutpp) + 1
00546         ny = int(maxval(cr(2,:))/maxcutpp) + 1
00547         nz = int(maxval(cr(3,:))/maxcutpp) + 1
00548
00549         numcells = nx * ny * nz
00550
00551     ENDIF
00552
00553     ALLOCATE(totpercell(numcells), cellatomid(4, nats, numcells))
00554
00555     ! Put all the atoms in their cell
00556
00557     totpercell = 0
00558
00559     IF (pbcon .EQ. 1) THEN
00560
00561         DO ii = -xrange, xrange
00562             DO jj = -yrange, yrange
00563                 DO kk = -zrange, zrange
00564
00565                     DO i = 1, nats
00566
00567                         IF (cr(1,i) + REAL(ii)*BOXDIMS(1) .LT. ZERO .AND. &
00568                             CR(1,i) + REAL(ii)*BOXDIMS(1) .GT. -MAXCUTPP) then
00569                             xcell = 0
00570                         ELSEIF (cr(1,i) + REAL(ii)*BOXDIMS(1) .GE. ZERO) then
00571                             xcell = int((cr(1,i) + REAL(ii)*BOXDIMS(1))/maxcutpp) + 1
00572                         ELSE
00573                             xcell = -1
00574                         ENDIF
00575
00576                         IF (cr(2,i) + REAL(jj)*BOXDIMS(2) .LT. ZERO .AND. &
00577                             CR(2,i) + REAL(jj)*BOXDIMS(2) .GT. -MAXCUTPP) then
00578                             ycell = 0
00579                         ELSEIF (cr(2,i) + REAL(jj)*BOXDIMS(2) .GE. ZERO) then
00580                             ycell = int((cr(2,i) + REAL(jj)*BOXDIMS(2))/maxcutpp) + 1
00581                         ELSE
00582                             ycell = -1
00583                         ENDIF
00584
00585                         IF (cr(3,i) + REAL(kk)*BOXDIMS(3) .LT. ZERO .AND. &
00586                             CR(3,i) + REAL(kk)*BOXDIMS(3) .GT. -MAXCUTPP) then
00587                             zcell = 0
00588                         ELSEIF (cr(3,i) + REAL(kk)*BOXDIMS(3) .GE. ZERO) then
00589                             zcell = int((cr(3,i) + REAL(kk)*BOXDIMS(3))/maxcutpp) + 1
00590                         ELSE
00591                             zcell = -1
00592                         ENDIF
00593
00594                         IF (xcell .GE. 0 .AND. ycell .GE. 0 .AND. zcell .GE. 0 &
00595                             .AND. xcell .LE. nx .AND. ycell .LE. ny .AND. &
00596                             zcell .LE. nz) THEN
00597
00598                             cellid = xcell + (nx+1)*(ycell-1) + (nx+1)*(ny+1)*(zcell-1)
00599                             cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00600
00601                             totpercell(cellid) = totpercell(cellid) + 1
00602                             cellatomid(1, totpercell(cellid), cellid) = i
00603                             cellatomid(2, totpercell(cellid), cellid) = ii
00604                             cellatomid(3, totpercell(cellid), cellid) = jj
00605                             cellatomid(4, totpercell(cellid), cellid) = kk
00606
00607
00608                             ENDDO
00609
00610                         ENDDO
00611
00612                     ENDDO
00613                 ENDDO

```

```

00614
00615     ELSE
00616
00617         DO i = 1, nats
00618
00619             xcell = int(cr(1,i)/maxcutpp) + 1
00620             ycell = int(cr(2,i)/maxcutpp) + 1
00621             zcell = int(cr(3,i)/maxcutpp) + 1
00622
00623             cellid = xcell + nx*(ycell-1) + ny*nx*(zcell-1)
00624
00625             totpercell(cellid) = totpercell(cellid) + 1
00626             cellatomid(1, totpercell(cellid), cellid) = i
00627             cellatomid(2, totpercell(cellid), cellid) = 0
00628             cellatomid(3, totpercell(cellid), cellid) = 0
00629             cellatomid(4, totpercell(cellid), cellid) = 0
00630
00631         ENDDO
00632
00633     ENDIF
00634
00635     print *, "8"
00636     !PRINT*, "MAX CELL = ", MAXVAL(TOTPERCELL)
00637     ! First pass through - figure out dimension of neighborlist array
00638
00639     IF (amiallo .EQ. 0) THEN
00640
00641         IF (pbcon .EQ. 1) THEN
00642
00643             print *, "MAX CELL =", maxval(totpercell)
00644
00645             !$OMP PARALLEL DO DEFAULT(NONE) &
00646             !$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
00647             !$OMP SHARED(NX, NY, NZ, MAXCUTPP) &
00648             !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00649             !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2, NEB) &
00650             !$OMP REDUCTION(MAX: MAXDIMP)
00651
00652             DO i = 1, nats
00653
00654                 neb = 0
00655
00656                 ! Find its cell
00657
00658                 xcell = int(cr(1,i)/maxcutpp) + 1
00659                 ycell = int(cr(2,i)/maxcutpp) + 1
00660                 zcell = int(cr(3,i)/maxcutpp) + 1
00661
00662                 ! Loop over all neighboring cells including its own cell
00663
00664                 DO kk = zcell - 1, zcell + 1
00665                     DO jj = ycell - 1, ycell + 1
00666                         DO ii = xcell - 1, xcell + 1
00667
00668                             cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
00669                             cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00670
00671                             DO j = 1, totpercell(cellid)
00672
00673                                 atomj = cellatomid(1, j, cellid)
00674                                 pbcj = cellatomid(2, j, cellid)
00675                                 pbcj = cellatomid(3, j, cellid)
00676                                 pbck = cellatomid(4, j, cellid)
00677
00678                                 x = cr(1,atomj) + REAL(pbcj)*BOXDIMS(1) - CR(1,i)
00679                                 y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)
00680                                 z = cr(3,atomj) + REAL(pbck)*BOXDIMS(3) - CR(3,i)
00681
00682                                 r2 = x*x + y*y + z*z
00683
00684                                 IF (r2 .LT. maxcutpp*maxcutpp .AND. &
00685                                     r2 .GT. minr) neb = neb + 1
00686
00687                             ENDDO
00688                         ENDDO
00689                     ENDDO
00690                 ENDDO
00691                 maxdimp = max(neb, maxdimp)
00692
00693             ENDDO
00694
00695             !$OMP END PARALLEL DO
00696
00697             ELSE ! NO PBC
00700

```



```

00701 !!$OMP PARALLEL DO DEFAULT(NONE) &
00702 !!$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
00703 !!$OMP SHARED(NX, NY, NZ, MAXCUTPP) &
00704 !!$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00705 !!$OMP PRIVATE(X, Y, Z, R2, NEB) &
00706 !!$OMP REDUCTION(MAX: MAXDIMPP)
00707
00708     DO i = 1, nats
00709
00710         neb = 0
00711
00712         xcell = int(cr(1,i)/maxcutpp) + 1
00713         ycell = int(cr(2,i)/maxcutpp) + 1
00714         zcell = int(cr(3,i)/maxcutpp) + 1
00715
00716         DO kk = zcell - 1, zcell + 1
00717             DO jj = ycell - 1, ycell + 1
00718                 DO ii = xcell - 1, xcell + 1
00719
00720                     IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
00721                         .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
00722                         kk .LE. nz) THEN
00723
00724
00725                         cellid = ii + nx*(jj-1) + ny*nx*(kk-1)
00726
00727                         DO j = 1, totpercell(cellid)
00728
00729                             atomj = cellatomid(1, j, cellid)
00730
00731                             x = cr(1,atomj) - cr(1,i)
00732                             y = cr(2,atomj) - cr(2,i)
00733                             z = cr(3,atomj) - cr(3,i)
00734
00735                             r2 = x*x + y*y + z*z
00736
00737                             IF (r2 .LT. maxcutpp*maxcutpp .AND. &
00738                                 r2 .GT. minr) neb = neb + 1
00739
00740                         ENDDO
00741                     ENDIF
00742                 ENDDO
00743             ENDDO
00744         ENDDO
00745
00746         maxdimpp = max(neb, maxdimpp)
00747
00748     ENDDO
00749
00750 !!$OMP END PARALLEL DO
00751
00752     ENDIF
00753
00754     maxdimpp = int(1.5*REAL(maxdimpp))
00755
00756     ALLOCATE( nebpp(4, maxdimpp, nats) )
00757
00758     ENDIF
00759
00760     print*, "maxdimpp = ", maxdimpp
00761     print*, "9"
00762
00763     ! Start with atom 1
00764
00765     totnebpp = 0
00766
00767     IF (pbcon .EQ. 1) THEN
00768
00769 !!$OMP PARALLEL DO DEFAULT(NONE) &
00770 !!$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTPP, TOTPERCELL, CELLATOMID) &
00771 !!$OMP SHARED(NEBPP, TOTNEBPP, NX, NY, NZ, MAXDIMPP) &
00772 !!$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00773 !!$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2)
00774
00775     DO i = 1, nats
00776
00777         ! Find its cell
00778
00779         xcell = int(cr(1,i)/maxcutpp) + 1
00780         ycell = int(cr(2,i)/maxcutpp) + 1
00781         zcell = int(cr(3,i)/maxcutpp) + 1
00782
00783         ! Loop over all neighboring cells including its own cell
00784
00785         DO kk = zcell - 1, zcell + 1
00786             DO jj = ycell - 1, ycell + 1
00787                 DO ii = xcell - 1, xcell + 1

```

```

00788
00789      cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
00790      cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00791
00792      DO j = 1, totpercell(cellid)
00793
00794          atomj = cellatomid(1, j, cellid)
00795          pbci = cellatomid(2, j, cellid)
00796          pbcj = cellatomid(3, j, cellid)
00797          pbck = cellatomid(4, j, cellid)
00798
00799          x = cr(1,atomj) + REAL(pbci)*BOXDIMS(1) - CR(1,i)
00800          y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)
00801          z = cr(3,atomj) + REAL(pbck)*BOXDIMS(3) - CR(3,i)
00802
00803          r2 = x*x + y*y + z*z
00804
00805          IF (r2 .LT. maxcutpp*maxcutpp .AND. r2 .GT. minr) THEN
00806
00807              totnebpp(i) = totnebpp(i) + 1
00808
00809              IF (totnebpp(i) .GT. maxdimpp) THEN
00810                  CALL errors("neblast_cell","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
00811 (PP) ")
00812
00813                  ENDDIF
00814
00815                  nebpp( 1, totnebpp(i), i ) = atomj
00816                  nebpp( 2, totnebpp(i), i ) = pbci
00817                  nebpp( 3, totnebpp(i), i ) = pbcj
00818                  nebpp( 4, totnebpp(i), i ) = pbck
00819
00820              ENDDIF
00821
00822          ENDDO
00823      ENDDO
00824  ENDDO
00825  ENDDO
00826  ENDDO
00827  !!$OMP END PARALLEL DO
00828
00829      ELSE ! NO PBC
00830
00831          !$OMP PARALLEL DO DEFAULT(NONE) &
00832          !$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTPP, TOTPERCELL, CELLATOMID) &
00833          !$OMP SHARED(NEBPP, TOTNEBPP, NX, NY, NZ, MAXDIMPP) &
00834          !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
00835          !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2)
00836
00837          DO i = 1, nats
00838
00839              xcell = int(cr(1,i)/maxcutpp) + 1
00840              ycell = int(cr(2,i)/maxcutpp) + 1
00841              zcell = int(cr(3,i)/maxcutpp) + 1
00842
00843              DO kk = zcell - 1, zcell + 1
00844                  DO jj = ycell - 1, ycell + 1
00845                      DO ii = xcell - 1, xcell + 1
00846
00847                          IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
00848                              .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
00849                              kk .LE. nz) THEN
00850
00851                              cellid = ii + nx*(jj-1) + ny*nx*(kk-1)
00852
00853                              DO j = 1, totpercell(cellid)
00854
00855                                  atomj = cellatomid(1, j, cellid)
00856
00857                                  x = cr(1,atomj) - cr(1,i)
00858                                  y = cr(2,atomj) - cr(2,i)
00859                                  z = cr(3,atomj) - cr(3,i)
00860
00861                                  r2 = x*x + y*y + z*z
00862
00863                                  IF (r2 .LT. maxcutpp*maxcutpp .AND. r2 .GT. minr) THEN
00864
00865                                      totnebpp(i) = totnebpp(i) + 1
00866
00867                                      IF (totnebpp(i) .GT. maxdimpp) THEN
00868                                          CALL errors("neblast_cell","NUMBER OF NEIGHBORS EXCEEDS ARRAY
00869 DIMENSION (PP) ")
00870
00871                                      ENDDIF
00872
00873                                      nebpp( 1, totnebpp(i), i ) = atomj

```

```

00873         nebpp( 2, totnebpp(i), i ) = 0
00874         nebpp( 3, totnebpp(i), i ) = 0
00875         nebpp( 4, totnebpp(i), i ) = 0
00876
00877         ENDIF
00878
00879         ENDDO
00880     ENDIF
00881     ENDDO
00882     ENDDO
00883     ENDDO
00884     ENDDO
00885
00886     !$OMP END PARALLEL DO
00887
00888     ENDIF
00889
00890     DEALLOCATE(totpercell, cellatomid)
00891
00892     ENDIF
00893     print*, "9"
00894     ! Coulomb now...
00895
00896     IF (electro .EQ. 1) THEN
00897
00898         IF (pbcon .EQ. 1) THEN
00899
00900             xrange = int(maxcutcoul/boxdims(1)) + 1
00901             yrange = int(maxcutcoul/boxdims(2)) + 1
00902             zrange = int(maxcutcoul/boxdims(3)) + 1
00903
00904             nx = int(boxdims(1)*(one + REAL(xrange))/maxcutcoul) + 1
00905             ny = int(boxdims(2)*(one + REAL(yrange))/maxcutcoul) + 1
00906             nz = int(boxdims(3)*(one + REAL(zrange))/maxcutcoul) + 1
00907
00908             numcells = (nx+1) * (ny+1) * (nz+1)
00909
00910         ELSE
00911
00912             xrange = 0
00913             yrange = 0
00914             zrange = 0
00915
00916             nx = int(maxval(cr(1,:))/maxcutcoul) + 1
00917             ny = int(maxval(cr(2,:))/maxcutcoul) + 1
00918             nz = int(maxval(cr(3,:))/maxcutcoul) + 1
00919
00920             numcells = nx * ny * nz
00921             print*, numcells
00922         ENDIF
00923
00924         ALLOCATE(totpercell(numcells), cellatomid(4, nats, numcells))
00925
00926         ! Put all the atoms in their cell
00927
00928         totpercell = 0
00929
00930         IF (pbcon .EQ. 1) THEN
00931
00932             DO ii = -xrange, xrange
00933                 DO jj = -yrange, yrange
00934                     DO kk = -zrange, zrange
00935
00936                         DO i = 1, nats
00937
00938                             IF (cr(1,i) + REAL(ii)*BOXDIMS(1) .LT. ZERO .AND. &
00939                                 CR(1,i) + REAL(ii)*BOXDIMS(1) .GT. -MAXCUTCoul) then
00940                                 xcell = 0
00941                             ELSEIF (cr(1,i) + REAL(ii)*BOXDIMS(1) .GE. ZERO) then
00942                                 xcell = int((cr(1,i) + REAL(ii)*BOXDIMS(1))/maxcutcoul) + 1
00943                             ELSE
00944                                 xcell = -1
00945                             ENDIF
00946
00947                             IF (cr(2,i) + REAL(jj)*BOXDIMS(2) .LT. ZERO .AND. &
00948                                 CR(2,i) + REAL(jj)*BOXDIMS(2) .GT. -MAXCUTCoul) then
00949                                 ycell = 0
00950                             ELSEIF (cr(2,i) + REAL(jj)*BOXDIMS(2) .GE. ZERO) then
00951                                 ycell = int((cr(2,i) + REAL(jj)*BOXDIMS(2))/maxcutcoul) + 1
00952                             ELSE
00953                                 ycell = -1
00954                             ENDIF
00955
00956                             IF (cr(3,i) + REAL(kk)*BOXDIMS(3) .LT. ZERO .AND. &
00957                                 CR(3,i) + REAL(kk)*BOXDIMS(3) .GT. -MAXCUTCoul) then
00958                                 zcell = 0
00959                             ELSEIF (cr(3,i) + REAL(kk)*BOXDIMS(3) .GE. ZERO) then

```

```

00960         zcell = int((cr(3,i) + REAL(kk)*BOXDIMS(3))/maxcutcoul) + 1
00961     ELSE
00962         zcell = -1
00963     ENDIF
00964
00965     IF (xcell .GE. 0 .AND. ycell .GE. 0 .AND. zcell .GE. 0 &
00966        .AND. xcell .LE. nx .AND. ycell .LE. ny .AND. &
00967        zcell .LE. nz) THEN
00968
00969         cellid = xcell + (nx+1)*(ycell-1) + (nx+1)*(ny+1)*(zcell-1)
00970         cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
00971
00972         totpercell(cellid) = totpercell(cellid) + 1
00973         cellatomid(1, totpercell(cellid), cellid) = i
00974         cellatomid(2, totpercell(cellid), cellid) = ii
00975         cellatomid(3, totpercell(cellid), cellid) = jj
00976         cellatomid(4, totpercell(cellid), cellid) = kk
00977
00978
00979     ENDIF
00980
00981     ENDDO
00982     ENDDO
00983     ENDDO
00984     ENDDO
00985 ELSE
00986
00987     DO i = 1, nats
00988
00989         xcell = int(cr(1,i)/maxcutcoul) + 1
00990         ycell = int(cr(2,i)/maxcutcoul) + 1
00991         zcell = int(cr(3,i)/maxcutcoul) + 1
00992
00993
00994         cellid = xcell + nx*(ycell-1) + ny*nx*(zcell-1)
00995
00996         totpercell(cellid) = totpercell(cellid) + 1
00997         cellatomid(1, totpercell(cellid), cellid) = i
00998         cellatomid(2, totpercell(cellid), cellid) = 0
00999         cellatomid(3, totpercell(cellid), cellid) = 0
01000         cellatomid(4, totpercell(cellid), cellid) = 0
01001
01002     ENDDO
01003
01004 ENDIF
01005
01006 ! First pass through - figure out dimension of neighborlist array
01007
01008 IF (amiallo .EQ. 0) THEN
01009
01010     IF (pbcon .EQ. 1) THEN
01011
01012         !$OMP PARALLEL DO DEFAULT(NONE) &
01013         !$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
01014         !$OMP SHARED(NX, NY, NZ, MAXCUTCOUL) &
01015         !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
01016         !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2, NEB) &
01017         !$OMP REDUCTION(MAX: MAXDIMCOUL)
01018
01019         DO i = 1, nats
01020
01021             neb = 0
01022
01023             ! Find its cell
01024
01025             xcell = int(cr(1,i)/maxcutcoul) + 1
01026             ycell = int(cr(2,i)/maxcutcoul) + 1
01027             zcell = int(cr(3,i)/maxcutcoul) + 1
01028
01029             ! Loop over all neighboring cells including its own cell
01030
01031             DO kk = zcell - 1, zcell + 1
01032                 DO jj = ycell - 1, ycell + 1
01033                     DO ii = xcell - 1, xcell + 1
01034
01035                         cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
01036                         cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
01037
01038                         DO j = 1, totpercell(cellid)
01039
01040                             atomj = cellatomid(1, j, cellid)
01041                             pbci = cellatomid(2, j, cellid)
01042                             pbcj = cellatomid(3, j, cellid)
01043                             pbck = cellatomid(4, j, cellid)
01044
01045                             x = cr(1,atomj) + REAL(pbci)*BOXDIMS(1) - CR(1,i)
01046                             y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)

```

```

01047         z = cr(3,atomj) + REAL(pbck)*BOXDIMS(3) - CR(3,i)
01048
01049         r2 = x*x + y*y + z*z
01050
01051         IF (r2 .LT. maxcutcoul*maxcutcoul .AND. &
01052             r2 .GT. minr) neb = neb + 1
01053
01054         ENDDO
01055
01056     ENDDO
01057 ENDDO
01058 ENDDO
01059
01060     maxdimcoul = max(neb, maxdimcoul)
01061
01062 ENDDO
01063
01064 !$OMP END PARALLEL DO
01065
01066 ELSE ! NO PBC
01067
01068     !$OMP PARALLEL DO DEFAULT(NONE) &
01069     !$OMP SHARED(NATS, BOXDIMS, CR, TOTPERCELL, CELLATOMID) &
01070     !$OMP SHARED(NX, NY, NZ, MAXCUTCoul) &
01071     !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
01072     !$OMP PRIVATE(X, Y, Z, R2, NEB) &
01073     !$OMP REDUCTION(MAX: MAXDIMCOUL)
01074
01075     DO i = 1, nats
01076
01077         neb = 0
01078
01079         xcell = int(cr(1,i)/maxcutcoul) + 1
01080         ycell = int(cr(2,i)/maxcutcoul) + 1
01081         zcell = int(cr(3,i)/maxcutcoul) + 1
01082
01083         DO kk = zcell - 1, zcell + 1
01084             DO jj = ycell - 1, ycell + 1
01085                 DO ii = xcell - 1, xcell + 1
01086
01087                     IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
01088                         .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
01089                         kk .LE. nz) THEN
01090
01091                         cellid = ii + nx*(jj-1) + ny*nx*(kk-1)
01092
01093                         DO j = 1, totpercell(cellid)
01094
01095                             atomj = cellatomid(1, j, cellid)
01096
01097                             x = cr(1,atomj) - cr(1,i)
01098                             y = cr(2,atomj) - cr(2,i)
01099                             z = cr(3,atomj) - cr(3,i)
01100
01101                             r2 = x*x + y*y + z*z
01102
01103                             IF (r2 .LT. maxcutcoul*maxcutcoul .AND. &
01104                                 r2 .GT. minr) neb = neb + 1
01105
01106                         ENDDO
01107                     ENDDO
01108                 ENDDO
01109             ENDDO
01110         ENDDO
01111
01112         maxdimcoul = max(neb, maxdimcoul)
01113
01114     ENDDO
01115
01116 !$OMP END PARALLEL DO
01117
01118 ENDF
01119
01120     maxdimcoul = int(1.5*REAL(maxdimcoul))
01121
01122     ALLOCATE( nebcoul(4, maxdimcoul, nats) )
01123
01124 ENDF
01125
01126
01127 ! Start with atom 1
01128
01129     totnebcoul = 0
01130
01131     IF (pbcon .EQ. 1) THEN

```

```

01134
01135      !$OMP PARALLEL DO DEFAULT(NONE) &
01136      !$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTCOU, TOTPERCELL, CELLATOMID) &
01137      !$OMP SHARED(NEBCOU, TOTNEBCOU, NX, NY, NZ, MAXDIMCOU, PBCON) &
01138      !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
01139      !$OMP PRIVATE(PBCI, PBCJ, PBCK, X, Y, Z, R2)
01140
01141      DO i = 1, nats
01142
01143          ! Find its cell
01144
01145          xcell = int(cr(1,i)/maxcutcou) + 1
01146          ycell = int(cr(2,i)/maxcutcou) + 1
01147          zcell = int(cr(3,i)/maxcutcou) + 1
01148
01149          ! Loop over all neighboring cells including its own cell
01150
01151          DO kk = zcell - 1, zcell + 1
01152              DO jj = ycell - 1, ycell + 1
01153                  DO ii = xcell - 1, xcell + 1
01154
01155                      cellid = ii + (nx+1)*(jj-1) + (nx+1)*(ny+1)*(kk-1)
01156                      cellid = cellid + (nx+1) + (nx+1)*(ny+1) + 1
01157
01158                      DO j = 1, totpercell(cellid)
01159
01160                          atomj = cellatomid(1, j, cellid)
01161                          pbcj = cellatomid(2, j, cellid)
01162                          pbcj = cellatomid(3, j, cellid)
01163                          pbck = cellatomid(4, j, cellid)
01164
01165                          x = cr(1,atomj) + REAL(pbcj)*BOXDIMS(1) - CR(1,i)
01166                          y = cr(2,atomj) + REAL(pbcj)*BOXDIMS(2) - CR(2,i)
01167                          z = cr(3,atomj) + REAL(pbck)*BOXDIMS(3) - CR(3,i)
01168
01169                          r2 = x*x + y*y + z*z
01170
01171                          IF (r2 .LT. maxcutcou*maxcutcou .AND. r2 .GT. minr) THEN
01172
01173                              totnebcou(i) = totnebcou(i) + 1
01174
01175                              IF (totnebcou(i) .GT. maxdimcou) THEN
01176                                  CALL errors("neblast_cell","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
(COU) ")
01177
01178                                  ENDDO
01179
01180                                  nebcou( 1, totnebcou(i), i ) = atomj
01181                                  nebcou( 2, totnebcou(i), i ) = pbcj
01182                                  nebcou( 3, totnebcou(i), i ) = pbcj
01183                                  nebcou( 4, totnebcou(i), i ) = pbck
01184
01185                                  ENDDO
01186
01187                                  ENDDO
01188
01189                                  ENDDO
01190
01191                                  ENDDO
01192
01193                                  !$OMP END PARALLEL DO
01194
01195                              ELSE ! NO PBC
01196
01197                                  !$OMP PARALLEL DO DEFAULT(NONE) &
01198                                  !$OMP SHARED(NATS, BOXDIMS, CR, MAXCUTCOU, TOTPERCELL, CELLATOMID) &
01199                                  !$OMP SHARED(NEBCOU, TOTNEBCOU, NX, NY, NZ, MAXDIMCOU) &
01200                                  !$OMP PRIVATE(I, XCELL, YCELL, ZCELL, II, JJ, KK, CELLID, J, ATOMJ) &
01201                                  !$OMP PRIVATE(X, Y, Z, R2)
01202
01203                                  DO i = 1, nats
01204
01205                                      xcell = int(cr(1,i)/maxcutcou) + 1
01206                                      ycell = int(cr(2,i)/maxcutcou) + 1
01207                                      zcell = int(cr(3,i)/maxcutcou) + 1
01208
01209                                      DO kk = zcell - 1, zcell + 1
01210                                          DO jj = ycell - 1, ycell + 1
01211                                              DO ii = xcell - 1, xcell + 1
01212
01213                                                  IF (ii .GE. 1 .AND. jj .GE. 1 .AND. kk .GE. 1 &
01214                                                  .AND. ii .LE. nx .AND. jj .LE. ny .AND. &
01215                                                  kk .LE. nz) THEN
01216
01217                                                      cellid = ii + nx*(jj-1) + ny*nx*(kk-1)
01218
01219                                                      DO j = 1, totpercell(cellid)

```

```

01220
01221             atomj = cellatomid(1, j, cellid)
01222
01223             x = cr(1,atomj) - cr(1,i)
01224             y = cr(2,atomj) - cr(2,i)
01225             z = cr(3,atomj) - cr(3,i)
01226
01227             r2 = x*x + y*y + z*z
01228
01229             IF (r2 .LT. maxcutcoul*maxcutcoul .AND. r2 .GT. minr) THEN
01230
01231                 totnebcoul(i) = totnebcoul(i) + 1
01232
01233                 IF (totnebcoul(i) .GT. maxdimcoul) THEN
01234                     CALL errors("neblast_cell","NUMBER OF NEIGHBORS EXCEEDS ARRAY
01235 DIMENSION (COUL) ")
01236
01237                     ENDF
01238
01239                     nebcoul( 1, totnebcoul(i), i ) = atomj
01240                     nebcoul( 2, totnebcoul(i), i ) = 0
01241                     nebcoul( 3, totnebcoul(i), i ) = 0
01242                     nebcoul( 4, totnebcoul(i), i ) = 0
01243
01244                     ENDF
01245
01246                     ENDDO
01247                 ENDF
01248             ENDDO
01249         ENDDO
01250
01251         !$OMP END PARALLEL DO
01252
01253     ENDF
01254
01255     DEALLOCATE(totpercell, cellatomid)
01256
01257 ENDF
01258
01259 DO i = 1, nats
01260     cr(1,i) = cr(1,i) + mincrx
01261     cr(2,i) = cr(2,i) + mincry
01262     cr(3,i) = cr(3,i) + mincrz
01263 ENDDO
01264
01265 ! Check whether we need to redimension arrays
01266
01267 maxnebtb = maxval(totnebtb)
01268 IF (ppoton .EQ. 1) maxnebpp = maxval(totnebpp)
01269 IF (electro .EQ. 1) maxnebcoul = maxval(totnebcoul)
01270
01271 prevdimtb = maxdimtb
01272 prevdimpp = maxdimpp
01273 prevdimcoul = maxdimcoul
01274
01275 ! If we have more neighbors this time around increase the allocation
01276
01277 ! Allocate more storage to be safe
01278
01279 IF (maxnebtb .GT. prevnebtb) THEN
01280     prevnebtb = maxnebtb
01281     maxdimtb = int(REAL(maxnebtb)*1.5)
01282 ENDF
01283
01284 IF (maxnebpp .GT. prevnebpp) THEN
01285     prevnebpp = maxnebpp
01286     maxdimpp = int(REAL(maxnebpp)*1.5)
01287 ENDF
01288
01289 IF (electro .EQ. 1 .AND. maxnebcoul .GT. prevnebcoul) THEN
01290     prevnebcoul = maxnebcoul
01291     maxdimcoul = int(REAL(maxnebcoul)*1.5)
01292 ENDF
01293
01294 RETURN
01295
01296 END SUBROUTINE neblists

```

8.261 neblistarray.f90 File Reference

Modules

- module [neblastarray](#)

Variables

- integer [neblastarray::allocst](#)
- integer [neblastarray::dimlist](#)
- real(latteprec) [neblastarray::maxcut](#)
- integer [neblastarray::maxdimcoul](#)
- integer [neblastarray::maxdimpp](#)
- integer [neblastarray::maxdimtb](#)
- integer, dimension(:), allocatable [neblastarray::molid](#)
- integer, dimension(:, :, :), allocatable [neblastarray::nebcoul](#)
- integer, dimension(:, :, :), allocatable [neblastarray::nebpp](#)
- integer, dimension(:, :, :), allocatable [neblastarray::nebtb](#)
- integer [neblastarray::nomol](#)
- real(latteprec) [neblastarray::ppmax2](#)
- real(latteprec) [neblastarray::rcutcoul2](#)
- real(latteprec) [neblastarray::rcuttb2](#)
- real(latteprec) [neblastarray::skin](#)
- integer, dimension(:), allocatable [neblastarray::totnebcoul](#)
- integer, dimension(:), allocatable [neblastarray::totnebpp](#)
- integer, dimension(:), allocatable [neblastarray::totnebtb](#)
- integer [neblastarray::udneigh](#)

8.262 neblastarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it       !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE neblastarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER, ALLOCATABLE :: totnebtb(:), totnebpp(:), totnebcoul(:)
00030   INTEGER, ALLOCATABLE :: nebtb(:, :, :), nebpp(:, :, :), nebcoul(:, :, :)
00031   INTEGER, ALLOCATABLE :: molid(:)
00032   INTEGER :: udneigh, nomol
00033   INTEGER :: maxdimtb, maxdimpp, maxdimcoul
00034   INTEGER :: dimlist, allocst
00035   REAL(LATTEPREC) :: skin
00036   REAL(LATTEPREC) :: rcuttb2, ppmax2, rcutcoul2, maxcut
00037
00038 END MODULE neblastarray

```


8.263 neblists.f90 File Reference

Functions/Subroutines

- subroutine [neblists](#) (AMIALLO)

8.263.1 Function/Subroutine Documentation

8.263.1.1 subroutine neblists (integer, intent(in) *AMIALLO*)

Definition at line [23](#) of file [neblists.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE neblists(AMIALLO)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE ppotarray
00028   USE neblastarray
00029   USE coulombarray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, L, M
00035   INTEGER :: A, B, C, MYATOMI, MYATOMJ
00036   INTEGER :: PBCX, PBCY, PBCZ
00037   INTEGER :: BOXX, BOXY, BOXZ
00038   INTEGER :: II, JJ, KK
00039   INTEGER, INTENT(IN) :: AMIALLO
00040   INTEGER :: X RANGE, Y RANGE, Z RANGE
00041   INTEGER :: MAXNEBTB, MAXNEBPP, MAXNEBCOUL
00042   INTEGER :: NCELL(3), NUMCELL
00043   INTEGER :: IPIV(3), INFO
00044   INTEGER :: MYCELL, BOXID(3), COUNT
00045   ! INTEGER, SAVE :: ALLOCEST
00046   INTEGER, ALLOCATABLE :: TOTINCELL(:), CELLLIST(:,:)
00047   ! INTEGER, ALLOCATABLE :: DIMTB(:), DIMPP(:), DIMCOUL(:)
00048   REAL(LATTEPREC) :: RIJ(3), MAGR2
00049   REAL(LATTEPREC) :: MAGA(3)
00050   REAL(LATTEPREC) :: RCUTTB, RCUTCUL, PPMAX, MAXCUT2
00051   REAL(LATTEPREC) :: WORK(3), BOXINV(3,3), S(3)
00052   REAL(LATTEPREC), PARAMETER :: MINR = 0.01
00053   IF (existererror) RETURN
00054
00055
00056   IF (pbcon .EQ. 1) CALL pbc
00057
00058   totnebtb = 0
00059   IF (ppoton .GT. 0) totnebpp = 0
00060   IF (electro .EQ. 1) totnebcoul = 0
00061
00062   IF (amiallo .NE. 0) THEN
00063
00064     ! Reallocate the neighbor lists based on their size the last time
00065     DEALLOCATE(nebtb)
00066     ALLOCATE(nebtb( 4, maxdimtb, nats ))
00067
00068     IF (ppoton .NE. 0) THEN
00069       DEALLOCATE(nebpp)
00070       ALLOCATE(nebpp( 4, maxdimpp, nats ))
00071     ENDIF
00072
00073     IF (electro .EQ. 1) THEN
00074       DEALLOCATE(nebcoul)
00075       ALLOCATE(nebcoul( 4, maxdimcoul, nats))
00076     ENDIF
00077
00078   ELSE
00079
00080     ! This bit is only done on the first neighbor list build
00081
00082     ! Let's get the cut-offs for our interactions
00083
00084     rcuttb = zero
00085     ppmax = zero
00086
00087     ! Find the maximum cut off
00088
00089     DO k = 1, noint
00090
00091       IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00092
00093       IF (basistype .EQ. "NONORTHO") THEN

```

```

00094         IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00095     ENDIF
00096
00097 ENDDO
00098
00099 IF (ppoton .GT. 0) THEN
00100     DO k = 1, nopps
00101         DO k = 1, nopps
00102             IF (ppoton .EQ. 1 .AND. potcoef(10,k) .GT. ppmax ) ppmax =
00103 potcoef(10,k)
00104
00105             IF (ppoton .EQ. 2 .AND. ppr(pptablenth(k), k) .GT. ppmax) &
00106 ppmax = ppr(pptablenth(k), k)
00107
00108             IF (ppoton .EQ. 3 .AND. pprk(1,k) .GT. ppmax) ppmax = pprk(1,k)
00109
00110         ENDDO
00111     ENDIF
00112
00113     rcuttb = rcuttb + skin
00114     rcuttb2 = rcuttb*rcuttb
00115
00116     IF (ppoton .GT. 0) THEN
00117         ppmax = ppmax + skin
00118         ppmax2 = ppmax*ppmax
00119     ELSE
00120         ppmax = zero
00121         ppmax2 = zero
00122     ENDIF
00123
00124     rcutcou = coulcoul + skin
00125     rcutcou2 = rcutcou * rcutcou
00126
00127     IF (electro .EQ. 0) rcutcou = zero
00128
00129     maxcut = max(rcuttb, ppmax, rcutcou)
00130
00131     maxcut2 = maxcut*maxcut
00132
00133     ! Now let's estimate the size of the arrays we need for to
00134     ! store the neighbor lists, plus some
00135
00136     IF (pbcon .EQ. 1) THEN
00137         xrange = int(maxcut/box(1,1)) + 1
00138         yrange = int(maxcut/box(2,2)) + 1
00139         zrange = int(maxcut/box(3,3)) + 1
00140         ! print*, maxcut, xrange, yrange, zrange
00141
00142         ! Here we're hoping atom 1 is in a typical environment
00143
00144         count = 0
00145         DO j = 1, nats
00146             DO ii = -xrange, xrange
00147                 DO jj = -yrange, yrange
00148                     DO kk = -zrange, zrange
00149
00150                         rij(1) = cr(1,j) + REAL(ii)*BOX(1,1) + &
00151 REAL(jj)*BOX(2,1) + REAL(kk)*BOX(3,1) - CR(1,1)
00152
00153                         rij(2) = cr(2,j) + REAL(ii)*BOX(1,2) + &
00154 REAL(jj)*BOX(2,2) + REAL(kk)*BOX(3,2) - CR(2,1)
00155
00156                         rij(3) = cr(3,j) + REAL(ii)*BOX(1,3) + &
00157 REAL(jj)*BOX(2,3) + REAL(kk)*BOX(3,3) - CR(3,1)
00158
00159                         magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00160
00161                         IF (magr2 .LE. maxcut2) count = count + 1
00162
00163                     ENDDO
00164                 ENDDO
00165             ENDDO
00166         ENDDO
00167
00168         maxdimtb = 2*count
00169         maxdimpp = 2*count
00170         maxdimcoul = 2*count
00171
00172     ELSEIF (pbcon .EQ. 0) THEN
00173         dimlist = 0
00174         DO i = 1, nats
00175             count = 0
00176             DO j = 1, nats

```

```

00180
00181         rij(1) = cr(1,j) - cr(1,i)
00182         rij(2) = cr(2,j) - cr(2,i)
00183         rij(3) = cr(3,j) - cr(3,i)
00184
00185         magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00186
00187         IF (magr2 .LE. maxcut2) count = count + 1
00188
00189     ENDDO
00190
00191     IF (count .GT. dimlist) dimlist = count
00192
00193     ENDDO
00194
00195     maxdimtb = dimlist
00196     maxdimpp = dimlist
00197     maxdimcoul = dimlist
00198
00199 ENDIF
00200
00201 IF ( ALLOCATED(nebtb) ) DEALLOCATE(nebtb)
00202 ALLOCATE ( nebtb( 4, maxdimtb, nats ) )
00203
00204 IF (ppoton .GT. 0) THEN
00205     IF ( ALLOCATED(nebpp) ) DEALLOCATE(nebpp)
00206     ALLOCATE(nebpp( 4, maxdimpp, nats ) )
00207 ENDIF
00208
00209 IF (electro .EQ. 1) THEN
00210     IF ( ALLOCATED(nebcoul) ) DEALLOCATE(nebcoul)
00211     ALLOCATE(nebcoul( 4, maxdimcoul, nats ) )
00212 ENDIF
00213
00214 ENDIF
00215
00216 ! Now build the neighbor list
00217
00218 ! With periodic boundaries first:
00219
00220 IF (pbcon .EQ. 1) THEN
00221
00222     maga(1) = sqrt(box(1,1)*box(1,1) + box(1,2)*box(1,2) + box(1,3)*box(1,3))
00223     maga(2) = sqrt(box(2,1)*box(2,1) + box(2,2)*box(2,2) + box(2,3)*box(2,3))
00224     maga(3) = sqrt(box(3,1)*box(3,1) + box(3,2)*box(3,2) + box(3,3)*box(3,3))
00225
00226     xrange = int(maxcut/maga(1)) + 1
00227     yrange = int(maxcut/maga(2)) + 1
00228     zrange = int(maxcut/maga(3)) + 1
00229
00230     ! This gives the number of sub-cells along each lattice vector
00231
00232     ncell(1) = max(int(maga(1)/maxcut),1)
00233     ncell(2) = max(int(maga(2)/maxcut),1)
00234     ncell(3) = max(int(maga(3)/maxcut),1)
00235
00236     numcell = ncell(1)*ncell(2)*ncell(3)
00237
00238     ! PRINT*, NCELL(1), NCELL(2), NCELL(3), NUMCELL
00239
00240     IF (amiallo .EQ. 0) allocest = 2*nats/numcell
00241
00242     ALLOCATE(totincell(numcell), celllist(allocest, numcell))
00243
00244     totincell = 0
00245
00246     boxinv = box
00247
00248     CALL dgetrf(3, 3, boxinv, 3, ipiv, info)
00249
00250     CALL dgetri(3, boxinv, 3, ipiv, work, 3, info)
00251
00252     ! Put the atoms into the sub-cells
00253
00254     DO i = 1, nats
00255
00256         CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,i), 1, zero, s, 1)
00257
00258         ! Dangerous condition caught below (MJC)
00259
00260         IF (s(1) .GE. one) s(1) = zero
00261         IF (s(2) .GE. one) s(2) = zero
00262         IF (s(3) .GE. one) s(3) = zero
00263
00264         boxid(1) = int(s(1)*ncell(1))
00265         boxid(2) = int(s(2)*ncell(2))
00266         boxid(3) = int(s(3)*ncell(3))

```

```

00267
00268      mycell = boxid(1) + ncell(1)*boxid(2) + ncell(1)*ncell(2)*boxid(3) + 1
00269
00270      totincell(mycell) = totincell(mycell) + 1
00271      celllist(totincell(mycell), mycell) = i
00272
00273      ENDDO
00274
00275      allocest = 2*maxval(totincell)
00276
00277      ! Loop over the subcells and build the lists
00278
00279      DO ii = 1, numcell
00280
00281          ! Indices of subcell II
00282
00283          boxz = (ii - 1)/(ncell(1)*ncell(2))
00284          boxy = (ii - 1 - boxz*ncell(1)*ncell(2))/ncell(1)
00285          boxx = (ii - 1 - ncell(1)*boxy - ncell(1)*ncell(2)*boxz)
00286          !      print*, boxx, boxy, boxz, totincell(ii), celllist(1,ii), NUMCELL
00287
00288          ! Loop over atoms in cell II
00289
00290          DO i = 1, totincell(ii)
00291
00292              myatomi = celllist(i,ii)
00293
00294              ! Loop over the neighboring subcells
00295
00296              DO a = boxz - zrange, boxz + zrange
00297                  DO b = boxy - yrange, boxy + yrange
00298                      DO c = boxx - xrange, boxx + xrange
00299
00300                          pbcx = 0
00301                          pbcy = 0
00302                          pbcz = 0
00303
00304                          boxid(1) = c
00305                          boxid(2) = b
00306                          boxid(3) = a
00307
00308                          IF (a .LT. 0) THEN
00309                              pbcz = a
00310                              boxid(3) = ncell(3) - 1
00311                          ELSEIF (a .GE. ncell(3)) THEN
00312                              pbcz = a - ncell(3) + 1
00313                              boxid(3) = 0
00314                          ENDIF
00315
00316                          IF (b .LT. 0) THEN
00317                              pbcy = b
00318                              boxid(2) = ncell(2) - 1
00319                          ELSEIF (b .GE. ncell(2)) THEN
00320                              pbcy = b - ncell(2) + 1
00321                              boxid(2) = 0
00322                          ENDIF
00323
00324                          IF (c .LT. 0) THEN
00325                              pbcx = c
00326                              boxid(1) = ncell(1) - 1
00327                          ELSEIF (c .GE. ncell(1)) THEN
00328                              pbcx = c - ncell(1) + 1
00329                              boxid(1) = 0
00330                          ENDIF
00331
00332                          mycell = boxid(1) + ncell(1)*boxid(2) + ncell(1)*ncell(2)*boxid(3) + 1
00333
00334                          ! Loop over the atoms in the neighboring cell
00335
00336                          DO j = 1, totincell(mycell)
00337
00338                              myatomj = celllist(j, mycell)
00339
00340                              rij(1) = cr(1,myatomj) + REAL(pbcx)*BOX(1,1) + &
00341                                  REAL(pbcy)*BOX(2,1) + REAL(pbcz)*BOX(3,1) - CR(1,myatomi)
00342
00343                              rij(2) = cr(2,myatomj) + REAL(pbcx)*BOX(1,2) + &
00344                                  REAL(pbcy)*BOX(2,2) + REAL(pbcz)*BOX(3,2) - CR(2,myatomi)
00345
00346                              rij(3) = cr(3,myatomj) + REAL(pbcx)*BOX(1,3) + &
00347                                  REAL(pbcy)*BOX(2,3) + REAL(pbcz)*BOX(3,3) - CR(3,myatomi)
00348
00349                              magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00350
00351                              IF (magr2 .GT. minr .AND. magr2 .LT. rcuttb2) THEN
00352                                  totnebtb(myatomi) = totnebtb(myatomi) + 1
00353

```

```

00354
00355         IF (totnebtb(myatomi) .GT. maxdimtb) THEN
00356             CALL errors("neblists","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
(TB) ")
00357             RETURN
00358         ENDIF
00359
00360         nebtb( 1, totnebtb(myatomi), myatomi ) = myatomj
00361         nebtb( 2, totnebtb(myatomi), myatomi ) = pbcx
00362         nebtb( 3, totnebtb(myatomi), myatomi ) = pbcy
00363         nebtb( 4, totnebtb(myatomi), myatomi ) = pbcz
00364
00365     ENDIF
00366
00367     IF (ppoton .NE. 0) THEN
00368
00369         IF (magr2 .GT. minr .AND. magr2 .LT. ppmax2) THEN
00370
00371             totnebpp(myatomi) = totnebpp(myatomi) + 1
00372             IF (totnebpp(myatomi) .GT. maxdimpp) THEN
00373                 CALL errors("neblists","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
(PP) ")
00374                 RETURN
00375             ENDIF
00376
00377             nebpp( 1, totnebpp(myatomi), myatomi ) = myatomj
00378             nebpp( 2, totnebpp(myatomi), myatomi ) = pbcx
00379             nebpp( 3, totnebpp(myatomi), myatomi ) = pbcy
00380             nebpp( 4, totnebpp(myatomi), myatomi ) = pbcz
00381
00382         ENDIF
00383
00384     ENDIF
00385
00386     IF (electro .NE. 0) THEN
00387
00388         IF (magr2 .GT. minr .AND. magr2 .LT. rcutcou2) THEN
00389
00390             totnebcoul(myatomi) = totnebcoul(myatomi) + 1
00391
00392             IF (totnebcoul(myatomi) .GT. maxdimcoul) THEN
00393                 CALL errors("neblists","NUMBER OF NEIGHBORS EXCEEDS ARRAY DIMENSION
(COUL) ")
00394                 RETURN
00395             ENDIF
00396
00397             nebcoul( 1, totnebcoul(myatomi), myatomi ) = myatomj
00398             nebcoul( 2, totnebcoul(myatomi), myatomi ) = pbcx
00399             nebcoul( 3, totnebcoul(myatomi), myatomi ) = pbcy
00400             nebcoul( 4, totnebcoul(myatomi), myatomi ) = pbcz
00401
00402         ENDIF
00403
00404     ENDIF
00405
00406
00407         ENDDO
00408     ENDDO
00409 ENDDO
00410
00411     ENDDO
00412
00413     ENDDO
00414
00415     ENDDO
00416
00417     DEALLOCATE(totincell, celllist)
00418
00419     ELSEIF (pbcon .EQ. 0) THEN
00420
00421         ! Now we're doing building the neighbor lists for gas-phase systems
00422
00423         DO i = 1, nats
00424             DO j = 1, nats
00425
00426                 rij(1) = cr(1,j) - cr(1,i)
00427
00428                 rij(2) = cr(2,j) - cr(2,i)
00429
00430                 rij(3) = cr(3,j) - cr(3,i)
00431
00432                 magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00433
00434                 IF (magr2 .GT. minr .AND. magr2 .LT. rcuttb2) THEN
00435
00436                     totnebtb(i) = totnebtb(i) + 1
00437                     nebtb( 1, totnebtb(i), i ) = j

```

```

00438         nebtb( 2, totnebtb(i), i ) = 0
00439         nebtb( 3, totnebtb(i), i ) = 0
00440         nebtb( 4, totnebtb(i), i ) = 0
00441
00442     ENDIF
00443
00444     IF (ppoton .NE. 0) THEN
00445
00446         IF (magr2 .GT. minr .AND. magr2 .LT. ppxmax2) THEN
00447
00448             totnebpp(i) = totnebpp(i) + 1
00449             nebpp( 1, totnebpp(i), i ) = j
00450             nebpp( 2, totnebpp(i), i ) = 0
00451             nebpp( 3, totnebpp(i), i ) = 0
00452             nebpp( 4, totnebpp(i), i ) = 0
00453
00454         ENDIF
00455
00456     ENDIF
00457
00458     IF (electro .NE. 0) THEN
00459
00460         IF (magr2 .GT. minr .AND. magr2 .LT. rcutcoul2) THEN
00461
00462             totnebcoul(i) = totnebcoul(i) + 1
00463             nebcoul( 1, totnebcoul(i), i ) = j
00464             nebcoul( 2, totnebcoul(i), i ) = 0
00465             nebcoul( 3, totnebcoul(i), i ) = 0
00466             nebcoul( 4, totnebcoul(i), i ) = 0
00467
00468         ENDIF
00469
00470     ENDIF
00471
00472     ENDDO
00473 ENDDO
00474
00475 ENDIF
00476
00477 ! Let's get the dimensions of the arrays about right for the next
00478 ! loop through here
00479
00480 maxdimtb = 2*maxval(totnebtb)
00481 IF (ppoton .NE. 0) maxdimpp = 2*maxval(totnebpp)
00482 IF (electro .NE. 0) maxdimcoul = 2*maxval(totnebcoul)
00483
00484 RETURN
00485
00486 END SUBROUTINE neblists

```

8.265 newkbldnewh.f90 File Reference

Functions/Subroutines

- subroutine [kbldnewh](#)

8.265.1 Function/Subroutine Documentation

8.265.1.1 subroutine kbldnewh ()

Definition at line 23 of file [newkbldnewh.f90](#).

8.266 newkbldnewh.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !

```



```

00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE kbldnewh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE neblistarray
00027   USE xboarray
00028   USE nonoarray
00029   USE univarray
00030   USE kspacearray
00031   USE myprecision
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, J, NEWJ, K, L, II, JJ, KK, MM, MP, NN, SUBI
00036   INTEGER :: IBRA, IKET, LBRA, LKET, MBRA, MKET
00037   INTEGER :: INDEX, INDI, INDJ
00038   INTEGER :: SWITCH, PREVJ
00039   INTEGER :: PBCI, PBCJ, PBCK
00040   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00041   INTEGER :: NORBI, KCOUNT
00042   INTEGER :: KX, KY, KZ
00043   REAL(LATTEPREC) :: KX0, KY0, KZ0
00044   REAL(LATTEPREC) :: ALPHA, BETA, COSBETA, PHI, TMP, PERM
00045   REAL(LATTEPREC) :: RIJ(3), MAGR2, MAGR, MAGRP, RCUTTB
00046   REAL(LATTEPREC) :: MAXARRAY(20), MAXRCUT, MAXRCUT2
00047   REAL(LATTEPREC) :: ANGFACTOR
00048   REAL(LATTEPREC) :: KPOINT(3), KDOTL
00049   REAL(LATTEPREC) :: AMMBRA, WIGLBRAMBRA
00050   COMPLEX(LATTEPREC) :: BLOCH, KHTMP
00051   REAL(LATTEPREC), EXTERNAL :: UNIVSCALE, WIGNERD, SLMP, TLMMP, AM, BM
00052
00053   hk = cmplx(zero)
00054
00055   index = 0
00056
00057   ! Build diagonal elements
00058   DO i = 1, nats
00059
00060     SELECT CASE(basis(elempointer(i)))
00061
00062     CASE("s")
00063
00064       index = index + 1
00065       hk(index, index, 1) = cmplx(hes(elempointer(i)))
00066
00067     CASE("p")
00068
00069       DO subi = 1, 3
00070         index = index + 1
00071         hk(index, index, 1) = cmplx(hep(elempointer(i)))
00072       ENDDO
00073
00074     CASE("d")
00075
00076       DO subi = 1, 5
00077         index = index + 1
00078         hk(index, index, 1) = cmplx(hed(elempointer(i)))
00079       ENDDO
00080
00081     CASE("f")
00082
00083       DO subi = 1, 7
00084         index = index + 1
00085         hk(index, index, 1) = cmplx(hef(elempointer(i)))
00086       ENDDO
00087
00088     CASE("sp")
00089
00090       DO subi = 1, 4
00091

```

```

00092         index = index + 1
00093         IF (subi .EQ. 1) THEN
00094             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00095         ELSE
00096             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00097         ENDIF
00098     ENDDO
00099
00100     CASE("sd")
00101
00102     DO subi = 1, 6
00103
00104         index = index + 1
00105         IF (subi .EQ. 1) THEN
00106             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00107         ELSE
00108             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00109         ENDIF
00110     ENDDO
00111
00112     CASE("sf")
00113
00114     DO subi = 1, 8
00115
00116         index = index + 1
00117         IF (subi .EQ. 1) THEN
00118             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00119         ELSE
00120             hk(index,index, 1) = cmplx(hef(elempointer(i)))
00121         ENDIF
00122     ENDDO
00123
00124     CASE("pd")
00125
00126     DO subi = 1, 8
00127
00128         index = index + 1
00129         IF (subi .LE. 3) THEN
00130             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00131         ELSE
00132             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00133         ENDIF
00134     ENDDO
00135
00136     CASE("pf")
00137
00138     DO subi = 1, 10
00139
00140         index = index + 1
00141         IF (subi .LE. 3) THEN
00142             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00143         ELSE
00144             hk(index,index, 1) = cmplx(hef(elempointer(i)))
00145         ENDIF
00146     ENDDO
00147
00148     CASE("df")
00149
00150     DO subi = 1, 12
00151
00152         index = index + 1
00153         IF (subi .LE. 5) THEN
00154             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00155         ELSE
00156             hk(index,index, 1) = cmplx(hef(elempointer(i)))
00157         ENDIF
00158     ENDDO
00159
00160     CASE("spd")
00161
00162     DO subi = 1, 9
00163
00164         index = index + 1
00165         IF (subi .EQ. 1) THEN
00166             hk(index,index, 1) = cmplx(hes(elempointer(i)))
00167         ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00168             hk(index,index, 1) = cmplx(hep(elempointer(i)))
00169         ELSE
00170             hk(index,index, 1) = cmplx(hed(elempointer(i)))
00171         ENDIF
00172     ENDDO
00173
00174
00175
00176
00177
00178

```

```

00179      ENDDO
00180
00181      CASE("spf")
00182
00183          DO subi = 1, 11
00184
00185              index = index + 1
00186              IF (subi .EQ. 1) THEN
00187                  hk(index,index, 1) = cmplx(hes(elempointer(i)))
00188              ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00189                  hk(index,index, 1) = cmplx(hep(elempointer(i)))
00190              ELSE
00191                  hk(index,index, 1) = cmplx(hef(elempointer(i)))
00192              ENDIF
00193          ENDDO
00194
00195      CASE("sdf")
00196
00197          DO subi = 1, 13
00198
00199              index = index + 1
00200              IF (subi .EQ. 1) THEN
00201                  hk(index,index, 1) = cmplx(hes(elempointer(i)))
00202              ELSEIF (subi .GT. 1 .AND. subi .LE. 6) THEN
00203                  hk(index,index, 1) = cmplx(hed(elempointer(i)))
00204              ELSE
00205                  hk(index,index, 1) = cmplx(hef(elempointer(i)))
00206              ENDIF
00207          ENDDO
00208
00209      CASE("pdf")
00210
00211          DO subi = 1, 15
00212
00213              index = index + 1
00214              IF (subi .LE. 3) THEN
00215                  hk(index,index, 1) = cmplx(hep(elempointer(i)))
00216              ELSEIF (subi .GT. 3 .AND. subi .LE. 8) THEN
00217                  hk(index,index, 1) = cmplx(hed(elempointer(i)))
00218              ELSE
00219                  hk(index,index, 1) = cmplx(hef(elempointer(i)))
00220              ENDIF
00221          ENDDO
00222
00223      CASE("spdf")
00224
00225          DO subi = 1, 16
00226
00227              index = index + 1
00228              IF (subi .EQ. 1) THEN
00229                  hk(index, index, 1) = cmplx(hes(elempointer(i)))
00230              ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00231                  hk(index, index, 1) = cmplx(hep(elempointer(i)))
00232              ELSEIF (subi .GT. 1 .AND. subi .LE. 4) THEN
00233                  hk(index, index, 1) = cmplx(hep(elempointer(i)))
00234              ELSEIF (subi .GT. 4 .AND. subi .LE. 9) THEN
00235                  hk(index, index, 1) = cmplx(hed(elempointer(i)))
00236              ELSE
00237                  hk(index, index, 1) = cmplx(hef(elempointer(i)))
00238              ENDIF
00239          ENDDO
00240
00241      END SELECT
00242
00243      ENDDO
00244
00245      DO i = 2, nktot
00246          DO j = 1, hdim
00247              hk(j,j,i) = hk(j,j,1)
00248          ENDDO
00249      ENDDO
00250
00251      ! We assign the diagonal elements in ADDQDEP
00252
00253      ! IF (BASISTYPE .EQ. "NONORTHO") THEN
00254
00255      !     SMAT = ZERO
00256
00257      !     DO I = 1, HDIM
00258      !         SMAT(I,I) = ONE
00259      !     ENDDO
00260
00261  
```

```

00266      !   ENDIF
00267
00268      ! Loop over Kpoints
00269
00270
00271      kx0 = pi*(one - REAL(nkx))/(REAL(nkx)*BOXDIMS(1)) - pi*kshift(1)
00272      ky0 = pi*(one - REAL(nky))/(REAL(nky)*BOXDIMS(2)) - pi*kshift(2)
00273      kz0 = pi*(one - REAL(nkz))/(REAL(nkz)*BOXDIMS(3)) - pi*kshift(3)
00274
00275      !   KCOUNT = 0
00276
00277      !   DO KX = 1, NKX
00278
00279      !       KPOINT(1) = KX0 + TWO*PI*REAL(KX-1)/(BOXDIMS(1)*REAL(NKX))
00280
00281      !       DO KY = 1, NKY
00282
00283      !           KPOINT(2) = KY0 + TWO*PI*REAL(KY-1)/(BOXDIMS(2)*REAL(NKY))
00284
00285      !           DO KZ = 1, NKZ
00286
00287      !               KPOINT(3) = KZ0 + TWO*PI*REAL(KZ-1)/(BOXDIMS(3)*REAL(NKZ))
00288
00289      !               KCOUNT = KCOUNT+1
00290
00291      !$OMP PARALLEL DO DEFAULT (NONE) &
00292      !$OMP SHARED(KPOINT, KCOUNT, NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00293      !$OMP SHARED(CR, BOXDIMS, HK, NOINT, ATELE, ELE1, ELE2) &
00294      !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00295      !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00296      !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP, PHI, ALPHA, BETA, COSBETA) &
00297      !$OMP PRIVATE(LBRAINC, LBRA, MBRA, L, LKETINC, LKET, MKET) &
00298      !$OMP PRIVATE(BLOCH, KDOTL) &
00299      !$OMP PRIVATE(RCUTTB, IBRA, IKET, AMMBRA, WIGLBRAMBRA, ANGFACTOR, MP)
00300
00301      DO i = 1, nats
00302
00303      ! Build the lists of orbitals on each atom
00304
00305      SELECT CASE(basis(elempointer(i)))
00306
00307      CASE("s")
00308          basisi(1) = 0
00309          basisi(2) = -1
00310      CASE("p")
00311          basisi(1) = 1
00312          basisi(2) = -1
00313      CASE("d")
00314          basisi(1) = 2
00315          basisi(2) = -1
00316      CASE("f")
00317          basisi(1) = 3
00318          basisi(2) = -1
00319      CASE("sp")
00320          basisi(1) = 0
00321          basisi(2) = 1
00322          basisi(3) = -1
00323      CASE("sd")
00324          basisi(1) = 0
00325          basisi(2) = 2
00326          basisi(3) = -1
00327      CASE("sf")
00328          basisi(1) = 0
00329          basisi(2) = 3
00330          basisi(3) = -1
00331      CASE("pd")
00332          basisi(1) = 1
00333          basisi(2) = 2
00334          basisi(3) = -1
00335      CASE("pf")
00336          basisi(1) = 1
00337          basisi(2) = 3
00338          basisi(3) = -1
00339      CASE("df")
00340          basisi(1) = 2
00341          basisi(2) = 3
00342          basisi(3) = -1
00343      CASE("spd")
00344          basisi(1) = 0
00345          basisi(2) = 1
00346          basisi(3) = 2
00347          basisi(4) = -1
00348      CASE("spf")
00349          basisi(1) = 0
00350          basisi(2) = 1
00351          basisi(3) = 3
00352          basisi(4) = -1

```

```

00353      CASE("sdf")
00354        basisi(1) = 0
00355        basisi(2) = 2
00356        basisi(3) = 3
00357        basisi(4) = -1
00358      CASE("pdf")
00359        basisi(1) = 1
00360        basisi(2) = 2
00361        basisi(3) = 3
00362        basisi(4) = -1
00363      CASE("spdf")
00364        basisi(1) = 0
00365        basisi(2) = 1
00366        basisi(3) = 2
00367        basisi(4) = 3
00368        basisi(5) = -1
00369      END SELECT
00370
00371      indi = matindlist(i)
00372
00373      ! open loop over neighbors J of atom I
00374      DO newj = 1, totnebtb(i)
00375
00376        j = nebtb(1, newj, i)
00377
00378        !
00379                                IF ( J .GE. I ) THEN
00380
00381          pbcj = nebtb(3, newj, i)
00382          pbck = nebtb(4, newj, i)
00383
00384          rij(1) = cr(1,j) + REAL(PBCI) * BOXDIMS(1) &
00385                - CR(1,i)
00386          rij(2) = cr(2,j) + REAL(PBCJ) * BOXDIMS(2) &
00387                - CR(2,i)
00388          rij(3) = cr(3,j) + REAL(PBCK) * BOXDIMS(3) &
00389                - CR(3,i)
00390
00391          magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00392
00393          rcuttb = zero
00394
00395          DO k = 1, noint
00396
00397            IF ( (atele(i) .EQ. ele1(k) .AND. &
00398                  atele(j) .EQ. ele2(k)) .OR. &
00399                  (atele(j) .EQ. ele1(k) .AND. &
00400                    atele(i) .EQ. ele2(k) )) THEN
00401
00402              IF (bond(8,k) .GT. rcuttb ) rcuttb = bond(8,k)
00403
00404              IF (basistype .EQ. "NONORTHO") THEN
00405                IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00406              ENDIF
00407
00408            ENDIF
00409
00410          ENDDO
00411
00412          IF (magr2 .LT. rcuttb*rcuttb) THEN
00413
00414            !
00415            !          KDOTL = KPOINT(1)*RIJ(1) + KPOINT(2)*RIJ(2) + &
00416            !          KPOINT(3)*RIJ(3)
00417
00418            !          BLOCH = EXP (CMPLX(ZERO,KDOTL) )
00419
00420            magr = sqrt(magr2)
00421
00422            SELECT CASE(basis(elempointer(j)))
00423            CASE("s")
00424              basisj(1) = 0
00425              basisj(2) = -1
00426            CASE("p")
00427              basisj(1) = 1
00428              basisj(2) = -1
00429            CASE("d")
00430              basisj(1) = 2
00431              basisj(2) = -1
00432            CASE("f")
00433              basisj(1) = 3
00434              basisj(2) = -1
00435            CASE("sp")
00436              basisj(1) = 0
00437              basisj(2) = 1
00438              basisj(3) = -1
00439            CASE("sd")
00440              basisj(1) = 0

```

```

00440         basisj(2) = 2
00441         basisj(3) = -1
00442     CASE("sf")
00443         basisj(1) = 0
00444         basisj(2) = 3
00445         basisj(3) = -1
00446     CASE("pd")
00447         basisj(1) = 1
00448         basisj(2) = 2
00449         basisj(3) = -1
00450     CASE("pf")
00451         basisj(1) = 1
00452         basisj(2) = 3
00453         basisj(3) = -1
00454     CASE("df")
00455         basisj(1) = 2
00456         basisj(2) = 3
00457         basisj(3) = -1
00458     CASE("spd")
00459         basisj(1) = 0
00460         basisj(2) = 1
00461         basisj(3) = 2
00462         basisj(4) = -1
00463     CASE("spf")
00464         basisj(1) = 0
00465         basisj(2) = 1
00466         basisj(3) = 3
00467         basisj(4) = -1
00468     CASE("sdf")
00469         basisj(1) = 0
00470         basisj(2) = 2
00471         basisj(3) = 3
00472         basisj(4) = -1
00473     CASE("pdf")
00474         basisj(1) = 1
00475         basisj(2) = 2
00476         basisj(3) = 3
00477         basisj(4) = -1
00478     CASE("spdf")
00479         basisj(1) = 0
00480         basisj(2) = 1
00481         basisj(3) = 2
00482         basisj(4) = 3
00483         basisj(5) = -1
00484     END SELECT
00485
00486     indj = matindlist(j)
00487
00488     magrp = sqrt(rij(1) * rij(1) + rij(2) * rij(2))
00489
00490     ! transform to system in which z-axis is aligned with RIJ,
00491     IF (abs(rij(1)) .GT. 1e-12) THEN
00492
00493         IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00494             phi = zero
00495         ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00496             phi = two * pi
00497         ELSE
00498             phi = pi
00499         ENDIF
00500         alpha = atan(rij(2) / rij(1)) + phi
00501
00502     ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00503
00504         IF (rij(2) .GT. 1e-12) THEN
00505             alpha = pi / two
00506         ELSE
00507             alpha = three * pi / two
00508         ENDIF
00509
00510     ELSE
00511         ! pathological case: beta=0 and alpha undefined, but
00512         ! this doesn't matter for matrix elements
00513
00514         alpha = zero
00515     ENDIF
00516
00517     cosbeta = rij(3)/magr
00518     beta = acos(rij(3) / magr)
00519
00520     ! Build matrix elements using eqns (1)-(9) in PRB 72 165107
00521
00522     ! The loops over LBRA and LKET need to take into account
00523     ! the orbitals assigned to each atom, e.g., sd rather than
00524     ! spd...
00525
00526

```

```

00527         ibra = indi + 1
00528
00529         lbrainc = 1
00530         DO WHILE (basisi(lbrainc) .NE. -1)
00531
00532             lbra = basisi(lbrainc)
00533             lbrainc = lbrainc + 1
00534
00535             DO mbra = -lbra, lbra
00536
00537                 ! We can calculate these two outside the
00538                 ! MKET loop...
00539
00540                 ammbra = am(mbra, alpha)
00541                 wiglbrambra = wignerd(lbra, abs(mbra), 0, cosbeta)
00542
00543                 iket = indj + 1
00544
00545                 lketinc = 1
00546                 DO WHILE (basisj(lketinc) .NE. -1)
00547
00548                     lket = basisj(lketinc)
00549                     lketinc = lketinc + 1
00550
00551                     DO mket = -lket, lket
00552
00553                         ! This is the sigma bonds (mp = 0)
00554
00555                         ! Hamiltonian build
00556
00557                         ! Pre-compute the angular part so we can use it
00558                         ! again later if we're building the S matrix too
00559
00560                         angfactor = two * ammbra * &
00561                             am(mket, alpha) * &
00562                             wiglbrambra * &
00563                             wignerd(lket, abs(mket), 0, cosbeta)
00564
00565                         khtmp = cmplx(angfactor * &
00566                             univscale(i, j, lbra, lket, &
00567                                 0, magr, "H"))
00568
00569                         kcount = 0
00570
00571                         DO kx = 1, nkx
00572                             kpoint(1) = kx0 + two*pi*REAL(kx-1)/(boxdims(1)*REAL(
00573                                 nkx))
00574
00575                             DO ky = 1, nky
00576                                 kpoint(2) = ky0 + two*pi*REAL(ky-1)/(boxdims(2)*REAL(
00577                                     nky))
00578
00579                                 DO kz = 1, nkz
00580                                     kpoint(3) = kz0 + two*pi*REAL(kz-1)/(boxdims(3)*REAL(
00581                                         nkz))
00582
00583                                     kcount = kcount+1
00584
00585                                     kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00586                                         kpoint(3)*rij(3)
00587
00588                                     bloch = exp(cmplx(zero,kdot1))
00589
00590                                     hk(ibra, iket, kcount) = &
00591                                         hk(ibra, iket, kcount) + &
00592                                         bloch*khtmp
00593
00594                                     ENDDO
00595                                 ENDDO
00596                             ENDDO
00597
00598                         ! Overlap matrix build
00599
00600                         !
00601                         IF (BASISTYPE .EQ. "NONORTHO") THEN
00602                             SMAT(IBRA, IKET) = SMAT(IBRA, IKET) + &
00603                                 BLOCH*ANGFACTOR * &
00604                                 UNIVSCALE(I, J, LBRA, LKET, 0, MAGR, "S")
00605                         ENDIF
00606
00607                         ! everything else
00608
00609                         DO mp = 1, min(lbra, lket)
00610                             angfactor = &
00611                                 slmmp(lbra, mbra, mp, alpha, cosbeta)* &
00612                                 slmmp(lket, mket, mp, alpha, cosbeta)+ &

```

```

00611          tlmmp(lbra, mbra, mp, alpha, cosbeta)* &
00612          tlmmp(lket, mket, mp, alpha, cosbeta)
00613
00614          khtmp = cmplx(angfactor * &
00615          univscale(i, j, lbra, lket, &
00616          mp, magr, "H"))
00617
00618          kcount = 0
00619          DO kx = 1, nkx
00620              kpoint(1) = kx0 + two*pi*REAL(kx-1)/(boxdims(1)*REAL(
nkx))
00621
00622              DO ky = 1, nky
00623                  kpoint(2) = ky0 + two*pi*REAL(ky-1)/(boxdims(2)*REAL(
nky))
00624
00625                  DO kz = 1, nkz
00626                      kpoint(3) = kz0 + two*pi*REAL(kz-1)/(boxdims(3)*REAL(
nkz))
00627
00628                      kcount = kcount+1
00629
00630                      kdot1 = kpoint(1)*rij(1) + kpoint(2)*rij(2) + &
00631                      kpoint(3)*rij(3)
00632
00633                      bloch = exp(cmplx(zero,kdot1))
00634
00635                      hk(ibra, iket, kcount) = &
00636                      hk(ibra, iket, kcount) + &
00637                      bloch*khtmp
00638
00639                      ENDDO
00640                  ENDDO
00641              ENDDO
00642
00643              ! HK(IBRA, IKET, KCOUNT) = &
00644              ! HK(IBRA, IKET, KCOUNT) + &
00645              ! BLOCH*CMPLX(ANGFACTOR * &
00646              ! UNIVSCALE(I, J, LBRA, LKET, &
00647              ! MP, MAGR, "H"))
00648
00649              ! IF (BASISTYPE .EQ. "NONORTHO") THEN
00650
00651              ! SMAT(IBRA, IKET) = SMAT(IBRA, IKET) + &
00652              ! BLOCH*ANGFACTOR * &
00653              ! UNIVSCALE(I, J, LBRA, LKET, MP, MAGR, "S")
00654
00655              ! ENDF
00656
00657              ENDDO
00658
00659              ! HK(IKET, IBRA, KCOUNT) = CONJG(HK(IBRA, IKET, KCOUNT))
00660              ! IF (BASISTYPE .EQ. "NONORTHO") &
00661              ! SMAT(IKET, IBRA) = SMAT(IBRA, IKET)
00662
00663              iket = iket + 1
00664
00665              ENDDO
00666
00667              ENDDO
00668
00669              ibra = ibra + 1
00670
00671              ENDDO
00672          ENDDO
00673      ENDF
00674      ! ENDF
00675      ENDDO
00676      ! INDI = INDI + NORBI
00677
00678      ENDDO
00679      !$OMP END PARALLEL DO
00680      ! ENDDO
00681      ! ENDDO
00682      ! ENDDO
00683
00684      ! Save the diagonal elements: it will help a lot when we add in the partial charges
00685
00686      DO i = 1, nktot
00687          DO j = 1, hdim
00688              hkdiag(j,i) = hk(j,j,i)
00689          ENDDO
00690      ENDDO
00691
00692
00693      ! IF (BASISTYPE .EQ. "NONORTHO") THEN
00694

```



```

00695 !      H0 = H
00696 !      CALL GENX
00697
00698 !      IF (DEBUGON .EQ. 1) THEN
00699
00700 !          OPEN(UNIT=30, STATUS="UNKNOWN", FILE="myS.dat")
00701 !          OPEN(UNIT=31, STATUS="UNKNOWN", FILE="myH0.dat")
00702
00703 !          PRINT*, "Caution - the Slater-Koster H and overlap matrices are being written to file"
00704
00705 !          DO I = 1, HDIM
00706 !              WRITE(30,10) (SMAT(I,J), J = 1, HDIM)
00707 !              WRITE(31,10) (H0(I,J), J = 1, HDIM)
00708 !          ENDDO
00709
00710 !          CLOSE(30)
00711 !          CLOSE(31)
00712
00713 !10      FORMAT(100F12.6)
00714
00715 !      ENDIF
00716
00717 !  ENDIF
00718
00719 RETURN
00720
00721 END SUBROUTINE kbldnewh

```

8.267 nnz.f90 File Reference

Functions/Subroutines

- integer function [nnzend](#) (M, HDIM)
- integer function [nnzstart](#) (MSPARSE, HDIM)

8.267.1 Function/Subroutine Documentation

8.267.1.1 integer function nnzend (integer, intent(in) M, integer, intent(in) HDIM)

Definition at line 40 of file [nnz.f90](#).

8.267.1.2 integer function nnzstart (integer, intent(in) MSPARSE, integer, intent(in) HDIM)

Definition at line 23 of file [nnz.f90](#).

8.268 nnz.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !

```

```

00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 INTEGER FUNCTION nnzstart(MSPARSE, HDIM)
00023
00024   IMPLICIT NONE
00025   INTEGER, INTENT(IN) :: MSPARSE, HDIM
00026   INTEGER :: M
00027
00028   m = msparse
00029   IF (msparse .LE. 0) m = hdim
00030   m = m + (32 - mod(m,32))
00031   IF (m > hdim) m = hdim
00032
00033   nnzstart = m
00034
00035   RETURN
00036
00037 END FUNCTION nnzstart
00038
00039 INTEGER FUNCTION nnzend(M, HDIM)
00040
00041   IMPLICIT NONE
00042   INTEGER, INTENT(IN) :: M, HDIM
00043   INTEGER :: NEWM
00044
00045   newm = 4 * m
00046   IF (newm >= hdim) newm = hdim
00047
00048   nnzend = newm
00049
00050   RETURN
00051
00052 END FUNCTION nnzend
00053

```

8.269 noelec.f90 File Reference

Functions/Subroutines

- subroutine [noelec](#) (NUMEL)

8.269.1 Function/Subroutine Documentation

8.269.1.1 subroutine noelec (real(latteprec) NUMEL)

Definition at line 23 of file [noelec.f90](#).

8.270 noelec.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !

```

```

00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE noelec (NUMEL)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I
00031   REAL(LATTEPREC) :: NUMEL
00032   IF (existerror) RETURN
00033
00034   numel = zero
00035
00036   DO i = 1, hdim
00037
00038     numel = numel + bo(i,i)
00039
00040   ENDDO
00041
00042   RETURN
00043
00044 END SUBROUTINE noelec

```

8.271 nonoarray.f90 File Reference

Modules

- module [nonoarray](#)

Variables

- real(latteprec), dimension(:), allocatable [nonoarray::hjj](#)
- real(latteprec), dimension(:), allocatable [nonoarray::nono_evals](#)
- integer, dimension(:), allocatable [nonoarray::nono_iwork](#)
- integer [nonoarray::nono_liwork](#)
- integer [nonoarray::nono_lwork](#)
- real(latteprec), dimension(:), allocatable [nonoarray::nono_work](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::nonotmp](#)
- integer [nonoarray::nonzero](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::orthoh](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::orthohdown](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::orthohup](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::sh2](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::smat](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::spintmp](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::umat](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::x2hrho](#)
- real(latteprec), dimension(:,:), allocatable [nonoarray::xmat](#)

8.272 nonoarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE nonoarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   ! Work arrays are for DSYEV when generating  $s^{-1/2}$ 
00030   ! SMAT contains the overlap matrix S
00031   ! UMAT contains eigenvectors of S and NONO_EVALS its eigenvalues
00032   ! XMAT contains transformation matrix  $X^{\dagger} S X = 1$ 
00033   ! NONOTMP is an array for storing AB in the product ABC...
00034   ! HORTHO contains  $X^{\dagger} dA H X$ 
00035
00036   INTEGER :: nono_lwork, nono_liwork
00037   INTEGER :: nonzero
00038   INTEGER, ALLOCATABLE :: nono_iwork(:)
00039   REAL(LATTEPREC), ALLOCATABLE :: nono_work(:)
00040   REAL(LATTEPREC), ALLOCATABLE :: umat(:, :), nono_evals(:)
00041   REAL(LATTEPREC), ALLOCATABLE :: xmat(:, :), smat(:, :), nonotmp(:, :)
00042   REAL(LATTEPREC), ALLOCATABLE :: orthoh(:, :)
00043   REAL(LATTEPREC), ALLOCATABLE :: orthohup(:, :), orthohdown(:, :)
00044   REAL(LATTEPREC), ALLOCATABLE :: hjj(:, :), sh2(:, :)
00045
00046   ! For the Pulay force =  $2\text{Tr}[S^{-1} H \rho dS/dR]$ 
00047
00048   REAL(LATTEPREC), ALLOCATABLE :: x2hrho(:, :), spintmp(:, :)
00049
00050 END MODULE nonoarray

```

8.273 norms.f90 File Reference

Functions/Subroutines

- subroutine [norms](#)

8.273.1 Function/Subroutine Documentation

8.273.1.1 subroutine norms ()

Definition at line 23 of file [norms.f90](#).

8.274 norms.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE norms
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE nonoarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, J, K, LWORK, INFO
00032   REAL(LATTEPREC), ALLOCATABLE :: HP(:, :), X2(:, :), DBRHO(:, :), DBH(:, :)
00033   REAL(LATTEPREC), ALLOCATABLE :: XTX(:, :), EVALS(:, :), WORK(:)
00034   REAL(LATTEPREC), ALLOCATABLE :: TRACEARR(:)
00035   REAL(LATTEPREC) :: FROBCOM, FROBIDEM, TRERR, TRTMP
00036   REAL(LATTEPREC) :: TWOCOM, TWOIDEM, TRRHOS
00037
00038
00039   lwork = 3*hdim - 1
00040
00041   ALLOCATE(dbrho(hdim, hdim), dbh(hdim, hdim))
00042   ALLOCATE(evals(hdim), work(lwork))
00043
00044   IF (basistype .EQ. "ORTHO") THEN
00045
00046     IF (latteprec .EQ. 8) THEN
00047
00048       dbrho = bo/2.0d0
00049       dbh = h
00050
00051     ELSEIF (latteprec .EQ. 4) THEN
00052
00053       dbrho = dble(bo)/2.0d0
00054       dbh = dble(h)
00055
00056     ENDIF
00057
00058
00059   ALLOCATE(hp(hdim, hdim))
00060
00061   CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00062     dbh, hdim, dbrho, hdim, 0.0d0, hp, hdim)
00063
00064   CALL dgemm('N', 'N', hdim, hdim, hdim, -1.0d0, &
00065     dbrho, hdim, dbh, hdim, 1.0d0, hp, hdim)
00066
00067   frobcom = 0.0d0
00068
00069   DO i = 1, hdim
00070     DO j = 1, hdim
00071
00072       frobcom = frobcom + hp(j,i)*hp(j,i)
00073
00074     ENDDO
00075   ENDDO
00076
00077   frobcom = sqrt(frobcom)
00078
00079   ALLOCATE(xtx(hdim, hdim))
00080
00081   CALL dgemm('T', 'N', hdim, hdim, hdim, 1.0d0, &
00082     hp, hdim, hp, hdim, 0.0d0, xtx, hdim)
00083
00084   CALL dsyev('N', 'U', hdim, xtx, hdim, evals, work, lwork, info)

```

```

00085
00086     twocom = sqrt(maxval(evals))
00087
00088     DEALLOCATE (hp)
00089
00090     ALLOCATE (x2(hdim, hdim))
00091
00092     x2 = dbrho
00093
00094     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00095             dbrho, hdim, dbrho, hdim, -1.0d0, x2, hdim)
00096
00097     frobidem = zero
00098
00099     DO i = 1, hdim
00100         DO j = 1, hdim
00101
00102             frobidem = frobidem + x2(j,i)*x2(j,i)
00103
00104         ENDDO
00105     ENDDO
00106
00107     frobidem = sqrt(frobidem)
00108
00109     ! Here X2 = P^2 - P
00110
00111     ! For the 2-norm we need the largest eigenvalue of X2^T X2
00112
00113     CALL dgemm('T', 'N', hdim, hdim, hdim, 1.0d0, &
00114             x2, hdim, x2, hdim, 0.0d0, txt, hdim)
00115
00116     CALL dsyev('N', 'U', hdim, txt, hdim, evals, work, lwork, info)
00117
00118     twoidem = sqrt(maxval(evals))
00119
00120     DEALLOCATE (x2, txt)
00121
00122
00123     ! ALLOCATE (TRACEARR (HDIM) )
00124     ! DO I = 1, HDIM
00125     !     TRACEARR(I) = DBRHO(I,I)
00126     ! ENDDO
00127
00128     ! TRTMP = SUM(TRACEARR)
00129
00130     trtmp = 0.0d0
00131
00132     DO i = 1, hdim
00133
00134         trtmp = trtmp + dbrho(i,i)
00135
00136     ENDDO
00137
00138     trerr = abs(trtmp - dble(bndfil)*dble(hdim))
00139
00140
00141
00142
00143     ! DEALLOCATE (DBRHO, DBH, EVALS, WORK)
00144     WRITE(6, '( " " )')
00145     WRITE(6, '( "HDIM = ", I6 )') hdim
00146     WRITE(6, '( "Occupation error = ", G26.16 )') trerr
00147     WRITE(6, '( "Frobenius norms:" )')
00148     WRITE(6, '( "Commutation error = ", G26.16 )') frobcom
00149     WRITE(6, '( "Idempotency error = ", G26.16 )') frobidem
00150     WRITE(6, '( "Two norms:" )')
00151     WRITE(6, '( "Commutation error = ", G26.16 )') twocom
00152     WRITE(6, '( "Idempotency error = ", G26.16 )') twoidem
00153     WRITE(6, 10) hdim, trerr, frobcom, frobidem, twocom, twoidem
00154
00155 10  FORMAT(i8, 5g26.16)
00156
00157     ELSEIF (basistype .EQ. "NONORTHO") THEN
00158
00159         ! Check rho S rho:
00160
00161
00162         IF (latteprec .EQ. 8) THEN
00163
00164             dbrho = bo/2.0d0
00165             dbh = h
00166
00167         ELSEIF (latteprec .EQ. 4) THEN
00168
00169             dbrho = dble(bo)/2.0d0
00170             dbh = dble(h)
00171

```

```

00172      ENDIF
00173
00174      ALLOCATE (xtx(hdim, hdim))
00175      ALLOCATE (x2(hdim, hdim))
00176
00177      !      X2 = DBRHO
00178
00179      CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00180               dbrho, hdim, smat, hdim, 0.0d0, xtx, hdim)
00181
00182      CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00183               xtx, hdim, dbrho, hdim, 0.0d0, x2, hdim)
00184
00185      x2 = x2 - dbrho
00186
00187
00188      CALL dgemm('T', 'N', hdim, hdim, hdim, 1.0d0, &
00189               x2, hdim, x2, hdim, 0.0d0, xtx, hdim)
00190
00191      CALL dsyev('N', 'U', hdim, xtx, hdim, evals, work, lwork, info)
00192
00193      twoidem = sqrt(maxval(evals))
00194
00195      trrhos = 0.0d0
00196
00197      DO i = 1, hdim
00198         DO j = 1, hdim
00199            trrhos = trrhos + dbrho(j,i)*smat(i,j)
00200         ENDDO
00201      ENDDO
00202
00203      print*, "Idempotency error: || rSr - r ||_2 =", twoidem
00204      print*, "Tr(rho S) =", trrhos
00205
00206      DEALLOCATE(x2, xtx)
00207      ENDIF
00208
00209      DEALLOCATE(dbrho, dbh, evals, work)
00210      RETURN
00211
00212 END SUBROUTINE norms

```

8.275 nptrescale.f90 File Reference

Functions/Subroutines

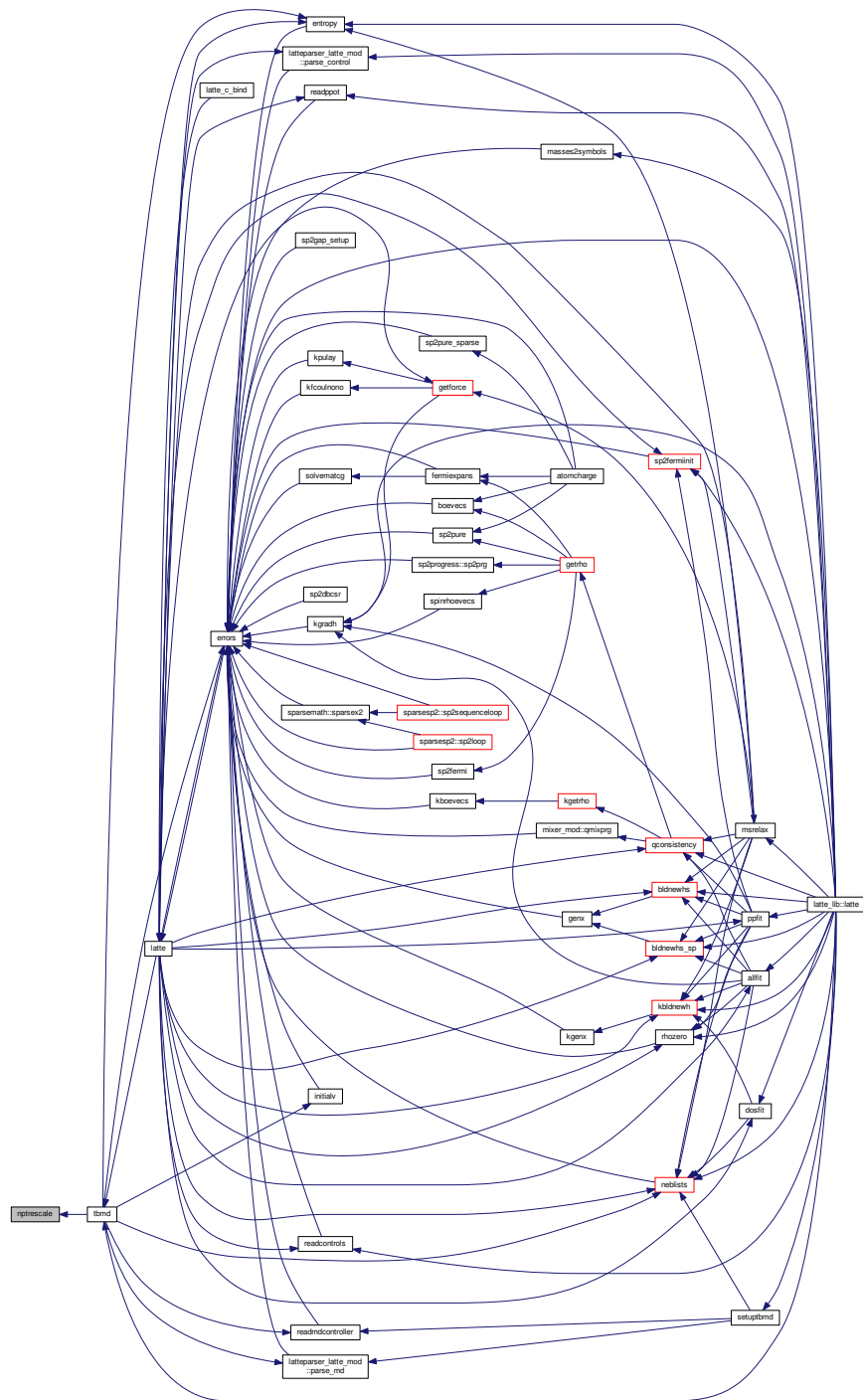
- subroutine [nptrescale](#)

8.275.1 Function/Subroutine Documentation

8.275.1.1 subroutine nptrescale ()

Definition at line 24 of file [nptrescale.f90](#).

Here is the caller graph for this function:



8.276 **nptrescale.f90**

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022
00023 SUBROUTINE nptrescale
00024
00025   USE constants_mod
00026   USE setuparray
00027   USE mdarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, K
00033   REAL(LATTEPREC) :: PREFACTOR, PREPREFACT
00034   REAL(LATTEPREC) :: CHI, TMPTEMP, MYTEMP
00035   REAL(LATTEPREC) :: MAXCHI = 1.05d0, maxdbox = 0.001d0
00036   REAL(LATTEPREC) :: PSCALE = 100.0d0
00037   REAL(LATTEPREC) :: TMPPRESS, MYPRESS, DBOX
00038   REAL(LATTEPREC) :: TMPPRESSX, TMPPRESSY, TMPPRESSZ
00039   REAL(LATTEPREC) :: DBOXX, DBOXY, DBOXZ, MYPRESSX, MYPRESSY, MYPRESSZ
00040   IF (existerror) RETURN
00041
00042   preprefact = half*f2v*dt
00043
00044   tmptemp = zero
00045   ! Isotropic
00046   tmppress = zero
00047   ! non-isotropic
00048   tmppressx = zero
00049   tmppressy = zero
00050   tmppressz = zero
00051
00052   DO i = 1, aveper
00053
00054     IF (npttype .EQ. "ISO") THEN
00055       tmppress = tmppress + phist(i)
00056     ELSE
00057       tmppressx = tmppressx + phistx(i)
00058       tmppressy = tmppressy + phisty(i)
00059       tmppressz = tmppressz + phistz(i)
00060     ENDIF
00061
00062   ENDDO
00063
00064   mypress = tmppress/REAL(aveper)
00065
00066   mypressx = tmppressx/REAL(aveper)
00067   mypressy = tmppressy/REAL(aveper)
00068   mypressz = tmppressz/REAL(aveper)
00069
00070   ! Change box dimensions depending on the average pressure
00071
00072   IF (npttype .EQ. "ISO") THEN
00073
00074     dbox = min(abs(mypress - ptarget)*maxdbox, maxdbox)
00075
00076     dbox = sign(dbox, mypress - ptarget)
00077
00078     box = box * (one + dbox)
00079
00080     cr = cr * (one + dbox)
00081
00082   ELSE ! Allow the three vectors to change length independently
00083
00084     dboxx = min(abs(mypressx - ptarget)*maxdbox, maxdbox)
00085     dboxy = min(abs(mypressy - ptarget)*maxdbox, maxdbox)
00086     dboxz = min(abs(mypressz - ptarget)*maxdbox, maxdbox)
00087
00088     dboxx = sign(dboxx, mypressx - ptarget)
00089     dboxy = sign(dboxy, mypressy - ptarget)
00090     dboxz = sign(dboxz, mypressz - ptarget)
00091
00092     box(1,1) = box(1,1) * (one + dboxx)
00093     box(2,2) = box(2,2) * (one + dboxy)

```

```
00094      box(3,3) = box(3,3) * (one + dboxz)
00095
00096      DO i = 1, nats
00097
00098          cr(1,i) = cr(1,i) * (one + dboxx)
00099          cr(2,i) = cr(2,i) * (one + dboxy)
00100          cr(3,i) = cr(3,i) * (one + dboxz)
00101
00102      ENDDO
00103
00104      ENDIF
00105
00106      RETURN
00107
00108 END SUBROUTINE nptrescale
00109
```

8.277 nvtandersen.f90 File Reference

Functions/Subroutines

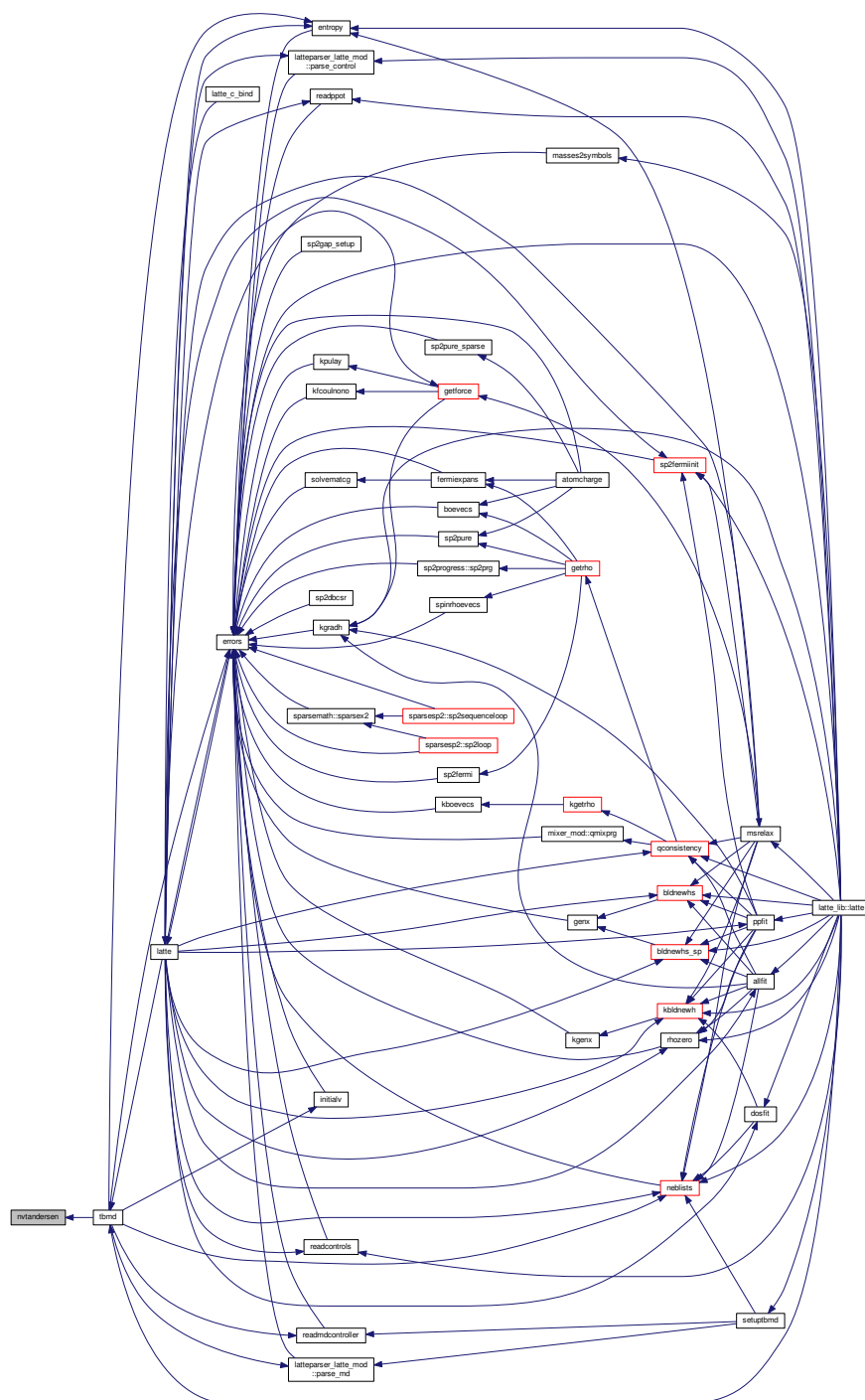
- subroutine [nvtandersen](#)

8.277.1 Function/Subroutine Documentation

8.277.1.1 subroutine [nvtandersen](#) ()

Definition at line 23 of file [nvtandersen.f90](#).

Here is the caller graph for this function:



8.278 nvtandersen.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\
00021
00022 SUBROUTINE nvtandersen
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I
00032   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00033   REAL(LATTEPREC) :: RN, MEAN, STDDEV
00034   REAL(LATTEPREC) :: BOLTZ, MYMASS
00035   IF (existererror) RETURN
00036
00037   setth = 0
00038   boltz = one/ke2t
00039
00040   DO i = 1, nats
00041
00042     CALL random_number(rn)
00043
00044     IF (rn .LT. cumdt/friction) THEN
00045
00046       mymass = mass(elempointer(i))
00047       stddev = sqrt(boltz*ttarget/(mymass*mvv2ke))
00048
00049       v(1,i) = gaussrn(zero, stddev)
00050       v(2,i) = gaussrn(zero, stddev)
00051       v(3,i) = gaussrn(zero, stddev)
00052
00053       cumdt = zero
00054
00055     ENDIF
00056
00057   ENDDO
00058
00059 END SUBROUTINE nvtandersen

```

8.279 nvtlangevin.f90 File Reference

Functions/Subroutines

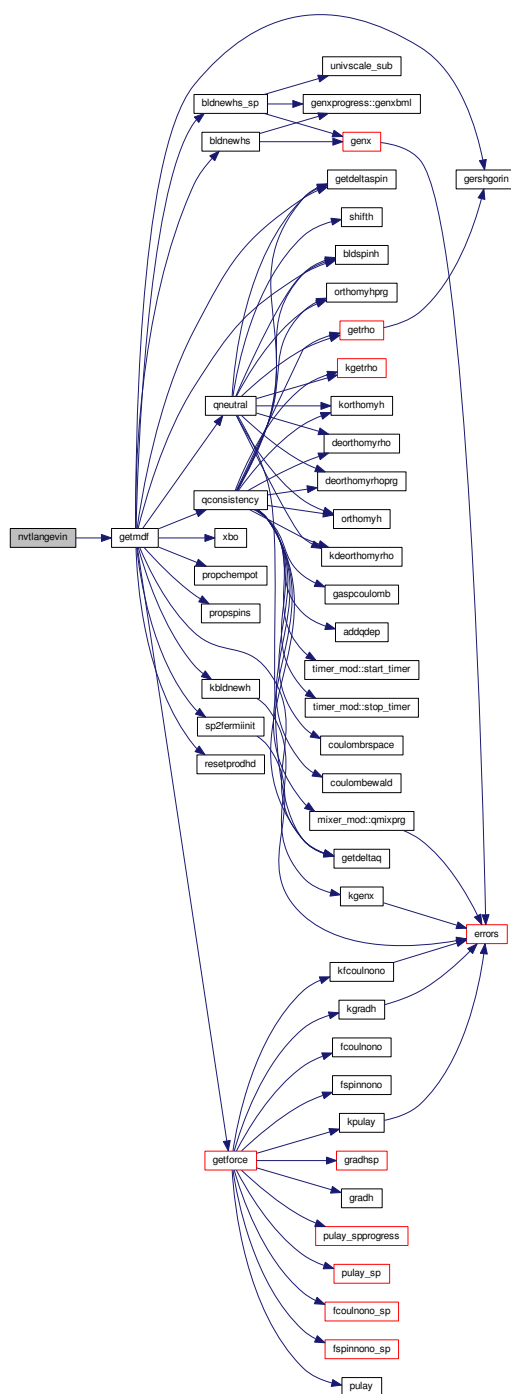
- subroutine [nvtlangevin](#) (ITER)

8.279.1 Function/Subroutine Documentation

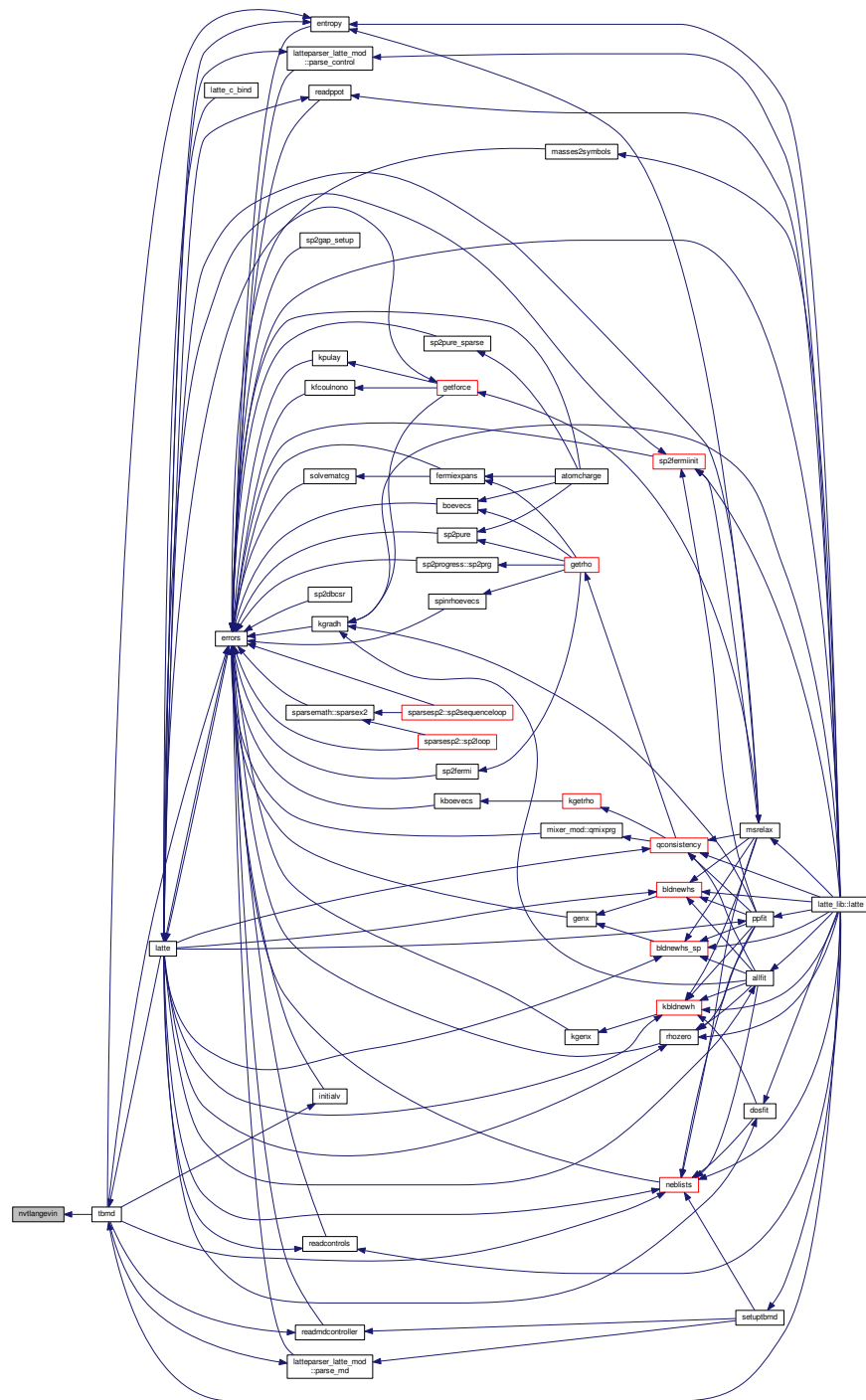
8.279.1.1 subroutine [nvtlangevin](#) (integer ITER)

Definition at line 28 of file [nvtlangevin.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.280 nvtlangevin.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\
00021
00022 !!$The algorithm developed by Gronbech-Jensen and Farago is implemented
00023 !!$Molecular Physics 111 (2013) 983-991
00024 !!$
00025 !!$Contribution from Enrique Martinez
00026
00027 SUBROUTINE nvtlangevin(ITER)
00028
00029   USE constants_mod
00030   USE setuparray
00031   USE mdarray
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, K, N
00037   INTEGER :: ITER
00038   REAL(LATTEPREC) :: BETA, b, A
00039   REAL(LATTEPREC), EXTERNAL :: GAUSSRN
00040   REAL(LATTEPREC) :: MEAN, STDDEV
00041   REAL(LATTEPREC) :: BOLTZ, SQMVV2KE
00042   REAL(LATTEPREC) :: PREFRIC, PREB, V1TERM, V2TERM, V3TERM
00043   REAL(LATTEPREC) :: GAMMA1, GAMMA2, TSQRT, DTF, DTV, DTFM
00044   REAL(LATTEPREC) :: PREF1, PREF2, MELPREF, MULTFACT, MYMASS
00045   INTEGER :: OPTION
00046   IF (existerror) RETURN
00047
00048   setth = 0
00049
00050   boltz = one/ke2t
00051
00052   mean = zero
00053
00054   DO i = 1, nats
00055
00056     !!$      Update positions
00057
00058     mymass = mass(elempointer(i))
00059
00060     gammal = mymass/friction
00061
00062     stddev = sqrt(two*gammal*boltz*ttarget*f2v*dt)
00063
00064     prefric = (gammal*dt)/(two*mymass)
00065     b = one / (one + prefric)
00066     a = (one - prefric)/(one + prefric)
00067     preb = (b*dt)/(two*mymass)
00068
00069     cr(1,i) = cr(1,i) + b*dt*v(1,i) + preb*dt*f2v*ftot(1,i) + &
00070       preb*gaussrn(mean,stddev)
00071
00072     cr(2,i) = cr(2,i) + b*dt*v(2,i) + preb*dt*f2v*ftot(2,i) + &
00073       preb*gaussrn(mean,stddev)
00074
00075     cr(3,i) = cr(3,i) + b*dt*v(3,i) + preb*dt*f2v*ftot(3,i) + &
00076       preb*gaussrn(mean,stddev)
00077
00078     ! Update velocities
00079
00080     multfact = (dt*f2v)/(two*mymass)
00081
00082     v(1,i) = a * (v(1,i) + multfact * ftot(1,i)) + &
00083       (b/mymass)*gaussrn(mean,stddev)
00084
00085     v(2,i) = a * (v(2,i) + multfact * ftot(2,i)) + &
00086       (b/mymass)*gaussrn(mean,stddev)
00087
00088     v(3,i) = a * (v(3,i) + multfact * ftot(3,i)) + &
00089       (b/mymass)*gaussrn(mean,stddev)
00090
00091   ENDDO
00092
00093   !

```

```
00094      ! Get new force to complete advance in V
00095      !
00096
00097      CALL getmdf(1, 100)
00098
00099      DO i = 1, nats
00100
00101          multfact = (dt*f2v)/(two*mass(elempointer(i)))
00102
00103          v(1,i) = v(1,i) + multfact*ftot(1,i)
00104          v(2,i) = v(2,i) + multfact*ftot(2,i)
00105          v(3,i) = v(3,i) + multfact*ftot(3,i)
00106
00107      ENDDO
00108
00109      RETURN
00110
00111 END SUBROUTINE nvtlangevin
00112
00113
00114
```

8.281 nvtNH.f90 File Reference

Functions/Subroutines

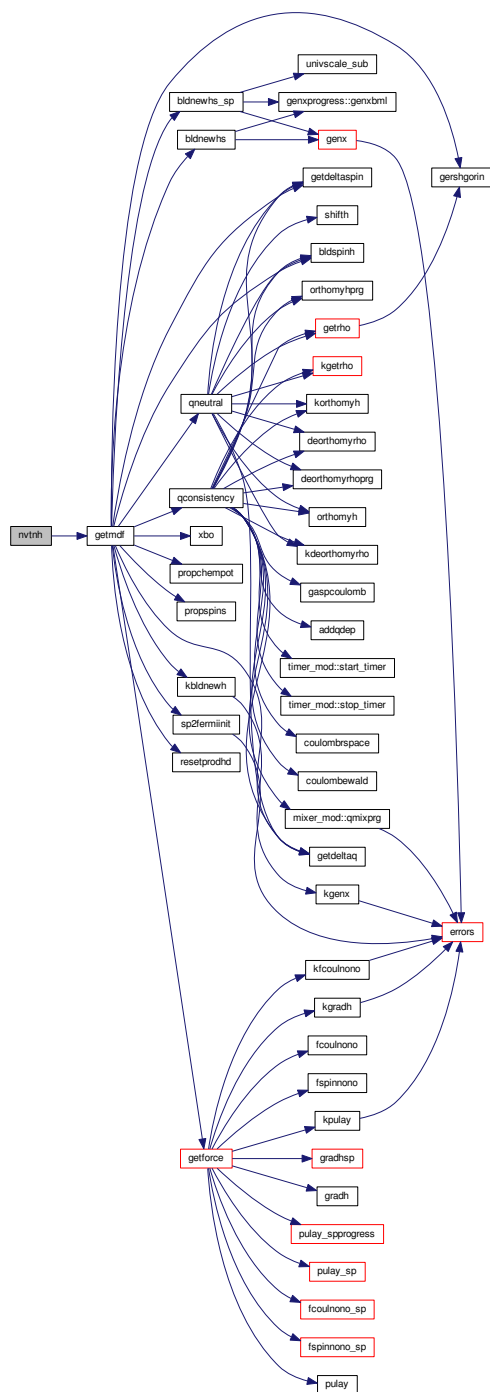
- subroutine [nvtnh](#)

8.281.1 Function/Subroutine Documentation

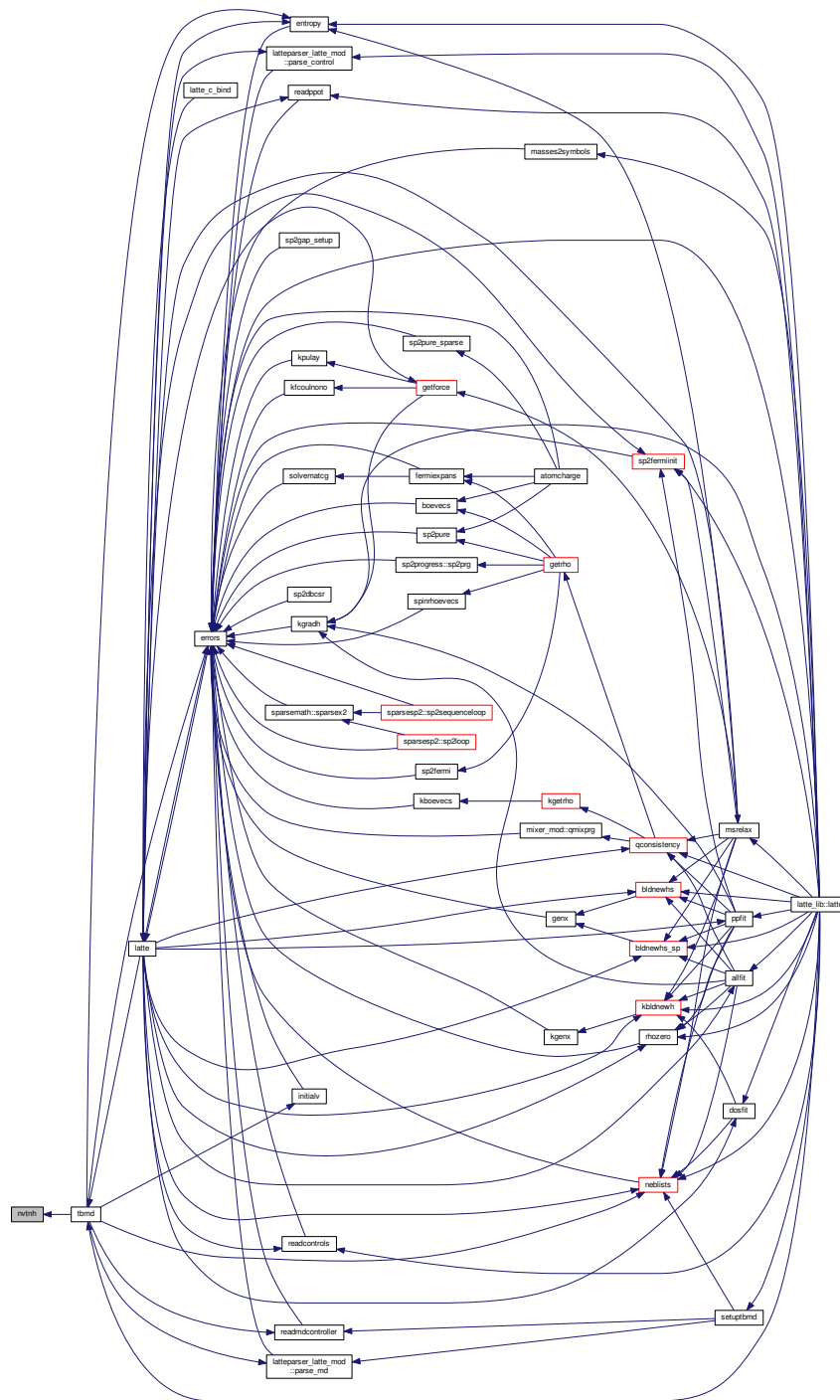
8.281.1.1 subroutine [nvtnh](#) ()

Definition at line 26 of file [nvtNH.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.282 **nvtNH.f90**

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\
00021
00022 !!$ Nose thermostat from Anders Niklasson derivation
00023 !!$ Contributing author: Enrique Martinez
00024
00025 SUBROUTINE nvtnh
00026
00027   USE constants_mod
00028   USE setuparray
00029   USE mdarray
00030   USE myprecision
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, J, K, N
00035   INTEGER :: ITER
00036   REAL(LATTEPREC) :: KEV, KO
00037   REAL(LATTEPREC) :: PREF1, PREF2, NOSEPREF
00038   IF (existerror) RETURN
00039
00040   kev = zero
00041   DO i = 1, nats
00042
00043     kev = kev + mass(elempointer(i)) * &
00044       & (v(1,i)*v(1,i) + v(2,i)*v(2,i) + v(3,i)*v(3,i))
00045
00046   ENDDO
00047
00048   kev = mvv2ke*kev/two
00049
00050   ko = three*REAL(nats)*TTARGET/(two*ke2t)
00051
00052   dgamma = dgamma + (dt/friction)*(kev - ko)/two
00053   gamma = gamma + dt*dgamma/two
00054   dgamma = dgamma + (dt/friction)*(kev - ko)/two
00055
00056   pref1 = f2v*dt/two
00057   nosepref = one / (one + (dt*dgamma/two))
00058
00059   DO i = 1, nats
00060
00061     pref2 = pref1/mass(elempointer(i))
00062
00063     v(1,i) = nosepref*( v(1,i) + pref2*ftot(1,i) )
00064     v(2,i) = nosepref*( v(2,i) + pref2*ftot(2,i) )
00065     v(3,i) = nosepref*( v(3,i) + pref2*ftot(3,i) )
00066
00067     cr(1,i) = cr(1,i) + dt*v(1,i)
00068     cr(2,i) = cr(2,i) + dt*v(2,i)
00069     cr(3,i) = cr(3,i) + dt*v(3,i)
00070
00071   ENDDO
00072
00073   CALL getmdf(1, 100)
00074
00075
00076   nosepref = one - dt*dgamma/two
00077
00078   DO i = 1, nats
00079
00080     pref2 = pref1/mass(elempointer(i))
00081
00082     v(1,i) = nosepref*v(1,i) + pref2*ftot(1,i)
00083     v(2,i) = nosepref*v(2,i) + pref2*ftot(2,i)
00084     v(3,i) = nosepref*v(3,i) + pref2*ftot(3,i)
00085
00086   ENDDO
00087
00088   kev = zero
00089   DO i = 1, nats
00090     kev = kev + mass(elempointer(i)) * &
00091       & (v(1,i)*v(1,i) + v(2,i)*v(2,i) + v(3,i)*v(3,i))
00092   ENDDO
00093

```

```
00094   kev = mvv2ke*kev/two
00095
00096   dgamma = dgamma + (dt/friction)*(kev - ko)/two
00097   gamma = gamma + dt*dgamma/two
00098   dgamma = dgamma + (dt/friction)*(kev - ko)/two
00099
00100   consmot = friction*dgamma*dgamma/two + two*ko*
gamma
00101
00102   RETURN
00103
00104 END SUBROUTINE nvtanh
```

8.283 nvtrescale.f90 File Reference

Functions/Subroutines

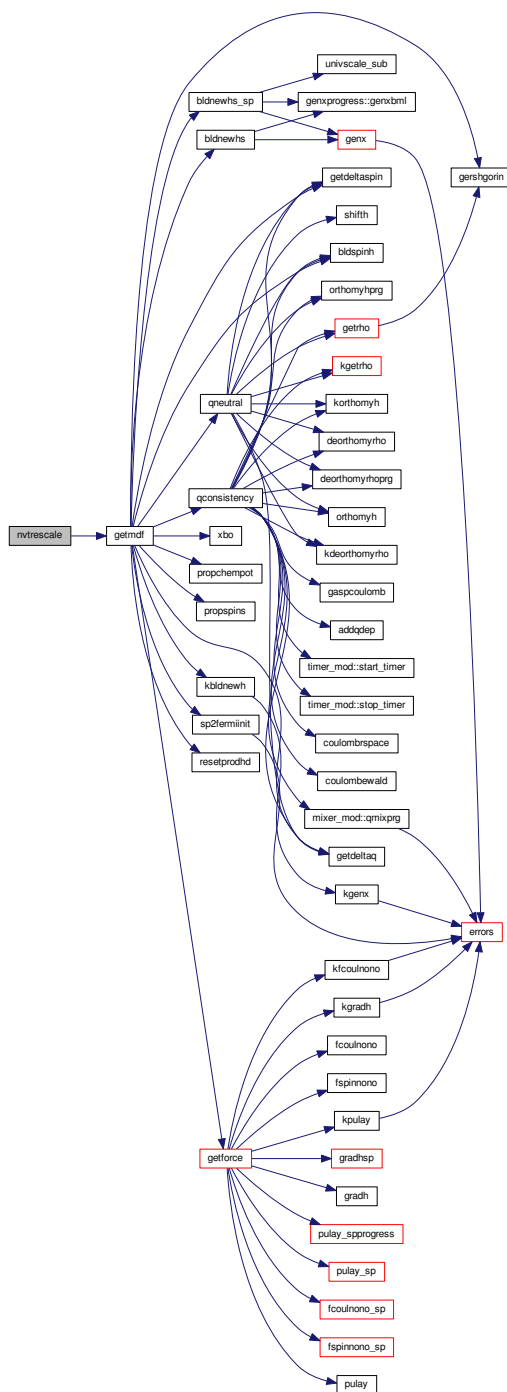
- subroutine [nvtrescale](#)

8.283.1 Function/Subroutine Documentation

8.283.1.1 subroutine [nvtrescale](#) ()

Definition at line 23 of file [nvtrescale.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE nvtrescale
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, J, K
00032   REAL(LATTEPREC) :: PREFACTOR, PREPREFACT
00033   REAL(LATTEPREC) :: CHI, TMPTEMP, MYTEMP, MOMENTUM(3)
00034   REAL(LATTEPREC) :: MAXCHI = 1.05d0
00035   IF (existerror) RETURN
00036
00037   preprefact = half*f2v*dt
00038
00039   tmptemp = zero
00040
00041   DO i = 1, aveper
00042     tmptemp = tmptemp + thist(i)
00043   ENDDO
00044
00045   mytemp = tmptemp/float(aveper)
00046
00047   ! CHI = SQRT(TTARGET/MYTEMP)
00048
00049   ! CHI = MIN(CHI, MAXCHI)
00050   chi = min(sqrt(ttarget/mytemp), maxchi)
00051
00052   ! PRINT*, "MYTEMP = ", MYTEMP, "CHI = ", CHI
00053
00054   DO i = 1, nats
00055
00056     prefactor = preprefact/mass(elempointer(i))
00057
00058     !
00059     ! Half timestep advance in V with velocity rescale
00060     !
00061
00062     v(1,i) = chi*v(1,i) + prefactor*ftot(1,i)
00063     v(2,i) = chi*v(2,i) + prefactor*ftot(2,i)
00064     v(3,i) = chi*v(3,i) + prefactor*ftot(3,i)
00065
00066   ENDDO
00067
00068   !
00069   ! Whole timestep advance in positions
00070   !
00071
00072   cr = cr + dt*v
00073
00074   !
00075   ! Get new force to complete advance in V
00076   !
00077
00078   CALL getmdf(1, 100)
00079
00080   !
00081   ! Now finish advancing V with F(t + dt)
00082   !
00083
00084   DO i = 1, nats
00085
00086     prefactor = preprefact/mass(elempointer(i))
00087
00088     v(1,i) = v(1,i) + prefactor*ftot(1,i)
00089     v(2,i) = v(2,i) + prefactor*ftot(2,i)
00090     v(3,i) = v(3,i) + prefactor*ftot(3,i)
00091
00092   ENDDO
00093

```

```

00094      ! Remove drift
00095
00096      momentum = zero
00097
00098      DO i = 1, nats
00099          momentum(1) = momentum(1) + v(1,i)*mass(elempointer(i))
00100          momentum(2) = momentum(2) + v(2,i)*mass(elempointer(i))
00101          momentum(3) = momentum(3) + v(3,i)*mass(elempointer(i))
00102      ENDDO
00103
00104      momentum = momentum/summass
00105
00106      DO i = 1, nats
00107          v(1,i) = v(1,i) - momentum(1)
00108          v(2,i) = v(2,i) - momentum(2)
00109          v(3,i) = v(3,i) - momentum(3)
00110      ENDDO
00111
00112      RETURN
00113
00114 END SUBROUTINE nvtrescale
00115

```

8.285 openfiles_mod.f90 File Reference

Modules

- module [openfiles_mod](#)
Module to handle input output files.

Functions/Subroutines

- integer function, public [openfiles_mod::get_file_unit](#) (IO_MAX)
Returns a unit number that is not in use.
- subroutine, public [openfiles_mod::open_file](#) (IO, NAME)
Opens a file to write.
- subroutine, public [openfiles_mod::open_file_to_read](#) (IO, NAME)
Opens a file to read.

8.286 openfiles_mod.f90

```

00001
00003 MODULE openfiles_mod
00004
00005     IMPLICIT NONE
00006
00007     PRIVATE
00008
00009     PUBLIC :: get_file_unit, open_file, open_file_to_read
00010
00011 CONTAINS
00012
00017 INTEGER FUNCTION get_file_unit(IO_MAX)
00018     IMPLICIT NONE
00019     INTEGER :: IO_MAX, IO, M, IOSTAT
00020     LOGICAL :: OPENED
00021
00022     m = io_max ; IF (m < 1) m = 97
00023     DO io = m,1,-1
00024         INQUIRE (unit=io, opened=opened, iostat=iostat)
00025         IF(iostat.NE.0) cycle
00026         IF(.NOT.opened) EXIT
00027     END DO
00028     get_file_unit = io
00029
00030 END FUNCTION get_file_unit

```



```

00031
00036 SUBROUTINE open_file(IO,NAME)
00037 IMPLICIT NONE
00038 CHARACTER(LEN=*) :: NAME
00039 CHARACTER(100) :: IO_NAME
00040 INTEGER :: IO
00041
00042 io=get_file_unit(100)
00043 io_name=trim(name)
00044 OPEN(io,file=io_name)
00045
00046 END SUBROUTINE open_file
00047
00052 SUBROUTINE open_file_to_read(IO,NAME)
00053 IMPLICIT NONE
00054 CHARACTER(LEN=*) :: NAME
00055 CHARACTER(100) :: IO_NAME
00056 INTEGER :: IO
00057 LOGICAL :: EXISTS
00058
00059 io=get_file_unit(100)
00060 io_name=trim(name)
00061
00062 INQUIRE(file=io_name, exist=exists)
00063 IF (.NOT.exists) THEN
00064     WRITE(*,*) "FILE ",io_name," DOES NOT EXIST ..."
00065     stop
00066 ENDIF
00067 OPEN(io,file=io_name,status="OLD")
00068
00069 END SUBROUTINE open_file_to_read
00070
00071 END MODULE openfiles_mod

```

8.287 orthomyH.f90 File Reference

Functions/Subroutines

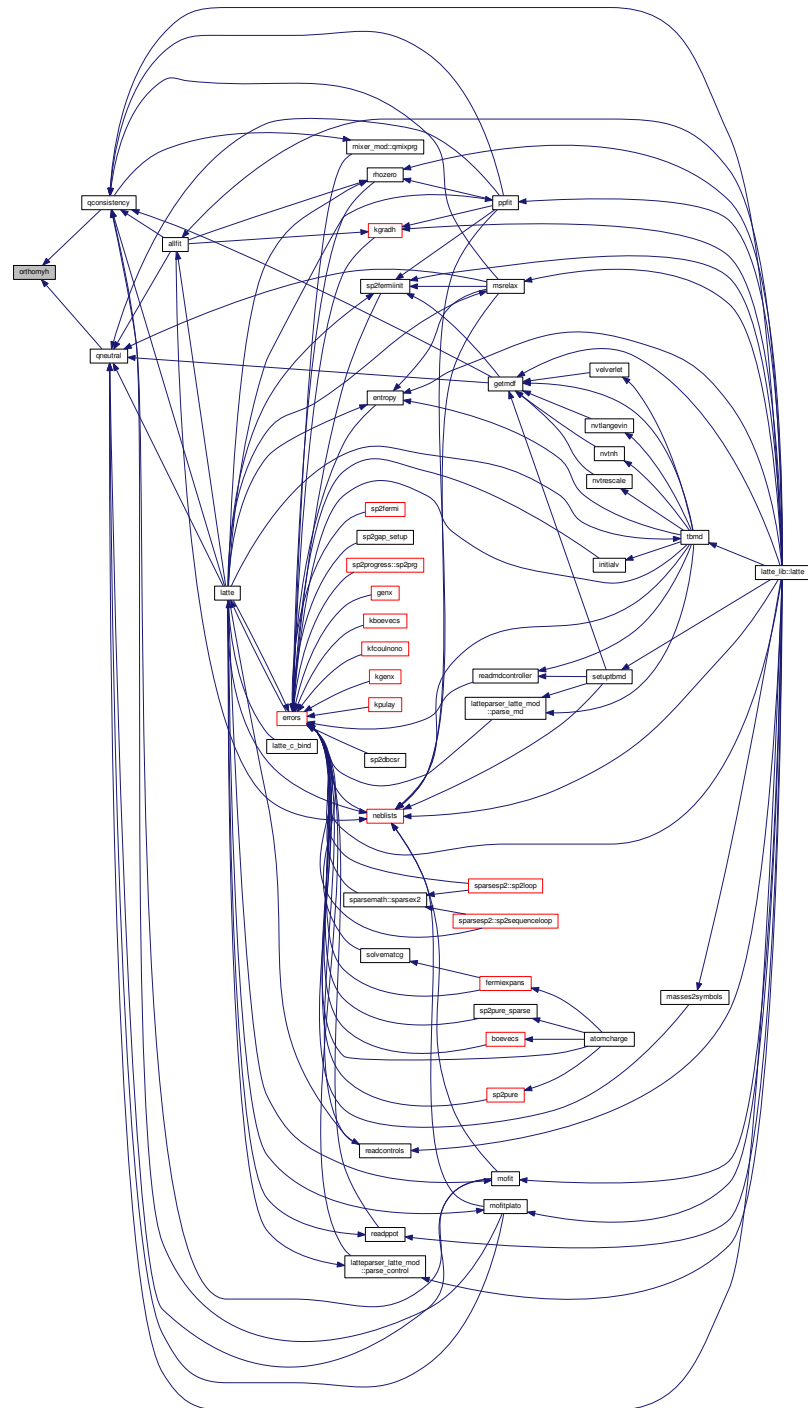
- subroutine [orthomyh](#)

8.287.1 Function/Subroutine Documentation

8.287.1.1 subroutine orthomyh ()

Definition at line 23 of file [orthomyH.f90](#).

Here is the caller graph for this function:



8.288 orthomyH.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE orthomyh
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE nonoarray
00027   USE spinarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER I, J
00033   IF (existerror) RETURN
00034
00035   !
00036   ! ORTHOH = X^dag H X
00037   !
00038   !
00039   !
00040   ! Don't overwrite H - we update this with new charges and build it
00041   ! from scratch only after moving the atoms
00042   !
00043
00044   IF (spinon .EQ. 0) THEN
00045
00046 #ifdef DOUBLEPREC
00047
00048     CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00049       xmat, hdim, h, hdim, zero, nonotmp, hdim)
00050     CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00051       nonotmp, hdim, xmat, hdim, zero, orthoh,
00052       hdim)
00053 #elif defined(SINGLEPREC)
00054
00055     CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00056       xmat, hdim, h, hdim, zero, nonotmp, hdim)
00057     CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00058       nonotmp, hdim, xmat, hdim, zero, orthoh,
00059       hdim)
00060 #endif
00061
00062     IF (debugon .EQ. 1) THEN
00063
00064       OPEN (unit=33, status="UNKNOWN", file="myXHX.dat")
00065
00066       DO i = 1, hdim
00067
00068         WRITE(33,10) (orthoh(i,j), j = 1, hdim)
00069
00070       ENDDO
00071
00072       CLOSE(33)
00073
00074 10    FORMAT(100g18.9)
00075
00076     ENDIF
00077
00078   ELSE
00079
00080 #ifdef DOUBLEPREC
00081
00082     CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00083       xmat, hdim, hup, hdim, zero, nonotmp, hdim)
00084     CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00085       nonotmp, hdim, xmat, hdim, zero, orthohup,
00086       hdim)
00087
00088     CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00089       xmat, hdim, hdown, hdim, zero, nonotmp,
00090       hdim)
00091     CALL dgemm('N', 'N', hdim, hdim, hdim, one, &

```

```

00090          nonotmp, hdim, xmat, hdim, zero, orthohdown,
00091          hdim)
00092 #elif defined(SINGLEPREC)
00093
00094          CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00095          xmat, hdim, hup, hdim, zero, nonotmp, hdim)
00096          CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00097          nonotmp, hdim, xmat, hdim, zero, orthohup,
00098          hdim)
00099          CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00100          xmat, hdim, hdown, hdim, zero, nonotmp,
00101          hdim)
00102          CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00103          nonotmp, hdim, xmat, hdim, zero, orthohdown,
00104          hdim)
00105 #endif
00106          ENDIF
00107
00108          RETURN
00109
00110 END SUBROUTINE orthomyh
00111

```

8.289 orthomyHprogress.f90 File Reference

Functions/Subroutines

- subroutine [orthomyhprg](#)

8.289.1 Function/Subroutine Documentation

8.289.1.1 subroutine [orthomyhprg](#) ()

Definition at line 23 of file [orthomyHprogress.f90](#).

[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE orthomyhprg
00023
00024 #ifdef PROGRESSON
00025
00026     USE constants_mod
00027     USE setuparray
00028     USE nonoarray
00029     USE spinarray
00030     USE myprecision
00031
00032     USE bml
00033     USE prg_nonortho_mod
00034     USE prg_extras_mod
00035     USE genxprogress
00036     USE latteparser_latte_mod
00037
00038     IMPLICIT NONE
00039
00040     INTEGER I, J
00041     REAL(LATTEPREC) :: MLSI
00042
00043     TYPE(bml_matrix_t) :: ORTHOH_BML, HAM_BML
00044     IF (existerror) RETURN
00045
00046     !
00047     ! ORTHOH = X^dag H X
00048     !
00049     mlsi = mls()
00050     IF (spinon .EQ. 0) THEN
00051         !! Convert Hamiltonian to bml format
00052         IF (lt%MDIM == -1) lt%MDIM = hdim
00053
00054         CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00055             latteprec, hdim, lt%MDIM, orthoh_bml)
00056
00057         CALL bml_import_from_dense(lt%BML_TYPE, &
00058             h, ham_bml, lt%THRESHOLD, lt%MDIM)
00059
00060         CALL prg_orthogonalize(ham_bml, zmat_bml, orthoh_bml, &
00061             lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)
00062
00063         CALL bml_export_to_dense(orthoh_bml, orthoh)
00064
00065         CALL bml_deallocate(ham_bml)
00066         CALL bml_deallocate(orthoh_bml)
00067
00068         IF (debugon .EQ. 1) THEN
00069
00070             OPEN (unit=33, status="UNKNOWN", file="myXHX.dat")
00071
00072             DO i = 1, hdim
00073
00074                 WRITE(33,10) (orthoh(i,j), j = 1, hdim)
00075
00076             ENDDO
00077
00078             CLOSE(33)
00079
00080 10      FORMAT(100g18.9)
00081
00082         ENDIF
00083
00084     ELSE
00085
00086         CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00087             latteprec, hdim, lt%MDIM, orthoh_bml)
00088
00089         CALL bml_import_from_dense(lt%BML_TYPE, &
00090             hup, ham_bml, zero, lt%MDIM)
00091
00092         CALL prg_orthogonalize(ham_bml, zmat_bml, orthoh_bml, &
00093             lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)

```

```

00094
00095     CALL bml_export_to_dense(orthoh_bml, orthohup)
00096
00097     CALL bml_import_from_dense(lt%BML_TYPE, &
00098         hdown, ham_bml, zero, lt%MDIM)
00099
00100     CALL prg_orthogonalize(ham_bml, zmat_bml, orthoh_bml, &
00101         lt%THRESHOLD, lt%BML_TYPE, lt%VERBOSE)
00102
00103     CALL bml_export_to_dense(orthoh_bml, orthohdown)
00104
00105     CALL bml_deallocate(ham_bml)
00106     CALL bml_deallocate(orthoh_bml)
00107
00108     ENDIF
00109
00110     WRITE(*,*) "Time for ORTHOMYH =", mls()-mlsi
00111
00112     RETURN
00113
00114 #endif
00115
00116 END SUBROUTINE orthomyhprg

```

8.291 orthomyrho.f90 File Reference

Functions/Subroutines

- subroutine [orthomyrho](#)

8.291.1 Function/Subroutine Documentation

8.291.1.1 subroutine orthomyrho ()

Definition at line 23 of file [orthomyrho.f90](#).

8.292 orthomyrho.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE orthomyrho
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE nonoarray
00027     USE spinarray
00028     USE myprecision
00029
00030     IMPLICIT NONE

```

```

00031  IF (existerror) RETURN
00032
00033  !
00034  ! ORTHORHO = X^dag RHO X
00035  !
00036
00037  IF (spinon .EQ. 0) THEN
00038
00039  #ifdef DOUBLEPREC
00040
00041      CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00042               xmat, hdim, bo, hdim, zero, nonotmp, hdim)
00043
00044      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00045               nonotmp, hdim, xmat, hdim, zero, bo, hdim)
00046
00047  #elif defined(SINGLEPREC)
00048
00049      CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00050               xmat, hdim, bo, hdim, zero, nonotmp, hdim)
00051
00052      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00053               nonotmp, hdim, xmat, hdim, zero, bo, hdim)
00054
00055  #endif
00056
00057  ELSE
00058
00059  #ifdef DOUBLEPREC
00060
00061      CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00062               xmat, hdim, rhoup, hdim, zero, nonotmp,
00063               hdim)
00064
00065      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00066               nonotmp, hdim, xmat, hdim, zero, rhoup,
00067               hdim)
00068
00069      CALL dgemm('T', 'N', hdim, hdim, hdim, one, &
00070               xmat, hdim, rhodown, hdim, zero, nonotmp,
00071               hdim)
00072
00073  #elif defined(SINGLEPREC)
00074
00075      CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00076               xmat, hdim, rhoup, hdim, zero, nonotmp,
00077               hdim)
00078
00079      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00080               nonotmp, hdim, xmat, hdim, zero, rhoup,
00081               hdim)
00082
00083      CALL sgemm('T', 'N', hdim, hdim, hdim, one, &
00084               xmat, hdim, rhodown, hdim, zero, nonotmp,
00085               hdim)
00086
00087  #endif
00088
00089  ENDIF
00090
00091  RETURN
00092
00093  END SUBROUTINE orthomyrho
00094

```

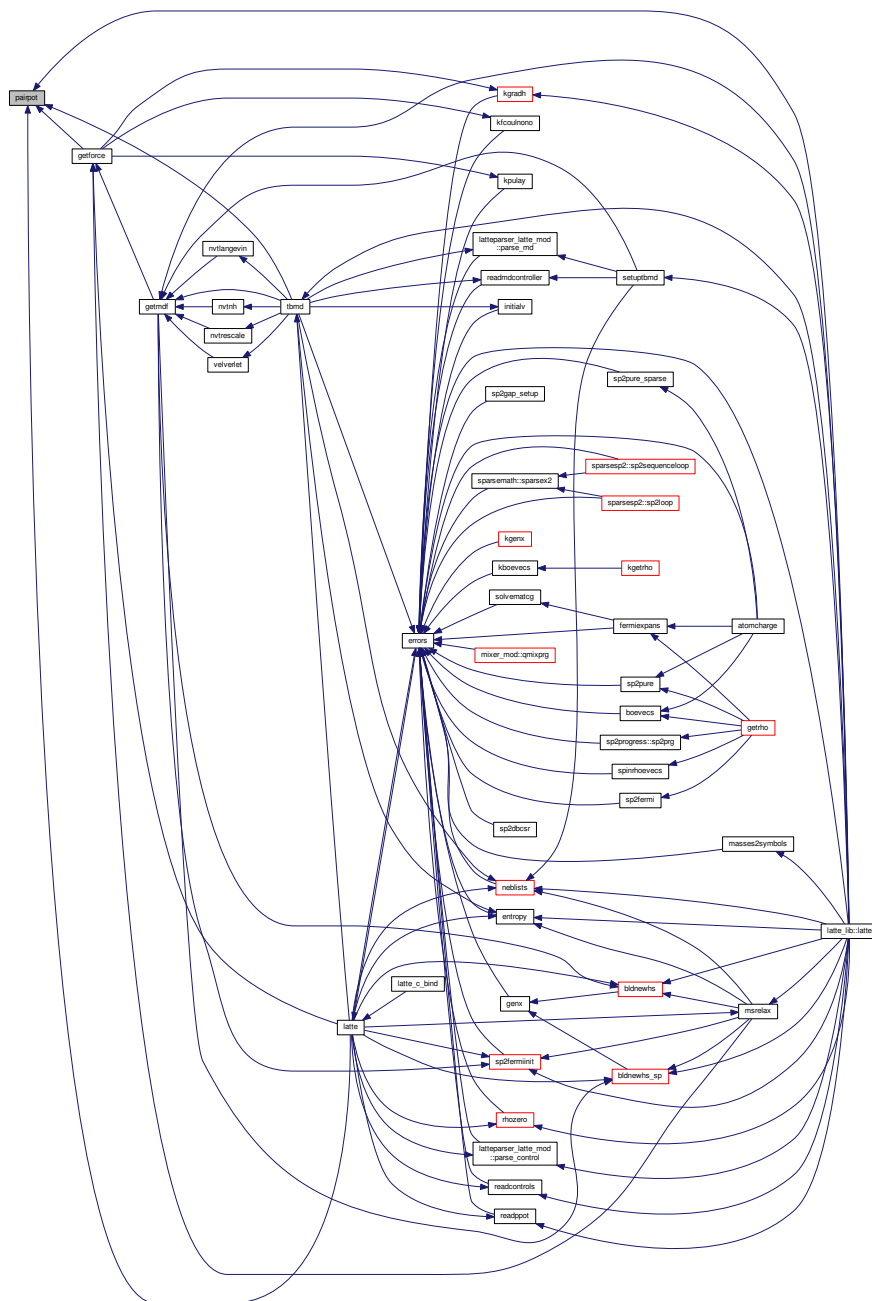
8.293 pairpot.f90 File Reference

Functions/Subroutines

- subroutine [pairpot](#)

8.293.1.1 subroutine pairpot ()

Here is the caller graph for this function:

[illegible]

```

00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pairpot
00023
00024 USE constants_mod
00025 USE setuparray
00026 USE ppotarray
00027 USE neblistarray
00028 USE virialarray
00029 USE myprecision
00030
00031 IMPLICIT NONE
00032
00033 INTEGER :: I, NEWJ, J, K, PPSEL, BREAKLOOP
00034 INTEGER :: PBCI, PBCJ, PBCK
00035 REAL(LATTEPREC) :: JR1, JRCUT, R1, RCUT2, RCUT
00036 REAL(LATTEPREC) :: FORCE, DC(3), RIJ(3)
00037 REAL(LATTEPREC) :: MYR, MYR2, MYR3, MYR4, MAGR2, MAGR
00038 REAL(LATTEPREC) :: UNIVPHI, JOINPHI, VDWPHI, CUTPHI, TMP
00039 REAL(LATTEPREC) :: VIRUNIV(6), VIRJOIN(6), VIRVDW(6), VIRCUT(6)
00040 REAL(LATTEPREC) :: FUNIV(3), FJOIN(3), FVDW(3), FCUT(3)
00041 REAL(LATTEPREC) :: PHI, DPHI(3), EXPTMP, R6, FTMP(3)
00042 REAL(LATTEPREC) :: POLYNOM, DPOLYNOM
00043 IF (existererror) RETURN
00044
00045
00046 univphi = zero
00047 cutphi = zero
00048
00049 viruniv = zero
00050 vircut = zero
00051
00052 IF (ppoton .EQ. 1) THEN
00053
00054 DO i = 1, nats
00055
00056 funiv = zero
00057 fcut = zero
00058
00059 ! Loop over all neighbors of I
00060
00061 DO newj = 1, totnebpp(i)
00062
00063 j = nebpp(1, newj, i)
00064
00065 pbc1 = nebpp(2, newj, i)
00066 pbcj = nebpp(3, newj, i)
00067 pbck = nebpp(4, newj, i)
00068
00069 DO k = 1, nopps
00070
00071 IF ((atele(i) .EQ. ppele1(k) .AND. atele(j) .EQ.
00072 ppele2(k)) &
00073 .OR. (atele(j) .EQ. ppele1(k) .AND. &
00074 atele(i) .EQ. ppele2(k))) THEN
00075
00076 ppsel = k
00077
00078 r1 = potcoef(9, ppsel)
00079 rcut = potcoef(10, ppsel)
00080 rcut2 = rcut*rcut
00081
00082 ENDIF
00083
00084 ENDDO
00085
00086 rij(1) = cr(1, j) + REAL(pbc1)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00087 REAL(pbck)*BOX(3,1) - CR(1,i)

```

```

00088      rij(2) = cr(2,j) + REAL(pbcj)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00089      REAL(pbck)*BOX(3,2) - CR(2,i)
00090
00091      rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00092      REAL(pbck)*BOX(3,3) - CR(3,i)
00093
00094      magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00095
00096      IF (magr2 .LE. rcut2) THEN
00097
00098          magr = sqrt(magr2)
00099
00100          ! Direction cosines
00101
00102          dc = rij/magr
00103
00104          IF (magr .LT. r1) THEN
00105
00106              magr = magr - potcoef(6,ppsel)
00107
00108              ! CALL DUNIVSCALE(MAGR, POTCOEF(:,PPSEL), DC, PHI, DPHI)
00109
00110              polynom = magr*(potcoef(2,ppsel) + magr*(potcoef(3,ppsel) + &
00111              magr*(potcoef(4,ppsel) + magr*(potcoef(5,ppsel))))
00112
00113              phi = potcoef(1,ppsel)*exp(polynom)
00114
00115              dpolynom = potcoef(2,ppsel) + magr*(two*potcoef(3,ppsel) + &
00116              magr*(three*potcoef(4,ppsel) + &
00117              four*potcoef(5,ppsel)*magr))
00118
00119              dphi = -dc*phi*dpolynom
00120
00121              ! Hack!
00122              ! EXPTMP = POTCOEF(6,PPSEL)*&
00123              ! EXP( POTCOEF(7,PPSEL) * (MAGR - POTCOEF(8,PPSEL)) )
00124              ! R6 = MAGR2*MAGR2*MAGR2
00125
00126              exptmp = zero
00127
00128              ! UNIVPHI = UNIVPHI + PHI + EXPTMP - POTCOEF(8,PPSEL)/R6
00129
00130              univphi = univphi + phi + exptmp
00131
00132              ftmp = dc*potcoef(7,ppsel)*exptmp
00133              ! FTMP = DC*(POTCOEF(7,PPSEL)*EXPTMP + &
00134              ! SIX*POTCOEF(8,PPSEL)/(MAGR*R6))
00135
00136              funiv = funiv - dphi + ftmp
00137
00138              viruniv(1) = viruniv(1) - rij(1)*(dphi(1) - ftmp(1))
00139              viruniv(2) = viruniv(2) - rij(2)*(dphi(2) - ftmp(2))
00140              viruniv(3) = viruniv(3) - rij(3)*(dphi(3) - ftmp(3))
00141              viruniv(4) = viruniv(4) - rij(1)*(dphi(2) - ftmp(2))
00142              viruniv(5) = viruniv(5) - rij(2)*(dphi(3) - ftmp(3))
00143              viruniv(6) = viruniv(6) - rij(3)*(dphi(1) - ftmp(1))
00144
00145          ELSE
00146
00147              myr = magr - r1
00148
00149              cutphi = cutphi + potcoef(11,ppsel) + &
00150              myr*(potcoef(12,ppsel) + myr*(potcoef(13,ppsel) + &
00151              myr*(potcoef(14,ppsel) + myr*(potcoef(15,ppsel) + &
00152              myr*(potcoef(16,ppsel)))))
00153
00154              force = potcoef(12,ppsel) + myr*(two*potcoef(13,ppsel) + &
00155              myr*(three*potcoef(14,ppsel) + &
00156              myr*(four*potcoef(15,ppsel) + &
00157              myr*(five*potcoef(16,ppsel)))))
00158
00159              fcut = fcut + dc*force
00160
00161              vircut(1) = vircut(1) + rij(1)*dc(1)*force
00162              vircut(2) = vircut(2) + rij(2)*dc(2)*force
00163              vircut(3) = vircut(3) + rij(3)*dc(3)*force
00164              vircut(4) = vircut(4) + rij(1)*dc(2)*force
00165              vircut(5) = vircut(5) + rij(2)*dc(3)*force
00166              vircut(6) = vircut(6) + rij(3)*dc(1)*force
00167
00168          ENDIF
00169
00170      ENDIF
00171
00172      ENDDO
00173
00174      fpp(1,i) = funiv(1) + fcut(1)

```

```

00175         fpp(2,i) = funiv(2) + fcut(2)
00176         fpp(3,i) = funiv(3) + fcut(3)
00177
00178     ENDDO
00179
00180     erep = half*(univphi + cutphi)
00181
00182     ! PRINT*, "EREP ", EREP
00183     virpair = half*(viruniv + vircut)
00184
00185 ELSE
00186
00187     fpp = zero
00188     erep = zero
00189     virpair = zero
00190
00191 ENDIF
00192
00193 RETURN
00194
00195 END SUBROUTINE pairpot
00196

```

8.295 pairpot_noneb.f90 File Reference

Functions/Subroutines

- subroutine [pairpotnoneb](#)

8.295.1 Function/Subroutine Documentation

8.295.1.1 subroutine pairpotnoneb ()

Definition at line 23 of file [pairpot_noneb.f90](#).

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pairpotnoneb
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE neblistarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, NEWJ, J, K, PPSEL, BREAKLOOP
00034   INTEGER :: PBCI, PBCJ, PBCK
00035   REAL(LATTEPREC) :: JRI, JRCUT, R1, RCUT2, RCUT
00036   REAL(LATTEPREC) :: FORCE, DC(3), RIJ(3)
00037   REAL(LATTEPREC) :: MYR, MYR2, MYR3, MYR4, MAGR2, MAGR
00038   REAL(LATTEPREC) :: UNIVPHI, JOINPHI, VDWPHI, CUTPHI, TMP
00039   REAL(LATTEPREC) :: VIRUNIV(6), VIRJOIN(6), VIRVDW(6), VIRCUT(6)
00040   REAL(LATTEPREC) :: FUNIV(3), FJOIN(3), FVDW(3), FCUT(3)
00041   REAL(LATTEPREC) :: PHI, DPHI(3), EXPTMP, R6, FTMP(3)
00042   REAL(LATTEPREC) :: POLYNOM, DPOLYNOM
00043   IF (existerror) RETURN
00044
00045   !
00046   ! In this subroutine we add contributions in a strange way to ensure
00047   ! numerical accuracy when switching between single and double precision.
00048   ! If we don't do this, we get errors associated with adding very small
00049   ! numbers to very large ones, and energies can be off by 0.01% or more.
00050   !
00051
00052   !
00053   ! There are 4 different parts to the pair potential:
00054   !
00055   ! 1) Short range repulsion fitting to give bond lengths etc
00056   ! 2) The joining function from JOINR1 TO JOINRCUT
00057   ! 3) The vdW-type pair potential from JOINCUT to PPR1
00058   ! 4) The final cut off tail from PPR1 TO PPRCUT
00059   !
00060
00061   univphi = zero
00062   cutphi = zero
00063
00064   viruniv = zero
00065   vircut = zero
00066
00067   IF (ppoton.EQ. 1) THEN
00068
00069     DO i = 1, nats
00070
00071       funiv = zero
00072       fcut = zero
00073
00074       ! Loop over all neighbors of I
00075
00076       DO j = 1, nats
00077
00078         !           DO NEWJ = 1, TOTNEBPP(I)
00079
00080         !           J = NEBPP(1, NEWJ, I)
00081
00082         !           PBCI = NEBPP(2, NEWJ, I)
00083         !           PBCJ = NEBPP(3, NEWJ, I)
00084         !           PBCK = NEBPP(4, NEWJ, I)
00085
00086         DO k = 1, nopps
00087
00088           IF ((atele(i) .EQ. ppele1(k) .AND. atele(j) .EQ.
00089 ppele2(k)) &
00089           .OR. (atele(j) .EQ. ppele1(k) .AND. &
00090           atele(i) .EQ. ppele2(k))) THEN
00091
00092             ppsel = k

```

```

00093
00094       r1 = potcoef(9,ppsel)
00095       rcut = potcoef(10,ppsel)
00096       rcut2 = rcut*rcut
00097
00098       ENDIF
00099
00100       ENDDO
00101
00102       rij(1) = cr(1,j) - cr(1,i)
00103       rij(2) = cr(2,j) - cr(2,i)
00104       rij(3) = cr(3,j) - cr(3,i)
00105
00106       !           RIJ(1) = CR(1,J) + REAL(PBCI)*BOX(1,1) + REAL(PBCJ)*BOX(2,1) + &
00107       !           REAL(PBCK)*BOX(3,1) - CR(1,I)
00108
00109       !           RIJ(2) = CR(2,J) + REAL(PBCI)*BOX(1,2) + REAL(PBCJ)*BOX(2,2) + &
00110       !           REAL(PBCK)*BOX(3,2) - CR(2,I)
00111
00112       !           RIJ(3) = CR(3,J) + REAL(PBCI)*BOX(1,3) + REAL(PBCJ)*BOX(2,3) + &
00113       !           REAL(PBCK)*BOX(3,3) - CR(3,I)
00114
00115       magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00116
00117       IF (magr2 .LE. rcut2 .AND. j .NE. i) THEN
00118
00119         magr = sqrt(magr2)
00120
00121         ! Direction cosines
00122
00123         dc = rij/magr
00124
00125         IF (magr .LT. r1) THEN
00126
00127           !           CALL DUNIVSCALE(MAGR, POTCOEF(:,PPSEL), DC, PHI, DPHI)
00128
00129           polynom = magr*(potcoef(2,ppsel) + magr*(potcoef(3,ppsel) + &
00130             magr*(potcoef(4,ppsel) + magr*potcoef(5,ppsel))))
00131
00132           phi = potcoef(1,ppsel)*exp(polynom)
00133
00134           dpolynom = potcoef(2,ppsel) + magr*(two*potcoef(3,ppsel) + &
00135             magr*(three*potcoef(4,ppsel) + &
00136             four*potcoef(5,ppsel)*magr))
00137
00138           dphi = -dc*phi*dpolynom
00139
00140           ! Hack!
00141           exptmp = potcoef(6,ppsel)*&
00142             exp( potcoef(7,ppsel) * (magr - potcoef(8,ppsel)) )
00143           !           R6 = MAGR2*MAGR2*MAGR2
00144
00145           !           UNIVPHI = UNIVPHI + PHI + EXPTMP - POTCOEF(8,PPSEL)/R6
00146
00147           univphi = univphi + phi + exptmp
00148
00149           ftmp = dc*potcoef(7,ppsel)*exptmp
00150           !           FTMP = DC*(POTCOEF(7,PPSEL)*EXPTMP + &
00151           !           SIX*POTCOEF(8,PPSEL) / (MAGR*R6) )
00152
00153           funiv = funiv - dphi + ftmp
00154
00155           viruniv(1) = viruniv(1) - rij(1)*(dphi(1) - ftmp(1))
00156           viruniv(2) = viruniv(2) - rij(2)*(dphi(2) - ftmp(2))
00157           viruniv(3) = viruniv(3) - rij(3)*(dphi(3) - ftmp(3))
00158           viruniv(4) = viruniv(4) - rij(1)*(dphi(2) - ftmp(2))
00159           viruniv(5) = viruniv(5) - rij(2)*(dphi(3) - ftmp(3))
00160           viruniv(6) = viruniv(6) - rij(3)*(dphi(1) - ftmp(1))
00161
00162         ELSE
00163
00164           myr = magr - r1
00165
00166           cutphi = cutphi + potcoef(11,ppsel) + &
00167             myr*(potcoef(12,ppsel) + myr*(potcoef(13,ppsel) + &
00168             myr*(potcoef(14,ppsel) + myr*(potcoef(15,ppsel) + &
00169             myr*potcoef(16,ppsel)))))
00170
00171           force = potcoef(12,ppsel) + myr*(two*potcoef(13,ppsel) + &
00172             myr*(three*potcoef(14,ppsel) + &
00173             myr*(four*potcoef(15,ppsel) + &
00174             myr*five*potcoef(16,ppsel))))
00175
00176           fcut = fcut + dc*force
00177
00178           vircut(1) = vircut(1) + rij(1)*dc(1)*force
00179           vircut(2) = vircut(2) + rij(2)*dc(2)*force

```

```

00180          vircut(3) = vircut(3) + rij(3)*dc(3)*force
00181          vircut(4) = vircut(4) + rij(1)*dc(2)*force
00182          vircut(5) = vircut(5) + rij(2)*dc(3)*force
00183          vircut(6) = vircut(6) + rij(3)*dc(1)*force
00184
00185          ENDIF
00186
00187          ENDIF
00188
00189          ENDDO
00190
00191          fpp(1,i) = funiv(1) + fcut(1)
00192          fpp(2,i) = funiv(2) + fcut(2)
00193          fpp(3,i) = funiv(3) + fcut(3)
00194
00195          ENDDO
00196
00197          erep = half*(univphi + cutphi)
00198
00199          ! PRINT*, "EREP ", EREP
00200          virpair = half*(viruniv + vircut)
00201
00202          ELSE
00203
00204          fpp = zero
00205          erep = zero
00206          virpair = zero
00207
00208          ENDIF
00209
00210          RETURN
00211
00212 END SUBROUTINE pairpotnoneb
00213

```

8.297 pairpotspline.f90 File Reference

Functions/Subroutines

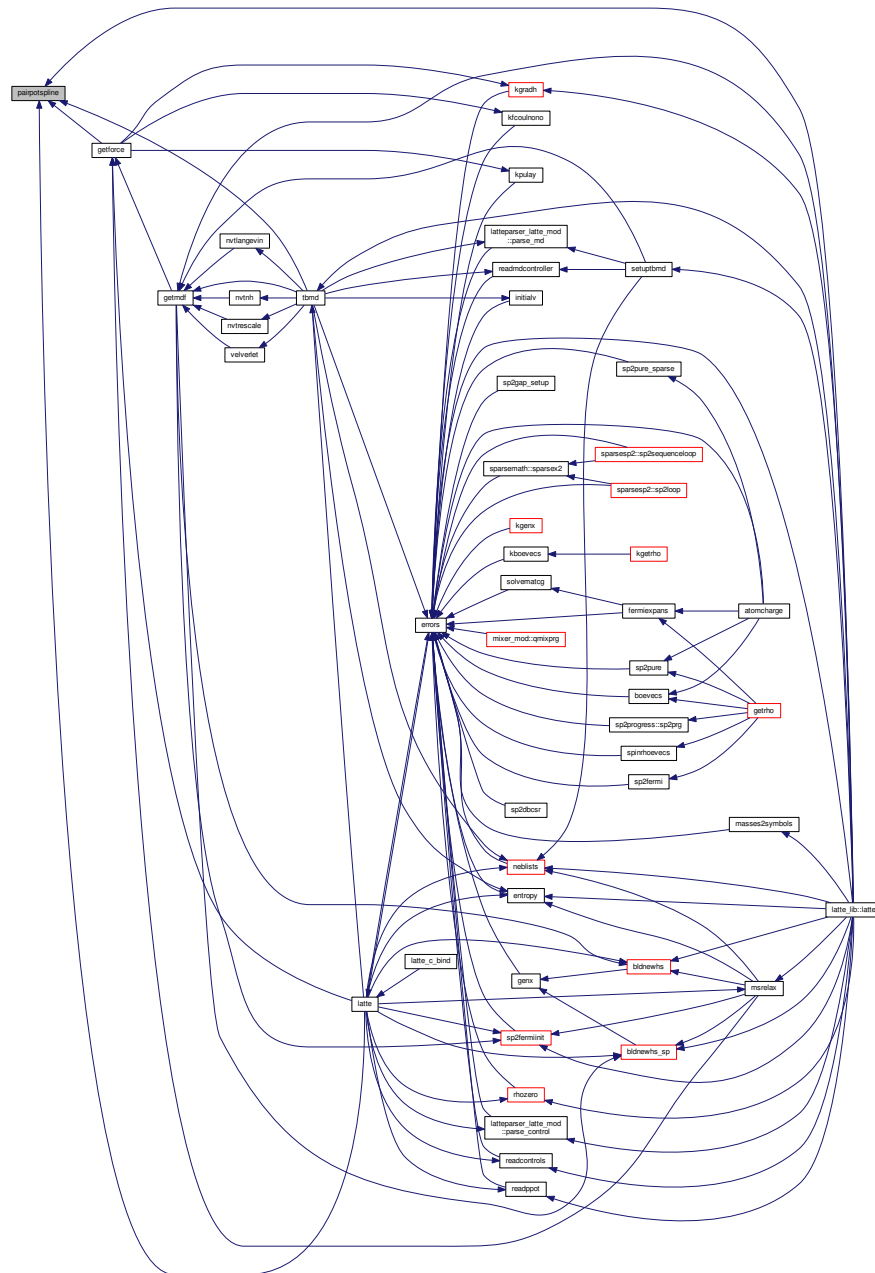
- subroutine [pairpotspline](#)
- real(latteprec) function [ppheavi](#) (RK, R)

8.297.1 Function/Subroutine Documentation

8.297.1.1 subroutine pairpotspline ()

Definition at line 23 of file [pairpotspline.f90](#).

Here is the caller graph for this function:



8.297.1.2 `real(latteprec) function ppheavi (real(latteprec) RK, real(latteprec) R)`

Definition at line 127 of file `pairpotspline.f90`.

8.298 `pairpotspline.f90`

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was
```

```

00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pairpotspline
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE neblastarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, NEWJ, J, K, PPSEL, BREAKLOOP
00034   INTEGER :: PBCI, PBCJ, PBCK, COUNT
00035   REAL(LATTEPREC) :: RCUT2, RCUT, MAGR, MAGR2
00036   REAL(LATTEPREC) :: FORCE(3), DC(3), RIJ(3)
00037   REAL(LATTEPREC) :: GRAD, PPTMP
00038   REAL(LATTEPREC), EXTERNAL :: PPHEAVI
00039
00040   erep = zero
00041   fpp = zero
00042   virpair = zero
00043
00044   DO i = 1, nats
00045
00046     ! Loop over all neighbors of I
00047
00048     DO newj = 1, totnebpp(i)
00049
00050       j = nebpp(1, newj, i)
00051
00052       pbc_i = nebpp(2, newj, i)
00053       pbc_j = nebpp(3, newj, i)
00054       pbck = nebpp(4, newj, i)
00055
00056       DO k = 1, nopps
00057
00058         IF ((atele(i) .EQ. ppele1(k) .AND. atele(j) .EQ.
00059 ppele2(k)) &
00060           .OR. (atele(j) .EQ. ppele1(k) .AND. &
00061             atele(i) .EQ. ppele2(k))) THEN
00062
00063           ppsel = k
00064           rcut = pprk(1, ppsel)
00065           rcut2 = rcut*rcut
00066
00067         ENDIF
00068
00069       ENDDO
00070
00071       rij(1) = cr(1,j) + REAL(pbc_i)*BOX(1,1) + REAL(pbc_j)*BOX(2,1) + &
00072         REAL(pbck)*BOX(3,1) - CR(1,i)
00073
00074       rij(2) = cr(2,j) + REAL(pbc_i)*BOX(1,2) + REAL(pbc_j)*BOX(2,2) + &
00075         REAL(pbck)*BOX(3,2) - CR(2,i)
00076
00077       rij(3) = cr(3,j) + REAL(pbc_i)*BOX(1,3) + REAL(pbc_j)*BOX(2,3) + &
00078         REAL(pbck)*BOX(3,3) - CR(3,i)
00079
00080       magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00081
00082       IF (magr2 .LE. rcut2) THEN
00083
00084         magr = sqrt(magr2)
00085
00086         dc = rij/magr
00087
00088         grad = zero

```

```

00089         DO k = 1, ppnk(ppsel)
00090
00091             pptmp = ppak(k,ppsel)*ppheavi(pprk(k,ppsel), magr)* &
00092                 (pprk(k,ppsel) - magr)*(pprk(k,ppsel) - magr)
00093
00094             erep = erep + pptmp*(pprk(k,ppsel) - magr)
00095
00096             grad = grad - three*pptmp
00097
00098         ENDDO
00099
00100         force = grad*dc
00101
00102         fpp(1,i) = fpp(1,i) + force(1)
00103         fpp(2,i) = fpp(2,i) + force(2)
00104         fpp(3,i) = fpp(3,i) + force(3)
00105
00106         virpair(1) = virpair(1) + rij(1)*force(1)
00107         virpair(2) = virpair(2) + rij(2)*force(2)
00108         virpair(3) = virpair(3) + rij(3)*force(3)
00109         virpair(4) = virpair(4) + rij(1)*force(2)
00110         virpair(5) = virpair(5) + rij(2)*force(3)
00111         virpair(6) = virpair(6) + rij(3)*force(1)
00112
00113     ENDIF
00114
00115     ENDDO
00116
00117     ENDDO
00118
00119     erep = erep/two
00120     virpair = virpair/two
00121
00122     RETURN
00123
00124 END SUBROUTINE pairpotspline
00125
00126 FUNCTION ppheavi (RK,R)
00127
00128     USE constants_mod
00129
00130     IMPLICIT NONE
00131
00132     REAL(LATTEPREC) :: RK, R, PPHEAVI
00133
00134     IF ((rk - r) .LE. zero) THEN
00135         ppheavi = zero
00136     ELSE
00137         ppheavi = one
00138     ENDIF
00139
00140     RETURN
00141
00142 END FUNCTION ppheavi

```

8.299 pairpottab.f90 File Reference

Functions/Subroutines

- subroutine [pairpottab](#)

8.299.1 Function/Subroutine Documentation

8.299.1.1 subroutine pairpottab ()

Definition at line 23 of file [pairpottab.f90](#).

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 !

```

```

00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General    !
00019 ! Public License for more details.                                             !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pairpottab
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE neblistarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, NEWJ, J, K, PPSEL, BREAKLOOP
00034   INTEGER :: PBCI, PBCJ, PBCK, COUNT
00035   INTEGER :: KLO, KHI
00036   REAL(LATTEPREC) :: RCUT2, RCUT, MAGR, MAGR2
00037   REAL(LATTEPREC) :: FORCE(3), DC(3), RIJ(3)
00038   REAL(LATTEPREC) :: GRAD, TMPE
00039   REAL(LATTEPREC) :: A, B, DX
00040   IF (existerror) RETURN
00041
00042   erep = zero
00043   fpp = zero
00044   virpair = zero
00045
00046   DO i = 1, nats
00047
00048       !     FORCE = ZERO
00049
00050       ! Loop over all neighbors of I
00051
00052       DO newj = 1, totnebpp(i)
00053
00054           j = nebpp(1, newj, i)
00055
00056           pbcj = nebpp(2, newj, i)
00057           pbcj = nebpp(3, newj, i)
00058           pbck = nebpp(4, newj, i)
00059
00060           DO k = 1, nopps
00061
00062               IF ((atele(i) .EQ. ppele1(k) .AND. atele(j) .EQ.
00063 ppele2(k)) &
00064                   .OR. (atele(j) .EQ. ppele1(k) .AND. &
00065 atele(i) .EQ. ppele2(k))) THEN
00066
00067                   ppsel = k
00068                   rcut = ppr(pptablenth(ppsel), ppsel)
00069                   rcut2 = rcut*rcut
00070
00071               ENDIF
00072
00073           ENDDO
00074
00075           rij(1) = cr(1,j) + REAL(pbcj)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00076 REAL(pbck)*BOX(3,1) - CR(1,i)
00077
00078           rij(2) = cr(2,j) + REAL(pbcj)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00079 REAL(pbck)*BOX(3,2) - CR(2,i)
00080
00081           rij(3) = cr(3,j) + REAL(pbcj)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00082 REAL(pbck)*BOX(3,3) - CR(3,i)
00083
00084           magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00085
00086           IF (magr2 .LE. rcut2) THEN
00087
00088               magr = sqrt(magr2)
00089
00090               dc = rij/magr
00091
00092               klo = 1
00093               khi = pptablenth(ppsel)
00094
00095               DO WHILE (khi - klo .GT. 1)
00096
00097                   k = (khi + klo)/2
00098
00099                   IF (ppr(k,ppsel) .GT. magr) THEN

```

```

00099             khi = k
00100             ELSE
00101                 klo = k
00102             ENDIF
00103
00104         ENDDO
00105
00106         dx = ppr(khi,ppsel) - ppr(klo,ppsel)
00107
00108         a = (ppr(khi, ppsel) - magr)/dx
00109         b = (magr - ppr(klo, ppsel))/dx
00110
00111         tmpe = a*ppval(klo,ppsel) + b*ppval(khi, ppsel) + &
00112             ((a*a*a - a)*ppspl(klo,ppsel) + &
00113             (b*b*b - b)*ppspl(khi,ppsel))*(dx*dx/six)
00114
00115         erep = erep + tmpe
00116
00117         grad = (ppval(khi,ppsel) - ppval(klo,ppsel))/dx + &
00118             ((one - three*a*a)*ppspl(klo,ppsel) + &
00119             (three*b*b - one)*ppspl(khi,ppsel))*(dx/six)
00120
00121         !             GRAD = ZERO
00122
00123         force = grad*dc
00124
00125         fpp(1,i) = fpp(1,i) + force(1)
00126         fpp(2,i) = fpp(2,i) + force(2)
00127         fpp(3,i) = fpp(3,i) + force(3)
00128
00129         virpair(1) = virpair(1) + rij(1)*force(1)
00130         virpair(2) = virpair(2) + rij(2)*force(2)
00131         virpair(3) = virpair(3) + rij(3)*force(3)
00132         virpair(4) = virpair(4) + rij(1)*force(2)
00133         virpair(5) = virpair(5) + rij(2)*force(3)
00134         virpair(6) = virpair(6) + rij(3)*force(1)
00135
00136     ENDIF
00137
00138     ENDDO
00139
00140 ENDDO
00141
00142 erep = erep/two
00143
00144 virpair = virpair/two
00145
00146 RETURN
00147
00148 END SUBROUTINE pairpottab
00149

```

8.301 panic.f90 File Reference

Functions/Subroutines

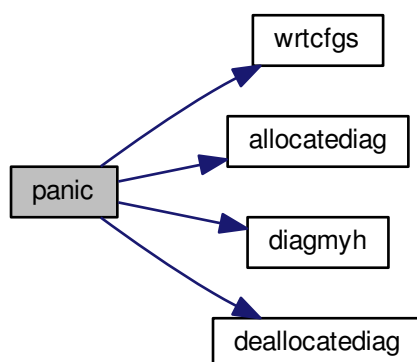
- subroutine [panic](#)

8.301.1 Function/Subroutine Documentation

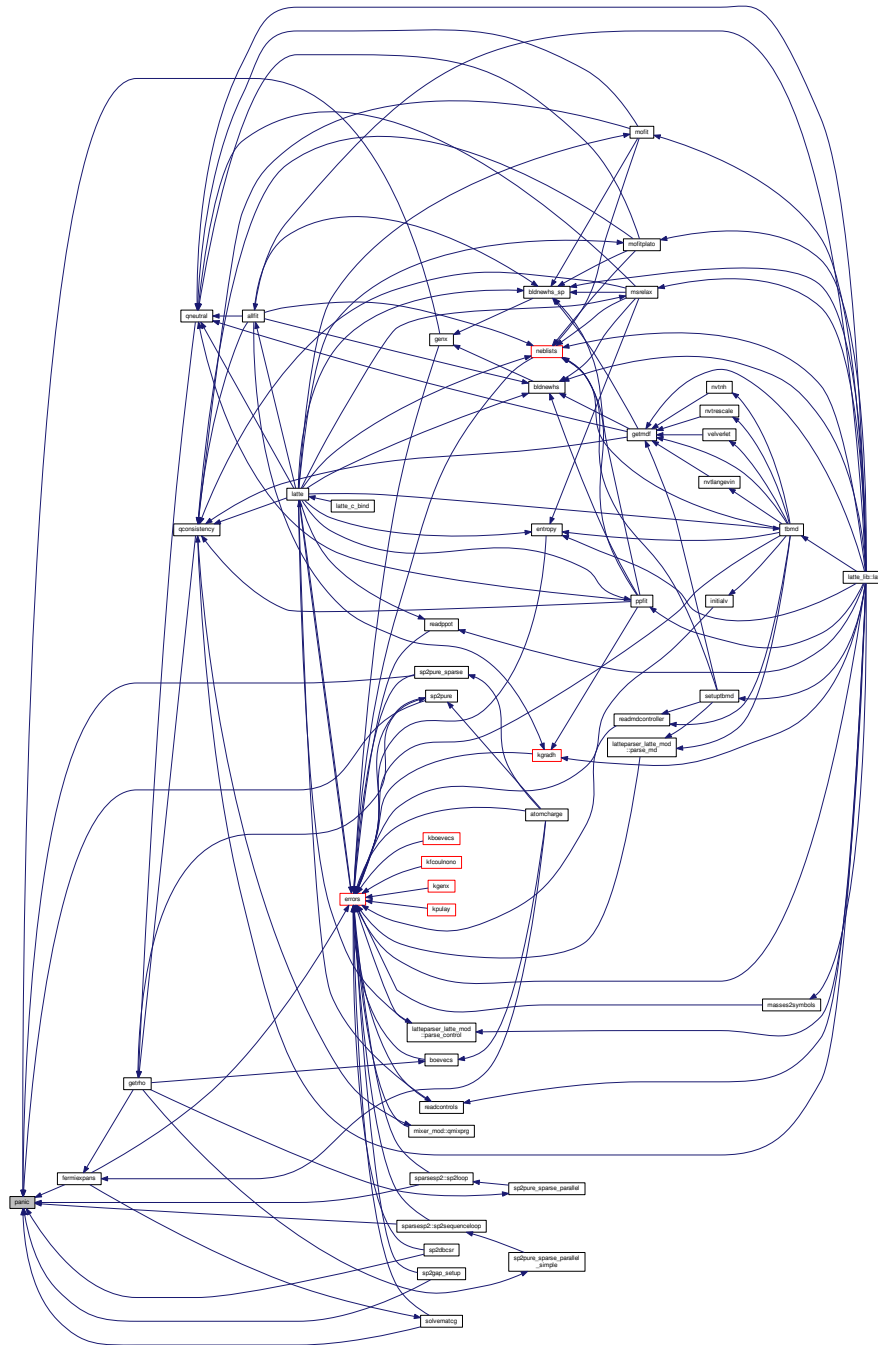
8.301.1.1 subroutine panic ()

Definition at line 23 of file [panic.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.302 panic.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```



```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE panic
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE ppotarray
00028   USE purearray
00029   USE sparsearray
00030   USE coulombarray
00031   USE spinarray
00032   USE myprecision
00033   USE diagarray
00034
00035   IMPLICIT NONE
00036
00037   INTEGER :: I
00038   IF (existerror) RETURN
00039
00040
00041   IF (mdon .EQ. 1) THEN
00042
00043     !
00044     ! Write a .cfg file with our problematic configuration
00045     !
00046
00047     CALL wrtcfgs(-1)
00048
00049   ELSE
00050
00051     CALL wrtcfgs(-999)
00052
00053   ENDIF
00054
00055   OPEN(unit=24, status="UNKNOWN", file="myPANICfile.dat")
00056
00057   WRITE(6, '# PANIC: SOMETHING BAD HAS HAPPENED! Check myPANICfile.dat')
00058   WRITE(24, 'PANIC: SOMETHING BAD HAS HAPPENED! SOME CLUES TO DIAGNOSE THE PROBLEM FOLLOW')
00059
00060   IF (mdon .EQ. 1) THEN
00061     WRITE(24, 'Molecular dynamics calculation')
00062   ENDIF
00063
00064
00065   IF (electro .EQ. 1) THEN
00066     WRITE(24, 'Self-consistent charge transfer: on')
00067     WRITE(24, 'SCF tolerance = ', G8.3) elec_qtol
00068     IF (elec_meth .EQ. 0) THEN
00069       WRITE(24, 'Using Ewald summation')
00070       WRITE(24, 'Coulomb accuracy = ', G8.3) coulacc
00071       WRITE(24, 'Real space cut-off for Ewald sum = ', F6.2) coulcut
00072     ELSEIF (elec_meth .EQ. 1) THEN
00073       WRITE(24, 'Real space electrostatics')
00074       WRITE(24, 'Cut-off tail applied between ', F6.2, F6.2) &
00075         coulrl, coulcut
00076     ENDIF
00077   ENDIF
00078
00079   IF (spinon .EQ. 1) THEN
00080     WRITE(24, 'Spin-polarized calculation')
00081   ENDIF
00082
00083   IF (sparseon .EQ. 1) THEN
00084     WRITE(24, 'Sparse matrix calculation')
00085     WRITE(24, 'Numerical threshold = ', G8.3) numthresh
00086   ENDIF
00087
00088
00089
00090   IF (control .EQ. 1) THEN
00091     WRITE(24, 'Diagonalization')
00092   ELSEIF (control .EQ. 2) THEN
00093     WRITE(24, 'SP2 purification at zero temperature')
00094   ELSEIF (control .EQ. 3) THEN
00095     WRITE(24, 'Recursive expansion of the Fermi operator')
00096   ELSEIF (control .EQ. 4) THEN

```

```

00097     WRITE(24, '("SP2 purification at finite temperature")')
00098 ELSEIF (control .EQ. 5) THEN
00099     WRITE(24, '("SP2/Fermi method at finite temperature")')
00100 ENDIF
00101
00102 IF (control .NE. 2) THEN
00103     WRITE(24, '("KBT (in eV) = ", F16.8)') kbt
00104 ENDIF
00105
00106 IF (control .EQ. 5) THEN
00107     WRITE(24, '("Gershgorin: MAXEVAL, MINEVAL = ", 2F16.8)') maxeval,
mineval
00108 ENDIF
00109
00110 IF (latteprec .EQ. kind(0.0d0)) THEN
00111     WRITE(24, '("Double precision arithmetic")')
00112 ELSEIF (latteprec .EQ. kind(0.0)) THEN
00113     WRITE(24, '("Single precision arithmetic")')
00114 ENDIF
00115
00116 IF (entropykind .EQ. 0) THEN
00117     WRITE(24, '("Entropy set = 0")')
00118 ELSEIF (entropykind .EQ. 1) THEN
00119     WRITE(24, '("Using exact ln form for entropy")')
00120 ELSEIF (entropykind .EQ. 2) THEN
00121     WRITE(24, '("Using the close-to-exact expansion of exact entropy (2)")')
00122 ELSEIF (entropykind .EQ. 3) THEN
00123     WRITE(24, '("Using 4th order approximation for entropy")')
00124 ELSEIF (entropykind .EQ. 4) THEN
00125     WRITE(24, '("Using 8th order approximation for entropy")')
00126 ENDIF
00127
00128 WRITE(24, '("Tr[ rho*H ] = ", F16.8)') trrhoh
00129 WRITE(24, '("Pairwise energy = ", F16.8)') erep
00130
00131 IF (electro .EQ. 1) THEN
00132     WRITE(24, '("Coulombic + onsite E = ", F16.8)') ecoul
00133 ENDIF
00134
00135 IF (control .NE. 2) THEN
00136     WRITE(24, '("Electron entropy TS = ", F16.8)') ente
00137 ENDIF
00138
00139 IF (control .EQ. 1 .OR. control .EQ. 3 .OR. control .EQ. 5) THEN
00140     WRITE(24, '("Chemical potential = ", F16.8)') chempot
00141 ENDIF
00142
00143 IF (spinon .EQ. 1) THEN
00144     WRITE(24, '("Self-consistent spin energy = ", F16.8)') espin
00145     WRITE(24, '("Free atom spin energy = ", F16.8)') espin_zero
00146 ENDIF
00147
00148 IF (spinon .EQ. 0) THEN
00149     WRITE(24, '("Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul
00150     WRITE(24, '(" ")')
00151     WRITE(24, '("FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente
00152     WRITE(24, '(" ")')
00153 ELSEIF (spinon .EQ. 1) THEN
00154     WRITE(24, '("Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul + &
    espin - espin_zero
00155     WRITE(24, '(" ")')
00156     WRITE(24, '("FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente + &
    espin - espin_zero
00157     WRITE(24, '(" ")')
00158     WRITE(24, '(" ")')
00159 ENDIF
00160
00161 IF (electro .EQ. 1) THEN
00162     WRITE(24, '("Partial charges")')
00163     WRITE(24, '(" Atom Charge")')
00164     DO i = 1, nats
00165         WRITE(24, 50) i, deltaq(i)
00166     ENDDO
00167
00168 50 FORMAT(i6, 9x, f11.8)
00169
00170 ENDIF
00171
00172 IF (spinon .EQ. 1) THEN
00173     WRITE(24, '(" ")')
00174     WRITE(24, '("Orbital spin densitites")')
00175     WRITE(24, '("Orbital index Spin density")')

```

```

00179      DO i = 1, deltadim
00180          WRITE(24,51) i, deltaspin(i)
00181      ENDDO
00182
00183 51  FORMAT(i6, 16x, f14.8)
00184
00185      IF (control .NE. 1) THEN
00186          CALL allocateddiag
00187      ENDIF
00188
00189      CALL diagmyh
00190
00191      WRITE(24,'(" ")')
00192      WRITE(24,'("Eigenvalues")')
00193      WRITE(24,'("          Up           :           Down")')
00194      DO i = 1, hdim
00195          WRITE(24, 52) i, upevals(i), downevals(i)
00196      ENDDO
00197
00198 52  FORMAT(i6, 2x, f14.8, 4x, f14.8)
00199
00200      CALL deallocateddiag
00201
00202  ENDIF
00203
00204
00205      WRITE(24,'(" ")')
00206      WRITE(24,'("Coordinates (in .xyz format!)")')
00207      DO i = 1, nats
00208          WRITE(24,54) atele(i), cr(1,i), cr(2,i), cr(3,i)
00209      ENDDO
00210
00211 53  FORMAT(i6, 1x, 3f18.9, 1x, a2)
00212 54  FORMAT(a2, 1x, 3g18.9)
00213
00214      IF (mdon .EQ. 1) THEN
00215
00216          WRITE(24,'(" ")')
00217          WRITE(24,'("Velocities")')
00218          DO i = 1, nats
00219              WRITE(24,53) i, v(1,i), v(2,i), v(3,i), atele(i)
00220          ENDDO
00221
00222      ENDIF
00223
00224      WRITE(24,'(" ")')
00225      WRITE(24,'("Forces")')
00226      DO i = 1, nats
00227          WRITE(24,53) i, ftot(1,i), ftot(2,i), ftot(3,i), atele(i)
00228      ENDDO
00229
00230      WRITE(24,'(" ")')
00231      WRITE(24,'("Band structure force")')
00232      DO i = 1, nats
00233          WRITE(24,53) i, two*f(1,i), two*f(2,i), two*f(3,i), atele(i)
00234      ENDDO
00235
00236      WRITE(24,'(" ")')
00237      WRITE(24,'("Pair potential force")')
00238      DO i = 1, nats
00239          WRITE(24,53) i, fpp(1,i), fpp(2,i), fpp(3,i), atele(i)
00240      ENDDO
00241
00242      IF (electro .EQ. 1) THEN
00243
00244          WRITE(24,'(" ")')
00245          WRITE(24,'("Coulomb force")')
00246          DO i = 1, nats
00247              WRITE(24,53) i, fcoul(1,i), fcoul(2,i), fcoul(3,i), atele(i)
00248          ENDDO
00249
00250      ENDIF
00251
00252      CLOSE(24)
00253
00254      RETURN
00255
00256 END SUBROUTINE panic

```

8.303 parafileopen.f90 File Reference

Functions/Subroutines

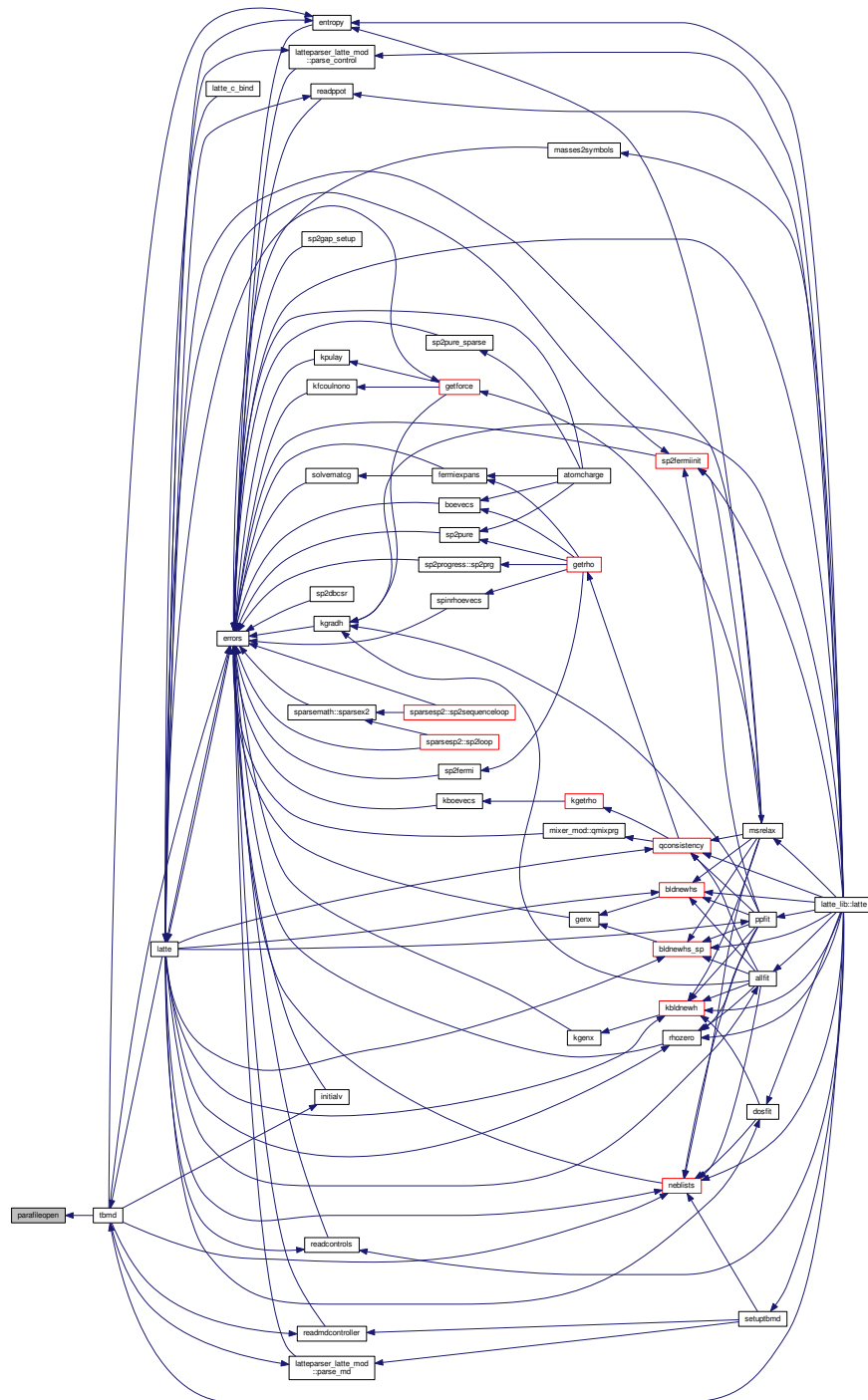
- subroutine [parafileopen](#)

8.303.1 Function/Subroutine Documentation

8.303.1.1 subroutine parafileopen ()

Definition at line 23 of file [parafileopen.f90](#).

Here is the caller graph for this function:



8.304 parafileopen.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE parafileopen
00023
00024 #ifdef MPI_ON
00025     USE mpi
00026 #endif
00027
00028     USE constants_mod, ONLY: existerror
00029
00030     IMPLICIT NONE
00031
00032     INTEGER :: MYID, IERR, MYUNIT
00033     CHARACTER(LEN=50) :: FLNM
00034
00035     IF (existerror) RETURN
00036
00037 #ifdef MPI_ON
00038     CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00039 #endif
00040
00041     myunit= 100 + myid
00042
00043     IF (myid .LT. 10) THEN
00044         WRITE(flnm,'(I1,"/pararep.dat")') myid
00045     ELSEIF (myid .GE. 10 .AND. myid .LT. 100) THEN
00046         WRITE(flnm,'(I2,"/pararep.dat")') myid
00047     ELSEIF (myid .GE. 100 .AND. myid .LT. 1000) THEN
00048         WRITE(flnm,'(I3,"/pararep.dat")') myid
00049     ENDIF
00050
00051     OPEN(unit=myunit, status="NEW", file=flnm)
00052
00053     RETURN
00054
00055 END SUBROUTINE parafileopen
00056

```

8.305 parawrite.f90 File Reference

Functions/Subroutines

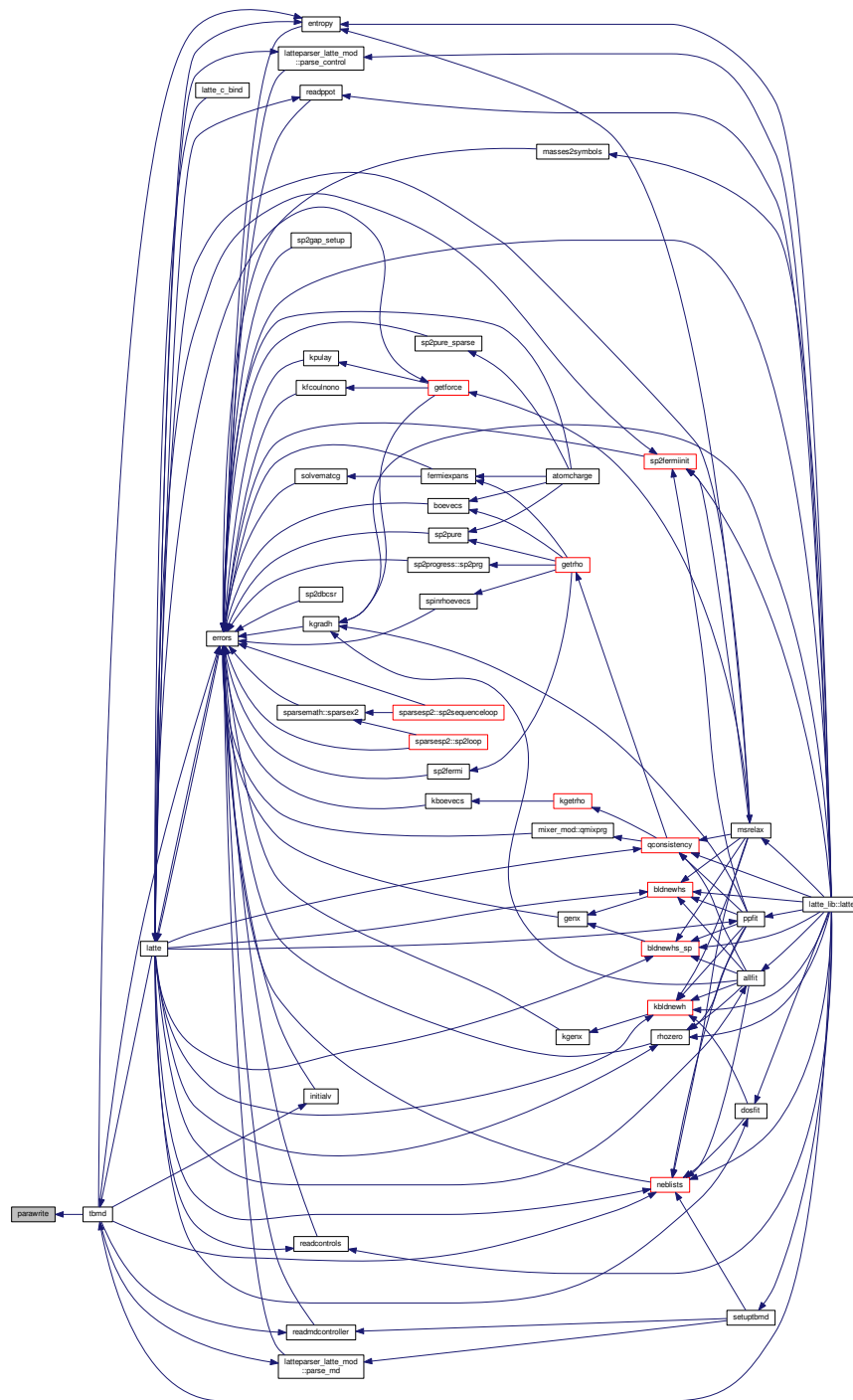
- subroutine [parawrite](#) (THETIME)

8.305.1 Function/Subroutine Documentation

8.305.1.1 subroutine parawrite (real(latteprec) THETIME)

Definition at line 23 of file [parawrite.f90](#).

Here is the caller graph for this function:



8.306 parawrite.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE parawrite (THETIME)
00023
00024   USE constants_mod
00025   USE virialarray
00026   USE mdarray
00027   USE myprecision
00028 #ifdef MPI_ON
00029   USE mpi
00030 #endif
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: MYID, IERR, MYUNIT
00035   REAL (LATTEPREC) :: THETIME
00036   IF (existerror) RETURN
00037
00038 #ifdef MPI_ON
00039   CALL mpi_comm_rank ( mpi_comm_world, myid, ierr )
00040 #endif
00041
00042   myunit= 100 + myid
00043
00044   WRITE(myunit, 10) thetime, tote, temperature, pressure,
00045   egap, chempot
00046
00047   FLUSH(myunit)
00048   10 FORMAT(6g18.9)
00049
00049   RETURN
00050
00051 END SUBROUTINE parawrite
00052

```

8.307 pbc.f90 File Reference

Functions/Subroutines

- subroutine [pbc](#)

8.307.1 Function/Subroutine Documentation

8.307.1.1 subroutine [pbc](#) ()

Definition at line 23 of file [pbc.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pbc
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I, IPIV(3), INFO
00031   REAL(LATTEPREC) :: WORK(3), BOXINV(3,3), S(3)
00032   IF (existerror) RETURN
00033
00034   ! reduced coordinates s = box^-1 X real coordinates
00035
00036   boxinv = box
00037
00038   #ifdef DOUBLEPREC
00039
00040   CALL dgetrf(3, 3, boxinv, 3, ipiv, info)
00041
00042   CALL dgetri(3, boxinv, 3, ipiv, work, 3, info)
00043
00044   #elif defined(SINGLEPREC)
00045
00046   CALL sgetrf(3, 3, boxinv, 3, ipiv, info)
00047
00048   CALL sgetri(3, boxinv, 3, ipiv, work, 3, info)
00049
00050   #endif
00051
00052   DO i = 1, nats
00053
00054   #ifdef DOUBLEPREC
00055
00056       CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,i), 1, zero, s, 1)
00057
00058   #elif defined(SINGLEPREC)
00059
00060       CALL sgemv('T', 3, 3, one, boxinv, 3, cr(1,i), 1, zero, s, 1)
00061
00062   #endif
00063
00064       ! If we're outside the box, add or subtract the corresponding vector accordingly
00065
00066       IF (s(1) .GT. one) cr(:,i) = cr(:,i) - box(1,:)
00067       IF (s(1) .LT. zero) cr(:,i) = cr(:,i) + box(1,:)
00068
00069       IF (s(2) .GT. one) cr(:,i) = cr(:,i) - box(2,:)
00070       IF (s(2) .LT. zero) cr(:,i) = cr(:,i) + box(2,:)
00071
00072       IF (s(3) .GT. one) cr(:,i) = cr(:,i) - box(3,:)
00073       IF (s(3) .LT. zero) cr(:,i) = cr(:,i) + box(3,:)
00074
00075   ENDDO
00076
00077   RETURN
00078
00080 END SUBROUTINE pbc

```

8.309 plot_ppot.f90 File Reference

Functions/Subroutines

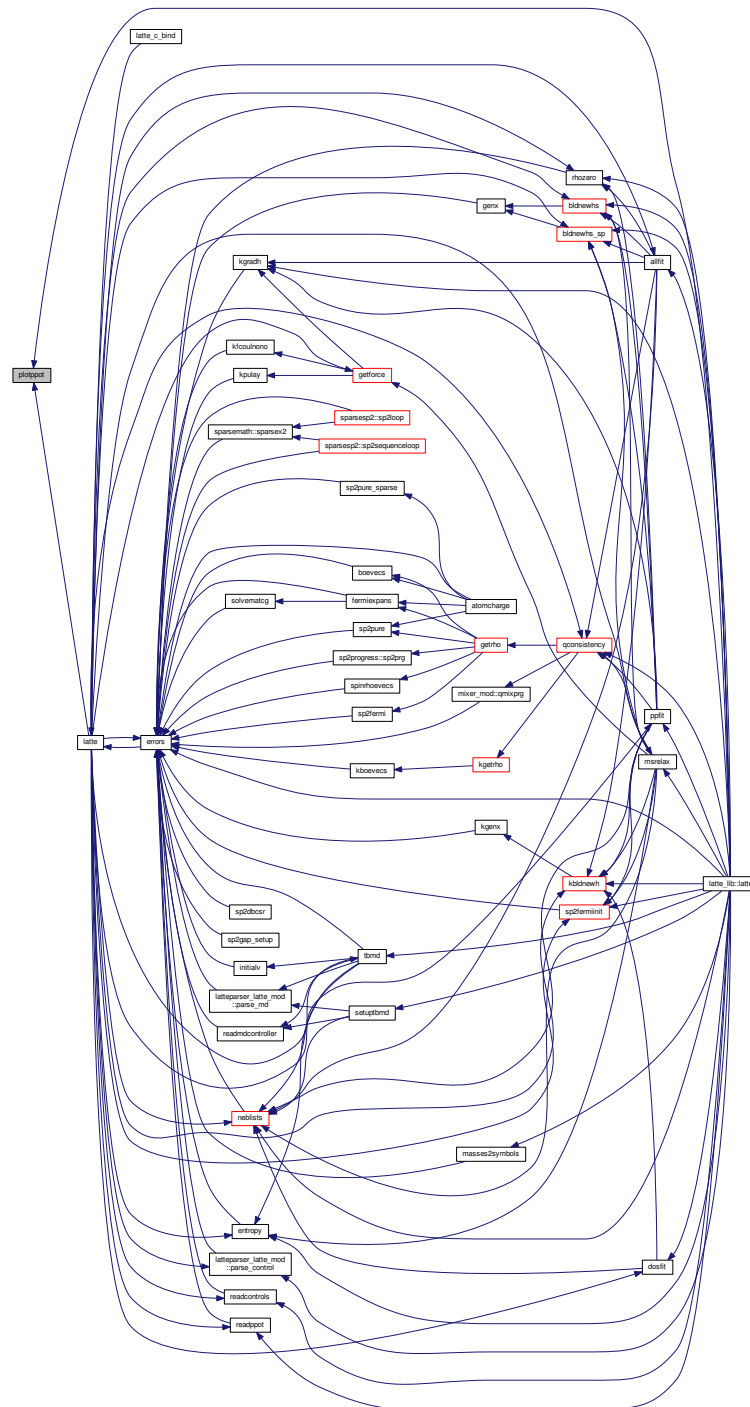
- subroutine [plotppot](#)

8.309.1 Function/Subroutine Documentation

8.309.1.1 subroutine plotppot ()

Definition at line 23 of file [plot_ppot.f90](#).

Here is the caller graph for this function:



8.310 plot_ppot.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE plotppot
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, K, II, PPSEL
00032   REAL(LATTEPREC) :: JR1, JRCUT, R1, RCUT2, RCUT
00033   REAL(LATTEPREC) :: PHI, FORCE
00034   REAL(LATTEPREC) :: MYR, MYR2, MYR3, MYR4, MAGR2, MAGR, R6
00035   REAL(LATTEPREC), ALLOCATABLE :: MYPAIRPOTS(:,,:), MYF(:,:)
00036   REAL(LATTEPREC) :: DC(3), DPHI(3), EXPTMP
00037   REAL(LATTEPREC) :: POLYNOM, DPOLYNOM
00038   CHARACTER(LEN=50) :: FLNM
00039   IF (existerror) RETURN
00040
00041   ! There are 4 different parts to the pair potential:
00042   !
00043   ! 1) Short range repulsion fitting to give bond lengths etc
00044   ! 2) The joining function from JOINR1 TO JOINRCUT
00045   ! 3) The vdW-type pair potential from JOINCUT to PPR1
00046   ! 4) The final cut off tail from PPR1 TO PPRCUT
00047   !
00048
00049   OPEN(unit=56, status="UNKNOWN", file = "vdW_scaling.dat")
00050   OPEN(unit=57, status="UNKNOWN", file = "vdW_F_scaling.dat")
00051
00052   ALLOCATE(mypairpots(1000,nopps), myf(1000,nopps))
00053
00054   mypairpots = zero
00055
00056   dc(1) = one
00057   dc(2) = zero
00058   dc(3) = zero
00059
00060   DO ppsel = 1, nopps
00061
00062     r1 = potcoef(9, ppsel)
00063     rcut = potcoef(10, ppsel)
00064
00065     DO ii = 1, 1000
00066
00067       magr = 0.7d0 + 4.3d0*REAL(ii-1)/THOUSAND
00068
00069       ! MAGR = MAGR - POTCOEF(6,PPSEL)
00070
00071       magr2 = magr*magr
00072
00073       IF (magr .LT. r1) THEN
00074
00075         magr = magr - potcoef(6, ppsel)
00076
00077         ! CALL DUNIVSCALE(MAGR, POTCOEF(:,PPSEL), DC, PHI, DPHI)
00078
00079         polynom = magr*(potcoef(2, ppsel) + magr*(potcoef(3, ppsel) + &
00080           magr*(potcoef(4, ppsel) + magr*potcoef(5, ppsel))))
00081
00082         phi = potcoef(1, ppsel)*exp(polynom)
00083
00084         dpolynom = potcoef(2, ppsel) + magr*(two*potcoef(3, ppsel) + &

```

```

00085         magr*(three*potcoef(4,ppsel) + &
00086         four*potcoef(5,ppsel)*magr))
00087
00088         dphi = -dc*phi*dpolynom
00089
00090         !           EXPTMP = POTCOEF(6,PPSEL)*EXP(POTCOEF(7,PPSEL) * &
00091         !           (MAGR - POTCOEF(8,PPSEL)))
00092
00093         exptmp = zero
00094
00095         mypairpots(ii, ppsel) = phi + exptmp
00096
00097         !           MYPAIRPOTS(II, PPSEL) = PHI + EXPTMP - &
00098         !           POTCOEF(8,PPSEL) / (MAGR2*MAGR2*MAGR2)
00099
00100         !           MYF(II,PPSEL) = -DPHI(1) + DC(1)*(POTCOEF(7,PPSEL)*EXPTMP + &
00101         !           SIX*POTCOEF(8,PPSEL) / (MAGR*MAGR2*MAGR2*MAGR2))
00102
00103         ELSEIF (magr .GE. r1 .AND. magr .LT. rcut) THEN
00104
00105         myr = magr - r1
00106
00107         mypairpots(ii, ppsel) = potcoef(11,ppsel) + &
00108         myr*(potcoef(12,ppsel) + myr*(potcoef(13,ppsel) + &
00109         myr*(potcoef(14,ppsel) + myr*(potcoef(15,ppsel) + &
00110         myr*potcoef(16,ppsel))))))
00111
00112         myf(ii,ppsel) = potcoef(12,ppsel) + myr*(two*potcoef(13,ppsel) + &
00113         myr*(three*potcoef(14,ppsel) + &
00114         myr*(four*potcoef(15,ppsel) + &
00115         myr*five*potcoef(16,ppsel))))))
00116
00117         ENDIF
00118
00119         ENDDO
00120
00121     ENDDO
00122
00123     DO ii = 1, 1000
00124
00125         magr = 0.7d0 + 4.3d0*REAL(ii-1)/THOUSAND
00126
00127         WRITE(56, 10) magr, &
00128         (mypairpots(ii,i), i = 1, nopps)
00129
00130         WRITE(57, 10) magr , &
00131         (myf(ii,i), i = 1, nopps)
00132     ENDDO
00133
00134 10 FORMAT(100g18.9)
00135
00136     CLOSE(56)
00137     CLOSE(57)
00138
00139     DEALLOCATE(mypairpots, myf)
00140
00141     RETURN
00142
00143 END SUBROUTINE plotppot
00144

```

8.311 plot_univ.f90 File Reference

Functions/Subroutines

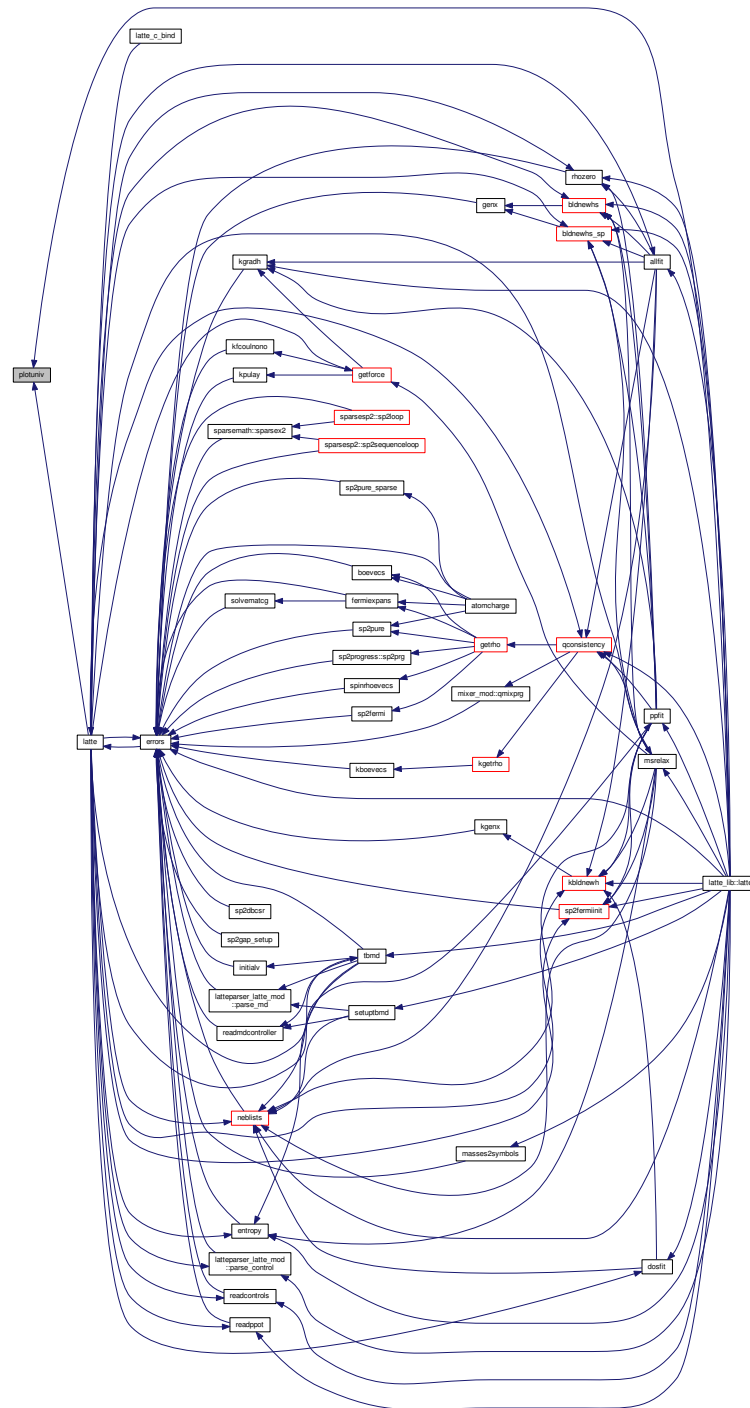
- subroutine [plotuniv](#)

8.311.1 Function/Subroutine Documentation

8.311.1.1 subroutine plotuniv ()

Definition at line 23 of file [plot_univ.f90](#).

Here is the caller graph for this function:



8.312 plot_univ.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE plotuniv
00023
00024   USE constants_mod
00025   ! USE GSPARRAY
00026   USE univarray
00027   USE ppotarray
00028   USE setuparray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: IGS, I, J
00034   INTEGER, PARAMETER :: NOSAMPLES = 1001
00035   REAL(LATTEPREC), ALLOCATABLE :: INTVALUE(:), DINTVALUE(:)
00036   REAL(LATTEPREC) :: MAGR, TMP, DTMP, POLYNOM, DPOLYNOM
00037   REAL(LATTEPREC) :: RMOD, RMINUSR1, MAXCUT
00038   IF (existerror) RETURN
00039
00040   ALLOCATE( intvalue(noint), dintvalue(noint) )
00041
00042   OPEN(unit=40, status="UNKNOWN", file="H_scaling.dat")
00043   OPEN(unit=41, status="UNKNOWN", file="dH_scaling.dat")
00044
00045   maxcut = zero
00046   DO i = 1, noint
00047     IF (bond(8,i) .GT. maxcut) maxcut = bond(8,i)
00048   ENDDO
00049
00050   DO i = 1, nosamples
00051
00052     magr = half + (maxcut - half)*REAL(i-1)/REAL(nosamples-1)
00053
00054     DO j = 1, noint
00055
00056       IF (magr .LE. bond(7,j)) THEN
00057
00058         rmod = magr - bond(6,j)
00059
00060         polynom = rmod*(bond(2,j) + rmod*(bond(3,j) + &
00061           rmod*(bond(4,j) + bond(5,j)*rmod)))
00062
00063         dpolynom = bond(2,j) + rmod*(two*bond(3,j) + &
00064           rmod*(three*bond(4,j) + four*bond(5,j)*rmod))
00065
00066         tmp = exp(polynom)
00067
00068         dtmp = dpolynom*tmp
00069
00070       ELSEIF (magr .GT. bond(7,j) .AND. magr .LE. bond(8,j)) THEN
00071
00072         rminusr1 = magr - bond(7,j)
00073
00074         tmp = bond(9,j) + rminusr1*(bond(10,j) + &
00075           rminusr1*(bond(11,j) + rminusr1*(bond(12,j) + &
00076             rminusr1*(bond(13,j) + rminusr1*bond(14,j)))))
00077
00078         dtmp = bond(10,j) + rminusr1*(two*bond(11,j) + &
00079           rminusr1*(three*bond(12,j) + rminusr1*(four*bond(13,j) + &
00080             rminusr1*five*bond(14,j)))))
00081
00082       ELSE
00083
00084         tmp = zero
00085         dtmp = zero
00086
00087       ENDIF
00088
00089       intvalue(j) = bond(1,j)*tmp
00090       dintvalue(j) = -bond(1,j)*dtmp
00091
00092     ENDDO
00093

```

```

00094      WRITE(40,10) magr, (intvalue(j), j = 1, noint)
00095      WRITE(41,10) magr, (dintvalue(j), j = 1, noint)
00096
00097      ENDDO
00098
00099      CLOSE(40)
00100      CLOSE(41)
00101
00102      IF (basistype .EQ. "NONORTHO") THEN
00103
00104          maxcut = zero
00105          DO i = 1, noint
00106              IF (overl(8,i) .GT.maxcut) maxcut = overl(8,i)
00107          ENDDO
00108
00109          OPEN(unit=40, status="UNKNOWN", file="S_scaling.dat")
00110
00111          DO i = 1, nosamples
00112
00113              magr = half + (maxcut - half)*REAL(i-1)/REAL(nosamples-1)
00114
00115              DO j = 1, noint
00116
00117                  IF (magr .LE. overl(7,j)) THEN
00118
00119                      rmod = magr - overl(6,j)
00120                      polynom = rmod*(overl(2,j) + rmod*(overl(3,j) + &
00121                          rmod*(overl(4,j) + overl(5,j)*rmod)))
00122
00123                      tmp = exp(polynom)
00124
00125                  ELSEIF (magr .GT. overl(7,j) .AND. magr .LE. overl(8,j)) THEN
00126
00127                      rminusr1 = magr - overl(7,j)
00128
00129                      tmp = overl(9,j) + rminusr1*(overl(10,j) + &
00130                          rminusr1*(overl(11,j) + rminusr1*(overl(12,j) + &
00131                          rminusr1*(overl(13,j) + rminusr1*overl(14,j))))))
00132
00133                  ELSE
00134
00135                      tmp = zero
00136
00137                  ENDIF
00138
00139                  intvalue(j) = overl(1,j)*tmp
00140
00141              ENDDO
00142
00143              WRITE(40,10) magr, (intvalue(j), j = 1, noint)
00144
00145          ENDDO
00146
00147          CLOSE(40)
00148
00149      ENDIF
00150
00151 10 FORMAT(f12.6, 1x, 500(g12.6, 1x))
00152      DEALLOCATE(intvalue, dintvalue)
00153
00154      RETURN
00155
00156 END SUBROUTINE plotuniv

```

8.313 ppfit.f90 File Reference

Functions/Subroutines

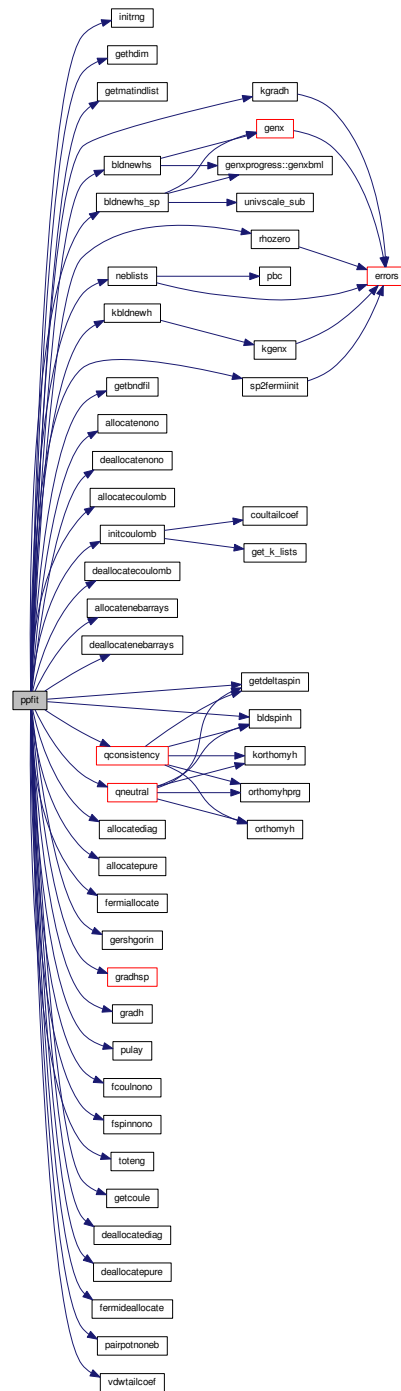
- subroutine [ppfit](#)

8.313.1 Function/Subroutine Documentation

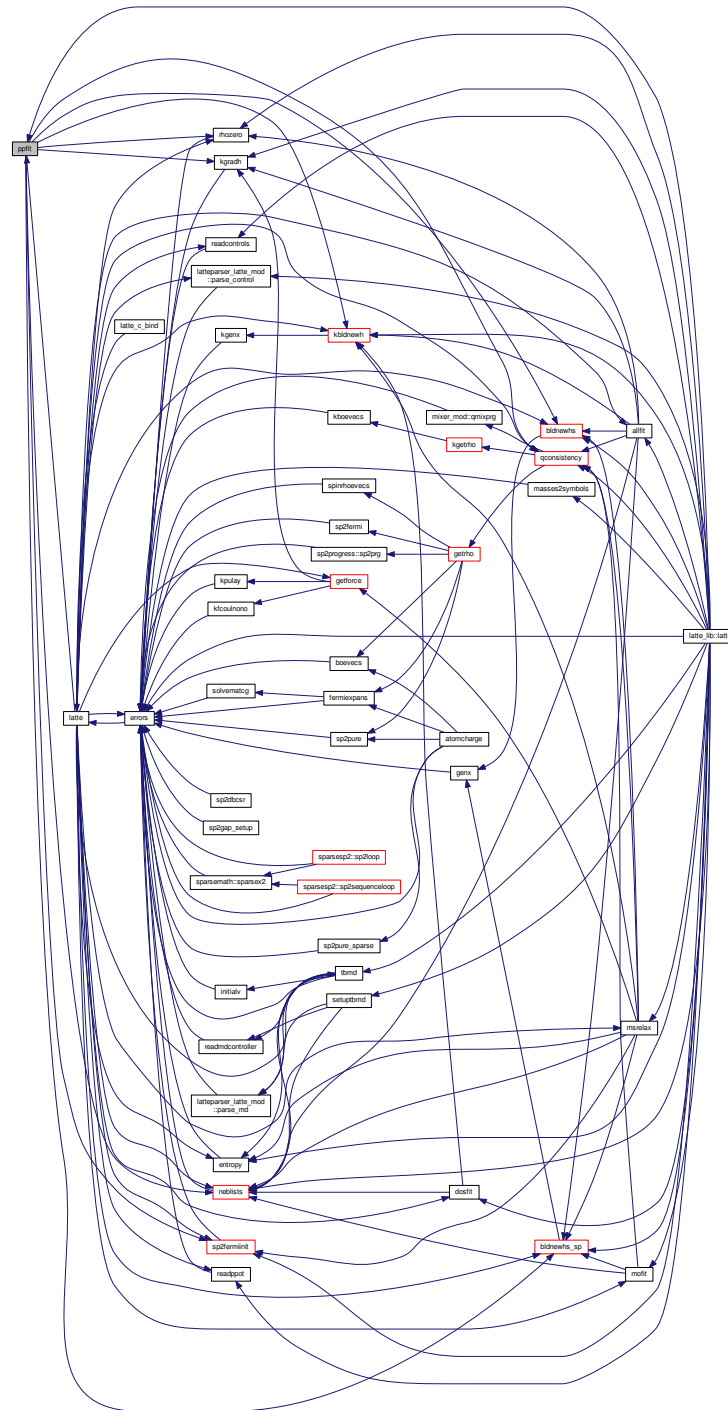
8.313.1.1 subroutine ppfit ()

Definition at line 23 of file [ppfit.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.314 ppfit.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE ppfit
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE kspacearray
00029   USE neblistarray
00030   USE univarray
00031   USE ppotarray
00032   USE myprecision
00033   ! USE MPI
00034
00035   IMPLICIT NONE
00036
00037   INTEGER :: I, J, K, Z, NSNAP, ATMOL, TOTAT, COUNT, II, COUNT2, ACC
00038   INTEGER :: PICK, PARAMPICK, COUNTSTART
00039   ! INTEGER, PARAMETER :: NMOL = 32, NGEOM = 200
00040   INTEGER, ALLOCATABLE :: ATINMOL(:), NC(:), NH(:)
00041   ! INTEGER :: PP2FIT
00042   REAL(LATTEPREC), ALLOCATABLE :: COORDS(:,:), FORCES(:,:)
00043   REAL(LATTEPREC), ALLOCATABLE :: BONDFORCES(:,:), FPAIR(:,:)
00044   REAL(LATTEPREC), ALLOCATABLE :: DFTENERGY(:)
00045   REAL(LATTEPREC) :: Y, ENERGY, MYERR, MYERRINIT, MINERR, PREVERR
00046   REAL(LATTEPREC) :: RN, TMPERR, ENERR
00047   ! REAL(LATTEPREC) :: MCBETA
00048   REAL(LATTEPREC), ALLOCATABLE :: PPBEST(:,:), PPOLD(:,:)
00049   REAL(LATTEPREC) :: DFTCE, DFTHE, TBCE, TBHE
00050   CHARACTER(LEN=2) :: X
00051   CHARACTER(LEN=2), ALLOCATABLE :: SPEC(:)
00052   IF (existerror) RETURN
00053
00054   dftce = -37.858854555371*27.211385
00055   dfthe = -0.502159745168*27.211385
00056
00057   tbce = -1.242
00058   tbhe = -0.89685
00059
00060
00061   ! Fit the atomization energies too
00062
00063   CALL initrng
00064
00065   ! Read the xyz files and gradients
00066
00067   OPEN(unit=60, status="OLD", file="forces")
00068
00069   nsnap = ppnmol*ppngeom
00070
00071   totat = 0
00072   DO i = 1, nsnap
00073     ! print*, I
00074     READ(60,*) atmol
00075     totat = totat+atmol
00076     READ(60,*) energy
00077     DO j = 1, atmol
00078       READ(60,*) x, y,y,y,y,y,y
00079     ENDDO
00080   ENDDO
00081
00082   rewind(60)
00083
00084   ALLOCATE(coords(3,totat), forces(3,totat), spec(totat), atinmol(nsnap))
00085   ALLOCATE(bondforces(3,totat), fpair(3,totat), dftenergy(nsnap))
00086   ALLOCATE(nc(nsnap), nh(nsnap))
00087
00088   count = 0
00089   count2 = 0
00090   DO i = 1, nsnap
00091     READ(60,*) atinmol(i)
00092     READ(60,*) dftenergy(i)
00093

```

```

00094      DO j = 1, atinmol(i)
00095          count = count + 1
00096          READ(60,*) spec(count), coords(1,count), coords(2,count), &
00097              coords(3,count), forces(1,count), forces(2,count), &
00098              forces(3,count)
00099
00100      ENDDO
00101  ENDDO
00102 10 FORMAT(a1,6f12.6)
00103
00104      dftenergy = dftenergy*27.211385
00105      forces = -forces*27.211385/0.591772
00106
00107      ! Turn the DFT energies into atomization energies
00108
00109      nc = 0
00110      nh = 0
00111      count = 0
00112      DO i = 1, nsnap
00113
00114          DO j = 1, atinmol(i)
00115
00116              count = count + 1
00117              IF (spec(count) .EQ. "C") nc(i) = nc(i) + 1
00118              IF (spec(count) .EQ. "H") nh(i) = nh(i) + 1
00119
00120          ENDDO
00121
00122          dftenergy(i) = dftenergy(i) - REAL(nc(i))*DFTCE - REAL(nh(i))*DFTHE
00123
00124      ENDDO
00125
00126      ! DO I = 1, NSNAP
00127      !     PRINT*, I, NC(I), NH(I), DFTENERGY(I)
00128      ! ENDDO
00129
00130
00131      ! First get the forces from the bond part and electrostatics
00132
00133      count = 0
00134
00135      DO ii = 1, nsnap
00136
00137          ! IF (MOD(II, NUMPROCS) .EQ. MYID) THEN
00138
00139          DEALLOCATE(cr, atele, f, fpp, ftot)
00140          DEALLOCATE(deltaq, mycharge)
00141          DEALLOCATE(elempointer)
00142          IF (electro .EQ. 0) DEALLOCATE(lcnshift)
00143          IF (kon .EQ. 1) DEALLOCATE(kf)
00144          IF (basistype .EQ. "NONORTHO") THEN
00145              IF (spinon .EQ. 0) THEN
00146                  DEALLOCATE(fpul, fscoul)
00147              ELSE
00148                  DEALLOCATE(fpul, fscoul, fsspin)
00149              ENDIF
00150          ENDIF
00151
00152          ! Put the coordinates into CR
00153
00154          nats = atinmol(ii)
00155
00156          ALLOCATE(cr(3,nats), atele(nats), f(3,nats), fpp(3,
nats), ftot(3,nats))
00157          ALLOCATE(deltaq(nats), mycharge(nats))
00158          ALLOCATE(elempointer(nats))
00159
00160          IF (electro .EQ. 0) THEN
00161              ALLOCATE(lcnshift(nats))
00162              lcnshift = zero
00163          ENDIF
00164
00165          IF (kon .EQ. 1) ALLOCATE(kf(3,nats))
00166
00167          IF (basistype .EQ. "NONORTHO") THEN
00168              IF (spinon .EQ. 0) THEN
00169                  ALLOCATE(fpul(3,nats), fscoul(3,nats))
00170              ELSE
00171                  ALLOCATE(fpul(3,nats), fscoul(3,nats), fsspin(3,
nats))
00172              ENDIF
00173          ENDIF
00174
00175          countstart = 0
00176          DO i = 1, ii-1
00177              countstart = countstart + atinmol(i)
00178          ENDDO

```

```

00179
00180     count = countstart
00181
00182     DO j = 1, nats
00183         count = count + 1
00184         atele(j) = spec(count)
00185         cr(1,j) = coords(1,count)
00186         cr(2,j) = coords(2,count)
00187         cr(3,j) = coords(3,count)
00188     ENDDO
00189
00190     ! Set up pointer to the data in TBparam/electrons.dat
00191
00192     DO i = 1, nats
00193         DO j = 1, noelem
00194             IF (atele(i) .EQ. ele(j)) elempointer(i) = j
00195         ENDDO
00196     ENDDO
00197
00198     ! For use when getting the partial charges
00199
00200     DEALLOCATE(qlist)
00201
00202     ! Allocate the Hamiltonian matrix
00203
00204     IF (kon .EQ. 0) THEN
00205
00206         ! Real space
00207         DEALLOCATE(h, hdiag)
00208
00209     ELSE ! k-space
00210
00211         DEALLOCATE(hk, hkdiag)
00212
00213     ENDIF
00214
00215     IF (basistype .EQ. "NONORTHO") DEALLOCATE(h0)
00216
00217     IF (spinon .EQ. 0) THEN
00218
00219         ! No spins: allocate 1 double-occupied bond order matrix
00220
00221         IF (kon .EQ. 0) THEN
00222             DEALLOCATE(bo)
00223
00224         ELSE
00225
00226             DEALLOCATE(kbo)
00227
00228         ENDIF
00229
00230     ELSEIF (spinon .EQ. 1) THEN
00231
00232         DEALLOCATE(hup, hdown)
00233         DEALLOCATE(rhoup, rhodown)
00234         DEALLOCATE(h2vect)
00235
00236         IF (basistype .EQ. "NONORTHO") DEALLOCATE(spinlist)
00237
00238         DEALLOCATE(deltaspin, olddeltaspin)
00239
00240     ENDIF
00241
00242     CALL gethdim
00243
00244     DEALLOCATE(matindlist)
00245
00246     IF (spinon .EQ. 1) DEALLOCATE(spinindlist)
00247
00248     CALL getmatindlist
00249
00250     IF (spinon .EQ. 0) THEN
00251         DEALLOCATE(bozero)
00252     ELSE
00253         DEALLOCATE(rhoupzero, rhodownzero)
00254     ENDIF
00255
00256     CALL rhozero
00257
00258     CALL getbndfil
00259
00260     IF (basistype .EQ. "NONORTHO") THEN
00261         IF (ii .EQ. 1) THEN
00262             CALL allocatenono
00263         ELSE
00264             CALL deallocatenono
00265

```

```

00266         CALL allocatenono
00267     ENDIF
00268 ENDIF
00269
00270
00271
00272     IF (ii .EQ. 1) THEN
00273
00274         CALL allocatcoulomb
00275         CALL initcoulomb
00276
00277     ELSE
00278
00279         CALL deallocatcoulomb
00280         CALL allocatcoulomb
00281         CALL initcoulomb
00282
00283     ENDIF
00284
00285     IF (ii .EQ. 1) THEN
00286
00287         CALL allocatenebarrays
00288
00289         CALL neblists(0)
00290
00291     ELSE
00292
00293         CALL deallocatenebarrays
00294
00295         CALL allocatenebarrays
00296
00297         DEALLOCATE(nebtb, nebpp)
00298         IF (electro .EQ. 1) DEALLOCATE(nebcoul)
00299
00300         CALL neblists(0)
00301
00302     ENDIF
00303
00304     ! Now we have the arrays set up we can get the energy and forces
00305
00306     ! Build the charge independent H matrix
00307
00308     IF (kon .EQ. 0) THEN
00309
00310         IF (sponly .EQ. 0) THEN
00311             CALL bldnewhs_sp
00312         ELSE
00313             CALL bldnewhs
00314         ENDIF
00315
00316     ELSE
00317
00318         CALL kbldnewh
00319
00320     ENDIF
00321
00322     IF (spinon .EQ. 1) THEN
00323         CALL getdeltaspin
00324         CALL bldspinh
00325     ENDIF
00326
00327     IF (control .EQ. 1) THEN
00328         CALL allocatediag
00329     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00330         CALL allocatpure
00331     ELSEIF (control .EQ. 3) THEN
00332         CALL fermiallocate
00333     ENDIF
00334
00335     IF (control .EQ. 5) THEN
00336
00337         CALL gershgorin
00338         CALL sp2fermiinit
00339
00340     ENDIF
00341
00342     IF (electro .EQ. 0) CALL qneutral(0,1) ! Local charge neutrality
00343
00344     IF (electro .EQ. 1) CALL qconsistency(0,1) ! Self-consistent charges
00345
00346
00347     IF (kon .EQ. 0) THEN
00348
00349         IF (sponly .EQ. 0) THEN
00350             CALL gradhsp
00351         ELSE
00352             CALL gradh

```

```

00353         ENDIF
00354
00355     ELSE
00356         CALL kgradh
00357     ENDIF
00358
00359     ftot = two * f
00360
00361     IF (electro .EQ. 1) ftot = ftot + fcoul
00362
00363     IF (basistype .EQ. "NONORTHO") THEN
00364
00365         CALL pulay
00366
00367         CALL fcoulnono
00368
00369         ftot = ftot - two*fpul + fscoul
00370
00371         IF (spinon .EQ. 1) THEN
00372             CALL fspinnono
00373             ftot = ftot + fsspin
00374         ENDIF
00375     ENDIF
00376
00377
00378     countstart = 0
00379     DO i = 1, ii-1
00380         countstart = countstart + atinmol(i)
00381     ENDDO
00382
00383     count2 = countstart
00384
00385     CALL toteng
00386     CALL getcoule
00387
00388     tote = trrhoh - ecoul - ente
00389
00390     ! Take off the contributions to the total energy from the bond term
00391
00392     dftenergy(ii) = dftenergy(ii) - (tote - REAL(nc(ii))*TBCE &
00393         - REAL(nh(ii))*TBHE)
00394
00395     DO i = 1, nats
00396         count2 = count2 + 1
00397         bondforces(1,count2) = ftot(1,i)
00398         bondforces(2,count2) = ftot(2,i)
00399         bondforces(3,count2) = ftot(3,i)
00400     ENDDO
00401
00402     IF (control .EQ. 1) THEN
00403         CALL deallocatediag
00404     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00405         CALL deallocatepure
00406     ELSEIF (control .EQ. 3) THEN
00407         CALL fermideallocate
00408     ENDIF
00409
00410     !ENDIF
00411
00412 ENDDO
00413
00414 ! CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
00415
00416 ! IF (MYID .NE. 0) THEN
00417
00418 !     CALL MPI_SEND(BONDFORCES(1,1), 3*TOTAT, MPI_DOUBLE_PRECISION, &
00419 !         0, MYID, MPI_COMM_WORLD, IERR)
00420
00421 ! ELSE
00422
00423 !     DO I = 1, NUMPROCS
00424
00425 !         CALL MPI_RECV(TMPBUFF(1,1), 3*TOTAT, MPI_DOUBLE_PRECISION, &
00426 !             MPI_ANY_SOURCE, MPI_ANY_TAG, MPI_COMM_WORLD, STATUS, IERR)
00427
00428 !
00429 !         BONDFORCES = BONDFORCE + TMPBUFF
00430
00431 !     ENDDO
00432
00433 ! ENDIF
00434
00435 ! CALL MPI_BARRIER(MPI_COMM_WORLD, IERR)
00436
00437 ! CALL MPI_BCAST(BONDFORCES, 3*TOTAT, MPI_DOUBLE_PRECISION, 0, &
00438 !     MPI_COMM_WORLD, IERR)
00439

```

```

00440 forces = forces - bondforces
00441
00442
00443 ! Now for the optimization
00444
00445 ! Initial error
00446
00447 count = 0
00448 count2 = 0
00449
00450 enerr = zero
00451
00452 DO ii = 1, nsnap
00453
00454     DEALLOCATE(cr, atele, fpp)
00455
00456     nats = atinmol(ii)
00457
00458     ALLOCATE(cr(3,nats), atele(nats), fpp(3,nats))
00459
00460     DO i = 1, nats
00461         count = count+1
00462         cr(1,i) = coords(1,count)
00463         cr(2,i) = coords(2,count)
00464         cr(3,i) = coords(3,count)
00465         atele(i) = spec(count)
00466     ENDDO
00467
00468     CALL pairpotnoneb
00469
00470     enerr = enerr + (dftenergy(ii) - erep)*(dftenergy(ii) - erep)/REAL(atinmol(ii))
00471
00472     DO i = 1, nats
00473         count2 = count2 + 1
00474         fpair(1,count2) = fpp(1,i)
00475         fpair(2,count2) = fpp(2,i)
00476         fpair(3,count2) = fpp(3,i)
00477     ENDDO
00478
00479 ENDDO
00480
00481 myerrinit = zero
00482 DO i = 1, nsnap
00483
00484     tmperr = zero
00485     DO j = 1, atinmol(i)
00486
00487         tmperr = tmperr + (forces(1,j) - fpair(1,j))*(forces(1,j) - fpair(1,j)) + &
00488             (forces(2,j) - fpair(2,j))*(forces(2,j) - fpair(2,j)) + &
00489             (forces(3,j) - fpair(3,j))*(forces(3,j) - fpair(3,j))
00490
00491     ENDDO
00492
00493     myerrinit = myerrinit + tmperr/REAL(atinmol(i))
00494
00495 ENDDO
00496
00497 enerr = enerr/REAL(nsnap)
00498 myerrinit = myerrinit/REAL(nsnap)
00499 ! PRINT*, MYERRINIT
00500
00501 myerrinit = myerrinit + enerr
00502
00503 ALLOCATE(ppbest(5,pp2fit), ppold(5,pp2fit))
00504
00505 DO i = 1, pp2fit
00506     DO j = 1, 5
00507         ppbest(j,i) = potcoef(j,i)
00508     ENDDO
00509 ENDDO
00510
00511 acc = 0
00512 DO z = 1, ppnfitstep
00513
00514     ! Change the pair potential and the cut-offs too
00515
00516     DO i = 1, pp2fit
00517         DO j = 1, 5
00518             ppold(j,i) = potcoef(j,i)
00519         ENDDO
00520     ENDDO
00521
00522     CALL random_number(rn)
00523
00524     pick = int(rn*REAL(pp2fit)) + 1
00525
00526     CALL random_number(rn)

```

```

00527
00528     parampick = int(rn*five) + 1
00529
00530     CALL random_number(rn)
00531
00532     potcoef(parampick,pick) = potcoef(parampick,pick) * &
00533         (one + ppsigma*(two*rn-one))
00534
00535     CALL vdwtailcoef
00536
00537     ! Here we get the contribution to the forces from the
00538     ! pair potential
00539
00540     count = 0
00541     count2 = 0
00542
00543     IF (z .EQ. 1) preverr = myerrinit
00544     IF (z .EQ. 1) minerr = myerrinit
00545
00546     enerr = zero
00547
00548     DO ii = 1, nsnap
00549
00550         DEALLOCATE(cr, atele, fpp)
00551
00552         nats = atinmol(ii)
00553
00554         ALLOCATE(cr(3,nats), atele(nats), fpp(3,nats))
00555
00556         DO i = 1, nats
00557             count = count+1
00558             cr(1,i) = coords(1,count)
00559             cr(2,i) = coords(2,count)
00560             cr(3,i) = coords(3,count)
00561             atele(i) = spec(count)
00562         ENDDO
00563
00564         CALL pairpotnoneb
00565
00566         enerr = enerr + (dftenergy(ii) - erep)*(dftenergy(ii) - erep)/REAL(atinmol(ii))
00567
00568         DO i = 1, nats
00569             count2 = count2 + 1
00570             fpair(1,count2) = fpp(1,i)
00571             fpair(2,count2) = fpp(2,i)
00572             fpair(3,count2) = fpp(3,i)
00573         ENDDO
00574
00575     ENDDO
00576
00577     myerr = zero
00578     DO i = 1, nsnap
00579
00580         tmperr = zero
00581
00582         DO j = 1, atinmol(i)
00583
00584             tmperr = tmperr + (forces(1,j) - fpair(1,j))*(forces(1,j) - fpair(1,j)) + &
00585                 (forces(2,j) - fpair(2,j))*(forces(2,j) - fpair(2,j)) + &
00586                 (forces(3,j) - fpair(3,j))*(forces(3,j) - fpair(3,j))
00587
00588         ENDDO
00589
00590         myerr = myerr + tmperr/REAL(atinmol(i))
00591
00592     ENDDO
00593
00594     enerr = enerr/REAL(nsnap)
00595     myerr = myerr/REAL(nsnap)
00596
00597     myerr = myerr + enerr
00598
00599     ! PRINT*, MYERR
00600
00601     IF (myerr .LT. preverr) THEN
00602
00603         acc = acc + 1
00604         preverr = myerr
00605
00606         print*, acc, myerr, enerr
00607
00608         IF (myerr .LT. minerr) THEN
00609
00610             minerr = myerr
00611
00612             DO i = 1, pp2fit
00613                 DO j = 1, 5

```



```

00614         ppbest(j,i) = potcoef(j,i)
00615     ENDDO
00616 ENDDO
00617
00618 ENDF
00619
00620 ELSE
00621
00622     CALL random_number(rn)
00623
00624     IF (exp(-(myerr - preverr)*ppbeta) .GT. rn) THEN
00625
00626         acc = acc+1
00627         preverr = myerr
00628
00629         print*, acc, myerr, enerr
00630
00631     ELSE
00632
00633         ! Put the original coefficients back
00634
00635         DO i = 1, pp2fit
00636             DO j = 1, 5
00637                 potcoef(j,i) = ppold(j,i)
00638             ENDDO
00639         ENDDO
00640
00641     ENDF
00642
00643 ENDF
00644
00645 END DO
00646
00647 DO i = 1, pp2fit
00648     WRITE(6,20) ppele1(i), ppele2(i), ppbest(1,i), ppbest(2,i), &
00649         ppbest(3,i), ppbest(4,i), ppbest(5,i), potcoef(6,i), potcoef(7,i), &
00650         potcoef(8,i), potcoef(9,i), potcoef(10,i)
00651 ENDDO
00652
00653
00654 20 FORMAT(a2, 1x, a2, 1x, 10f16.8)
00655
00656
00657 DEALLOCATE(ppbest, ppold)
00658
00659 RETURN
00660
00661 END SUBROUTINE ppfit

```

8.315 ppotarray.f90 File Reference

Modules

- module [ppotarray](#)

Variables

- integer [ppotarray::nopps](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::potcoef](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::ppak](#)
- character(len=2), dimension(:), allocatable [ppotarray::ppele1](#)
- character(len=2), dimension(:), allocatable [ppotarray::ppele2](#)
- integer, dimension(:), allocatable [ppotarray::ppnk](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::ppr](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::pprk](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::ppspl](#)
- integer, dimension(:), allocatable [ppotarray::pptablength](#)
- real(latteprec), dimension(:,:), allocatable [ppotarray::ppval](#)

8.316 ppotarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE ppotarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027
00028   SAVE
00029
00030   INTEGER :: nopps
00031   INTEGER, ALLOCATABLE :: pptablenth(:), ppnk(:)
00032   REAL(LATTEPREC), ALLOCATABLE :: potcoef(:, :)
00033   CHARACTER(LEN=2), ALLOCATABLE :: ppele1(:), ppele2(:)
00034   REAL(LATTEPREC), ALLOCATABLE :: ppr(:, :), ppval(:, :), ppspl(:, :)
00035   ! These ones are for the spline pps
00036   REAL(LATTEPREC), ALLOCATABLE :: prk(:, :), ppak(:, :)
00037
00038 END MODULE ppotarray

```

8.317 printspare.f90 File Reference

Functions/Subroutines

- subroutine [printspare](#)

8.317.1 Function/Subroutine Documentation

8.317.1.1 subroutine printspare ()

Definition at line 23 of file [printspare.f90](#).

8.318 printspare.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !

```

```

00012 !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE printsparse
00023
00024   USE constants_mod
00025   USE setuparray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J
00030   IF (existerror) RETURN
00031
00032   OPEN(unit=67, status="UNKNOWN", file="BOsparsitude.dat")
00033
00034   DO i = 1, hdim
00035     DO j = 1, hdim
00036
00037       IF (abs(bo(j,i)) .GT. 1.0d-12) THEN
00038
00039         WRITE(67,10) j,i
00040
00041       ENDIF
00042
00043     ENDDO
00044   ENDDO
00045
00046 10 FORMAT(i6,1x,i6)
00047
00048   CLOSE(67)
00049
00050   RETURN
00051
00052 END SUBROUTINE printsparse

```

8.319 propchempot_xbo.f90 File Reference

Functions/Subroutines

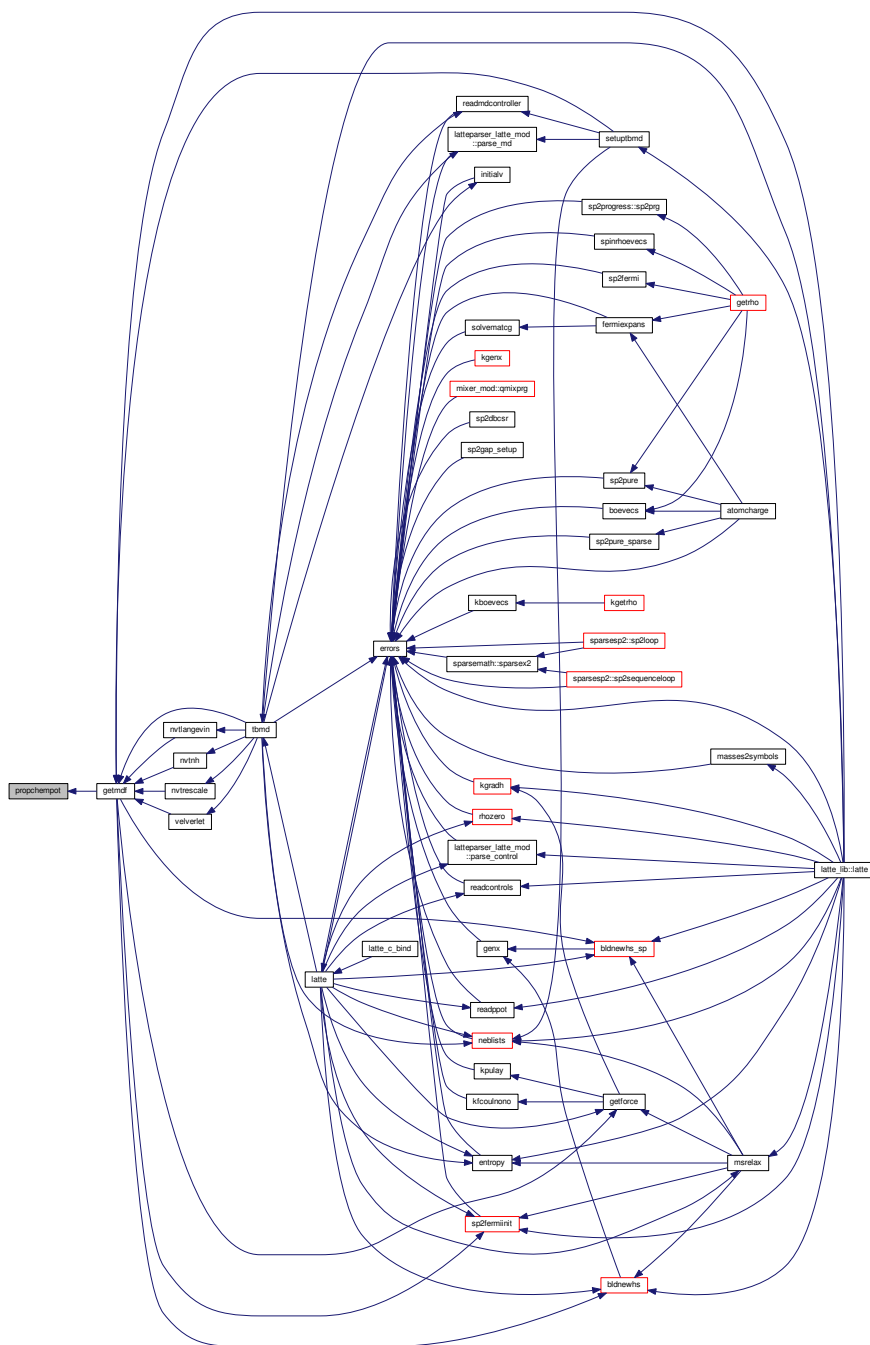
- subroutine [propchempot](#) (ITER)

8.319.1 Function/Subroutine Documentation

8.319.1.1 subroutine propchempot (integer *ITER*)

Definition at line 23 of file [propchempot_xbo.f90](#).

Here is the caller graph for this function:



8.320 propchempot_xbo.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE propchempot(ITER)
00023
00024   USE constants_mod
00025   USE xboarray
00026   USE setuparray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, ITER
00032   IF (existerror) RETURN
00033
00034   ! If iter = 1, then we just set up our arrays of previous guesses
00035
00036   IF (iter .EQ. 1) THEN
00037
00038     ! Dissipation off
00039
00040     IF (xbodison .EQ. 0) THEN
00041
00042       chempot_pnk(1) = chempot
00043       chempot_pnk(2) = chempot_pnk(1)
00044
00045       ! Dissipation on
00046
00047     ELSE
00048
00049       chempot_pnk(1) = chempot
00050
00051       DO i = 2, xbodisorder + 1
00052         chempot_pnk(i) = chempot_pnk(i-1)
00053       ENDDO
00054
00055     ENDIF
00056
00057   ELSE ! For ITER > 1
00058
00059     IF (xbodison .EQ. 0) THEN
00060
00061       chempot = two*chempot_pnk(1) - chempot_pnk(2) + &
00062         two*(chempot - chempot_pnk(1))
00063
00064       chempot_pnk(2) = chempot_pnk(1)
00065       chempot_pnk(1) = chempot
00066
00067     ELSEIF (xbodison .EQ. 1) THEN
00068
00069       chempot = two*chempot_pnk(1) - chempot_pnk(2) + &
00070         kappa_scale*kappa_xbo*(chempot -
00071         chempot_pnk(1))
00072
00073       DO i = 1, xbodisorder + 1
00074         chempot = chempot + alpha_xbo*cnk(i)*
00075         chempot_pnk(i)
00076       ENDDO
00077
00078       DO i = 1, xbodisorder
00079
00080         chempot_pnk(xbodisorder + 2 - i) = &
00081         chempot_pnk(xbodisorder + 1 - i)
00082       ENDDO
00083
00084       chempot_pnk(1) = chempot
00085
00086     ENDIF
00087
00088   ENDIF
00089
00090   RETURN
00091
00092 END SUBROUTINE propchempot
00093
00094

```

00095

8.321 propspins_xbo.f90 File Reference

Functions/Subroutines

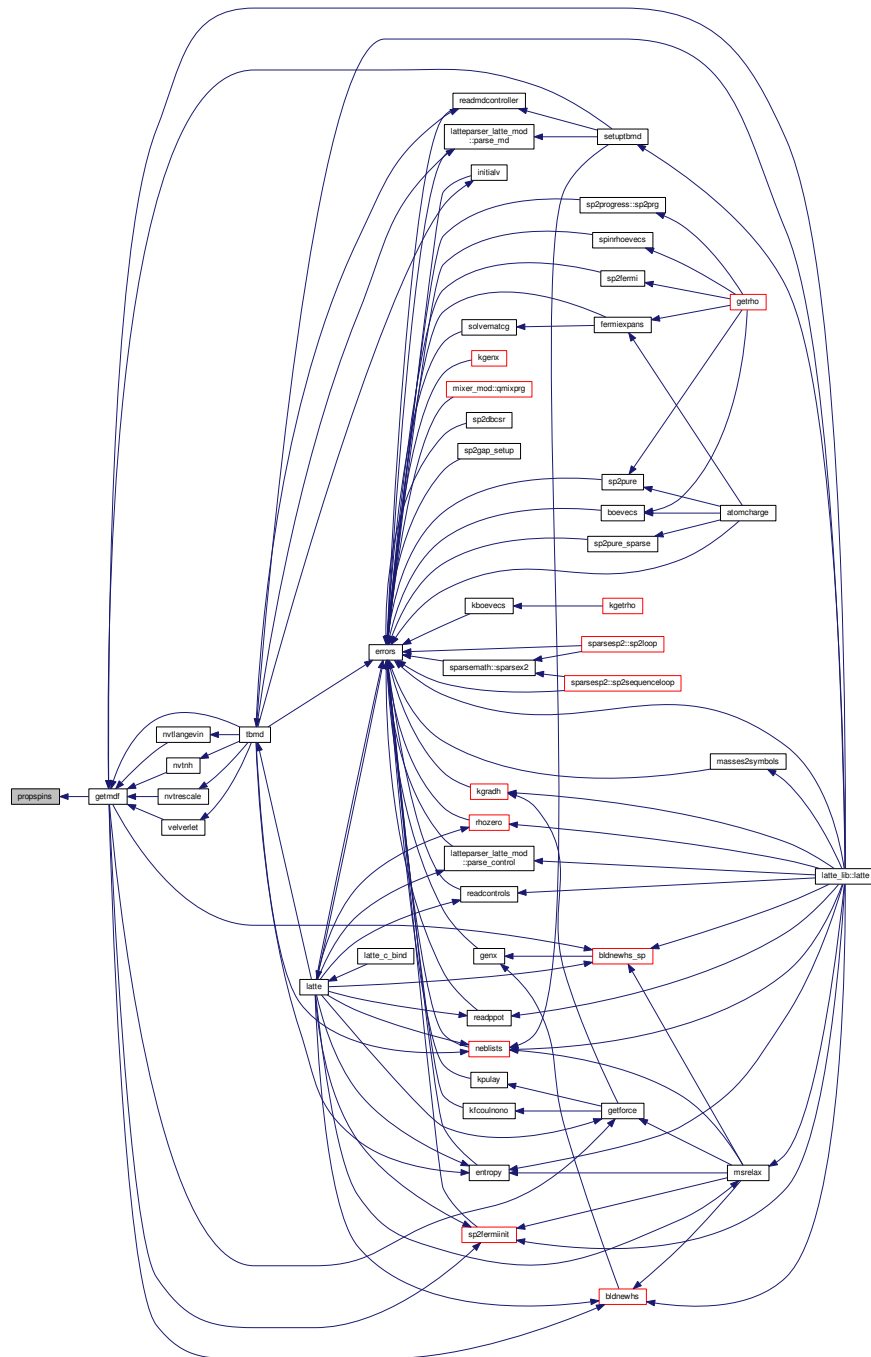
- subroutine [propspins](#) (ITER)

8.321.1 Function/Subroutine Documentation

8.321.1.1 subroutine [propspins](#) (integer *ITER*)

Definition at line [23](#) of file [propspins_xbo.f90](#).

Here is the caller graph for this function:



8.322 propspins_xbo.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos
00004 ! National Laboratory (LANL), which is operated by Los Alamos National
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS

```

```

00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE propspins(ITER)
00023
00024 USE constants_mod
00025 USE xboarray
00026 USE setuparray
00027 USE spinarray
00028 USE myprecision
00029
00030 IMPLICIT NONE
00031
00032 INTEGER :: I, J, ITER
00033 IF (existerror) RETURN
00034
00035 ! If iter = 1, then we just set up our arrays of previous guesses
00036
00037 IF (iter .EQ. 1) THEN
00038
00039     ! Dissipation off
00040
00041     IF (xbodison .EQ. 0) THEN
00042
00043         DO i = 1, deltadim
00044             spin_pnk(1,i) = deltaspin(i)
00045             spin_pnk(2,i) = spin_pnk(1,i)
00046         ENDDO
00047
00048         ! Dissipation on
00049
00050     ELSEIF (xbodison .EQ. 1) THEN
00051
00052         DO i = 1, deltadim
00053             spin_pnk(1,i) = deltaspin(i)
00054
00055             DO j = 2, xbodisorder + 1
00056                 spin_pnk(j,i) = spin_pnk(j-1,i)
00057             ENDDO
00058
00059         ENDDO
00060
00061     ENDIF
00062
00063 ELSE
00064     ! ITER > 1
00065
00066     IF (xbodison .EQ. 0) THEN
00067
00068         DO i = 1, deltadim
00069
00070             deltaspin(i) = two*spin_pnk(1,i) - spin_pnk(2,i) + &
00071                 two*(deltaspin(i) - spin_pnk(1,i))
00072
00073             spin_pnk(2,i) = spin_pnk(1,i)
00074             spin_pnk(1,i) = deltaspin(i)
00075
00076         ENDDO
00077
00078     ELSE ! Dissipation on (XBODISON .EQ. 1)
00079
00080         DO i = 1, deltadim
00081
00082             deltaspin(i) = two*spin_pnk(1,i) - spin_pnk(2,i) + &
00083                 kappa_scale*kappa_xbo*(deltaspin(i) -
00084 spin_pnk(1,i))
00085
00086             DO j = 1, xbodisorder + 1
00087                 deltaspin(i) = deltaspin(i) + &
00088                     alpha_xbo*cnk(j)*spin_pnk(j,i)
00089             ENDDO
00090
00091             DO j = 1, xbodisorder
00092                 spin_pnk(xbodisorder + 2 - j, i) = &

```



```
00095             spin_pnk(xbodisorder + 1 - j, i)
00096
00097             ENDDO
00098
00099             spin_pnk(1,i) = deltaspin(i)
00100
00101             ENDDO
00102
00103             ENDIF
00104
00105             ENDIF
00106
00107             RETURN
00108
00109 END SUBROUTINE propspins
00110
00111
```

8.323 pulay.f90 File Reference

Functions/Subroutines

- subroutine [pulay](#)

8.323.1 Function/Subroutine Documentation

8.323.1.1 subroutine [pulay](#) ()

Definition at line 23 of file [pulay.f90](#).

[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pulay
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE nonoarray
00028   USE neblastarray
00029   USE spinarray
00030   USE virialarray
00031   USE myprecision
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, J, K, L, M, N, KK, INDI, INDJ
00036   INTEGER :: LBRA, MBRA, LKET, MKET
00037   INTEGER :: PREVJ, NEWJ
00038   INTEGER :: PBCI, PBCJ, PBCK
00039   INTEGER :: BASISI(5), BASISJ(5), LBRAIN, LKETINC
00040   REAL(LATTEPREC) :: ALPHA, BETA, PHI, RHO, COSBETA
00041   REAL(LATTEPREC) :: RIJ(3), DC(3)
00042   REAL(LATTEPREC) :: MAGR, MAGR2, MAGRP, MAGRP2, FTMP(3)
00043   REAL(LATTEPREC) :: MAXRCUT, MAXRCUT2
00044   REAL(LATTEPREC) :: MYDFDA, MYDFDB, MYDFDR, RCUTTB
00045   REAL(LATTEPREC), EXTERNAL :: DFDA, DFDB, DFDR
00046   LOGICAL PATH
00047   IF (existerror) RETURN
00048
00049   fpul = zero
00050
00051   virpul = zero
00052
00053   ! We first have to make the matrix S^-1 H rho = X^2 H rho
00054
00055   IF (spinon .EQ. 0) THEN
00056       IF (kbt .GT. 0.0000001) THEN
00057
00058 #ifdef DOUBLEPREC
00059
00060         ! XMAT * XMAT = S^-1
00061
00062         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00063             xmat, hdim, xmat, hdim, zero, x2hrho,
00064             hdim)
00065
00066         ! S^-1 * H
00067
00068         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00069             x2hrho, hdim, h, hdim, zero, nonotmp,
00070             hdim)
00071
00072         ! (S^-1 * H)*RHO
00073
00074         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00075             nonotmp, hdim, bo, hdim, zero, x2hrho,
00076             hdim)
00077
00078 #elif defined(SINGLEPREC)
00079
00080         ! XMAT * XMAT = S^-1
00081
00082         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00083             xmat, hdim, xmat, hdim, zero, x2hrho,
00084             hdim)
00085
00086         ! S^-1 * H
00087
00088         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00089             x2hrho, hdim, h, hdim, zero, nonotmp,
00090             hdim)
00091
00092         ! (S^-1 * H)*RHO

```

```

00089
00090     CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00091             nonotmp, hdim, bo, hdim, zero, x2hrho,
             hdim)
00092
00093 #endif
00094
00095     ELSE
00096
00097         ! Te = 0 : Fp = 2Tr[rho H rho dS/dR]
00098
00099         ! Be careful - we're working with bo = 2rho, so we need
00100         ! the factor of 1/2...
00101
00102 #ifdef DOUBLEPREC
00103
00104     CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00105             bo, hdim, h, hdim, zero, nonotmp, hdim)
00106
00107     CALL dgemm('N', 'N', hdim, hdim, hdim, half, &
00108             nonotmp, hdim, bo, hdim, zero, x2hrho,
             hdim)
00109
00110 #elif defined(SINGLEPREC)
00111
00112     CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00113             bo, hdim, h, hdim, zero, nonotmp, hdim)
00114
00115     CALL sgemm('N', 'N', hdim, hdim, hdim, half, &
00116             nonotmp, hdim, bo, hdim, zero, x2hrho,
             hdim)
00117
00118 #endif
00119
00120     ENDIF
00121
00122     ELSE
00123
00124         IF (kbt .GT. 0.0000001) THEN
00125
00126 #ifdef DOUBLEPREC
00127
00128             ! XMAT * XMAT = S^-1
00129
00130             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00131                     xmat, hdim, xmat, hdim, zero, spintmp,
                     hdim)
00132
00133             !      Hup * rhoup
00134
00135             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00136                     hup, hdim, rhoup, hdim, zero, nonotmp,
                     hdim)
00137
00138             ! (Hup * rhoup) + Hdown*rhdown
00139
00140             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00141                     hdown, hdim, rhdown, hdim, one, nonotmp,
                     hdim)
00142
00143
00144             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00145                     spintmp, hdim, nonotmp, hdim, zero,
                     x2hrho, hdim)
00146
00147 #elif defined(SINGLEPREC)
00148
00149             ! XMAT * XMAT = S^-1
00150
00151             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00152                     xmat, hdim, xmat, hdim, zero, spintmp,
                     hdim)
00153
00154             !      Hup * rhoup
00155
00156             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00157                     hup, hdim, rhoup, hdim, zero, nonotmp,
                     hdim)
00158
00159             ! (Hup * rhoup) + Hdown*rhdown
00160
00161             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00162                     hdown, hdim, rhdown, hdim, one, nonotmp,
                     hdim)
00163
00164
00165             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &

```

```

00166         spintmp, hdim, nonotmp, hdim, zero,
00167         x2hrho, hdim)
00168 #endif
00169
00170         ELSE
00171
00172 #ifdef DOUBLEPREC
00173
00174         ! At zero K: tr(rho H rho)
00175
00176         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00177         rhoup, hdim, hup, hdim, zero, nonotmp,
00178         hdim)
00179
00180         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00181         nonotmp, hdim, rhoup, hdim, zero, x2hrho,
00182         hdim)
00183
00184         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00185         rhodown, hdim, hdown, hdim, zero, nonotmp,
00186         hdim)
00187
00188         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00189         nonotmp, hdim, rhodown, hdim, one, x2hrho,
00190         hdim)
00191
00192 #elif defined(SINGLEPREC)
00193
00194         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00195         rhoup, hdim, hup, hdim, zero, nonotmp,
00196         hdim)
00197
00198         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00199         nonotmp, hdim, rhoup, hdim, zero, x2hrho,
00200         hdim)
00201
00202         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00203         rhodown, hdim, hdown, hdim, zero, nonotmp,
00204         hdim)
00205
00206         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00207         nonotmp, hdim, rhodown, hdim, one, x2hrho,
00208         hdim)
00209
00210 #endif
00211
00212         ENDIF
00213
00214         ENDIF
00215
00216         !$OMP PARALLEL DO DEFAULT (NONE) &
00217         !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00218         !$OMP SHARED(CR, BOX, X2HRHO, NOINT, ATELE, ELE1, ELE2) &
00219         !$OMP SHARED(BOND, OVERL, MATINDLIST, BASISTYPE) &
00220         !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00221         !$OMP PRIVATE(RIJ, MAGR2, MAGR, MAGRP2, MAGRP, PATH, PHI, ALPHA, BETA, COSBETA, FTMP) &
00222         !$OMP PRIVATE(DC, LBRAIN, LBRA, MBRA, L, LKETINC, LKET, MKET, RHO) &
00223         !$OMP PRIVATE(MYDFDA, MYDFDB, MYDFDR, RCUTTB) &
00224         !$OMP REDUCTION(+:FPUL, VIRPUL)
00225
00226         DO i = 1, nats
00227
00228         ! Build list of orbitals on atom I
00229         SELECT CASE(basis(elempointer(i)))
00230
00231         CASE("s")
00232             basisi(1) = 0
00233             basisi(2) = -1
00234
00235         CASE("p")
00236             basisi(1) = 1
00237             basisi(2) = -1
00238
00239         CASE("d")
00240             basisi(1) = 2
00241             basisi(2) = -1
00242
00243         CASE("f")
00244             basisi(1) = 3
00245             basisi(2) = -1
00246
00247         CASE("sp")
00248             basisi(1) = 0
00249             basisi(2) = 1
00250             basisi(3) = -1
00251
00252         CASE("sd")
00253             basisi(1) = 0
00254             basisi(2) = 2
00255             basisi(3) = -1
00256

```

```

00244     CASE("sf")
00245         basisi(1) = 0
00246         basisi(2) = 3
00247         basisi(3) = -1
00248     CASE("pd")
00249         basisi(1) = 1
00250         basisi(2) = 2
00251         basisi(3) = -1
00252     CASE("pf")
00253         basisi(1) = 1
00254         basisi(2) = 3
00255         basisi(3) = -1
00256     CASE("df")
00257         basisi(1) = 2
00258         basisi(2) = 3
00259         basisi(3) = -1
00260     CASE("spd")
00261         basisi(1) = 0
00262         basisi(2) = 1
00263         basisi(3) = 2
00264         basisi(4) = -1
00265     CASE("spf")
00266         basisi(1) = 0
00267         basisi(2) = 1
00268         basisi(3) = 3
00269         basisi(4) = -1
00270     CASE("sdf")
00271         basisi(1) = 0
00272         basisi(2) = 2
00273         basisi(3) = 3
00274         basisi(4) = -1
00275     CASE("pdf")
00276         basisi(1) = 1
00277         basisi(2) = 2
00278         basisi(3) = 3
00279         basisi(4) = -1
00280     CASE("spdf")
00281         basisi(1) = 0
00282         basisi(2) = 1
00283         basisi(3) = 2
00284         basisi(4) = 3
00285         basisi(5) = -1
00286     END SELECT
00287
00288     indi = matindlist(i)
00289
00290     DO newj = 1, totnebtb(i)
00291
00292         j = nebtb(1, newj, i)
00293         pbci = nebtb(2, newj, i)
00294         pbcj = nebtb(3, newj, i)
00295         pbck = nebtb(4, newj, i)
00296
00297         rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00298             REAL(pbck)*BOX(3,1) - CR(1,i)
00299
00300         rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00301             REAL(pbck)*BOX(3,2) - CR(2,i)
00302
00303         rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00304             REAL(pbck)*BOX(3,3) - CR(3,i)
00305
00306         magr2 = rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3)
00307
00308         ! Building the forces is expensive - use the cut-off
00309
00310         rcuttb = zero
00311         DO k = 1, noint
00312
00313             IF ( (atele(i) .EQ. ele1(k) .AND. &
00314                 atele(j) .EQ. ele2(k)) .OR. &
00315                 (atele(j) .EQ. ele1(k) .AND. &
00316                 atele(i) .EQ. ele2(k) ) ) THEN
00317
00318                 IF (overl(8,k) .GT. rcuttb ) rcuttb = overl(8,k)
00319
00320             ENDIF
00321
00322         ENDDO
00323
00324         IF (magr2 .LT. rcuttb*rcuttb) THEN
00325
00326             magr = sqrt(magr2)
00327
00328             ! Build list of orbitals on atom J
00329
00330             SELECT CASE(basis(elempointer(j)))

```

```

00331      CASE("s")
00332          basisj(1) = 0
00333          basisj(2) = -1
00334      CASE("p")
00335          basisj(1) = 1
00336          basisj(2) = -1
00337      CASE("d")
00338          basisj(1) = 2
00339          basisj(2) = -1
00340      CASE("f")
00341          basisj(1) = 3
00342          basisj(2) = -1
00343      CASE("sp")
00344          basisj(1) = 0
00345          basisj(2) = 1
00346          basisj(3) = -1
00347      CASE("sd")
00348          basisj(1) = 0
00349          basisj(2) = 2
00350          basisj(3) = -1
00351      CASE("sf")
00352          basisj(1) = 0
00353          basisj(2) = 3
00354          basisj(3) = -1
00355      CASE("pd")
00356          basisj(1) = 1
00357          basisj(2) = 2
00358          basisj(3) = -1
00359      CASE("pf")
00360          basisj(1) = 1
00361          basisj(2) = 3
00362          basisj(3) = -1
00363      CASE("df")
00364          basisj(1) = 2
00365          basisj(2) = 3
00366          basisj(3) = -1
00367      CASE("spd")
00368          basisj(1) = 0
00369          basisj(2) = 1
00370          basisj(3) = 2
00371          basisj(4) = -1
00372      CASE("spf")
00373          basisj(1) = 0
00374          basisj(2) = 1
00375          basisj(3) = 3
00376          basisj(4) = -1
00377      CASE("sdf")
00378          basisj(1) = 0
00379          basisj(2) = 2
00380          basisj(3) = 3
00381          basisj(4) = -1
00382      CASE("pdf")
00383          basisj(1) = 1
00384          basisj(2) = 2
00385          basisj(3) = 3
00386          basisj(4) = -1
00387      CASE("spdf")
00388          basisj(1) = 0
00389          basisj(2) = 1
00390          basisj(3) = 2
00391          basisj(4) = 3
00392          basisj(5) = -1
00393      END SELECT
00394
00395      indj = matindlist(j)
00396
00397      magrp2 = rij(1)*rij(1) + rij(2)*rij(2)
00398      magrp = sqrt(magrp2)
00399
00400
00401      ! transform to system in which z-axis is aligned with RIJ
00402
00403      path = .false.
00404      IF (abs(rij(1)) .GT. 1e-12) THEN
00405          IF (rij(1) .GT. zero .AND. rij(2) .GE. zero) THEN
00406              phi = zero
00407          ELSEIF (rij(1) .GT. zero .AND. rij(2) .LT. zero) THEN
00408              phi = two * pi
00409          ELSE
00410              phi = pi
00411          ENDIF
00412          alpha = atan(rij(2) / rij(1)) + phi
00413      ELSEIF (abs(rij(2)) .GT. 1e-12) THEN
00414          IF (rij(2) .GT. 1e-12) THEN
00415              alpha = pi / two
00416          ELSE
00417              alpha = three * pi / two

```

```

00418         ENDIF
00419     ELSE
00420         ! pathological case: pole in alpha at beta=0
00421         path = .true.
00422     ENDIF
00423
00424     cosbeta = rij(3)/magr
00425     beta = acos(rij(3) / magr)
00426
00427     dc = rij/magr
00428
00429     ! build forces using PRB 72 165107 eq. (12) - the sign of the
00430     ! dfda contribution seems to be wrong, but gives the right
00431     ! answer(?)
00432
00433     ftmp = zero
00434     k = indi
00435
00436     lbrainc = 1
00437     DO WHILE (basisi(lbrainc) .NE. -1)
00438
00439         lbra = basisi(lbrainc)
00440         lbrainc = lbrainc + 1
00441
00442         DO mbra = -lbra, lbra
00443
00444             k = k + 1
00445             l = indj
00446
00447             lketinc = 1
00448             DO WHILE (basisj(lketinc) .NE. -1)
00449
00450                 lket = basisj(lketinc)
00451                 lketinc = lketinc + 1
00452
00453                 DO mket = -lket, lket
00454
00455                     l = l + 1
00456
00457                     rho = x2hrho(l, k)
00458
00459                     IF (.NOT. path) THEN
00460
00461                         ! Unroll loops and pre-compute
00462
00463                         mydfda = dfda(i, j, lbra, lket, mbra, &
00464                             mket, magr, alpha, cosbeta, "S")
00465
00466                         mydfdb = dfdb(i, j, lbra, lket, mbra, &
00467                             mket, magr, alpha, cosbeta, "S")
00468
00469                         mydfdr = dfdr(i, j, lbra, lket, mbra, &
00470                             mket, magr, alpha, cosbeta, "S")
00471
00472                         !
00473                         ! d/d_alpha
00474                         !
00475
00476                         ftmp(1) = ftmp(1) + rho * &
00477                             (-rij(2) / magrp2 * mydfda)
00478
00479                         ftmp(2) = ftmp(2) + rho * &
00480                             (rij(1)/ magrp2 * mydfda)
00481
00482                         !
00483                         ! d/d_beta
00484                         !
00485
00486                         ftmp(1) = ftmp(1) + rho * &
00487                             (((rij(3) * rij(1)) / &
00488                                 magr2)) / magrp) * mydfdb)
00489
00490                         ftmp(2) = ftmp(2) + rho * &
00491                             (((rij(3) * rij(2)) / &
00492                                 magr2)) / magrp) * mydfdb)
00493
00494                         ftmp(3) = ftmp(3) - rho * &
00495                             (((one - (rij(3) * rij(3)) / &
00496                                 magr2)) / magrp) * mydfdb)
00497
00498                         !
00499                         ! d/dR
00500                         !
00501
00502                         ftmp(1) = ftmp(1) - rho * dc(1) * &
00503                             mydfdr
00504

```



```

00505          ftmp(2) = ftmp(2) - rho * dc(2) * &
00506              mydfdr
00507
00508          ftmp(3) = ftmp(3) - rho * dc(3) * &
00509              mydfdr
00510
00511          ELSE
00512
00513              ! pathological configuration in which beta=0
00514              ! or pi => alpha undefined
00515
00516              ! fixed: MJC 12/17/13
00517
00518              mydfdb = dfdb(i, j, lbra, lket, &
00519                  mbra, mket, magr, zero, cosbeta, "S") / magr
00520
00521              ftmp(1) = ftmp(1) - rho * (cosbeta * mydfdb)
00522
00523              mydfdb = dfdb(i, j, lbra, lket, &
00524                  mbra, mket, magr, pi/two, cosbeta, "S") / magr
00525
00526              ftmp(2) = ftmp(2) - rho * (cosbeta * mydfdb)
00527
00528              mydfdr = dfdr(i, j, lbra, lket, mbra, &
00529                  mket, magr, zero, cosbeta, "S")
00530
00531              ftmp(3) = ftmp(3) - rho * cosbeta * mydfdr
00532
00533          ENDIF
00534
00535          ENDDO
00536      ENDDO
00537  ENDDO
00538  ENDDO
00539
00540      fpul(1,i) = fpul(1,i) + ftmp(1)
00541      fpul(2,i) = fpul(2,i) + ftmp(2)
00542      fpul(3,i) = fpul(3,i) + ftmp(3)
00543
00544      virpul(1) = virpul(1) + rij(1) * ftmp(1)
00545      virpul(2) = virpul(2) + rij(2) * ftmp(2)
00546      virpul(3) = virpul(3) + rij(3) * ftmp(3)
00547      virpul(4) = virpul(4) + rij(1) * ftmp(2)
00548      virpul(5) = virpul(5) + rij(2) * ftmp(3)
00549      virpul(6) = virpul(6) + rij(3) * ftmp(1)
00550
00551  ENDIF
00552
00553  ENDDO
00554
00555  ENDDO
00556
00557  !$OMP END PARALLEL DO
00558
00559  ! DO I= 1, NATS
00560  !   WRITE(6,10) I, FPUL(1,I), FPUL(2,I), FPUL(3,I)
00561  !   ENDDO
00562
00563  !10 FORMAT(I4, 3F12.6)
00564
00565  RETURN
00566
00567 END SUBROUTINE pulay

```

8.325 pulay_sp.f90 File Reference

Functions/Subroutines

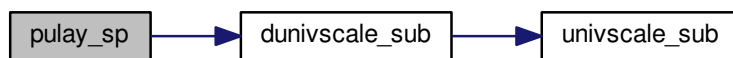
- subroutine [pulay_sp](#)

8.325.1 Function/Subroutine Documentation

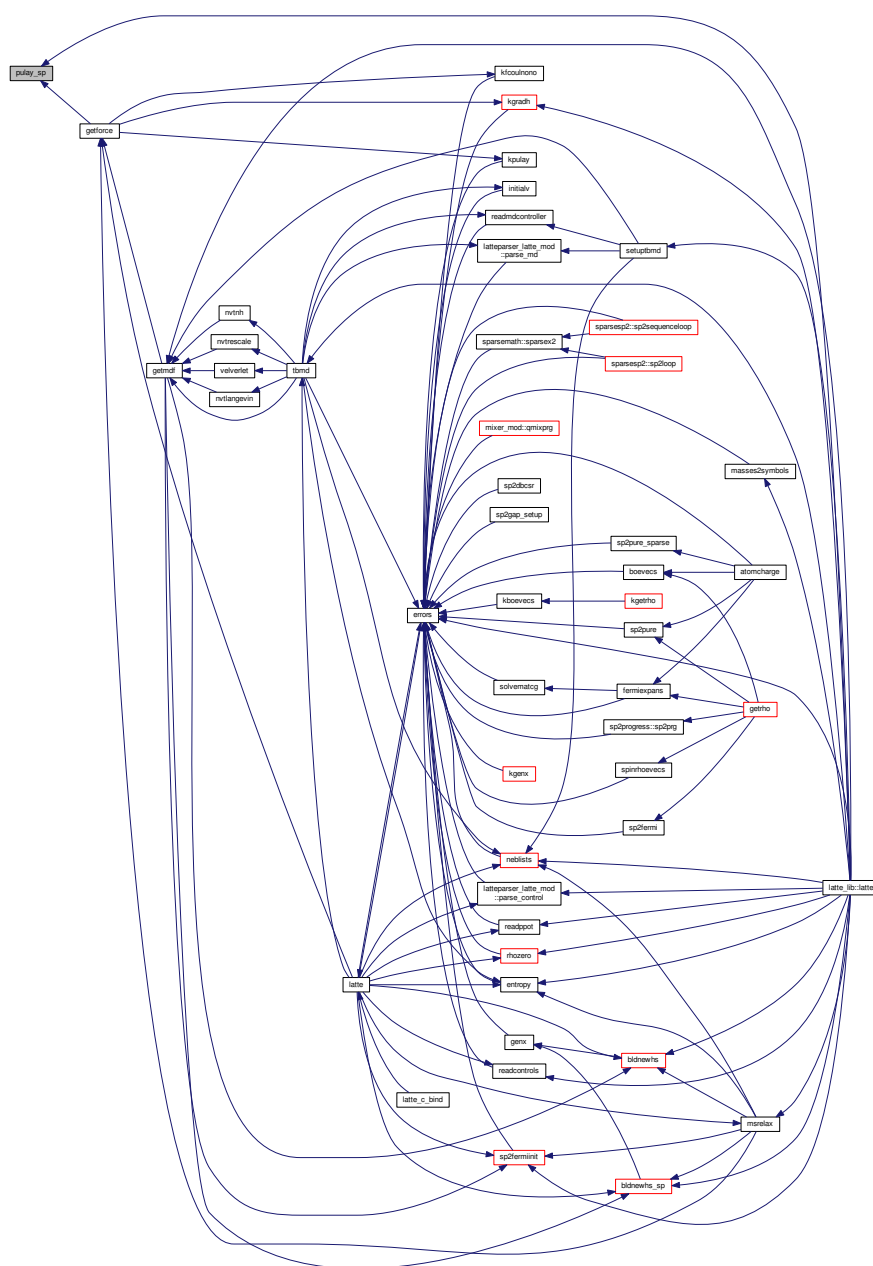
8.325.1.1 subroutine [pulay_sp](#) ()

Definition at line 23 of file [pulay_sp.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.326 pulay_sp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pulay_sp
00023
00024     USE constants_mod
00025     USE setuparray
00026     USE univarray
00027     USE nonoarray
00028     USE neblistarray
00029     USE spinarray
00030     USE virialarray
00031     USE myprecision
00032
00033     IMPLICIT NONE
00034
00035     INTEGER :: I, J, K, KK, INDI, INDJ
00036     INTEGER :: NEWJ
00037     INTEGER :: PBCI, PBCJ, PBCK
00038     REAL(LATTEPREC) :: HSSS, HSPS, HPSS, HPPS, HPPP
00039     REAL(LATTEPREC) :: RIJ(3), DC(3), SCLGSP, DGSPDR(3)
00040     REAL(LATTEPREC) :: L, M, N, L2, M2, N2, LM, LN, MN, LMN
00041     REAL(LATTEPREC) :: DSSSDR(3), DSPSDR(3), DPSSDR(3), DPPSDR(3), DPPPDR(3)
00042     REAL(LATTEPREC) :: MAGR, INVR, FTMP(3)
00043     REAL(LATTEPREC) :: PPSMPPP, PPSUBINVR
00044     CHARACTER(LEN=2) :: BASISI, BASISJ
00045     IF (existerror) RETURN
00046
00047     indi = 0
00048
00049     fpul = zero
00050
00051     virpul = zero
00052
00053     ! We first have to make the matrix S^-1 H rho = X^2 H rho
00054
00055     IF (spinon .EQ. 0) THEN
00056
00057         IF (kbt .GT. 0.0000001) THEN
00058
00059 #ifdef DOUBLEPREC
00060
00061             ! XMAT * XMAT = S^-1
00062
00063             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00064                 xmat, hdim, xmat, hdim, zero, x2hrho,
00065                 hdim)
00066
00067             ! S^-1 * H
00068
00069             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00070                 x2hrho, hdim, h, hdim, zero, nonotmp,
00071                 hdim)
00072
00073             ! (S^-1 * H)*RHO
00074
00075             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00076                 nonotmp, hdim, bo, hdim, zero, x2hrho,
00077                 hdim)
00078
00079 #elif defined(SINGLEPREC)
00080
00081             ! XMAT * XMAT = S^-1
00082
00083             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00084                 xmat, hdim, xmat, hdim, zero, x2hrho,

```

```

    hdim)
00082
00083     ! S^-1 * H
00084
00085     CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00086               x2hrho, hdim, h, hdim, zero, nonotmp,
    hdim)
00087
00088     ! (S^-1 * H)*RHO
00089
00090     CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00091               nonotmp, hdim, bo, hdim, zero, x2hrho,
    hdim)
00092
00093 #endif
00094
00095     ELSE
00096
00097         ! Te = 0 : Fp = 2Tr[rho H rho dS/dR]
00098
00099         ! Be careful - we're working with bo = 2rho, so we need
00100         ! the factor of 1/2...
00101
00102 #ifdef DOUBLEPREC
00103
00104         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00105               bo, hdim, h, hdim, zero, nonotmp, hdim)
00106
00107         CALL dgemm('N', 'N', hdim, hdim, hdim, half, &
00108               nonotmp, hdim, bo, hdim, zero, x2hrho,
    hdim)
00109
00110 #elif defined(SINGLEPREC)
00111
00112         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00113               bo, hdim, h, hdim, zero, nonotmp, hdim)
00114
00115         CALL sgemm('N', 'N', hdim, hdim, hdim, half, &
00116               nonotmp, hdim, bo, hdim, zero, x2hrho,
    hdim)
00117
00118 #endif
00119
00120     ENDIF
00121
00122     ELSE
00123
00124         IF (kbt .GT. 0.0000001) THEN
00125
00126 #ifdef DOUBLEPREC
00127
00128             ! XMAT * XMAT = S^-1
00129
00130             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00131                   xmat, hdim, xmat, hdim, zero, spintmp,
    hdim)
00132
00133             !      Hup * rhoup
00134
00135             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00136                   hup, hdim, rhoup, hdim, zero, nonotmp,
    hdim)
00137
00138             ! (Hup * rhoup) + Hdown*rhdown
00139
00140             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00141                   hdown, hdim, rhdown, hdim, one, nonotmp,
    hdim)
00142
00143
00144             CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00145                   spintmp, hdim, nonotmp, hdim, zero,
    x2hrho, hdim)
00146
00147 #elif defined(SINGLEPREC)
00148
00149             ! XMAT * XMAT = S^-1
00150
00151             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00152                   xmat, hdim, xmat, hdim, zero, spintmp,
    hdim)
00153
00154             !      Hup * rhoup
00155
00156             CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00157                   hup, hdim, rhoup, hdim, zero, nonotmp,
    hdim)

```

```

00158
00159      ! (Hup * rhoup) + Hdown*rhdown
00160
00161      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00162                hdown, hdim, rhdown, hdim, one, nonotmp,
00163                hdim)
00164
00165      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00166                spintmp, hdim, nonotmp, hdim, zero,
00167                x2hrho, hdim)
00168 #endif
00169
00170      ELSE
00171
00172      #ifdef DOUBLEPREC
00173
00174      ! At zero K: tr(rho H rho)
00175
00176      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00177                rhoup, hdim, hup, hdim, zero, nonotmp,
00178                hdim)
00179
00180      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00181                nonotmp, hdim, rhoup, hdim, zero, x2hrho,
00182                hdim)
00183
00184      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00185                rhodown, hdim, hdown, hdim, zero, nonotmp,
00186                hdim)
00187
00188      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00189                nonotmp, hdim, rhodown, hdim, one, x2hrho,
00190                hdim)
00191
00192      #elif defined(SINGLEPREC)
00193
00194      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00195                rhoup, hdim, hup, hdim, zero, nonotmp,
00196                hdim)
00197
00198      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00199                nonotmp, hdim, rhoup, hdim, zero, x2hrho,
00200                hdim)
00201
00202      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00203                rhodown, hdim, hdown, hdim, zero, nonotmp,
00204                hdim)
00205
00206      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00207                nonotmp, hdim, rhodown, hdim, one, x2hrho,
00208                hdim)
00209 #endif
00210
00211      ENDIF
00212
00213      !$OMP PARALLEL DO DEFAULT (NONE) &
00214      !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00215      !$OMP SHARED(CR, BOX, X2HRHO, NOINT, ATELE, ELE1, ELE2) &
00216      !$OMP SHARED(BOND, OVERL, MATINDLIST, BTYPE, BASISTYPE) &
00217      !$OMP PRIVATE(I, J, K, NEWJ, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00218      !$OMP PRIVATE(RIJ, DC, MAGR, INVR, FTMP) &
00219      !$OMP PRIVATE(DSSSDR, DSPSDR, DPSSDR, DPPSDR, PPPSDR, PPSMPPP, PPSUBINVR)&
00220      !$OMP PRIVATE( L, M, N, L2, M2, N2, LM, LN, MN, LMN) &
00221      !$OMP PRIVATE(HSSS, HSPS, HPSS, HPPS, HPPP)&
00222      !$OMP REDUCTION(+:FPUL, VIRPUL)
00223
00224      DO i = 1, nats
00225
00226      basisi = basis(elempointer(i))
00227      indl = matindlist(i)
00228
00229      ! Loop over all neighbors of I
00230
00231      DO newj = 1, totnebtb(i)
00232
00233      j = nebtb(1, newj, i)
00234      pbcj = nebtb(2, newj, i)
00235      pbcj = nebtb(3, newj, i)
00236      pbck = nebtb(4, newj, i)
00237
00238      indj = matindlist(j)
00239      basisj = basis(elempointer(j))

```

```

00235
00236     rij(1) = cr(1,j) + REAL(pbc1)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00237             REAL(pbc3)*BOX(3,1) - CR(1,i)
00238
00239     rij(2) = cr(2,j) + REAL(pbc1)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00240             REAL(pbc3)*BOX(3,2) - CR(2,i)
00241
00242     rij(3) = cr(3,j) + REAL(pbc1)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00243             REAL(pbc3)*BOX(3,3) - CR(3,i)
00244
00245     magr = sqrt(rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3))
00246
00247     invr = one/magr
00248
00249     ftmp = zero
00250
00251     !
00252     ! Direction cosines (DC)
00253     !
00254
00255     dc = rij/magr
00256
00257     l = dc(1)
00258     m = dc(2)
00259     n = dc(3)
00260
00261     IF (basisi .EQ. "s") THEN
00262
00263         IF (basisj .EQ. "s") THEN
00264
00265             DO k = 1, noint
00266                 IF ((atele(i) .EQ. ele1(k) .AND. &
00267                     atele(j) .EQ. ele2(k)) .OR. &
00268                     (atele(i) .EQ. ele2(k) .AND. &
00269                     atele(j) .EQ. ele1(k))) THEN
00270
00271                     IF (btype(k) .EQ. "sss") THEN
00272
00273                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssdr)
00274
00275                     ENDIF
00276
00277                 ENDIF
00278             ENDDO
00279
00280             ftmp = ftmp - x2rho(indi+1, indj+1)* dssdr
00281
00282         ELSEIF (basisj .EQ. "sp") THEN
00283
00284             DO k = 1, noint
00285
00286                 IF ((atele(i) .EQ. ele1(k) .AND. &
00287                     atele(j) .EQ. ele2(k)) .OR. &
00288                     (atele(i) .EQ. ele2(k) .AND. &
00289                     atele(j) .EQ. ele1(k))) THEN
00290
00291                     IF (btype(k) .EQ. "sss") THEN
00292
00293                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssdr)
00294
00295                     ELSEIF (btype(k) .EQ. "sps") THEN
00296
00297                         CALL dunivscale_sub(magr, overl(:,k), dc, hsp, dspdr)
00298
00299                     ENDIF
00300                 ENDIF
00301             ENDDO
00302
00303             l2 = l*l
00304             m2 = m*m
00305             n2 = n*n
00306             lm = l*m
00307             ln = l*n
00308             mn = m*n
00309
00310             ! E_s1,s2
00311
00312             ftmp = ftmp - x2rho(indi+1, indj+1)*dssdr
00313
00314             ! E_s1,x2
00315
00316             ftmp(1) = ftmp(1) - x2rho(indi+1, indj+2) * &
00317                     (1*dspdr(1) + (l2 - one)*invr*hsp)
00318
00319             ftmp(2) = ftmp(2) - x2rho(indi+1, indj+2) * &
00320                     (1*dspdr(2) + lm*invr*hsp)
00321

```

```

00322      ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+2) * &
00323      (l*dspedr(3) + ln*invr*hsps)
00324
00325      ! E_s1,y2
00326
00327      ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+3) * &
00328      (m*dspedr(1) + lm*invr*hsps)
00329
00330      ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+3) * &
00331      (m*dspedr(2) + (m2 - one)*invr*hsps)
00332
00333      ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+3) * &
00334      (m*dspedr(3) + mn*invr*hsps)
00335
00336      ! E_s1,z2
00337
00338      ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+4) * &
00339      (n*dspedr(1) + ln*invr*hsps)
00340
00341      ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+4) * &
00342      (n*dspedr(2) + mn*invr*hsps)
00343
00344      ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+4) * &
00345      (n*dspedr(3) + (n2 - one)*invr*hsps)
00346
00347      ENDIF
00348
00349      ELSEIF (basisi .EQ. "sp") THEN
00350
00351      IF (basisj .EQ. "s") THEN
00352
00353      DO k = 1, noint
00354
00355      IF ((atele(i) .EQ. ele1(k) .AND. &
00356      atele(j) .EQ. ele2(k)) .OR. &
00357      (atele(i) .EQ. ele2(k) .AND. &
00358      atele(j) .EQ. ele1(k))) THEN
00359
00360      IF (btype(k) .EQ. "sss") THEN
00361
00362      CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssedr)
00363
00364      ELSEIF (btype(k) .EQ. "sps") THEN
00365
00366      CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpssedr)
00367
00368      hpss = -hpss
00369      dpssedr = -dpssedr
00370
00371      ENDIF
00372      ENDIF
00373      ENDDO
00374
00375      l2 = l*l
00376      m2 = m*m
00377      n2 = n*n
00378      lm = l*m
00379      ln = l*n
00380      mn = m*n
00381
00382      ! E_s1,s2
00383
00384      ftmp = ftmp - dssedr*x2hrho(indi+1, indj+1)
00385
00386      ! E_x1,s2
00387
00388      ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+1) * &
00389      (l*dpssedr(1) + (l2 - one)*invr*hpss)
00390
00391      ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+1) * &
00392      (l*dpssedr(2) + lm*invr*hpss)
00393
00394      ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+1) * &
00395      (l*dpssedr(3) + ln*invr*hpss)
00396
00397      ! E_y1,s2
00398
00399      ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+1) * &
00400      (m*dpssedr(1) + lm*invr*hpss)
00401
00402      ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+1) * &
00403      (m*dpssedr(2) + (m2 - one)*invr*hpss)
00404
00405      ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+1) * &
00406      (m*dpssedr(3) + mn*invr*hpss)
00407
00408      ! E_z1,s2

```

```

00409
00410         ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+1) * &
00411             (n*dpssdr(1) + ln*invr*hpss)
00412
00413         ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+1) * &
00414             (n*dpssdr(2) + mn*invr*hpss)
00415
00416         ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+1) * &
00417             (n*dpssdr(3) + (n2 - one)*invr*hpss)
00418
00419     ELSEIF (basisj .EQ. "sp") THEN
00420
00421     IF (atele(i) .EQ. atele(j)) THEN
00422
00423         DO k = 1, noint
00424
00425             IF (atele(i) .EQ. ele1(k) .AND. &
00426                 atele(j) .EQ. ele2(k)) THEN
00427
00428                 IF (btype(k) .EQ. "sss") THEN
00429
00430                     CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00431
00432                 ELSEIF (btype(k) .EQ. "sps") THEN
00433
00434                     CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00435
00436                     dpssdr = -dpsssdr
00437
00438                     hpss = -hspss
00439
00440                 ELSEIF (btype(k) .EQ. "pps") THEN
00441
00442                     CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppssdr)
00443
00444                 ELSEIF (btype(k) .EQ. "ppp") THEN
00445
00446                     CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppsdr)
00447
00448                 ENDIF
00449             ENDIF
00450         ENDDO
00451
00452     ELSEIF (atele(i) .NE. atele(j)) THEN
00453
00454         DO k = 1, noint
00455
00456             IF (atele(i) .EQ. ele1(k) .AND. &
00457                 atele(j) .EQ. ele2(k)) THEN
00458
00459                 IF (btype(k) .EQ. "sss") THEN
00460
00461                     CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00462
00463                 ELSEIF (btype(k) .EQ. "sps") THEN
00464
00465                     CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00466
00467                 ELSEIF (btype(k) .EQ. "pps") THEN
00468
00469                     CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppssdr)
00470
00471                 ELSEIF (btype(k) .EQ. "ppp") THEN
00472
00473                     CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppsdr)
00474
00475                 ENDIF
00476
00477             ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00478                 atele(j) .EQ. ele1(k)) THEN
00479
00480                 IF (btype(k) .EQ. "sss") THEN
00481
00482                     CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00483
00484                 ELSEIF (btype(k) .EQ. "sps") THEN
00485
00486                     CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00487
00488                     dpssdr = -dpsssdr
00489                     hpss = -hspss
00490
00491                 ELSEIF (btype(k) .EQ. "pps") THEN
00492
00493                     CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppssdr)
00494
00495                 ELSEIF (btype(k) .EQ. "ppp") THEN

```



```

00496
00497         CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00498
00499         ENDIF
00500
00501         ENDIF
00502     ENDDO
00503
00504 ENDIF
00505
00506 ppsmppp = hpps - hppp
00507 ppsubinvr = ppsmppp * invr
00508
00509 l2 = l*l
00510 m2 = m*m
00511 n2 = n*n
00512 lm = l*m
00513 ln = l*n
00514 mn = m*n
00515 lmn = lm*n
00516
00517 ! E_s1,s2
00518
00519 ftmp = ftmp - dsssdr*x2hrho(indi+1, indj+1)
00520
00521 ! E_s1,x2
00522
00523 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+2) * &
00524     (l*dspssdr(1) + (l2 - one)*invr*hsps)
00525
00526 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+2) * &
00527     (l*dspssdr(2) + lm*invr*hsps)
00528
00529 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+2) * &
00530     (l*dspssdr(3) + ln*invr*hsps)
00531
00532 ! E_s1,y2
00533
00534 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+3) * &
00535     (m*dspssdr(1) + lm*invr*hsps)
00536
00537 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+3) * &
00538     (m*dspssdr(2) + (m2 - one)*invr*hsps)
00539
00540 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+3) * &
00541     (m*dspssdr(3) + mn*invr*hsps)
00542
00543 ! E_s1,z2
00544
00545 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+4) * &
00546     (n*dspssdr(1) + ln*invr*hsps)
00547
00548 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+4) * &
00549     (n*dspssdr(2) + mn*invr*hsps)
00550
00551 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+4) * &
00552     (n*dspssdr(3) + (n2 - one)*invr*hsps)
00553
00554 ! E_x1,s2
00555
00556 ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+1) * &
00557     (l*dpssdr(1) + (l2 - one)*invr*hpss)
00558
00559 ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+1) * &
00560     (l*dpssdr(2) + lm*invr*hpss)
00561
00562 ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+1) * &
00563     (l*dpssdr(3) + ln*invr*hpss)
00564
00565 ! E_x1,x2
00566
00567 ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+2) * &
00568     (l2*dppsdr(1) + (one - l2)*dpppdr(1) + &
00569     two*l*(l2 - one)*ppsubinvr)
00570
00571 ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+2) * &
00572     (l2*dppsdr(2) + (one - l2)*dpppdr(2) + &
00573     two*l2*m*ppsubinvr)
00574
00575 ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+2) * &
00576     (l2*dppsdr(3) + (one - l2)*dpppdr(3) + &
00577     two*l2*n*ppsubinvr)
00578
00579 ! E_x1,y2
00580
00581 ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+3) * &
00582     (lm*(dpssdr(1) - dpppdr(1)) + &

```

```

00583         m*(two*12 - one)*ppsubinvr)
00584
00585     ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+3) * &
00586         (lm*(dppsdr(2) - dpppdr(2)) + &
00587         1*(two*m2 - one)*ppsubinvr)
00588
00589     ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+3) * &
00590         (lm*(dppsdr(3) - dpppdr(3)) + &
00591         two*lmn*ppsubinvr)
00592
00593     ! E_x1,z2
00594
00595     ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+4) * &
00596         (ln*(dppsdr(1) - dpppdr(1)) + &
00597         n*(two*12 - one)*ppsubinvr)
00598
00599     ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+4) * &
00600         (ln*(dppsdr(2) - dpppdr(2)) + &
00601         two*lmn*ppsubinvr)
00602
00603     ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+4) * &
00604         (ln*(dppsdr(3) - dpppdr(3)) + &
00605         1*(two*n2 - one)*ppsubinvr)
00606
00607     ! E_y1,s2
00608
00609     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+1) * &
00610         (m*dppsdr(1) + lm*invr*hpss)
00611
00612     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+1) * &
00613         (m*dppsdr(2) + (m2 - one)*invr*hpss)
00614
00615     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+1) * &
00616         (m*dppsdr(3) + mn*invr*hpss)
00617
00618     ! E_y1,x2
00619
00620     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+2) * &
00621         (lm*(dppsdr(1) - dpppdr(1)) + &
00622         m*(two*12 - one)*ppsubinvr)
00623
00624     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+2) * &
00625         (lm*(dppsdr(2) - dpppdr(2)) + &
00626         1*(two*m2 - one)*ppsubinvr)
00627
00628     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+2) * &
00629         (lm*(dppsdr(3) - dpppdr(3)) + &
00630         two*lmn*ppsubinvr)
00631
00632     ! E_y1,y2
00633
00634     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+3) * &
00635         (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00636         two*1*m2*ppsubinvr)
00637
00638     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+3) * &
00639         (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00640         two*m*(m2 - one)*ppsubinvr)
00641
00642     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+3) * &
00643         (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00644         two*n*m2*ppsubinvr)
00645
00646     ! E_y1,z2
00647
00648     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+4) * &
00649         (mn*(dppsdr(1) - dpppdr(1)) + &
00650         two*lmn*ppsubinvr)
00651
00652     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+4) * &
00653         (mn*(dppsdr(2) - dpppdr(2)) + &
00654         n*(two*m2 - one)*ppsubinvr)
00655
00656     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+4) * &
00657         (mn*(dppsdr(3) - dpppdr(3)) + &
00658         m*(two*n2 - one)*ppsubinvr)
00659
00660     ! E_z1,s2
00661
00662     ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+1) * &
00663         (n*dppsdr(1) + ln*invr*hpss)
00664
00665     ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+1) * &
00666         (n*dppsdr(2) + mn*invr*hpss)
00667
00668     ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+1) * &
00669         (n*dppsdr(3) + (n2 - one)*invr*hpss)

```

```

00670
00671      ! E_z1,x2
00672
00673      ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+2) * &
00674        (ln*(dppsdr(1) - dpppdr(1)) + &
00675          n*(two*12 - one)*ppsubinvr)
00676
00677      ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+2) * &
00678        (ln*(dppsdr(2) - dpppdr(2)) + &
00679          two*lmn*ppsubinvr)
00680
00681      ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+2) * &
00682        (ln*(dppsdr(3) - dpppdr(3)) + &
00683          l*(two*n2 - one)*ppsubinvr)
00684
00685      ! E_z1,y2
00686
00687      ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+3) * &
00688        (mn*(dppsdr(1) - dpppdr(1)) + &
00689          two*lmn*ppsubinvr)
00690
00691      ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+3) * &
00692        (mn*(dppsdr(2) - dpppdr(2)) + &
00693          n*(two*m2 - one)*ppsubinvr)
00694
00695      ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+3) * &
00696        (mn*(dppsdr(3) - dpppdr(3)) + &
00697          m*(two*n2 - one)*ppsubinvr)
00698
00699      ! E_z1,z2
00700
00701      ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+4) * &
00702        (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00703          two*1*n2*ppsubinvr)
00704
00705      ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+4) * &
00706        (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00707          two*m*n2*ppsubinvr)
00708
00709      ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+4) * &
00710        (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00711          two*n*(n2 - one)*ppsubinvr)
00712
00713      ENDIF
00714
00715      ENDIF
00716
00717      fpul(1,i) = fpul(1,i) + ftmp(1)
00718      fpul(2,i) = fpul(2,i) + ftmp(2)
00719      fpul(3,i) = fpul(3,i) + ftmp(3)
00720
00721
00722      ! with the factor of 2...
00723
00724      virpul(1) = virpul(1) + rij(1)*ftmp(1)
00725      virpul(2) = virpul(2) + rij(2)*ftmp(2)
00726      virpul(3) = virpul(3) + rij(3)*ftmp(3)
00727      virpul(4) = virpul(4) + rij(1)*ftmp(2)
00728      virpul(5) = virpul(5) + rij(2)*ftmp(3)
00729      virpul(6) = virpul(6) + rij(3)*ftmp(1)
00730
00731      ENDDO
00732
00733      ENDDO
00734
00735      !$OMP END PARALLEL DO
00736
00737      RETURN
00738
00739  END SUBROUTINE pulay_sp

```

8.327 pulay_spprogress.f90 File Reference

Functions/Subroutines

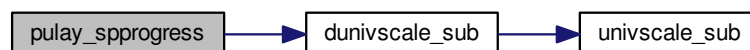
- subroutine [pulay_spprogress](#)

8.327.1 Function/Subroutine Documentation

8.327.1.1 subroutine pulay_spprogress ()

Definition at line 23 of file [pulay_spprogress.f90](#).

Here is the call graph for this function:



```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License.
```

```

00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE pulay_spprogress
00023
00024 #ifdef PROGRESSON
00025
00026   USE constants_mod
00027   USE setuparray
00028   USE univarray
00029   USE nonoarray
00030   USE neblistarray
00031   USE spinarray
00032   USE virialarray
00033   USE myprecision
00034   USE latteparser_latte_mod
00035   USE genxprogress
00036   USE prg_pulaycomponent_mod
00037   USE bml
00038
00039   IMPLICIT NONE
00040
00041   INTEGER :: I, J, K, KK, INDI, INDJ
00042   INTEGER :: NEWJ
00043   INTEGER :: PBCI, PBCJ, PBCK
00044   REAL(LATTEPREC) :: HSSS, HSPS, HPSS, HPPS, HPPP
00045   REAL(LATTEPREC) :: RIJ(3), DC(3), SCLGSP, DGSPDR(3)
00046   REAL(LATTEPREC) :: L, M, N, L2, M2, N2, LM, LN, MN, LMN
00047   REAL(LATTEPREC) :: DSSSDR(3), DSPSDR(3), DPSSDR(3), DPPSDR(3), DPPPDR(3)
00048   REAL(LATTEPREC) :: MAGR, INVR, FTMP(3)
00049   REAL(LATTEPREC) :: PPSMPPP, PPSUBINVR
00050   CHARACTER(LEN=2) :: BASISI, BASISJ
00051   TYPE(bml_matrix_t) :: H_BML, BO_BML, X2HRHO_BML
00052   TYPE(bml_matrix_t) :: HUP_BML, HDOWN_BML, RHOUP_BML, RHODOWN_BML, X2HRHOUP_BML
00053
00054   IF (existerror) RETURN
00055
00056   indi = 0
00057
00058   fpul = zero
00059
00060   virpul = zero
00061
00062   ! We first have to make the matrix  $S^{-1} H \rho = X^2 H \rho$ 
00063
00064   IF (spinon .EQ. 0) THEN
00065
00066     !! Convert Hamiltonian to bml format
00067     CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00068       latteprec, hdim, lt%MDIM, h_bml)
00069     CALL bml_import_from_dense(lt%BML_TYPE, &
00070       h, h_bml, zero, lt%MDIM)
00071
00072     !! Convert the density matrix to bml format
00073     CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00074       latteprec, hdim, lt%MDIM, bo_bml)
00075     CALL bml_import_from_dense(lt%BML_TYPE, &
00076       bo, bo_bml, zero, lt%MDIM)
00077
00078     !! Allocate a bml matrix for the Pulay component matrix
00079     CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00080       latteprec, hdim, lt%MDIM, x2hrho_bml)
00081
00082     IF (kbt .GT. 0.0000001) THEN
00083
00084       CALL prg_pulaycomponentt(bo_bml, h_bml, zmat_bml, x2hrho_bml, lt%THRESHOLD &
00085         &, lt%MDIM, lt%BML_TYPE, verbose)
00086
00087     ELSE
00088
00089       !  $T_e = 0 : F_p = 2\text{Tr}[\rho H \rho dS/dR]$ 
00090
00091       ! Be careful - we're working with  $bo = 2\rho$ , so we need
00092       ! the factor of  $1/2...$ 
00093
00094       CALL prg_pulaycomponent0(bo_bml, h_bml, x2hrho_bml, lt%THRESHOLD &
00095         &, lt%MDIM, lt%BML_TYPE, verbose)
00096
00097     ENDIF
00098
00099     CALL bml_deallocate(bo_bml)
00100     CALL bml_deallocate(h_bml)
00101     CALL bml_export_to_dense(x2hrho_bml, x2hrho)
00102

```

```

00103      CALL bml_deallocate(x2hrho_bml)
00104
00105      ELSE
00106
00107      !! Allocate a bml matrix for half of the Pulay component matrix
00108      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00109          latteprec, hdim, lt%MDIM, x2hrhoup_bml)
00110
00111      !! Convert Hamiltonian (Up) to bml format
00112      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00113          latteprec, hdim, lt%MDIM, hup_bml)
00114      CALL bml_import_from_dense(lt%BML_TYPE, &
00115          hup, hup_bml, zero, lt%MDIM)
00116      !! Convert the density matrix (RhoUp) to bml format
00117      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00118          latteprec, hdim, lt%MDIM, rhoup_bml)
00119      CALL bml_import_from_dense(lt%BML_TYPE, &
00120          rhoup, rhoup_bml, zero, lt%MDIM)
00121
00122      IF (kbt .GT. 0.0000001) THEN
00123          CALL prg_pulaycomponentt(rhoup_bml, hup_bml, zmat_bml, x2hrhoup_bml,
00124              lt%THRESHOLD &
00125              &, lt%MDIM, lt%BML_TYPE, verbose)
00126      ELSE
00127          CALL prg_pulaycomponent0(rhoup_bml, hup_bml, x2hrhoup_bml, lt%THRESHOLD &
00128              &, lt%MDIM, lt%BML_TYPE, verbose)
00129      ENDIF
00130
00131      CALL bml_deallocate(rhoup_bml)
00132      CALL bml_deallocate(hup_bml)
00133
00134      !! Convert Hamiltonian (Down) to bml format
00135      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00136          latteprec, hdim, lt%MDIM, hdown_bml)
00137      CALL bml_import_from_dense(lt%BML_TYPE, &
00138          hdown, hdown_bml, zero, lt%MDIM)
00139      !! Convert the density matrix (RhoDown) to bml format
00140      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00141          latteprec, hdim, lt%MDIM, rhodown_bml)
00142      CALL bml_import_from_dense(lt%BML_TYPE, &
00143          rhodown, rhodown_bml, zero, lt%MDIM)
00144
00145      !! Allocate a bml matrix for the other half pulay component matrix
00146      CALL bml_zero_matrix(lt%BML_TYPE, bml_element_real, &
00147          latteprec, hdim, lt%MDIM, x2hrho_bml)
00148
00149      IF (kbt .GT. 0.0000001) THEN
00150          CALL prg_pulaycomponentt(rhodown_bml, hdown_bml, zmat_bml, x2hrho_bml,
00151              lt%THRESHOLD &
00152              &, lt%MDIM, lt%BML_TYPE, verbose)
00153      ELSE
00154          CALL prg_pulaycomponent0(rhodown_bml, hdown_bml, x2hrho_bml, lt%THRESHOLD &
00155              &, lt%MDIM, lt%BML_TYPE, verbose)
00156      ENDIF
00157
00158      CALL bml_deallocate(rhodown_bml)
00159      CALL bml_deallocate(hdown_bml)
00160
00161      CALL bml_add_deprecated(2.0d0, x2hrho_bml, 2.0d0, x2hrhoup_bml)
00162
00163      CALL bml_deallocate(x2hrhoup_bml)
00164      CALL bml_export_to_dense(x2hrho_bml, x2hrho)
00165      CALL bml_deallocate(x2hrho_bml)
00166
00167      ENDIF
00168
00169      !$OMP PARALLEL DO DEFAULT (NONE) &
00170      !$OMP SHARED(NATS, BASIS, ELEMPONTER, TOTNEBTB, NEBTB) &
00171      !$OMP SHARED(CR, BOX, X2HRHO, NOINT, ATELE, ELE1, ELE2) &
00172      !$OMP SHARED(BOND, OVERL, MATINDLIST, BTYPE, BASISTYPE) &
00173      !$OMP PRIVATE(I, J, K, NEWJ, BASISI, BASISJ, INDI, INDJ, PBCI, PBCJ, PBCK) &
00174      !$OMP PRIVATE(RIJ, DC, MAGR, INVR, FTMP) &
00175      !$OMP PRIVATE(DSSSDR, DSPSDR, DPSSDR, DPPSDR, DPPPDR, PPSMPPP, PPSUBINVR) &
00176      !$OMP PRIVATE(L, M, N, L2, M2, N2, LM, LN, MN, LMN) &
00177      !$OMP PRIVATE(HSSS, HSPS, HPSS, HPPS, HPPP) &
00178      !$OMP REDUCTION(+:FPUL, VIRPUL)
00179
00180      DO i = 1, nats
00181          basisi = basis(elempointer(i))
00182          indi = matindlist(i)
00183
00184          ! Loop over all neighbors of I
00185          DO newj = 1, totnebtb(i)
00186              j = nebtb(1, newj, i)
00187

```

```

00188      pbci = nebtb(2, newj, i)
00189      pbcj = nebtb(3, newj, i)
00190      pbck = nebtb(4, newj, i)
00191
00192      indj = matindlist(j)
00193      basisj = basis(elempointer(j))
00194
00195      rij(1) = cr(1,j) + REAL(pbci)*BOX(1,1) + REAL(pbcj)*BOX(2,1) + &
00196              REAL(pbck)*BOX(3,1) - CR(1,i)
00197
00198      rij(2) = cr(2,j) + REAL(pbci)*BOX(1,2) + REAL(pbcj)*BOX(2,2) + &
00199              REAL(pbck)*BOX(3,2) - CR(2,i)
00200
00201      rij(3) = cr(3,j) + REAL(pbci)*BOX(1,3) + REAL(pbcj)*BOX(2,3) + &
00202              REAL(pbck)*BOX(3,3) - CR(3,i)
00203
00204      magr = sqrt(rij(1)*rij(1) + rij(2)*rij(2) + rij(3)*rij(3))
00205
00206      invr = one/magr
00207
00208      ftmp = zero
00209
00210      !
00211      ! Direction cosines (DC)
00212      !
00213
00214      dc = rij/magr
00215
00216      l = dc(1)
00217      m = dc(2)
00218      n = dc(3)
00219
00220      IF (basisi .EQ. "s") THEN
00221
00222          IF (basisj .EQ. "s") THEN
00223
00224              DO k = 1, noint
00225                  IF ((atele(i) .EQ. ele1(k) .AND. &
00226                      atele(j) .EQ. ele2(k)) .OR. &
00227                      (atele(i) .EQ. ele2(k) .AND. &
00228                      atele(j) .EQ. ele1(k))) THEN
00229
00230                      IF (btype(k) .EQ. "sss") THEN
00231
00232                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00233
00234                      ENDIF
00235
00236                  ENDIF
00237              ENDDO
00238
00239              ftmp = ftmp - x2hrho(indi+1, indj+1)* dssssdr
00240
00241          ELSEIF (basisj .EQ. "sp") THEN
00242
00243              DO k = 1, noint
00244
00245                  IF ((atele(i) .EQ. ele1(k) .AND. &
00246                      atele(j) .EQ. ele2(k)) .OR. &
00247                      (atele(i) .EQ. ele2(k) .AND. &
00248                      atele(j) .EQ. ele1(k))) THEN
00249
00250                      IF (btype(k) .EQ. "sss") THEN
00251
00252                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00253
00254                      ELSEIF (btype(k) .EQ. "sps") THEN
00255
00256                          CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dspssdr)
00257
00258                      ENDIF
00259                  ENDIF
00260              ENDDO
00261
00262              l2 = l*l
00263              m2 = m*m
00264              n2 = n*n
00265              lm = l*m
00266              ln = l*n
00267              mn = m*n
00268
00269              ! E_s1,s2
00270
00271              ftmp = ftmp - x2hrho(indi+1, indj+1)*dssssdr
00272
00273              ! E_s1,x2
00274

```



```

00275         ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+2) * &
00276             (l*dspedr(1) + (l2 - one)*invr*hsps)
00277
00278         ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+2) * &
00279             (l*dspedr(2) + lm*invr*hsps)
00280
00281         ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+2) * &
00282             (l*dspedr(3) + ln*invr*hsps)
00283
00284         ! E_s1,y2
00285
00286         ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+3) * &
00287             (m*dspedr(1) + lm*invr*hsps)
00288
00289         ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+3) * &
00290             (m*dspedr(2) + (m2 - one)*invr*hsps)
00291
00292         ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+3) * &
00293             (m*dspedr(3) + mn*invr*hsps)
00294
00295         ! E_s1,z2
00296
00297         ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+4) * &
00298             (n*dspedr(1) + ln*invr*hsps)
00299
00300         ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+4) * &
00301             (n*dspedr(2) + mn*invr*hsps)
00302
00303         ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+4) * &
00304             (n*dspedr(3) + (n2 - one)*invr*hsps)
00305
00306     ENDIF
00307
00308     ELSEIF (basisi .EQ. "sp") THEN
00309
00310         IF (basisj .EQ. "s") THEN
00311
00312             DO k = 1, noint
00313
00314                 IF ((atele(i) .EQ. ele1(k) .AND. &
00315                     atele(j) .EQ. ele2(k)) .OR. &
00316                     (atele(i) .EQ. ele2(k) .AND. &
00317                     atele(j) .EQ. ele1(k))) THEN
00318
00319                     IF (btype(k) .EQ. "sss") THEN
00320
00321                         CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssedr)
00322
00323                     ELSEIF (btype(k) .EQ. "sps") THEN
00324
00325                         CALL dunivscale_sub(magr, overl(:,k), dc, hpss, dpssedr)
00326
00327                         hpss = -hpss
00328                         dpssedr = -dpssedr
00329
00330                     ENDIF
00331                 ENDIF
00332             ENDDO
00333
00334             l2 = l*l
00335             m2 = m*m
00336             n2 = n*n
00337             lm = l*m
00338             ln = l*n
00339             mn = m*n
00340
00341             ! E_s1,s2
00342
00343             ftmp = ftmp - dssedr*x2hrho(indi+1, indj+1)
00344
00345             ! E_x1,s2
00346
00347             ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+1) * &
00348                 (l*dpssedr(1) + (l2 - one)*invr*hpss)
00349
00350             ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+1) * &
00351                 (l*dpssedr(2) + lm*invr*hpss)
00352
00353             ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+1) * &
00354                 (l*dpssedr(3) + ln*invr*hpss)
00355
00356             ! E_y1,s2
00357
00358             ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+1) * &
00359                 (m*dpssedr(1) + lm*invr*hpss)
00360
00361             ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+1) * &

```

```

00362          (m*dpssdr(2) + (m2 - one)*invr*hpss)
00363
00364      ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+1) * &
00365          (m*dpssdr(3) + mn*invr*hpss)
00366
00367      ! E_z1,s2
00368
00369      ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+1) * &
00370          (n*dpssdr(1) + ln*invr*hpss)
00371
00372      ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+1) * &
00373          (n*dpssdr(2) + mn*invr*hpss)
00374
00375      ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+1) * &
00376          (n*dpssdr(3) + (n2 - one)*invr*hpss)
00377
00378      ELSEIF (basisj .EQ. "sp") THEN
00379
00380          IF (atele(i) .EQ. atele(j)) THEN
00381
00382              DO k = 1, noint
00383
00384                  IF (atele(i) .EQ. ele1(k) .AND. &
00385                      atele(j) .EQ. ele2(k)) THEN
00386
00387                      IF (btype(k) .EQ. "sss") THEN
00388
00389                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00390
00391                      ELSEIF (btype(k) .EQ. "sps") THEN
00392
00393                          CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00394
00395                          dpssdr = -dpsssdr
00396
00397                          hpss = -hspss
00398
00399                      ELSEIF (btype(k) .EQ. "pps") THEN
00400
00401                          CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppssdr)
00402
00403                      ELSEIF (btype(k) .EQ. "ppp") THEN
00404
00405                          CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppsdr)
00406
00407                      ENDIF
00408                  ENDIF
00409              ENDDO
00410
00411          ELSEIF (atele(i) .NE. atele(j)) THEN
00412
00413              DO k = 1, noint
00414
00415                  IF (atele(i) .EQ. ele1(k) .AND. &
00416                      atele(j) .EQ. ele2(k)) THEN
00417
00418                      IF (btype(k) .EQ. "sss") THEN
00419
00420                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00421
00422                      ELSEIF (btype(k) .EQ. "sps") THEN
00423
00424                          CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00425
00426                      ELSEIF (btype(k) .EQ. "pps") THEN
00427
00428                          CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppssdr)
00429
00430                      ELSEIF (btype(k) .EQ. "ppp") THEN
00431
00432                          CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppsdr)
00433
00434                      ENDIF
00435
00436                  ELSEIF (atele(i) .EQ. ele2(k) .AND. &
00437                      atele(j) .EQ. ele1(k)) THEN
00438
00439                      IF (btype(k) .EQ. "sss") THEN
00440
00441                          CALL dunivscale_sub(magr, overl(:,k), dc, hsss, dssssdr)
00442
00443                      ELSEIF (btype(k) .EQ. "sps") THEN
00444
00445                          CALL dunivscale_sub(magr, overl(:,k), dc, hspss, dpsssdr)
00446
00447                          dpssdr = -dpsssdr
00448                          hpss = -hspss

```

```

00449
00450         ELSEIF (btype(k) .EQ. "pps") THEN
00451
00452             CALL dunivscale_sub(magr, overl(:,k), dc, hpps, dppsdr)
00453
00454         ELSEIF (btype(k) .EQ. "ppp") THEN
00455
00456             CALL dunivscale_sub(magr, overl(:,k), dc, hppp, dpppdr)
00457
00458         ENDIF
00459
00460     ENDIF
00461 ENDDO
00462
00463 ENDIF
00464
00465 ppsmppp = hpps - hppp
00466 ppsubinvr = ppsmppp * invr
00467
00468 l2 = l*l
00469 m2 = m*m
00470 n2 = n*n
00471 lm = l*m
00472 ln = l*n
00473 mn = m*n
00474 lmn = lm*n
00475
00476 ! E_s1,s2
00477
00478 ftmp = ftmp - dsssdr*x2hrho(indi+1, indj+1)
00479
00480 ! E_s1,x2
00481
00482 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+2) * &
00483     (l*dspssdr(1) + (l2 - one)*invr*hsps)
00484
00485 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+2) * &
00486     (l*dspssdr(2) + lm*invr*hsps)
00487
00488 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+2) * &
00489     (l*dspssdr(3) + ln*invr*hsps)
00490
00491 ! E_s1,y2
00492
00493 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+3) * &
00494     (m*dspssdr(1) + lm*invr*hsps)
00495
00496 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+3) * &
00497     (m*dspssdr(2) + (m2 - one)*invr*hsps)
00498
00499 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+3) * &
00500     (m*dspssdr(3) + mn*invr*hsps)
00501
00502 ! E_s1,z2
00503
00504 ftmp(1) = ftmp(1) - x2hrho(indi+1, indj+4) * &
00505     (n*dspssdr(1) + ln*invr*hsps)
00506
00507 ftmp(2) = ftmp(2) - x2hrho(indi+1, indj+4) * &
00508     (n*dspssdr(2) + mn*invr*hsps)
00509
00510 ftmp(3) = ftmp(3) - x2hrho(indi+1, indj+4) * &
00511     (n*dspssdr(3) + (n2 - one)*invr*hsps)
00512
00513 ! E_x1,s2
00514
00515 ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+1) * &
00516     (l*dpssdr(1) + (l2 - one)*invr*hpss)
00517
00518 ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+1) * &
00519     (l*dpssdr(2) + lm*invr*hpss)
00520
00521 ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+1) * &
00522     (l*dpssdr(3) + ln*invr*hpss)
00523
00524 ! E_x1,x2
00525
00526 ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+2) * &
00527     (l2*dppsdr(1) + (one - l2)*dpppdr(1) + &
00528     two*l*(l2 - one)*ppsubinvr)
00529
00530 ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+2) * &
00531     (l2*dppsdr(2) + (one - l2)*dpppdr(2) + &
00532     two*l2*m*ppsubinvr)
00533
00534 ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+2) * &
00535     (l2*dppsdr(3) + (one - l2)*dpppdr(3) + &

```

```

00536         two*12*n*ppsubinvr)
00537
00538     ! E_x1,y2
00539
00540     ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+3) * &
00541         (lm*(dppsdr(1) - dpppdr(1)) + &
00542         m*(two*12 - one)*ppsubinvr)
00543
00544     ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+3) * &
00545         (lm*(dppsdr(2) - dpppdr(2)) + &
00546         l*(two*m2 - one)*ppsubinvr)
00547
00548     ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+3) * &
00549         (lm*(dppsdr(3) - dpppdr(3)) + &
00550         two*lmn*ppsubinvr)
00551
00552     ! E_x1,z2
00553
00554     ftmp(1) = ftmp(1) - x2hrho(indi+2, indj+4) * &
00555         (ln*(dppsdr(1) - dpppdr(1)) + &
00556         n*(two*12 - one)*ppsubinvr)
00557
00558     ftmp(2) = ftmp(2) - x2hrho(indi+2, indj+4) * &
00559         (ln*(dppsdr(2) - dpppdr(2)) + &
00560         two*lmn*ppsubinvr)
00561
00562     ftmp(3) = ftmp(3) - x2hrho(indi+2, indj+4) * &
00563         (ln*(dppsdr(3) - dpppdr(3)) + &
00564         l*(two*n2 - one)*ppsubinvr)
00565
00566     ! E_y1,s2
00567
00568     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+1) * &
00569         (m*dppsdr(1) + lm*invr*hpss)
00570
00571     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+1) * &
00572         (m*dppsdr(2) + (m2 - one)*invr*hpss)
00573
00574     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+1) * &
00575         (m*dppsdr(3) + mn*invr*hpss)
00576
00577     ! E_y1,x2
00578
00579     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+2) * &
00580         (lm*(dppsdr(1) - dpppdr(1)) + &
00581         m*(two*12 - one)*ppsubinvr)
00582
00583     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+2) * &
00584         (lm*(dppsdr(2) - dpppdr(2)) + &
00585         l*(two*m2 - one)*ppsubinvr)
00586
00587     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+2) * &
00588         (lm*(dppsdr(3) - dpppdr(3)) + &
00589         two*lmn*ppsubinvr)
00590
00591     ! E_y1,y2
00592
00593     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+3) * &
00594         (m2*dppsdr(1) + (one - m2)*dpppdr(1) + &
00595         two*1*m2*ppsubinvr)
00596
00597     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+3) * &
00598         (m2*dppsdr(2) + (one - m2)*dpppdr(2) + &
00599         two*m*(m2 - one)*ppsubinvr)
00600
00601     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+3) * &
00602         (m2*dppsdr(3) + (one - m2)*dpppdr(3) + &
00603         two*n*m2*ppsubinvr)
00604
00605     ! E_y1,z2
00606
00607     ftmp(1) = ftmp(1) - x2hrho(indi+3, indj+4) * &
00608         (mn*(dppsdr(1) - dpppdr(1)) + &
00609         two*lmn*ppsubinvr)
00610
00611     ftmp(2) = ftmp(2) - x2hrho(indi+3, indj+4) * &
00612         (mn*(dppsdr(2) - dpppdr(2)) + &
00613         n*(two*m2 - one)*ppsubinvr)
00614
00615     ftmp(3) = ftmp(3) - x2hrho(indi+3, indj+4) * &
00616         (mn*(dppsdr(3) - dpppdr(3)) + &
00617         m*(two*n2 - one)*ppsubinvr)
00618
00619     ! E_z1,s2
00620
00621     ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+1) * &
00622         (n*dppsdr(1) + ln*invr*hpss)

```

```

00623
00624         ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+1) * &
00625             (n*dppsdr(2) + mn*invr*hpss)
00626
00627         ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+1) * &
00628             (n*dppsdr(3) + (n2 - one)*invr*hpss)
00629
00630         ! E_z1,x2
00631
00632         ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+2) * &
00633             (ln*(dppsdr(1) - dpppdr(1)) + &
00634             n*(two*12 - one)*ppsubinvr)
00635
00636         ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+2) * &
00637             (ln*(dppsdr(2) - dpppdr(2)) + &
00638             two*lmn*ppsubinvr)
00639
00640         ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+2) * &
00641             (ln*(dppsdr(3) - dpppdr(3)) + &
00642             l*(two*n2 - one)*ppsubinvr)
00643
00644         ! E_z1,y2
00645
00646         ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+3) * &
00647             (mn*(dppsdr(1) - dpppdr(1)) + &
00648             two*lmn*ppsubinvr)
00649
00650         ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+3) * &
00651             (mn*(dppsdr(2) - dpppdr(2)) + &
00652             n*(two*m2 - one)*ppsubinvr)
00653
00654         ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+3) * &
00655             (mn*(dppsdr(3) - dpppdr(3)) + &
00656             m*(two*n2 - one)*ppsubinvr)
00657
00658         ! E_z1,z2
00659
00660         ftmp(1) = ftmp(1) - x2hrho(indi+4, indj+4) * &
00661             (n2*dppsdr(1) + (one - n2)*dpppdr(1) + &
00662             two*1*n2*ppsubinvr)
00663
00664         ftmp(2) = ftmp(2) - x2hrho(indi+4, indj+4) * &
00665             (n2*dppsdr(2) + (one - n2)*dpppdr(2) + &
00666             two*m*n2*ppsubinvr)
00667
00668         ftmp(3) = ftmp(3) - x2hrho(indi+4, indj+4) * &
00669             (n2*dppsdr(3) + (one - n2)*dpppdr(3) + &
00670             two*n*(n2 - one)*ppsubinvr)
00671
00672         ENDIF
00673
00674     ENDIF
00675
00676
00677     fpul(1,i) = fpul(1,i) + ftmp(1)
00678     fpul(2,i) = fpul(2,i) + ftmp(2)
00679     fpul(3,i) = fpul(3,i) + ftmp(3)
00680
00681     ! with the factor of 2...
00682
00683     virpul(1) = virpul(1) + rij(1)*ftmp(1)
00684     virpul(2) = virpul(2) + rij(2)*ftmp(2)
00685     virpul(3) = virpul(3) + rij(3)*ftmp(3)
00686     virpul(4) = virpul(4) + rij(1)*ftmp(2)
00687     virpul(5) = virpul(5) + rij(2)*ftmp(3)
00688     virpul(6) = virpul(6) + rij(3)*ftmp(1)
00689
00690     ENDDO
00691
00692     ENDDO
00693
00694     !$OMP END PARALLEL DO
00695
00696 #endif
00697
00698     RETURN
00699
00700 END SUBROUTINE pulay_spprogress

```

8.329 purearray.f90 File Reference

Modules

- module [purearray](#)

Variables

- real(latteprec) [purearray::beta0](#)
- integer [purearray::maxdim](#)
- integer [purearray::nr_sp2_iter](#)
- integer, dimension(100) [purearray::pp](#)
- integer, dimension(:), allocatable [purearray::signlist](#)
- real(latteprec), dimension(:,:), allocatable [purearray::twoxx2](#)
- real(latteprec), dimension(100) [purearray::vv](#)
- real(latteprec), dimension(:,:), allocatable [purearray::x2](#)
- real(latteprec), dimension(:,:), allocatable [purearray::x2down](#)
- real(latteprec), dimension(:,:), allocatable [purearray::x2up](#)

8.330 purearray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE purearray
00023
00024     USE myprecision
00025
00026     IMPLICIT NONE
00027     SAVE
00028
00029     INTEGER :: maxdim
00030     INTEGER, ALLOCATABLE :: signlist(:)
00031     REAL(LATTEPREC) :: beta0
00032     REAL(LATTEPREC), ALLOCATABLE :: x2(:,,:), twoxx2(:,,:)
00033     REAL(LATTEPREC), ALLOCATABLE :: x2up(:,,:), x2down(:,,:)
00034
00035     ! New stuff so we can do the SP2 with the gap estimate
00036
00037     INTEGER :: nr_sp2_iter, pp(100)
00038     REAL(LATTEPREC) :: vv(100)
00039
00040 END MODULE purearray

```

8.331 qconsistency.f90 File Reference

Functions/Subroutines

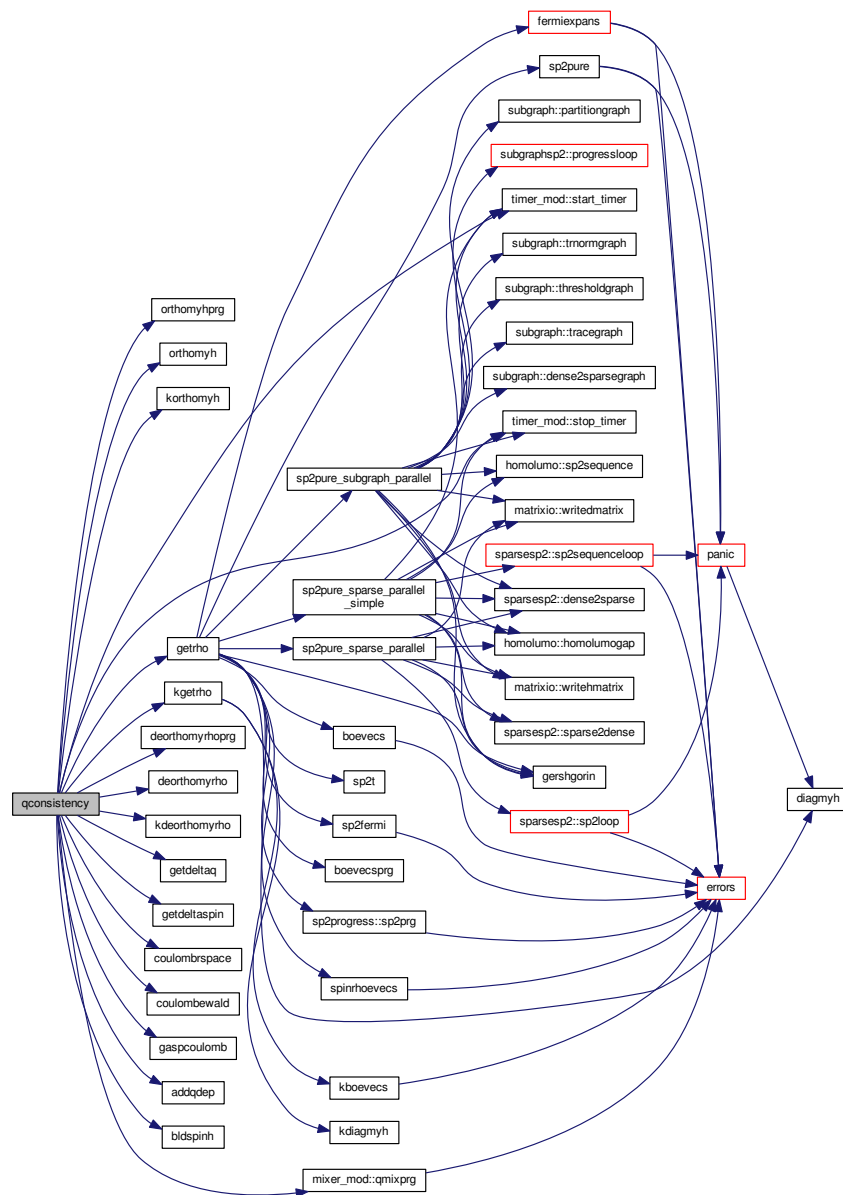
- subroutine [qconsistency](#) (SWITCH, MDITER)

8.331.1 Function/Subroutine Documentation

8.331.1.1 subroutine qconsistency (integer *SWITCH*, integer *MDITER*)

Definition at line 23 of file [qconsistency.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE qconsistency (SWITCH, MDITER)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE coulombarray
00028   USE timer_mod
00029   USE myprecision
00030   USE mixer_mod
00031
00032   IMPLICIT NONE
00033
00034   INTEGER :: I, SWITCH, MDITER, ITER, II
00035   INTEGER :: ALLOKQ, ALLOKM, ALLOK
00036   INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX, ITERACC
00037   REAL(4) :: TIMEACC
00038   REAL(LATTEPREC) :: MAXDQ
00039   REAL(LATTEPREC), ALLOCATABLE :: QDIFF(:), SPINDIFF(:)
00040   IF (existerror) RETURN
00041
00042   !
00043   ! If FULLQCONV = 1, then we're going to iterate until all charges are within
00044   ! QTOL.
00045   !
00046   ! If FULLQCONV = 0, then we're going to run only a user specified number
00047   ! of iterations (= QITER)
00048   !
00049   ! If SWITCH = 0, then we don't have any partial charges defined yet so
00050   ! we'll have to get these from our charge-independent H matrix first
00051   !
00052
00053   ente = zero
00054
00055   timeacc = 0.0
00056   iteracc = 0
00057
00058   IF (fullqconv .EQ. 1 .OR. mditer .LE. 10) THEN
00059
00060     IF (switch .EQ. 0) THEN
00061
00062       IF (basistype .EQ. "NONORTHO") THEN
00063         IF (kon .EQ. 0) THEN
00064
00065           #ifdef PROGRESSION
00066             IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00067               CALL orthomyhprg
00068             ELSE
00069               CALL orthomyh
00070             ENDIF
00071           #else
00072             CALL orthomyh
00073           #endif
00074
00075           ELSEIF (kon .EQ. 1) THEN
00076             CALL korthomyh
00077           ENDIF
00078         ENDIF
00079
00080         ! Compute the density matrix
00081
00082         tx = start_timer(dmbuild_timer)
00083
00084         IF (kon .EQ. 0) THEN
00085           CALL getrho(mditer)
00086         ELSE
00087           CALL kgetrho
00088         ENDIF
00089
00090         tx = stop_timer(dmbuild_timer)
00091
00092         !
00093         ! Now we have our bond-order/density matrices,

```

```

00094      ! we can get the charges and spins
00095      !
00096
00097      ! The partial charges and spin densities are computed from the
00098      ! density matrices computed from the orthogonalized H matrices
00099
00100      IF (basistype .EQ. "NONORTHO") THEN
00101          IF (kon .EQ. 0) THEN
00102              #ifdef PROGRESSON
00103                  IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00104                      CALL deorthomyrhoprpg
00105                  ELSE
00106                      CALL deorthomyrho
00107                  ENDIF
00108              #else
00109                  CALL deorthomyrho
00110              #endif
00111          ELSEIF (kon .EQ. 1) THEN
00112              CALL kdeorthomyrho
00113          ENDIF
00114      ENDIF
00115
00116      IF(verbose >= 1)WRITE(*,*)"In GETDELTAQ ..."
00117      CALL getdeltaq
00118
00119      IF (spinon .EQ. 1) CALL getdeltaspin
00120
00121  ENDIF
00122
00123      !
00124      ! Now we're going to run our iterations for self-consistency
00125      !
00126
00127      allok = 1
00128      iter = 0
00129
00130      ALLOCATE(qdiff(nats))
00131      IF (spinon .EQ. 1) ALLOCATE(spindiff(deltadim))
00132
00133      DO WHILE (allok .GT. 0)
00134
00135          iter = iter + 1
00136          IF(verbose >= 1)WRITE(*,*)"SCF ITER =", iter
00137          CALL flush(6)
00138
00139
00140          IF (elecmmeth .EQ. 0) THEN
00141
00142              !
00143              ! First do the real space part of the electrostatics
00144              ! This subroutine is based on Sanville's work
00145              !
00146
00147              CALL coulombspace
00148
00149              !
00150              ! And now the long range bit (this is also a modified version
00151              ! of Ed's code).
00152              !
00153
00154              CALL coulombewald
00155
00156          ELSE
00157
00158              !
00159              ! Doing the electrostatics all in real space
00160              ! help tremendously when getting the virial
00161              !
00162
00163              CALL gaspcoulomb
00164
00165          ENDIF
00166
00167      !
00168      ! Now let's modify the diagonal elements of our H matrix according
00169      ! to the electrostatic potential experienced by each atom
00170      !
00171
00172      CALL addqdep
00173
00174      ! Got to add the electrostatic potential to
00175      ! the Slater-Koster H before adding H_2 to form
00176      ! H_up and H_down
00177
00178
00179      !
00180      ! Calculate the spin-dependent H matrix again

```

```

00181      !
00182
00183      IF (spinon .EQ. 1) CALL bldspinh
00184
00185
00186      ! We've made changes to the H matrix so we have to re-orthogonalize
00187
00188      IF (basistype .EQ. "NONORTHO") THEN
00189          IF (kon .EQ. 0) THEN
00190              #ifdef PROGRESSON
00191                  IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00192                      CALL orthomyhprg
00193                  ELSE
00194                      CALL orthomyh
00195                  ENDIF
00196              #else
00197                  CALL orthomyh
00198              #endif
00199              ELSEIF (kon .EQ. 1) THEN
00200                  CALL korthomyh
00201              ENDIF
00202          ENDIF
00203
00204      !
00205      ! New Hamiltonian: get the bond order
00206      !
00207
00208      ! Compute the density matrix
00209
00210      tx = start_timer(dmbuild_timer)
00211
00212      IF (kon .EQ. 0) THEN
00213          CALL getrho(mditer)
00214      ELSE
00215          CALL kgetrho
00216      ENDIF
00217
00218      tx = stop_timer(dmbuild_timer)
00219
00220      ! We have a density matrix computed in from the orthogonalized
00221      ! H matrix - we have to revert back
00222
00223      !
00224      ! Save our old charges/spins so we can mix them later
00225      !
00226
00227      IF (basistype.EQ. "NONORTHO") THEN
00228          IF (kon .EQ. 0) THEN
00229              #ifdef PROGRESSON
00230                  IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00231                      CALL deorthomyrhopr
00232                  ELSE
00233                      CALL deorthomyrho
00234                  ENDIF
00235              #else
00236                  CALL deorthomyrho
00237              #endif
00238              ELSEIF (kon .EQ. 1) THEN
00239                  CALL kdeorthomyrho
00240              ENDIF
00241          ENDIF
00242
00243      olddeltaqs = deltaq
00244
00245      !
00246      ! Get a new set of charges for our system
00247      !
00248
00249      CALL getdeltaq
00250
00251      !
00252      ! Let's check for convergence
00253      !
00254
00255      alloq = 0
00256
00257      qdiff = abs(deltaq - olddeltaqs)
00258
00259      maxdq = maxval(qdiff)
00260      IF (maxdq .GT. elec_qtol) alloq = 1
00261
00262      ! Mix new and old partial charges
00263
00264      IF (mditer .LE. 10) THEN
00265          #ifdef PROGRESSON
00266              IF (mx%MIXERON) THEN
00267                  CALL qmixprg(iter) !Alternative mixing scheme from PROGRESS

```

```

00268         ELSE
00269             deltaq = qmix*deltaq + (one - qmix)*
olddeltaqs
00270         ENDIF
00271 #else
00272         deltaq = qmix*deltaq + (one - qmix)*olddeltaqs
00273 #endif
00274
00275     ELSE
00276
00277 #ifdef PROGRESSON
00278         IF (mx%MIXERON) THEN
00279             CALL qmixprg(iter)      !Alternative mixing scheme from PROGRESS
00280         ELSE
00281             deltaq = mdmix*deltaq + (one - mdmix)*
olddeltaqs
00282         ENDIF
00283 #else
00284         deltaq = mdmix*deltaq + (one - mdmix)*
olddeltaqs
00285 #endif
00286
00287     ENDIF
00288
00289     IF(verbose >= 1) WRITE(*,*) "SCF error (MAXDQ) =",maxdq
00290
00291     allokm = 0
00292
00293     IF (spinon .EQ. 1) THEN
00294
00295         olddeltaspin = deltaspin
00296         CALL getdeltaspin
00297
00298         spindiff = abs(deltaspin - olddeltaspin)
00299
00300         IF (maxval(spindiff) .GT. spintol) allokm = 1
00301
00302         ! Mix new and old spin densities
00303
00304         deltaspin = spinmix*deltaspin + (one -
spinmix)*olddeltaspin
00305
00306     ENDIF
00307
00308     allok = allokq + allokm
00309
00310     IF (iter .EQ. maxscf) THEN
00311
00312         WRITE(6,*) "## WARNING - the SCF procedure has not converged"
00313         WRITE(6,*) "## to the tolerances defined in TBparam/control.in"
00314         WRITE(6,*) "## Continuing anyway, but be very careful... "
00315
00316         allok = 0
00317
00318     ENDIF
00319
00320     ENDDO
00321
00322     numscf = iter
00323
00324     DEALLOCATE(qdiff)
00325     IF (spinon .EQ. 1) DEALLOCATE(spindiff)
00326
00327     ! WRITE(6,99) "## HDIM, TIME PER RHO BUILD = ", HDIM, TIMEACC/REAL(ITERACC)
00328     !99 FORMAT(A29, I5, 1X, G18.6)
00329
00330     CALL flush(6)
00331
00332     ELSEIF (fullqconv .EQ. 0 .AND. mdon .EQ. 1 .AND. mditer .GT. 10) THEN
00333
00334         ! Now we're doing MD
00335
00336         DO ii = 1, qiter
00337
00338             IF (elecmeth .EQ. 0) THEN
00339
00340                 CALL coulombospace
00341
00342                 CALL coulombewald
00343
00344             ELSE
00345
00346                 CALL gaspcoulomb
00347
00348             ENDIF
00349
00350         ENDIF

```

```

00351
00352     CALL addqdep
00353
00354     !
00355     ! Building the spin up and spin down H's after we've
00356     ! added the electrostatic potential to the Slater-Koster one,
00357     ! as it should be.
00358     !
00359
00360     IF (spinon .EQ. 1) CALL bldspinh
00361
00362     IF (basistype .EQ. "NONORTHO") THEN
00363         IF (kon .EQ. 0) THEN
00364
00365             #ifdef PROGRESSON
00366                 IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00367                     CALL orthomyhprg
00368                 ELSE
00369                     CALL orthomyh
00370                 ENDIF
00371             #else
00372                 CALL orthomyh
00373             #endif
00374
00375             ELSEIF (kon .EQ. 1) THEN
00376                 CALL korthomyh
00377             ENDIF
00378         ENDIF
00379
00380
00381     !
00382     ! New Hamiltonian: get the bond order
00383     !
00384
00385     ! Compute the density matrix
00386
00387
00388     IF (kon .EQ. 0) THEN
00389         CALL getrho(mditer)
00390     ELSE
00391         CALL kgetrho
00392     ENDIF
00393
00394     olddeltaqs = deltaq
00395
00396     !
00397     ! Get a new set of charges for our system
00398     !
00399
00400     IF (basistype .EQ. "NONORTHO") THEN
00401         IF (kon .EQ. 0) THEN
00402             #ifdef PROGRESSON
00403                 IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00404                     CALL deorthomyrhoprgr
00405                 ELSE
00406                     CALL deorthomyrho
00407                 ENDIF
00408             #else
00409                 CALL deorthomyrho
00410             #endif
00411
00412             ELSEIF (kon .EQ. 1) THEN
00413                 CALL kdeorthomyrho
00414             ENDIF
00415         ENDIF
00416
00417         CALL getdeltaq
00418
00419     !
00420     ! Mix to get new charges
00421     !
00422     deltaq = mdmix*deltaq + (one - mdmix)*
olddeltaqs
00423
00424     !         PRINT*, DELTAQ(1)
00425
00426     IF (spinon .EQ. 1) THEN
00427
00428         olddeltaspin = deltaspin
00429         CALL getdeltaspin
00430         deltaspin = spinmix*deltaspin + (one -
spinmix)*olddeltaspin
00431
00432     ENDIF
00433
00434     ENDDO
00435

```

```

00436      ! Calculate the bond order one more time since we need the forces for
00437      ! that charge distribution
00438
00439      IF (elecsmeth .EQ. 0) THEN
00440
00441          CALL coulombbrspace
00442
00443          CALL coulombewald
00444
00445      ELSE
00446
00447          CALL gaspcoulomb
00448
00449      ENDIF
00450
00451      CALL addqdep
00452
00453      ! This is the right order
00454
00455      IF (spinon .EQ. 1) CALL bldspinh
00456
00457      IF (basistype .EQ. "NONORTHO") THEN
00458          IF (kon .EQ. 0) THEN
00459
00460      #ifdef PROGRESSON
00461          IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00462              CALL orthomyhprg
00463          ELSE
00464              CALL orthomyh
00465          ENDIF
00466      #else
00467          CALL orthomyh
00468      #endif
00469
00470
00471          ELSEIF (kon .EQ. 1) THEN
00472              CALL korthomyh
00473          ENDIF
00474      ENDIF
00475
00476      !
00477      ! New Hamiltonian: get the bond order/density matrices
00478      !
00479
00480      ! Compute the density matrix
00481
00482      IF (kon .EQ. 0) THEN
00483          CALL getrho(mditer)
00484      ELSE
00485          CALL kgetrho
00486      ENDIF
00487
00488      IF (basistype .EQ. "NONORTHO") THEN
00489          IF (kon .EQ. 0) THEN
00490
00491      #ifdef PROGRESSON
00492          IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00493              CALL deorthomyrhprg
00494          ELSE
00495              CALL deorthomyrho
00496          ENDIF
00497      #else
00498          CALL deorthomyrho
00499      #endif
00500
00501          ELSEIF (kon .EQ. 1) THEN
00502              CALL kdeorthomyrho
00503          ENDIF
00504
00505      ENDIF
00506
00507      RETURN
00508
00509  END SUBROUTINE qconsistency

```

8.333 qneutral.f90 File Reference

Functions/Subroutines

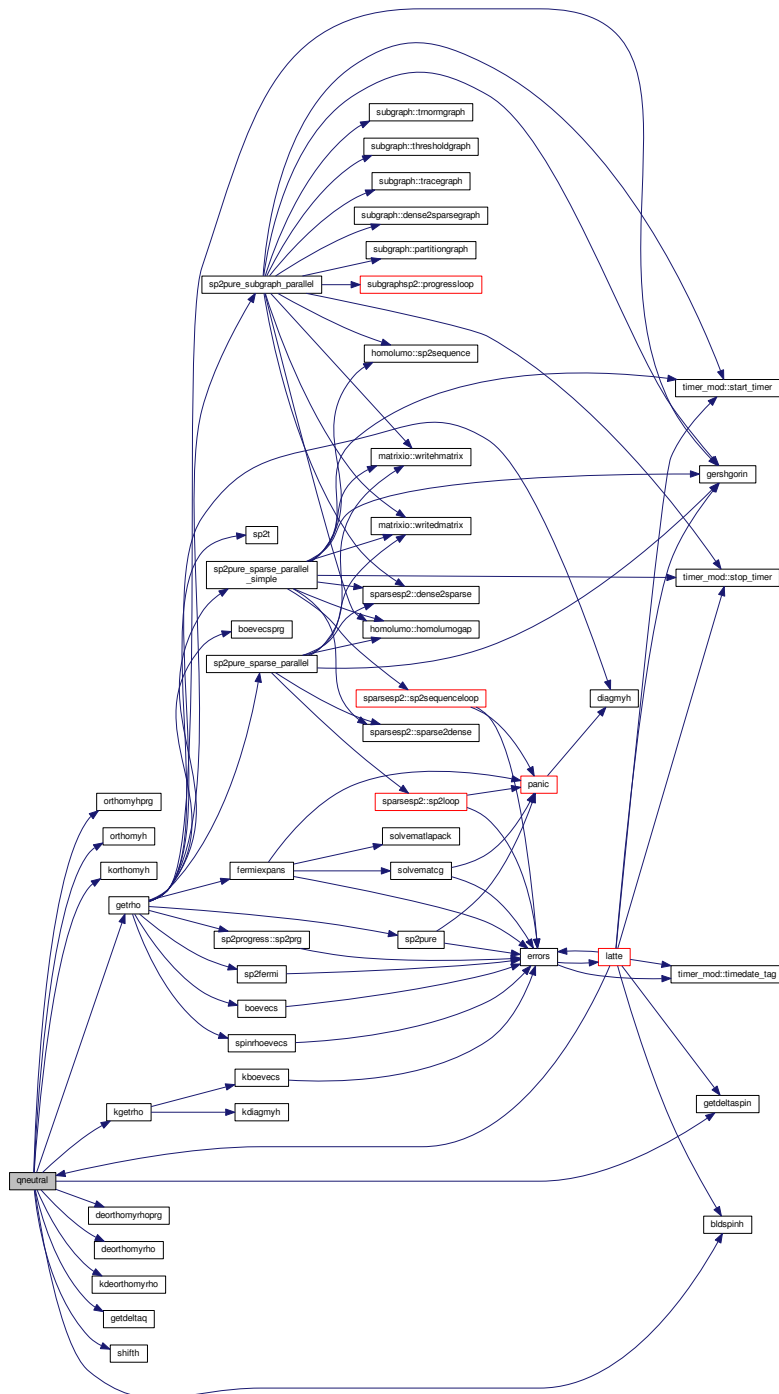
- subroutine [qneutral](#) (SWITCH, MDITER)

8.333.1 Function/Subroutine Documentation

8.333.1.1 subroutine qneutral (integer *SWITCH*, integer *MDITER*)

Definition at line 23 of file [qneutral.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE qneutral(SWITCH, MDITER)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE coulombarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, SWITCH, MDITER, ITER, II
00033   INTEGER :: ALLOKQ, ALLOKM, ALLOK
00034   INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00035   REAL(4) :: TIMEACC
00036   REAL(LATTEPREC), ALLOCATABLE :: SPINDIFF(:)
00037   IF (existerror) RETURN
00038
00039   !
00040   ! If FULLQCONV = 1, then we're going to iterate until all charges are within
00041   ! QTOL.
00042   !
00043   ! If FULLQCONV = 0, then we're going to run only a user specified number
00044   ! of iterations (= QITER)
00045   !
00046   ! If SWITCH = 0, then we don't have any partial charges defined yet so
00047   ! we'll have to get these from our charge-independent H matrix first
00048   !
00049
00050   ente = zero
00051
00052   IF (fullqconv .EQ. 1 .OR. mditer .LE. 10) THEN
00053
00054     IF (switch .EQ. 0) THEN
00055
00056       IF (basistype .EQ. "NONORTHO") THEN
00057         IF (kon .EQ. 0) THEN
00058 #ifdef PROGRESSON
00059           IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00060             CALL orthomyhprg
00061           ELSE
00062             CALL orthomyh
00063           ENDIF
00064 #else
00065             CALL orthomyh
00066 #endif
00067         ELSEIF (kon .EQ. 1) THEN
00068           CALL korthomyh
00069         ENDIF
00070       ENDIF
00071
00072       ! Compute the density matrix
00073
00074       IF (kon .EQ. 0) THEN
00075         CALL getrho(mditer)
00076       ELSE
00077         CALL kgetrho
00078       ENDIF
00079
00080       ! If we used diagonalization, we can compute the response function
00081       ! for updating the charges
00082
00083       !           IF (CONTROL .EQ. 1) CALL GETRESPF
00084
00085       !
00086       ! Now we have our bond-order/density matrices,
00087       ! we can get the charges and spins
00088       !
00089
00090       ! The partial charges and spin densities are computed from the
00091       ! density matrices computed from the orthogonalized H matrices
00092
00093

```

```

00094         IF (basistype .EQ. "NONORTHO") THEN
00095             IF (kon .EQ. 0) THEN
00096 #ifdef PROGRESSON
00097                 IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00098                     CALL deorthomyrhoprgr
00099                 ELSE
00100                     CALL deorthomyrho
00101                 ENDIF
00102 #else
00103                 CALL deorthomyrho
00104 #endif
00105             ELSEIF (kon .EQ. 1) THEN
00106                 CALL kdeorthomyrho
00107             ENDIF
00108         ENDIF
00109
00110         CALL getdeltaq
00111
00112         IF (spinon .EQ. 1) CALL getdeltaspin
00113
00114     ENDIF
00115
00116     !
00117     ! Now we're going to run our iterations for self-consistency
00118     !
00119
00120     allok = 1
00121     iter = 0
00122
00123     IF (spinon .EQ. 1) ALLOCATE(spindiff(deltadim))
00124
00125     DO WHILE (allok .GT. 0)
00126
00127         iter = iter + 1
00128
00129         ! Adjust on-site energies such that partial charges -> 0
00130
00131         CALL shifth(qmix)
00132
00133         !
00134         ! New Hamiltonian: get the bond order
00135         !
00136
00137         ! Compute the density matrix
00138
00139         IF (spinon .EQ. 1) CALL bldspinh
00140
00141         IF (basistype .EQ. "NONORTHO") THEN
00142             IF (kon .EQ. 0) THEN
00143 #ifdef PROGRESSON
00144                 IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00145                     CALL orthomyhprgr
00146                 ELSE
00147                     CALL orthomyh
00148                 ENDIF
00149 #else
00150                 CALL orthomyh
00151 #endif
00152             ELSEIF (kon .EQ. 1) THEN
00153                 CALL korthomyh
00154             ENDIF
00155         ENDIF
00156
00157
00158         IF (kon .EQ. 0) THEN
00159             CALL getrho(mditer)
00160         ELSE
00161             CALL kgetrho
00162         ENDIF
00163
00164         !             IF (CONTROL .EQ. 1) CALL GETRESPF
00165
00166         IF (basistype.EQ. "NONORTHO") THEN
00167             IF (kon .EQ. 0) THEN
00168 #ifdef PROGRESSON
00169                 IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00170                     CALL deorthomyrhoprgr
00171                 ELSE
00172                     CALL deorthomyrho
00173                 ENDIF
00174 #else
00175                 CALL deorthomyrho
00176 #endif
00177             ELSEIF (kon .EQ. 1) THEN
00178                 CALL kdeorthomyrho
00179             ENDIF
00180         ENDIF

```

```

00181
00182
00183     CALL getdeltaq
00184
00185     IF (abs(maxval(deltaq)) .LT. elec_qtol) alloc = 0
00186     !PRINT*, ABS(MAXVAL(DELTAQ)), ELEC_QTOL
00187
00188     IF (spinon .EQ. 1) THEN
00189
00190         olddeltaspin = deltaspin
00191         CALL getdeltaspin
00192
00193         spindiff = abs(deltaspin - olddeltaspin)
00194
00195         IF (maxval(spindiff) .LT. spintol) alloc = 0
00196
00197         ! Mix new and old spin densities
00198
00199         deltaspin = spinmix*deltaspin + (one -
spinmix)*olddeltaspin
00200
00201     ENDIF
00202
00203
00204     IF (iter .EQ. maxscf) THEN
00205
00206         WRITE(6,*) "WARNING - the SCF procedure has not converged"
00207         WRITE(6,*) "to the tolerances defined in TBparam/control.in"
00208         WRITE(6,*) "Continuing anyway, but be very careful... "
00209
00210         alloc = 0
00211
00212     ENDIF
00213
00214
00215     ENDDO
00216
00217     numscf = iter
00218
00219     IF (spinon .EQ. 1) DEALLOCATE(spindiff)
00220
00221     ELSEIF (fullqconv .EQ. 0 .AND. mdon .EQ. 1 .AND. mditer .GT. 10) THEN
00222
00223         ! Now we're doing MD
00224
00225         DO ii = 1, qiter
00226
00227             CALL shifth(mdmix)
00228
00229             IF (spinon .EQ. 1) CALL bldspinh
00230
00231             IF (basistype .EQ. "NONORTHO") THEN
00232                 IF (kon .EQ. 0) THEN
00233 #ifdef PROGRESSON
00234                     IF (latteinexists) THEN !orthogonalize from progress lib if latte.in exists
00235                         CALL orthomyhprg
00236                     ELSE
00237                         CALL orthomyh
00238                     ENDIF
00239 #else
00240                     CALL orthomyh
00241 #endif
00242                 ELSEIF (kon .EQ. 1) THEN
00243                     CALL korthomyh
00244                 ENDIF
00245             ENDIF
00246
00247
00248
00249             !
00250             ! New Hamiltonian: get the bond order
00251             !
00252
00253             ! Compute the density matrix
00254
00255             IF (kon .EQ. 0) THEN
00256                 CALL getrho(mditer)
00257             ELSE
00258                 CALL kgetrho
00259             ENDIF
00260
00261             IF (basistype .EQ. "NONORTHO") THEN
00262                 IF (kon .EQ. 0) THEN
00263 #ifdef PROGRESSON
00264                     IF (latteinexists) THEN !deorthogonalize from progress lib if latte.in exists
00265                         CALL deorthomyrhprg
00266                     ELSE

```

```

00267             CALL deorthomyrho
00268             ENDIF
00269 #else
00270             CALL deorthomyrho
00271 #endif
00272             ELSEIF (kon .EQ. 1) THEN
00273             CALL kdeorthomyrho
00274             ENDIF
00275             ENDIF
00276
00277             !           IF (CONTROL .EQ. 1) CALL GETRESPF
00278
00279             CALL getdeltaq
00280
00281             IF (spinon .EQ. 1) THEN
00282
00283             olddeltaspin = deltaspin
00284             CALL getdeltaspin
00285             deltaspin = spinmix*deltaspin + (one -
spinmix)*olddeltaspin
00286
00287             ENDIF
00288
00289             ENDDO
00290
00291             ENDIF
00292
00293             RETURN
00294
00295 END SUBROUTINE qneutral

```

8.335 readcontrols.f90 File Reference

Functions/Subroutines

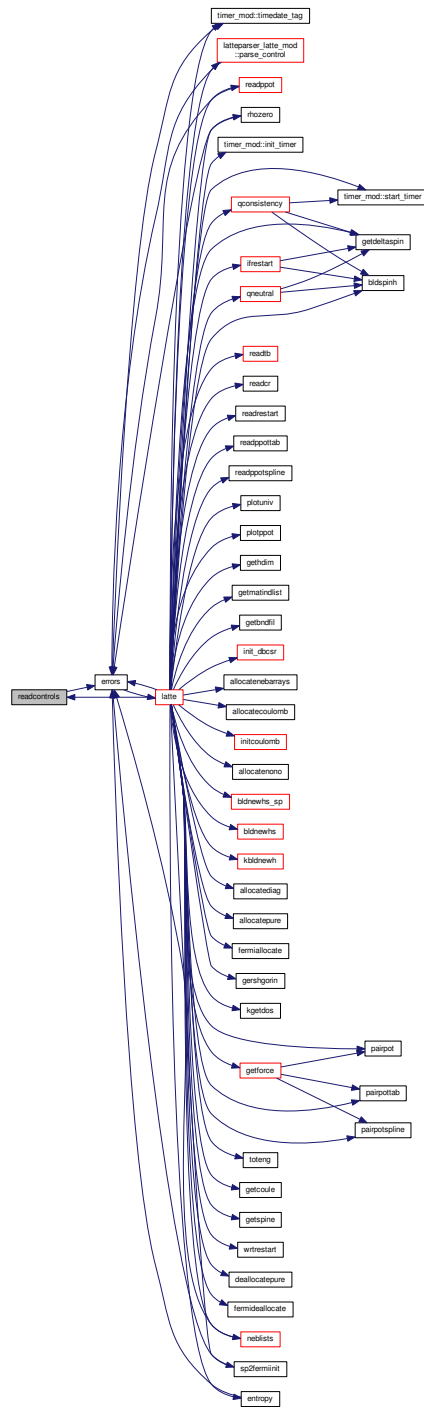
- subroutine [readcontrols](#)

8.335.1 Function/Subroutine Documentation

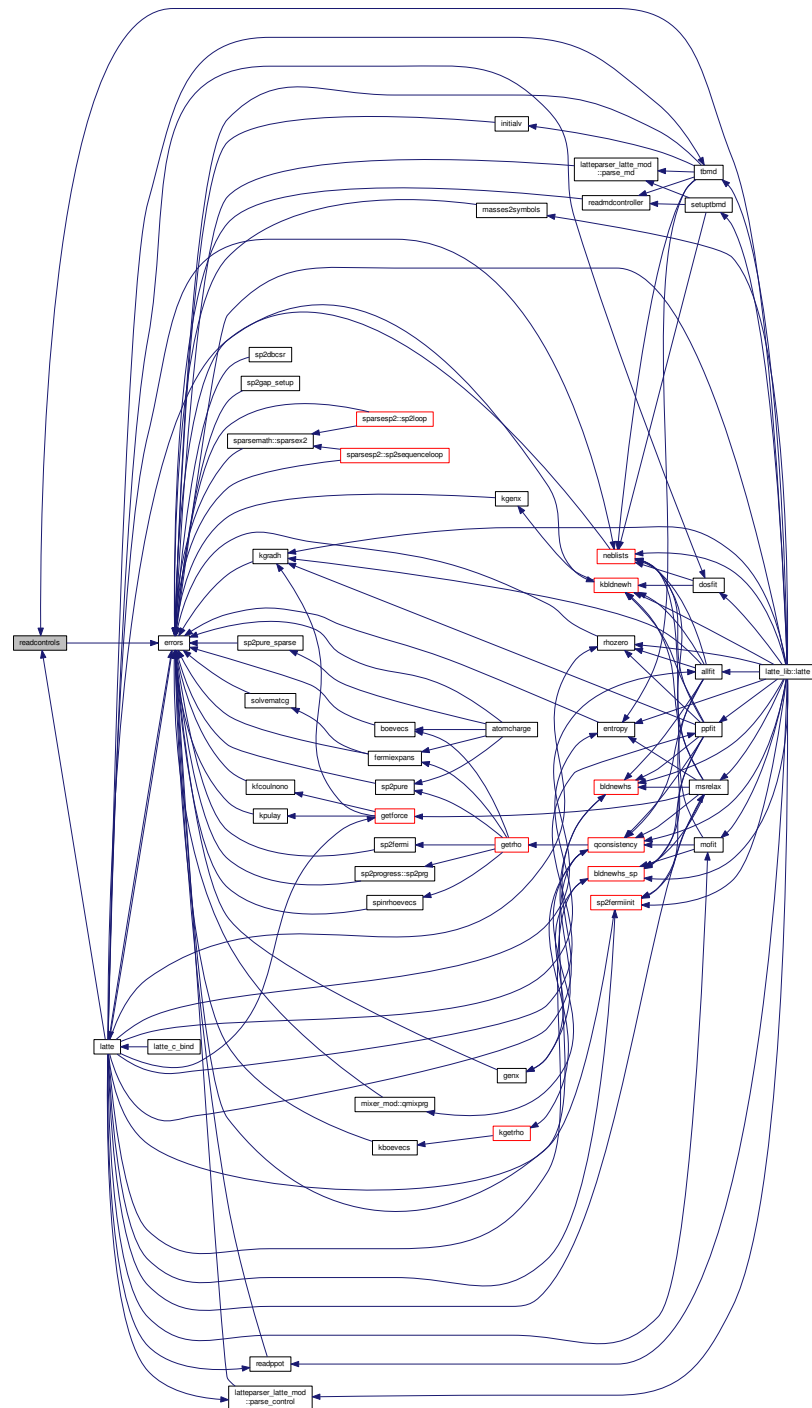
8.335.1.1 subroutine readcontrols ()

Definition at line 23 of file [readcontrols.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.336 readcontrols.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readcontrols
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE pspotarray
00027   USE neblastarray
00028   USE coulombarray
00029   USE fermicommon
00030   USE sparsearray
00031   USE relaxcommon
00032
00033   IMPLICIT NONE
00034
00035   CHARACTER(LEN=20) :: HD
00036
00037   IF (existerror) RETURN
00038
00039   OPEN(unit=13, status="OLD", file=trim(parampath)//"/control.in")
00040
00041   !
00042   ! CONTROL determines how the density matrix is going to be
00043   ! calculated: 1 = diagonalization, 2 = SP2 purification,
00044   ! 3 = recursive expansion of the Fermi operator, 4 = SP2T,
00045   ! 5 = SP2 Fermi (truncated SP2)
00046   !
00047
00048   READ(13,*) hd, control
00049
00050   !
00051   ! BASISTYPE can equal "ORTHO" OR "NONORTHO",
00052   !
00053
00054   READ(13,*) hd, basistype
00055
00056   IF (basistype .NE. "ORTHO" .AND. basistype .NE. "NONORTHO") THEN
00057     CALL errors("readcontrols","Error defining basis type (ortho/nonortho)")
00058   ENDIF
00059
00060   READ(13,*) hd, debugon
00061
00062   !
00063   ! Read the order of the recursion in the expansion of the Fermi
00064   ! operator, M.
00065   !
00066   !
00067
00068   READ(13,*) hd, fermim
00069
00070   ! If we're using the expansion of the Fermi operator, we can
00071   ! use a LAPACK routine or Niklasson's conjugate gradient method to
00072   ! solve AX = B. CGORLIB: 0 = LAPACK, 1 = conjugate gradient
00073   ! CGTOL = the user-supplied tolerance for the CG solution of AX = B
00074
00075   READ(13,*) hd, cgorlib, hd, cgtol
00076
00077   cgtol2 = cgtol*cgtol
00078
00079   ! Electronic temperature, in eV
00080
00081   READ(13,*) hd, kbt
00082
00083   !
00084   ! Read the number of recursions for the truncated, finite
00085   ! temperature SP2 algorithm
00086   !
00087
00088   READ(13,*) hd, norecs
00089
00090   !
00091   ! What kind of entropy are we going to use in a finite Te calculation
00092   !
00093   ! ENTROPYKIND = 0 : none

```

```
00094 ! ENTROPYKIND = 1 : exact for Fermi-Dirac occupation
00095 ! ENTROPYKIND = 2 : Different form of exact expression that may be useful
00096 ! when using CONTROL = 5
00097 ! ENTROPYKIND = 3 : 4th order expansion of exact form (no diag)
00098 ! ENTROPYKIND = 4 : 8th order expansion of exact form (no diag)
00099 !
00100
00101 READ(13,*) hd, entropykind
00102
00103 !
00104 ! Do we want long-range C/R^6 tails?
00105 !
00106 ! PPOTON = 1: Turn on pairwise interaction
00107 ! PPOTON = 0: Turn it off (useful for fitting)
00108 !
00109 ! VDWON = 0: No C/R^6 tails
00110 ! VDWON = 1: Use tails
00111 !
00112
00113 READ(13,*) hd, ppoton, hd, vdwon
00114
00115
00116 !
00117 ! Are we doing a spin-polarized calculation?
00118 ! SPINON = 1 = yes
00119 ! SPINON = 0 = no
00120
00121 READ(13,*) hd, spinon, hd, spintol
00122
00123 !
00124 ! Controls for electrostatics:
00125 !
00126 ! ELECTRO: 0 = LCN is applied, 1 = charge dependent TB on
00127 ! ELECMETH: 0 = Ewald summation, 1 = All real space
00128 ! ELEC_ETOL: Tolerance on energy when determining charges (not implemented)
00129 ! ELEC_QTOL: Tolerance on charges during self-consistent calc
00130 !
00131
00132 READ(13,*) hd, electro, hd, elecmeth, hd, elec_etol, hd,
elec_qtol
00133
00134 !
00135 ! COULACC: Accuracy for the Ewald method (1.0e-4 works)
00136 ! COULCUT: If we're using the Ewald method, this is the cut-off for the
00137 ! real space part. If we're doing it all in real space, this is the radial
00138 ! cut-off for the sum.
00139 ! COULR1: If we're doing it in real space, the cut-off tail on 1/R is
00140 ! applied here at terminated at COULCUT.
00141 !
00142
00143 READ(13,*) hd, coulacc, hd, coulcut, hd, coulrl
00144
00145 !
00146 ! MAXSCF: Maximum number of SCF cycles
00147 !
00148
00149 READ(13,*) hd, maxscf
00150
00151 !
00152 ! BREAKTOL: Tolerance for breaking SP2 loops
00153 ! MINSP2ITER: Minimum number of iterations during SP2 purification
00154 !
00155
00156 READ(13,*) hd, breaktol, hd, minsp2iter, hd, sp2conv
00157
00158 !
00159 ! FULLQCONV: 0 = We'll run QITER SCF cycles during MD, = 1, we'll run
00160 ! SCF cycles until we've reached ELEC_QTOL. Only important for MD
00161 ! QITER: Number of SCF cycles we're going to run at each MD time step
00162 !
00163
00164 READ(13,*) hd, fullqconv, hd, qiter
00165
00166 !
00167 ! QMIX AND SPINMIX are the coefficients for the linear mixing of
00168 ! new and old charge and spin densities, respectively, during SCF cycles
00169 !
00170
00171 READ(13,*) hd, qmix, hd, spinmix, hd, mdmix
00172
00173 !
00174 ! ORDERNMOL: Turn on molecule-ID-based density matrix blocking
00175 !
00176
00177 READ(13,*) hd, ordernmol
00178
00179 !
```



```

00180 ! SPARSEON: 0 = all dense matrix stuff, 1 = use CSR format and
00181 ! Gustavson's algorithm for matrix-matrix multiplication
00182 ! THRESHOLDON: 0 = do not throw away elements; 1 = throw away elements
00183 ! NUMTHRESH: If THRESHOLDON = 1 throw away element whose absolute value is
00184 ! smaller than NUMTHRESH
00185 ! FILLINSTOP: Number of purification cycles beyond which we stop allowing
00186 ! for further fill-in
00187 !
00188
00189 READ(13,*) hd, sparseon, hd, thresholdon, hd, numthresh, hd,
fillinstop, hd, blkksz
00190
00191 !
00192 ! MSPARSE: value for M when SPARSEON = 1, used by sp2 sparse algorithm
00193 !         0 = value for M is not known, defaults to N
00194 !
00195
00196 READ(13,*) hd, msparse
00197
00198 !
00199 ! LCNON: 0 = during charge neutral MD simulations we'll run LCNITER SCF
00200 ! cycles at each time step, 1 = we'll run SCF cycles until CHTOL is reached
00201 ! LCNITER: Number of SCF cycles to achieve LCN at each MD time step
00202 ! CHTOL: Tolerance on atomic charges (Mulliken) before LCN is declared
00203 !
00204
00205 READ(13,*) hd, lcnon, hd, lcniter, hd, chtol
00206
00207 !
00208 ! Read the SKIN for the neighbor list (Angstrom)
00209 !
00210
00211 READ(13,*) hd, skin
00212
00213 !
00214 ! RELAXME: 0 = Don't run relaxation, 1 = relax geometry
00215 ! RELTYPE: SD = steepest descent, CG = conjugate gradient
00216 ! MXRLX: Maximum number of steps in the geometry optimization
00217 ! RLXFTOT: Run optimization until all forces are less than RLXFTOL
00218 !
00219
00220 READ(13,*) hd, relaxme, hd, reltype, hd, mxrlx, hd, rlxftol
00221
00222 !
00223 ! MDON: 0 = Molecular dynamics off, 1 = Molecular dynamics on
00224 ! (MD is controlled using the file MDcontroller)
00225 !
00226
00227 READ(13,*) hd, mdon
00228
00229 !
00230 ! PBCON: 1 = full periodic boundary conditions, 0 = gas phase: no pbc and
00231 ! electrostatics done all in real space
00232 !
00233
00234 READ(13,*) hd, pbcon
00235
00236 READ(13,*) hd, restart
00237
00238 ! Add or remove electrons. 2+ -> charge = +2 since TOTNE = TOTNE - CHARGE
00239
00240 READ(13,*) hd, charge
00241
00242 !
00243 ! XBOON: 0 = Niklasson's extended Lagrangian Born-Oppenheimer MD off,
00244 ! 1 = on.
00245 !
00246
00247 READ(13,*) hd, xboon
00248
00249 !
00250 ! XBODISON: We have the option of turning on damping for the XBO
00251 ! to remedy the accumulation of noise. 0 = off, 1 = on.
00252 !
00253
00254 READ(13,*) hd, xbodison
00255
00256 !
00257 ! XBODISORDER: = Order of the damping function (1 - 9)
00258 !
00259
00260 READ(13,*) hd, xbodisorder
00261
00262 fiton = 0
00263
00264 !
00265 ! Read in the number of GPUs per node

```

```

00266      !
00267
00268      READ(13,*) hd, ngpu
00269
00270      ! Are we doing k-space?
00271
00272      READ(13,*) hd, kon
00273
00274      ! Do we want to calculate forces too (not always necessary when fitting)
00275
00276      READ(13,*) hd, compforce
00277
00278      ! Turn on the simulated annealing subroutine to fit DOS
00279
00280
00281      READ(13,*) hd, dosfiton, hd, int2fit, hd, mcbeta, hd,
nfitstep, hd, qfit, &
00282          hd, mcsigma
00283
00284      READ(13,*) hd, ppfiton
00285
00286      READ(13,*) hd, allfiton
00287
00288      READ(13,*) hd, ppnfitstep, hd, binfitstep, hd, pp2fit, hd,
bint2fit
00289
00290      READ(13,*) hd, ppbeta, hd, ppsigma, hd, ppnmol, hd, ppngeom
00291
00292      READ(13,*) hd, parrep
00293
00294      ! Dielectric constant
00295
00296      READ(13,*) hd, relperm
00297
00298      CLOSE(13)
00299
00300      !
00301      ! Summarize the calculation we're doing here
00302      !
00303
00304      ! OPEN(UNIT=99, STATUS="UNKNOWN", FILE="my_last_LATTE_calc")
00305
00306      ! IF (CONTROL .EQ. 1) THEN
00307      !     WRITE(99,*) "Diagonization used to calculate bond order"
00308      ! ELSEIF (CONTROL .EQ. 2 .AND. SPARSEON .EQ. 0) THEN
00309      !     WRITE(99,*) "Dense matrix SP2 used to calculate bond order"
00310      ! ELSEIF (CONTROL .EQ. 2 .AND. SPARSEON .EQ. 1) THEN
00311      !     WRITE(99,*) "Quasi-sparse matrix SP2 used to calculated bond order"
00312      ! ELSEIF (CONTROL .EQ. 3) THEN
00313      !     WRITE(99,*) "Recursive expansion of the Fermi operator"
00314      !     IF (CGORLIB .EQ. 0) THEN
00315      !         WRITE(99,*) "Dense matrix: using LAPACK routine to solve AX=B"
00316      !     ENDIF
00317      !     IF (CGORLIB .EQ. 1 .AND. SPARSEON .EQ. 0) THEN
00318      !         WRITE(99,*) "Dense matrix conjugate gradient scheme to solve AX=B"
00319      !     ELSEIF (CGORLIB .EQ. 1 .AND. SPARSEON .EQ. 1) THEN
00320      !         WRITE(99,*) "Sparse matrix conjugate gradient scheme to solve AX=B"
00321      !     ENDIF
00322      ! ENDIF
00323
00324      RETURN
00325
00326  END SUBROUTINE readcontrols

```

8.337 readcr.f90 File Reference

Functions/Subroutines

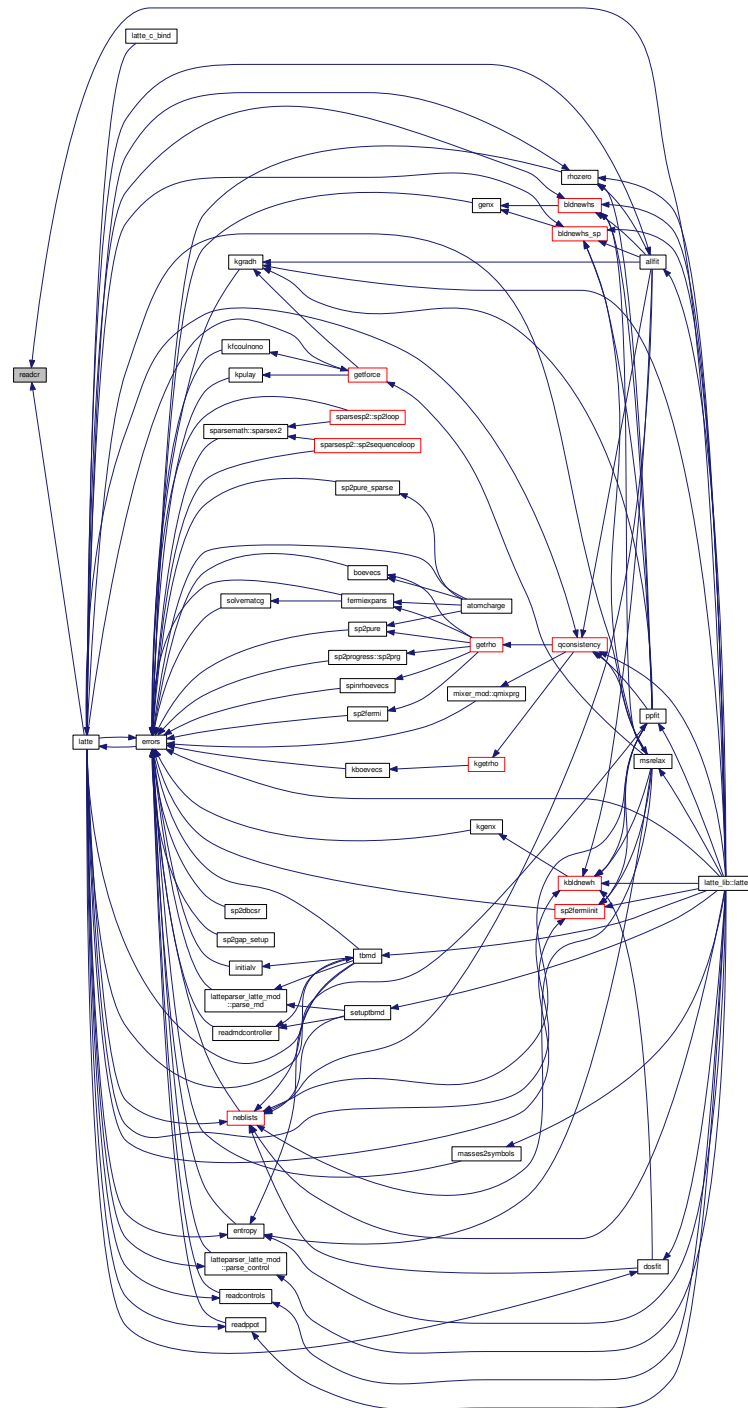
- subroutine [readcr](#)

8.337.1 Function/Subroutine Documentation

8.337.1.1 subroutine readcr ()

Definition at line 23 of file [readcr.f90](#).

Here is the caller graph for this function:



8.338 readcr.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE reader
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE neblistarray
00028   USE kspacearray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, MYINDEX
00034   REAL(LATTEPREC) :: LN(6)
00035   CHARACTER(LEN=20) :: HEADER
00036   IF (existerror) RETURN
00037
00038   IF (.NOT.ALLOCATED(cr)) THEN
00039
00040     OPEN(unit=12, status="OLD", file=coordsfile)
00041
00042     READ(12,*) nats
00043
00044     ALLOCATE(cr(3,nats), atele(nats), f(3,nats), fpp(3,
nats), ftot(3,nats))
00045     ALLOCATE(deltaq(nats), mycharge(nats))
00046     ALLOCATE(elempointer(nats))
00047
00048     IF (electro .EQ. 0) THEN
00049       ALLOCATE(lcnshift(nats))
00050       lcnshift = zero
00051     ENDIF
00052
00053     IF (kon .EQ. 1) ALLOCATE(kf(3,nats))
00054
00055     IF (basistype .EQ. "NONORTHO") THEN
00056       IF (spinon .EQ. 0) THEN
00057         ALLOCATE(fpul(3,nats), fscoul(3,nats))
00058       ELSE
00059         ALLOCATE(fpul(3,nats), fscoul(3,nats), fsspin(3,
nats))
00060       ENDIF
00061     ENDIF
00062
00063     READ(12,*) box(1,1), box(1,2), box(1,3)
00064     READ(12,*) box(2,1), box(2,2), box(2,3)
00065     READ(12,*) box(3,1), box(3,2), box(3,3)
00066
00067     DO i = 1, nats
00068       READ(12,*) atele(i), cr(1,i), cr(2,i), cr(3,i)
00069     ENDDO
00070
00071     CLOSE(12)
00072
00073   ELSE
00074
00075     ALLOCATE(f(3,nats), fpp(3,nats), ftot(3,nats))
00076     ALLOCATE(deltaq(nats), mycharge(nats))
00077     ALLOCATE(elempointer(nats))
00078
00079     IF (electro .EQ. 0) THEN
00080       ALLOCATE(lcnshift(nats))
00081       lcnshift = zero
00082     ENDIF
00083
00084     IF (kon .EQ. 1) ALLOCATE(kf(3,nats))
00085
00086     IF (basistype .EQ. "NONORTHO") THEN
00087       IF (spinon .EQ. 0) THEN
00088         ALLOCATE(fpul(3,nats), fscoul(3,nats))
00089       ELSE
00090         ALLOCATE(fpul(3,nats), fscoul(3,nats), fsspin(3,
nats))

```

```

00091         ENDIF
00092     ENDIF
00093
00094 ENDIF
00095
00096 ! Set up pointer to the data in TBparam/electrons.dat
00097 elempointer = 0
00098
00099 DO i = 1, nats
00100     DO j = 1, noelem
00101         IF (atele(i) .EQ. ele(j)) elempointer(i) = j
00102     ENDDO
00103 ENDDO
00104
00105 ! Let's compute the total mass - this comes in handy when computing the
00106 ! density later
00107
00108 summass = zero
00109
00110 DO i = 1, nats
00111     summass = summass + mass(elempointer(i))
00112 ENDDO
00113
00114 ! Let's check whether we have only sp elements. If so, we can
00115 ! use a much faster version of gradH
00116
00117 ! SPONLY = 0: use GRADHSP
00118 ! SPONLY = 1: use Josh Coe's implementation of the automatic H build
00119
00120 sponly = 0
00121 DO i = 1, nats
00122     IF (basis(elempointer(i)) .NE. "s" .AND. &
00123         basis(elempointer(i)) .NE. "sp") sponly = 1
00124 ENDDO
00125 ! Print a warning
00126
00127 IF (sponly .EQ. 1) THEN
00128     print*, "#FYI: d or f orbitals detected so we're using the"
00129     print*, "#slower, general SK expansions"
00130 ENDIF
00131
00132 ! SPONLY = 1
00133
00134 ! If we're enforcing LCN and we're using diagonalization
00135
00136 IF (control .EQ. 1 .AND. electro .EQ. 0) ALLOCATE(respchi(
nats))
00137
00138 RETURN
00139
00140 END SUBROUTINE readcr

```

8.339 readmdcontroller.f90 File Reference

Functions/Subroutines

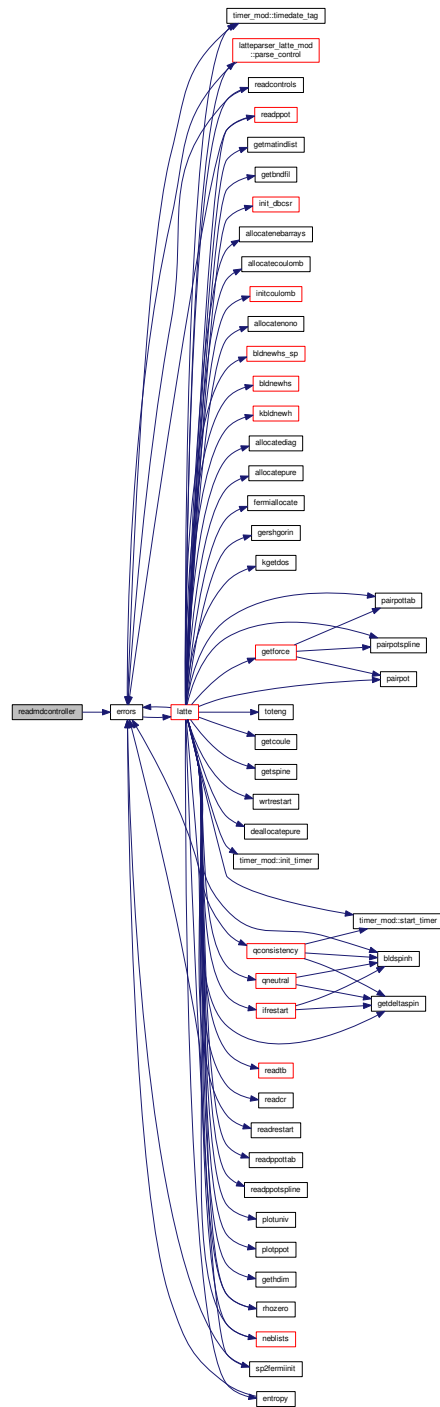
- subroutine [readmdcontroller](#)

8.339.1 Function/Subroutine Documentation

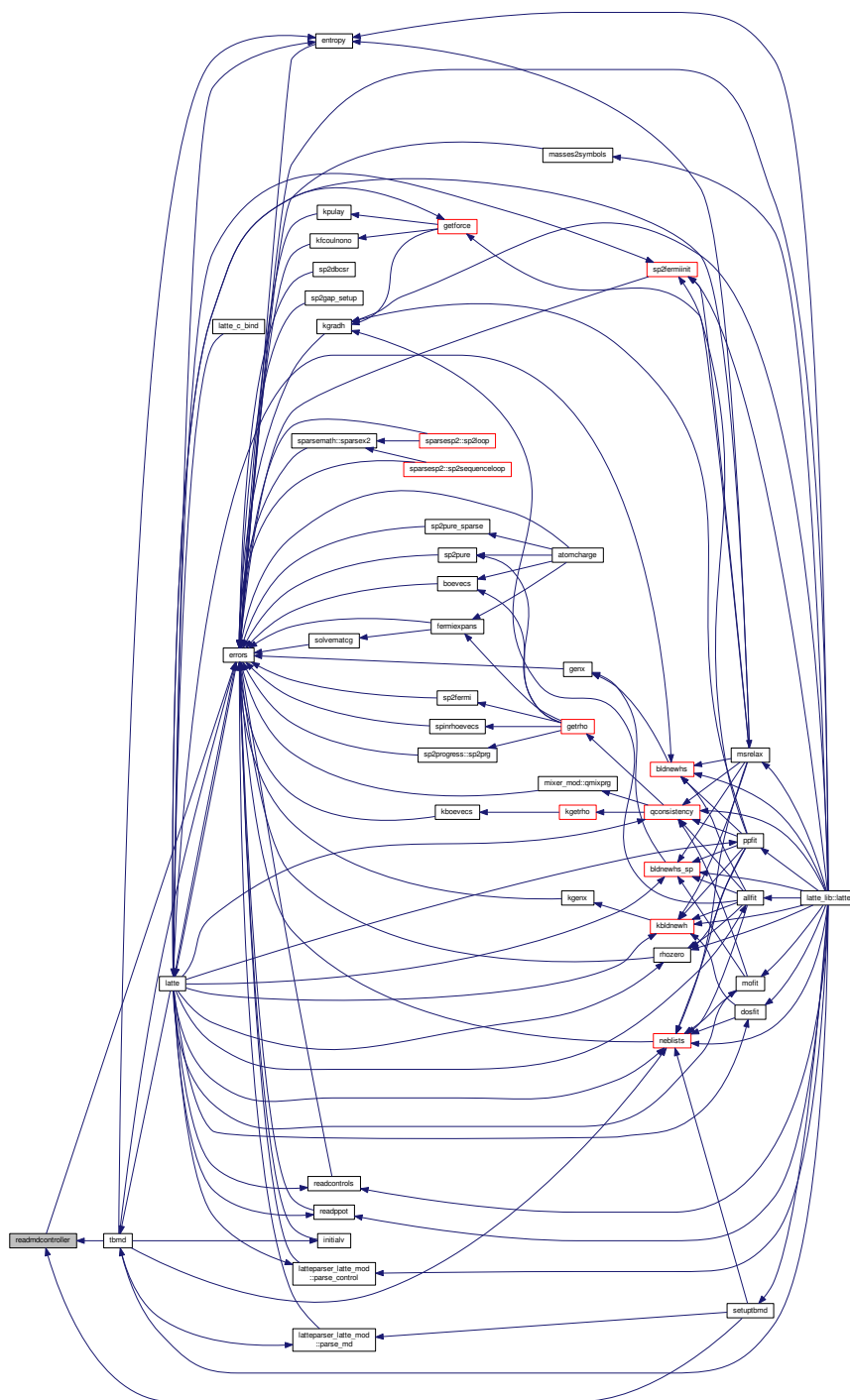
8.339.1.1 subroutine readmdcontroller ()

Definition at line 23 of file [readmdcontroller.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.340 readmdcontroller.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readmdcontroller
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE neblistarray
00028
00029   IMPLICIT NONE
00030
00031   CHARACTER(LEN=20) :: HD
00032   IF (existerror) RETURN
00033
00034   OPEN (unit=15, status="OLD", file="MDcontroller")
00035
00036   !
00037   ! MAXITER = run this many MD time steps
00038   !
00039
00040   READ(15,*) hd, maxiter
00041
00042   !
00043   ! UDNEIGH = update the neighbor lists every UDNEIGH time steps
00044   !
00045
00046   READ(15,*) hd, udneigh
00047
00048   !
00049   ! DT = size of the time step in fs
00050   !
00051
00052   READ(15,*) hd, dt
00053
00054   !
00055   ! TTARGET = temperature in K were initialize and aim for during NVT MD
00056   ! RNDIST = Type of distribution of random numbers used to initialize T
00057   !         = GAUSSIAN or UNIFORM
00058   ! SEEDINIT = Type of seed used in the generation of random numbers
00059   !         = RANDOM - seed changes every time
00060   !         = DEFAULT - use the same, default seed every time
00061   !
00062
00063   READ(15,*) hd, ttarget, hd, rndist, hd, seedinit
00064
00065   !
00066   ! DUMPFREQ: Write a dump file every DUMPFREQ time steps
00067   !
00068
00069   READ(15,*) hd, dumpfreq
00070
00071   !
00072   ! RSFREQ: Write a restart file every RSFREQ time steps
00073   !
00074
00075   READ(15,*) hd, rsfreq
00076
00077   !
00078   ! WRTFREQ: Output energy and temperature every WRTFREQ time steps
00079   !
00080
00081   READ(15,*) hd, wrtfreq
00082
00083   !
00084   ! TOINITTEMP: Whether or not we are going to initialize velocities
00085   ! using a random number generator (sometimes during a restart we
00086   ! may not want to reinitialize the temperature
00087   !
00088
00089   READ(15,*) hd, toinittemp
00090
00091   !
00092   ! THERMPER: If we're running NVT, rescale velocities every THERMPER
00093   ! time steps.

```



```

00094      !
00095
00096      READ(15,*) hd, thermper
00097
00098      !
00099      ! THERMRUN: Thermalize over this many time steps when NVT is on
00100      !
00101
00102      READ(15,*) hd, thermrun
00103
00104      !
00105      ! NVTON: 0 = running NVE MD, 1 = running NVT MD
00106      ! AVEPER: Average the temperature over AVEPER time steps when determining
00107      ! how to rescale velocities
00108      !
00109
00110      READ(15,*) hd, nvton, hd, npton, hd, aveper, hd, friction, &
00111             hd, seedth
00112
00113      IF (nvton .EQ. 1 .AND. npton .EQ. 1) THEN
00114          CALL errors("readmdcontroller", "You can't have NVTON = 1 and NPTON = 1")
00115      ENDIF
00116
00117      ! PTARGET = Target pressure (in GPa) when running NPT
00118      ! NPTTYPE = ISO or ANISO
00119
00120      READ(15,*) hd, ptarget, hd, npttype
00121
00122      !
00123      ! The following are for the Hugoniotstat
00124      !
00125
00126      ! On (1) or off (0)?
00127
00128      READ(15,*) hd, shockon
00129
00130      !
00131      ! SHOCKSTART = the MD iteration where we will start to compress
00132      ! the iteration when we stop depends on the size of the block and Us
00133      !
00134
00135      READ(15,*) hd, shockstart
00136
00137      !
00138      ! SHOCKDIR is the cartesian direction (1 = X, 2 = Y, 3 = Z),
00139      ! parallel to which we're going to compress uniaxially
00140      !
00141
00142      READ(15,*) hd, shockdir
00143
00144      !
00145      ! And finally, the particle and shock velocities
00146      ! IN UNITS OF METRES PER SECOND
00147      !
00148
00149      READ(15,*) hd, uparticle, hd, ushock, hd, c0
00150
00151      ! Adapt SCF on the fly?
00152
00153      READ(15,*) hd, mdadapt
00154
00155      ! Calculating Hugoniot points?
00156
00157      READ(15,*) hd, gethug, hd, e0, hd, v0, hd, p0
00158
00159
00160      CLOSE(15)
00161
00162      RETURN
00163
00164  END SUBROUTINE readmdcontroller

```

8.341 readppot.f90 File Reference

Functions/Subroutines

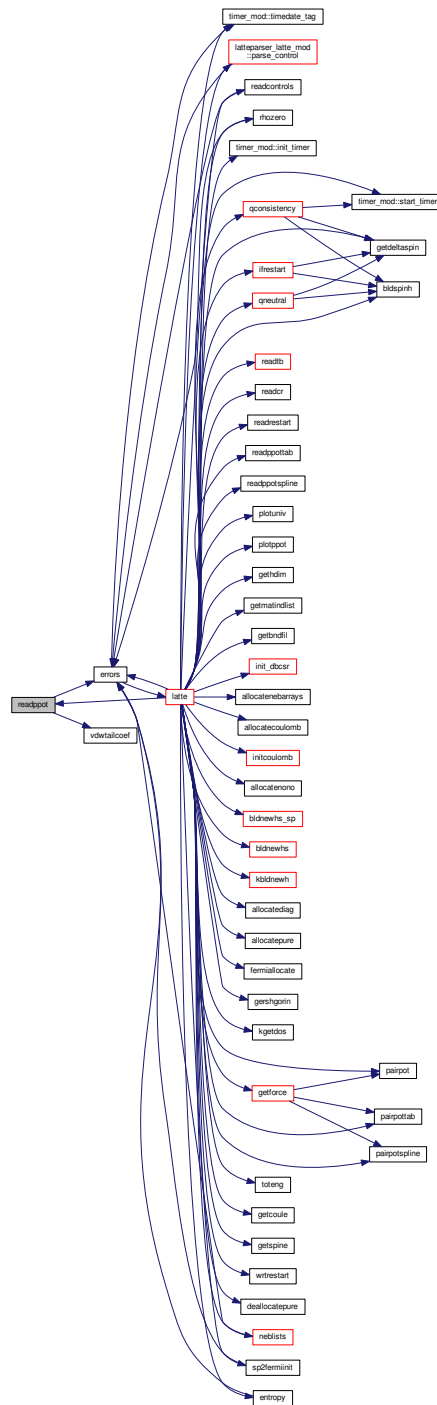
- subroutine [readppot](#) ()

8.341.1 Function/Subroutine Documentation

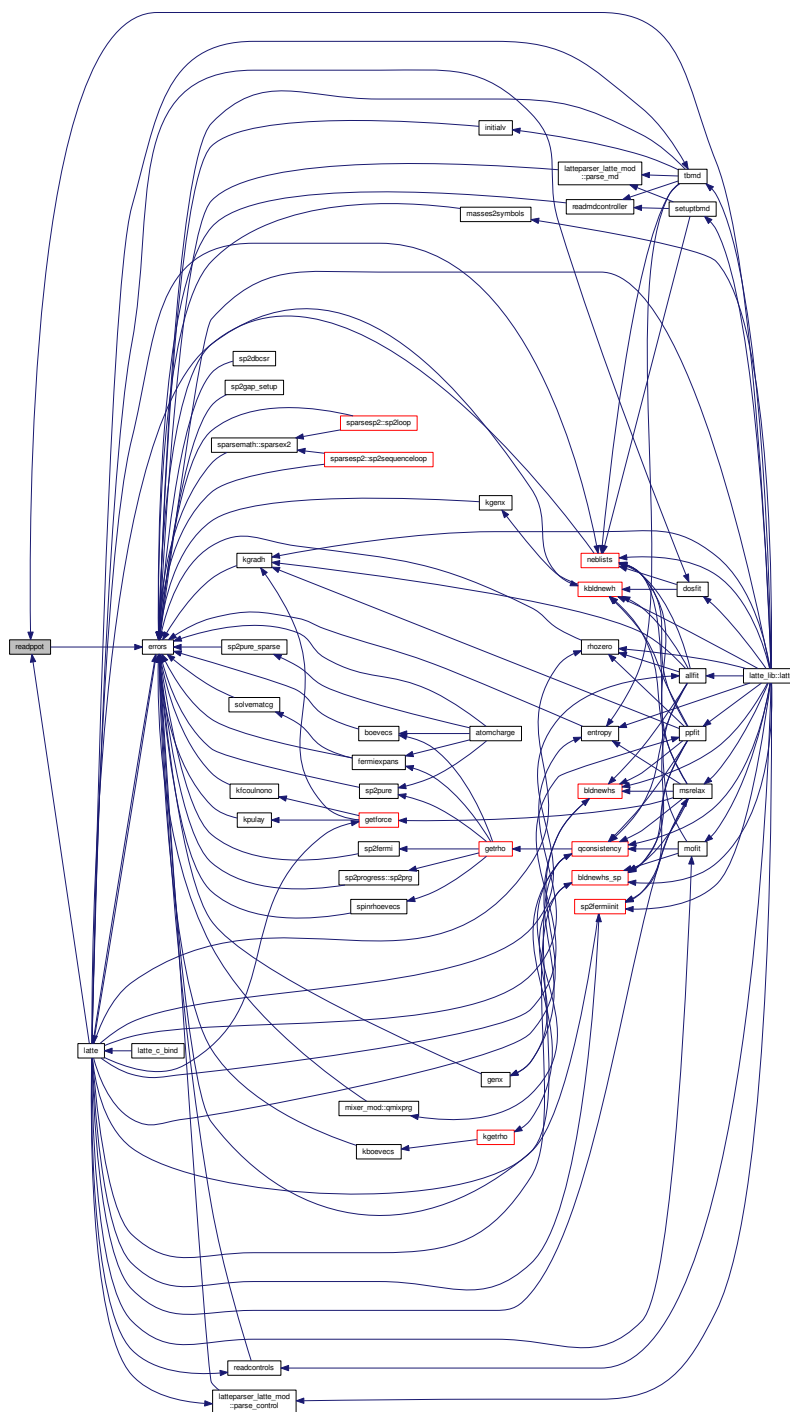
8.341.1.1 subroutine readppot ()

Definition at line 23 of file [readppot.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.342 readppot.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readppot()
00023
00024   USE constants_mod
00025   USE ppotarray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J, K
00030   CHARACTER(LEN=20) :: HD
00031   LOGICAL :: FILEEXISTS
00032
00033   IF (existererror) RETURN
00034
00035   IF (basistype .EQ. "ORTHO") THEN
00036     INQUIRE( file=trim(parampath)//"/ppots.ortho", exist=fileexists)
00037     IF (.NOT. fileexists) THEN
00038       CALL errors("readppot","ppot.ortho file does not exist. &
00039         & Please either set PPOTON= 0 or add a file for the pair potentials.")
00040     ELSE
00041       OPEN(unit=14,status="OLD", file=trim(parampath)//"/ppots.ortho")
00042     END IF
00043   ELSEIF (basistype .EQ. "NONORTHO") THEN
00044     INQUIRE( file=trim(parampath)//"/ppots.nonortho", exist=fileexists)
00045     IF (.NOT. fileexists) THEN
00046       CALL errors("readppot","ppot.ortho file does not exist. &
00047         & Please either set PPOTON= 0 or add a file for the pair potentials.")
00048     ELSE
00049       OPEN(unit=14, status="OLD", file=trim(parampath)//"/ppots.nonortho")
00050     END IF
00051   END IF
00052
00053   READ(14,*) hd, nopps
00054
00055   ! POTCOEF:
00056   ! 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
00057   ! A0 A1 A2 A3 A4 A5 A6 C R1 RCUT B1 B2 B3 B4 B5 B6
00058
00059   ! PHI = A0*EXP(A1*X + A2*X^2 + A3*X^3 + A4*X^4) + A5*EXP(A6*X) - C/X^6
00060
00061
00062   ALLOCATE(ppele1(nopps), ppele2(nopps), potcoef(16,
00063     nopps))
00064
00065   READ(14,*) hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd
00066
00067   DO i = 1, nopps
00068     READ(14,*) ppele1(i), ppele2(i), (potcoef(j,i), j = 1, 10)
00069   ENDDO
00070
00071   CLOSE(14)
00072
00073   ! Add the cut-off tails to our pair potentials
00074
00075   CALL vdwtailcoef
00076
00077
00078   RETURN
00079
00080
00081 END SUBROUTINE readppot

```

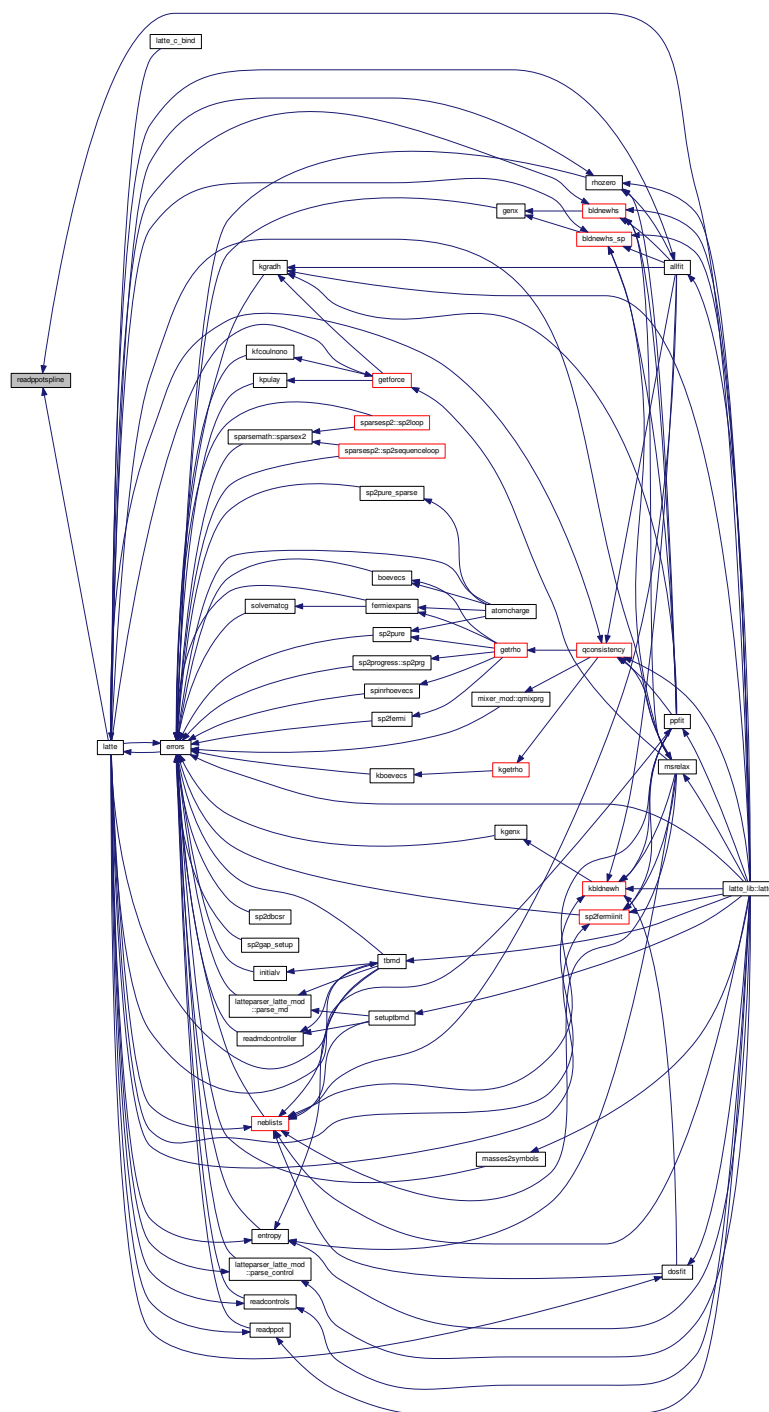
8.343 readppotspline.f90 File Reference

Functions/Subroutines

- subroutine [readppotspline](#)

8.343.1.1 subroutine readppotspline ()

Here is the caller graph for this function:



8.344 readppotspline.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readppotspline
00023
00024   USE constants_mod
00025   USE ppotarray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J, K, MAXENTRY, NUMENTRY, N
00030   REAL(LATTEPREC) :: JUNK, P, QN, SIG, UN
00031   REAL(LATTEPREC), ALLOCATABLE :: U(:)
00032   CHARACTER(LEN=20) :: HD, HD1, HD2
00033   IF (existerror) RETURN
00034
00035   OPEN(unit=14, status="OLD", file=trim(parampath)//"/ppots.spline")
00036
00037   READ(14,*) nopps
00038
00039   ! Figure out array dimensions for allocation
00040
00041   maxentry = 0
00042   DO i = 1, nopps
00043     READ(14,*) hd1, hd2
00044     READ(14,*) numentry
00045
00046     IF (numentry .GT. maxentry) maxentry = numentry
00047
00048     DO j = 1, numentry
00049       READ(14,*) junk, junk
00050     ENDDO
00051   ENDDO
00052
00053   rewind(14)
00054
00055   ! print*, MAXENTRY
00056   ! Now we can allocate
00057
00058   ALLOCATE(ppele1(nopps), ppele2(nopps), pprk(maxentry,
nopps), &
00059     ppak(maxentry,nopps), ppnk(nopps))
00060
00061   pprk = zero
00062   ppak = zero
00063
00064   READ(14,*) nopps
00065   DO i = 1, nopps
00066     READ(14,*) ppele1(i), ppele2(i)
00067     READ(14,*) ppnk(i)
00068     DO j = 1, ppnk(i)
00069       READ(14,*) pprk(j,i), ppak(j,i)
00070     ENDDO
00071   ENDDO
00072
00073   ! print *, ppak(1,1)
00074   CLOSE(14)
00075
00076   RETURN
00077
00078 END SUBROUTINE readppotspline

```

8.345 readppottab.f90 File Reference

Functions/Subroutines

- subroutine [readppottab](#)

8.345.1 Function/Subroutine Documentation

8.345.1.1 subroutine readppottab ()

Definition at line 23 of file [readppottab.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.     !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readppottab
00023
00024   USE constants_mod
00025   USE ppotarray
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I, J, K, MAXENTRY, NUMENTRY, N
00030   REAL(LATTEPREC) :: JUNK, P, QN, SIG, UN
00031   REAL(LATTEPREC), ALLOCATABLE :: U(:)
00032   CHARACTER(LEN=20) :: HD, HD1, HD2
00033   IF (existerror) RETURN
00034
00035   OPEN(unit=14,status="OLD", file=trim(parampath)//"/ppots.dftb")
00036
00037   READ(14,*) nopps
00038
00039   ! Figure out array dimensions for allocation
00040
00041   maxentry = 0
00042   DO i = 1, nopps
00043     READ(14,*) hd1, hd2
00044     READ(14,*) numentry
00045
00046     IF (numentry .GT. maxentry) maxentry = numentry
00047
00048     DO j = 1, numentry
00049       READ(14,*) junk, junk
00050     ENDDO
00051   ENDDO
00052
00053   rewind(14)
00054
00055   ! print*, MAXENTRY
00056   ! Now we can allocate
00057
00058   ALLOCATE(ppele1(nopps), ppele2(nopps), ppr(maxentry,
00059     nopps), &
00059     ppval(maxentry,nopps))
00060   ALLOCATE(pptablenth(nopps), ppspl(maxentry,nopps))
00061
00062   ppr = zero
00063   ppval = zero
00064
00065   READ(14,*) nopps
00066   DO i = 1, nopps
00067     READ(14,*) ppele1(i), ppele2(i)
00068     READ(14,*) pptablenth(i)
00069     DO j = 1, pptablenth(i)
00070       READ(14,*) ppr(j,i), ppval(j,i)
00071     ENDDO
00072   ENDDO
00073
00074   CLOSE(14)
00075
00076   ! Let's get the cubic spline coefficients
00077
00078   ALLOCATE(u(maxentry))
00079
00080   DO i = 1, nopps
00081
00082     n = pptablenth(i)
00083
00084     ppspl(1,i) = zero
00085     u(1) = zero
00086
00087     DO j = 2, n-1
00088       sig = (ppr(j,i) - ppr(j-1,i))/(ppr(j+1,i) - ppr(j-1,i))
00089       p = sig*ppspl(j-1,i) + two
00090       ppspl(j,i) = (sig - one)/p
00091       u(j) = (six*((ppval(j+1,i) - ppval(j,i)) / &
00092         (ppr(j+1,i) - ppr(j,i)) - (ppval(j,i) - ppval(j-1,i)) &

```

```

00093          / (ppr(j,i) - ppr(j-1,i)) / (ppr(j+1,i) - ppr(j-1,i)) &
00094          - sig*u(j-1))/p
00095      ENDDO
00096
00097      qn = zero
00098      un = zero
00099
00100      ppspl(n,i) = (un - qn*u(n-1)) / (qn*ppspl(n-1,i) + one)
00101
00102      DO k = n-1, 1, -1
00103          ppspl(k,i) = ppspl(k,i)*ppspl(k+1,i) + u(k)
00104      ENDDO
00105
00106      ENDDO
00107
00108      DEALLOCATE(u)
00109
00110      ! Test the interpolation
00111
00112      ! CALL TABTEST
00113
00114      RETURN
00115
00116 END SUBROUTINE readppottab

```

8.347 readrestart.f90 File Reference

Functions/Subroutines

- subroutine [readrestart](#)

8.347.1 Function/Subroutine Documentation

8.347.1.1 subroutine readrestart ()

Definition at line 23 of file [readrestart.f90](#).

[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readrestart
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE restartarray
00028   USE kspacearray
00029   #ifdef MPI_ON
00030     USE mpi
00031   #endif
00032   USE myprecision
00033
00034   IMPLICIT NONE
00035
00036   INTEGER :: I, J, MYINDEX
00037   INTEGER :: MYID, IERR
00038   REAL(LATTEPREC) :: LN(6), JUNK, TMP1, TMP2
00039   CHARACTER(LEN=20) :: HEADER
00040   CHARACTER(LEN=50) :: FLNM
00041   IF (existerror) RETURN
00042
00043
00044   IF (mdon .EQ. 1 .AND. parrep .EQ. 1) THEN
00045
00046     #ifdef MPI_ON
00047       CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00048     #endif
00049
00050
00051     IF (myid .LT. 10) THEN
00052       WRITE(flnm,'(I1,"/bl/restart.dat")') myid
00053     ELSEIF (myid .GE. 10 .AND. myid .LT. 100) THEN
00054       WRITE(flnm,'(I2,"/bl/restart.dat")') myid
00055     ELSEIF (myid .GE. 100 .AND. myid .LT. 1000) THEN
00056       WRITE(flnm,'(I3,"/bl/restart.dat")') myid
00057     ENDIF
00058
00059     print*, myid, flnm
00060
00061     OPEN(unit=12, status="OLD", file=flnm)
00062
00063   ELSE
00064
00065     ! Regular restart
00066
00067     OPEN(unit=12, status="OLD", file="bl/restart.dat")
00068
00069   ENDIF
00070
00071   IF (mdon .EQ. 1) READ(12,*) header, contiter
00072   READ(12,*) nats
00073
00074   ALLOCATE(cr(3,nats), atele(nats), f(3,nats), fpp(3,nats),
00075 ftot(3,nats))
00076   ALLOCATE(deltaq(nats), mycharge(nats))
00077   ALLOCATE(elpointer(nats))
00078
00079   IF (electro .EQ. 0) THEN
00080     ALLOCATE(lcnshift(nats))
00081     lcnshift = zero
00082   ENDIF
00083
00084   IF (basistype .EQ. "NONORTHO") THEN
00085     IF (spinon .EQ. 0) THEN
00086       ALLOCATE(fpul(3,nats), fscoul(3,nats))
00087     ELSE
00088       ALLOCATE(fpul(3,nats), fscoul(3,nats), fsspin(3,
00089 nats))
00090     ENDIF
00091   ENDIF
00092
00093   IF (kon .EQ. 1) ALLOCATE(kf(3,nats))

```

```

00092
00093 IF (mdon .EQ. 1) ALLOCATE(v(3,nats))
00094
00095 READ(12,*) box(1,1), box(1,2), box(1,3)
00096 READ(12,*) box(2,1), box(2,2), box(2,3)
00097 READ(12,*) box(3,1), box(3,2), box(3,3)
00098
00099
00100 ! READ(12,*) (ATELE(I), CR(1,I), CR(2,I), CR(3,I), I = 1, NATS)
00101
00102 DO i = 1, nats
00103     READ(12,*) atele(i), cr(1,i), cr(2,i), cr(3,i)
00104 ENDDO
00105
00106
00107 ! Set up pointer to the data in TBparam/electrons.dat
00108
00109 DO i = 1, nats
00110     DO j = 1, noelem
00111         IF (atele(i) .EQ. ele(j)) elempointer(i) = j
00112     ENDDO
00113 ENDDO
00114
00115 summass = zero
00116 DO i = 1, nats
00117     summass = summass + mass(elempointer(i))
00118 ENDDO
00119
00120 ! Let's check whether we have only sp elements. If so, we can
00121 ! use a much faster version of gradH
00122
00123 ! SPONLY = 0: use GRADHSP
00124 ! SPONLY = 1: use Josh Coe's implementation of the automatic H build
00125
00126 sponly = 0
00127 DO i = 1, nats
00128     IF (basis(elempointer(i)) .NE. "s" .AND. &
00129         basis(elempointer(i)) .NE. "sp") sponly = 1
00130 ENDDO
00131
00132
00133 READ(12,*) chempot
00134 ! PRINT*, CHEMPOT
00135
00136 IF (spinon .EQ. 0) THEN
00137
00138     READ(12,*) tmphdim
00139     ALLOCATE(tmpbodiag(tmphdim))
00140
00141     DO i = 1, tmphdim
00142         READ(12,*) tmp1, tmp2
00143         tmpbodiag(i) = tmp1 + tmp2
00144     ENDDO
00145
00146 ELSEIF (spinon .EQ. 1) THEN
00147
00148     READ(12,*) tmphdim
00149     ALLOCATE(tmprhoup(tmphdim), tmprhodown(tmphdim))
00150     READ(12,*) (tmprhoup(i), tmprhodown(i), i = 1, tmphdim)
00151
00152 ENDIF
00153
00154 IF (mdon .EQ. 1) THEN
00155     DO i = 1, nats
00156         READ(12,*) v(1,i), v(2,i), v(3,i)
00157     ENDDO
00158 ENDIF
00159
00160 CLOSE(12)
00161
00162 ! CR = ALAT * CR
00163
00164 IF (pbcon .EQ. 0) THEN
00165
00166     IF (contiter .LT. 10) THEN
00167         WRITE(flnm, '("animate/myXYZfile_restart.",I1,".xyz")') contiter
00168     ELSEIF (contiter .GE. 10 .AND. contiter .LT. 100) THEN
00169         WRITE(flnm, '("animate/myXYZfile_restart.",I2,".xyz")') contiter
00170     ELSEIF (contiter .GE. 100 .AND. contiter .LT. 1000) THEN
00171         WRITE(flnm, '("animate/myXYZfile_restart.",I3,".xyz")') contiter
00172     ELSEIF (contiter .GE. 1000 .AND. contiter .LT. 10000) THEN
00173         WRITE(flnm, '("animate/myXYZfile_restart.",I4,".xyz")') contiter
00174     ELSEIF (contiter .GE. 10000 .AND. contiter .LT. 100000) THEN
00175         WRITE(flnm, '("animate/myXYZfile_restart.",I5,".xyz")') contiter
00176     ELSEIF (contiter .GE. 100000 .AND. contiter .LT. 1000000) THEN
00177         WRITE(flnm, '("animate/myXYZfile_restart.",I6,".xyz")') contiter
00178     ELSEIF (contiter .GE. 1000000 .AND. contiter .LT. 10000000) THEN

```

```

00179         WRITE (flnm, ' ("animate/myXYZfile_restart.", I7, ".xyz")') contiter
00180     ENDIF
00181
00182     OPEN (unit=23, status="UNKNOWN", file=flnm)
00183
00184 ENDIF
00185
00186
00187 RETURN
00188
00189 END SUBROUTINE readrestart

```

8.349 readrestartlib.f90 File Reference

Functions/Subroutines

- subroutine [readrestartlib](#) (ITER)

8.349.1 Function/Subroutine Documentation

8.349.1.1 subroutine readrestartlib (integer *ITER*)

Definition at line 23 of file [readrestartlib.f90](#).

Here is the caller graph for this function:



8.350 readrestartlib.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readrestartlib(ITER)
00023

```

```

00024 USE constants_mod
00025 USE kspacearray
00026 USE neblistarray
00027 USE ppotarray
00028 USE restartarray
00029 USE sparsearray
00030 USE univarray
00031 USE xboarray
00032 USE coulombarray
00033 USE diagarray
00034 USE mdarray
00035 USE nonoarray
00036 USE purearray
00037 USE setuparray
00038 USE spinarray
00039 USE virialarray
00040 #ifdef MPI_ON
00041 USE mpi
00042 #endif
00043
00044 IMPLICIT NONE
00045
00046 INTEGER :: I, K, ITER
00047 INTEGER :: MYID, IERR, DUMMYN, DUMMYN1, DUMMYN2
00048 INTEGER :: CHARACTERCHECK(9), COMPLEXCHECK(15), INTEGERCHECK(20), REALCHECK(98)
00049 INTEGER :: CHECK(25)
00050 REAL(LATTEPREC) :: TMP1, TMP2
00051 REAL(LATTEPREC), ALLOCATABLE :: TMPBODIAGUP(:), TMPBODIAGDOWN(:)
00052 COMPLEX(LATTEPREC) :: KSUM
00053 CHARACTER(LEN=100) :: FLNM, NAME
00054 IF (existerror) RETURN
00055
00056 IF (verbose >= 1) write(*,*) "Restarting latte_lib from file ..."
00057
00058 OPEN(14, file="restart.latte.dat", form="UNFORMATTED", status="OLD")
00059
00060 READ(14) charactercheck
00061 READ(14) complexcheck
00062 READ(14) integercheck
00063 READ(14) realcheck
00064
00065 READ(14) iter
00066
00067 write(*,*) "ITER", iter
00068 libcalls = iter
00069
00070 IF(charactercheck( 2 ) == 1) THEN
00071   READ(14) dummyN
00072   IF(.NOT.ALLOCATED( atele )) ALLOCATE( atele(dummyN))
00073   READ(14) atele
00074 ENDIF
00075
00076 IF(charactercheck( 3 ) == 1) THEN
00077   READ(14) dummyN
00078   IF(.NOT.ALLOCATED( basis )) ALLOCATE( basis(dummyN))
00079   READ(14) basis
00080 ENDIF
00081
00082 IF(charactercheck( 4 ) == 1) THEN
00083   READ(14) dummyN
00084   IF(.NOT.ALLOCATED( btype )) ALLOCATE( btype(dummyN))
00085   READ(14) btype
00086 ENDIF
00087
00088 IF(charactercheck( 5 ) == 1) THEN
00089   READ(14) dummyN
00090   IF(.NOT.ALLOCATED( ele )) ALLOCATE( ele(dummyN))
00091   READ(14) ele
00092 ENDIF
00093
00094 IF(charactercheck( 6 ) == 1) THEN
00095   READ(14) dummyN
00096   IF(.NOT.ALLOCATED( ele1 )) ALLOCATE( ele1(dummyN))
00097   READ(14) ele1
00098 ENDIF
00099
00100 IF(charactercheck( 7 ) == 1) THEN
00101   READ(14) dummyN
00102   IF(.NOT.ALLOCATED( ele2 )) ALLOCATE( ele2(dummyN))
00103   READ(14) ele2
00104 ENDIF
00105
00106 IF(charactercheck( 8 ) == 1) THEN
00107   READ(14) dummyN
00108   IF(.NOT.ALLOCATED( ppele1 )) ALLOCATE( ppele1(dummyN))
00109   READ(14) ppele1
00110 ENDIF

```

```

00111
00112 IF(charactercheck( 9 ) == 1)THEN
00113   READ(14) dummyn
00114   IF(.NOT.ALLOCATED( ppele2 )) ALLOCATE( ppele2(dummyn) )
00115   READ(14) ppele2
00116 ENDIF
00117
00118 IF(complexcheck( 2 ) == 1)THEN
00119   READ(14) dummyn
00120   IF(.NOT.ALLOCATED( diag_zwork )) ALLOCATE( diag_zwork(dummyn) )
00121   READ(14) diag_zwork
00122 ENDIF
00123
00124 IF(complexcheck( 3 ) == 1)THEN
00125   READ(14) dummyn ,dummysn1 ,dummysn2
00126   IF(.NOT.ALLOCATED( hk )) ALLOCATE( hk(dummyn,dummysn1,dummysn2) )
00127   READ(14) hk
00128 ENDIF
00129
00130 IF(complexcheck( 4 ) == 1)THEN
00131   READ(14) dummyn ,dummysn1 ,dummysn2
00132   IF(.NOT.ALLOCATED( hk0 )) ALLOCATE( hk0(dummyn,dummysn1,dummysn2) )
00133   READ(14) hk0
00134 ENDIF
00135
00136 IF(complexcheck( 5 ) == 1)THEN
00137   READ(14) dummyn ,dummysn1
00138   IF(.NOT.ALLOCATED( hkdiag )) ALLOCATE( hkdiag(dummyn,dummysn1) )
00139   READ(14) hkdiag
00140 ENDIF
00141
00142 IF(complexcheck( 6 ) == 1)THEN
00143   READ(14) dummyn ,dummysn1 ,dummysn2
00144   IF(.NOT.ALLOCATED( kbo )) ALLOCATE( kbo(dummyn,dummysn1,dummysn2) )
00145   READ(14) kbo
00146 ENDIF
00147
00148 IF(complexcheck( 7 ) == 1)THEN
00149   READ(14) dummyn ,dummysn1 ,dummysn2
00150   IF(.NOT.ALLOCATED( kevecs )) ALLOCATE( kevecs(dummyn,dummysn1,dummysn2) )
00151   READ(14) kevecs
00152 ENDIF
00153
00154 IF(complexcheck( 8 ) == 1)THEN
00155   READ(14) dummyn ,dummysn1
00156   IF(.NOT.ALLOCATED( kf )) ALLOCATE( kf(dummyn,dummysn1) )
00157   READ(14) kf
00158 ENDIF
00159
00160 IF(complexcheck( 9 ) == 1)THEN
00161   READ(14) dummyn ,dummysn1
00162   IF(.NOT.ALLOCATED( khtmp )) ALLOCATE( khtmp(dummyn,dummysn1) )
00163   READ(14) khtmp
00164 ENDIF
00165
00166 IF(complexcheck( 10 ) == 1)THEN
00167   READ(14) dummyn ,dummysn1 ,dummysn2
00168   IF(.NOT.ALLOCATED( korthoh )) ALLOCATE( korthoh(dummyn,dummysn1,dummysn2) )
00169   READ(14) korthoh
00170 ENDIF
00171
00172 IF(complexcheck( 11 ) == 1)THEN
00173   READ(14) dummyn ,dummysn1 ,dummysn2
00174   IF(.NOT.ALLOCATED( kxmat )) ALLOCATE( kxmat(dummyn,dummysn1,dummysn2) )
00175   READ(14) kxmat
00176 ENDIF
00177
00178 IF(complexcheck( 12 ) == 1)THEN
00179   READ(14) dummyn ,dummysn1 ,dummysn2
00180   IF(.NOT.ALLOCATED( sk )) ALLOCATE( sk(dummyn,dummysn1,dummysn2) )
00181   READ(14) sk
00182 ENDIF
00183
00184 IF(complexcheck( 13 ) == 1)THEN
00185   READ(14) dummyn ,dummysn1
00186   IF(.NOT.ALLOCATED( zbo )) ALLOCATE( zbo(dummyn,dummysn1) )
00187   READ(14) zbo
00188 ENDIF
00189
00190 IF(complexcheck( 14 ) == 1)THEN
00191   READ(14) dummyn
00192   IF(.NOT.ALLOCATED( zheevd_work )) ALLOCATE( zheevd_work(dummyn) )
00193   READ(14) zheevd_work
00194 ENDIF
00195
00196 IF(complexcheck( 15 ) == 1)THEN
00197   READ(14) dummyn

```



```

00198         IF (.NOT.ALLOCATED( zhjj )) ALLOCATE( zhjj(dummysn))
00199         READ(14) zhjj
00200     ENDIF
00201
00202     IF(integercheck( 2 ) == 1)THEN
00203         READ(14) dummysn
00204         IF (.NOT.ALLOCATED( diag_iwork )) ALLOCATE( diag_iwork(dummysn))
00205         READ(14) diag_iwork
00206     ENDIF
00207
00208     IF(integercheck( 3 ) == 1)THEN
00209         READ(14) dummysn
00210         IF (.NOT.ALLOCATED( elempointer )) ALLOCATE( elempointer(dummysn))
00211         READ(14) elempointer
00212     ENDIF
00213
00214     IF(integercheck( 4 ) == 1)THEN
00215         READ(14) dummysn
00216         IF (.NOT.ALLOCATED( ifail )) ALLOCATE( ifail(dummysn))
00217         READ(14) ifail
00218     ENDIF
00219
00220     IF(integercheck( 5 ) == 1)THEN
00221         READ(14) dummysn
00222         IF (.NOT.ALLOCATED( matindlist )) ALLOCATE( matindlist(dummysn))
00223         READ(14) matindlist
00224     ENDIF
00225
00226     IF(integercheck( 6 ) == 1)THEN
00227         READ(14) dummysn
00228         IF (.NOT.ALLOCATED( molid )) ALLOCATE( molid(dummysn))
00229         READ(14) molid
00230     ENDIF
00231
00232     IF(integercheck( 7 ) == 1)THEN
00233         READ(14) dummysn ,dummysn1 ,dummysn2
00234         IF(ALLOCATED( nebcoul )) DEALLOCATE(nebcoul)
00235         ALLOCATE( nebcoul(dummysn,dummysn1,dummysn2))
00236         READ(14) nebcoul
00237     ENDIF
00238
00239     IF(integercheck( 8 ) == 1)THEN
00240         READ(14) dummysn ,dummysn1 ,dummysn2
00241         IF(ALLOCATED( nebpp )) DEALLOCATE( nebpp )
00242         ALLOCATE( nebpp(dummysn,dummysn1,dummysn2))
00243         READ(14) nebpp
00244     ENDIF
00245
00246     IF(integercheck( 9 ) == 1)THEN
00247         READ(14) dummysn ,dummysn1 ,dummysn2
00248         IF(ALLOCATED( nebtb )) DEALLOCATE( nebtb )
00249         ALLOCATE( nebtb(dummysn,dummysn1,dummysn2))
00250         READ(14) nebtb
00251     ENDIF
00252
00253     IF(integercheck( 10 ) == 1)THEN
00254         READ(14) dummysn
00255         IF (.NOT.ALLOCATED( nono_iwork )) ALLOCATE( nono_iwork(dummysn))
00256         READ(14) nono_iwork
00257     ENDIF
00258
00259     IF(integercheck( 11 ) == 1)THEN
00260         READ(14) dummysn
00261         IF (.NOT.ALLOCATED( pptablenth )) ALLOCATE( pptablenth(dummysn))
00262         READ(14) pptablenth
00263     ENDIF
00264
00265     IF(integercheck( 12 ) == 1)THEN
00266         READ(14) dummysn
00267         IF (.NOT.ALLOCATED( rx )) ALLOCATE( rx(dummysn))
00268         READ(14) rx
00269     ENDIF
00270
00271     IF(integercheck( 13 ) == 1)THEN
00272         READ(14) dummysn
00273         IF (.NOT.ALLOCATED( rxtmp )) ALLOCATE( rxtmp(dummysn))
00274         READ(14) rxtmp
00275     ENDIF
00276
00277     IF(integercheck( 14 ) == 1)THEN
00278         READ(14) dummysn
00279         IF (.NOT.ALLOCATED( signlist )) ALLOCATE( signlist(dummysn))
00280         READ(14) signlist
00281     ENDIF
00282
00283     IF(integercheck( 15 ) == 1)THEN
00284         READ(14) dummysn

```

```

00285         IF (.NOT.ALLOCATED( spinindlist )) ALLOCATE( spinindlist (dummyn) )
00286         READ(14) spinindlist
00287     ENDIF
00288
00289     IF(integercheck( 16 ) == 1) THEN
00290         READ(14) dummyn
00291         IF (.NOT.ALLOCATED( totnebcoul )) ALLOCATE( totnebcoul (dummyn) )
00292         READ(14) totnebcoul
00293     ENDIF
00294
00295     IF(integercheck( 17 ) == 1) THEN
00296         READ(14) dummyn
00297         IF (.NOT.ALLOCATED( totnebpp )) ALLOCATE( totnebpp (dummyn) )
00298         READ(14) totnebpp
00299     ENDIF
00300
00301     IF(integercheck( 18 ) == 1) THEN
00302         READ(14) dummyn
00303         IF (.NOT.ALLOCATED( totnebtb )) ALLOCATE( totnebtb (dummyn) )
00304         READ(14) totnebtb
00305     ENDIF
00306
00307     IF(integercheck( 19 ) == 1) THEN
00308         READ(14) dummyn
00309         IF (.NOT.ALLOCATED( xb )) ALLOCATE( xb (dummyn) )
00310         READ(14) xb
00311     ENDIF
00312
00313     IF(integercheck( 20 ) == 1) THEN
00314         READ(14) dummyn
00315         IF (.NOT.ALLOCATED( zheevd_iwork )) ALLOCATE( zheevd_iwork (dummyn) )
00316         READ(14) zheevd_iwork
00317     ENDIF
00318
00319     IF(realcheck( 2 ) == 1) THEN
00320         READ(14) dummyn
00321         IF (.NOT.ALLOCATED( atocc )) ALLOCATE( atocc (dummyn) )
00322         READ(14) atocc
00323     ENDIF
00324
00325     IF(realcheck( 3 ) == 1) THEN
00326         READ(14) dummyn ,dummysn1
00327         IF (.NOT.ALLOCATED( bo )) ALLOCATE( bo (dummyn,dummysn1) )
00328         READ(14) bo
00329     ENDIF
00330
00331     IF(realcheck( 4 ) == 1) THEN
00332         READ(14) dummyn ,dummysn1
00333         IF (.NOT.ALLOCATED( bond )) ALLOCATE( bond (dummyn,dummysn1) )
00334         READ(14) bond
00335     ENDIF
00336
00337     IF(realcheck( 5 ) == 1) THEN
00338         READ(14) dummyn
00339         IF (.NOT.ALLOCATED( bozero )) ALLOCATE( bozero (dummyn) )
00340         READ(14) bozero
00341     ENDIF
00342
00343     IF(realcheck( 6 ) == 1) THEN
00344         READ(14) dummyn ,dummysn1
00345         IF (.NOT.ALLOCATED( bo_padded )) ALLOCATE( bo_padded (dummyn,dummysn1) )
00346         READ(14) bo_padded
00347     ENDIF
00348
00349     IF(realcheck( 7 ) == 1) THEN
00350         READ(14) dummyn
00351         IF (.NOT.ALLOCATED( chempot_pnk )) ALLOCATE( chempot_pnk (dummyn) )
00352         READ(14) chempot_pnk
00353     ENDIF
00354
00355     IF(realcheck( 8 ) == 1) THEN
00356         READ(14) dummyn
00357         IF (.NOT.ALLOCATED( coslist )) ALLOCATE( coslist (dummyn) )
00358         READ(14) coslist
00359     ENDIF
00360
00361     IF(realcheck( 9 ) == 1) THEN
00362         READ(14) dummyn
00363         IF (.NOT.ALLOCATED( coulombv )) ALLOCATE( coulombv (dummyn) )
00364         READ(14) coulombv
00365     ENDIF
00366
00367     IF(realcheck( 10 ) == 1) THEN
00368         READ(14) dummyn
00369         IF (.NOT.ALLOCATED( cplist )) ALLOCATE( cplist (dummyn) )
00370         READ(14) cplist
00371     ENDIF

```

```

00372
00373 IF(realcheck( 11 ) == 1) THEN
00374   READ(14) dummyyn , dummyyn1
00375   IF(.NOT.ALLOCATED( cr )) ALLOCATE( cr(dummyyn, dummyyn1))
00376   READ(14) cr
00377 ENDIF
00378
00379 IF(realcheck( 12 ) == 1) THEN
00380   READ(14) dummyyn
00381   IF(.NOT.ALLOCATED( deltaq )) ALLOCATE( deltaq(dummyyn))
00382   READ(14) deltaq
00383 ENDIF
00384
00385 IF(realcheck( 13 ) == 1) THEN
00386   READ(14) dummyyn
00387   IF(.NOT.ALLOCATED( deltaspin )) ALLOCATE( deltaspin(dummyyn))
00388   READ(14) deltaspin
00389 ENDIF
00390
00391 IF(realcheck( 14 ) == 1) THEN
00392   READ(14) dummyyn
00393   IF(.NOT.ALLOCATED( diag_rwork )) ALLOCATE( diag_rwork(dummyyn))
00394   READ(14) diag_rwork
00395 ENDIF
00396
00397 IF(realcheck( 15 ) == 1) THEN
00398   READ(14) dummyyn
00399   IF(.NOT.ALLOCATED( diag_work )) ALLOCATE( diag_work(dummyyn))
00400   READ(14) diag_work
00401 ENDIF
00402
00403 IF(realcheck( 16 ) == 1) THEN
00404   READ(14) dummyyn
00405   IF(.NOT.ALLOCATED( downevals )) ALLOCATE( downevals(dummyyn))
00406   READ(14) downevals
00407 ENDIF
00408
00409 IF(realcheck( 17 ) == 1) THEN
00410   READ(14) dummyyn , dummyyn1
00411   IF(.NOT.ALLOCATED( downvecs )) ALLOCATE( downvecs(dummyyn, dummyyn1))
00412   READ(14) downvecs
00413 ENDIF
00414
00415 IF(realcheck( 18 ) == 1) THEN
00416   READ(14) dummyyn
00417   IF(.NOT.ALLOCATED( ehist )) ALLOCATE( ehist(dummyyn))
00418   READ(14) ehist
00419 ENDIF
00420
00421 IF(realcheck( 19 ) == 1) THEN
00422   READ(14) dummyyn
00423   IF(.NOT.ALLOCATED( evals )) ALLOCATE( evals(dummyyn))
00424   READ(14) evals
00425 ENDIF
00426
00427 IF(realcheck( 20 ) == 1) THEN
00428   READ(14) dummyyn , dummyyn1
00429   IF(.NOT.ALLOCATED( evecs )) ALLOCATE( evecs(dummyyn, dummyyn1))
00430   READ(14) evecs
00431 ENDIF
00432
00433 IF(realcheck( 21 ) == 1) THEN
00434   READ(14) dummyyn , dummyyn1
00435   IF(.NOT.ALLOCATED( f )) ALLOCATE( f(dummyyn, dummyyn1))
00436   READ(14) f
00437 ENDIF
00438
00439 IF(realcheck( 22 ) == 1) THEN
00440   READ(14) dummyyn , dummyyn1
00441   IF(.NOT.ALLOCATED( fcoul )) ALLOCATE( fcoul(dummyyn, dummyyn1))
00442   READ(14) fcoul
00443 ENDIF
00444
00445 IF(realcheck( 23 ) == 1) THEN
00446   READ(14) dummyyn , dummyyn1
00447   IF(.NOT.ALLOCATED( fpp )) ALLOCATE( fpp(dummyyn, dummyyn1))
00448   READ(14) fpp
00449 ENDIF
00450
00451 IF(realcheck( 24 ) == 1) THEN
00452   READ(14) dummyyn , dummyyn1
00453   IF(.NOT.ALLOCATED( fpul )) ALLOCATE( fpul(dummyyn, dummyyn1))
00454   READ(14) fpul
00455 ENDIF
00456
00457 IF(realcheck( 25 ) == 1) THEN
00458   READ(14) dummyyn , dummyyn1

```

```

00459         IF (.NOT.ALLOCATED( franprev )) ALLOCATE( franprev(dummy, dummy1))
00460         READ(14) franprev
00461     ENDIF
00462
00463     IF(realcheck( 26 ) == 1) THEN
00464         READ(14) dummy, dummy1
00465         IF (.NOT.ALLOCATED( fscoul )) ALLOCATE( fscoul(dummy, dummy1))
00466         READ(14) fscoul
00467     ENDIF
00468
00469     IF(realcheck( 27 ) == 1) THEN
00470         READ(14) dummy, dummy1
00471         IF (.NOT.ALLOCATED( fsspin )) ALLOCATE( fsspin(dummy, dummy1))
00472         READ(14) fsspin
00473     ENDIF
00474
00475     IF(realcheck( 28 ) == 1) THEN
00476         READ(14) dummy, dummy1
00477         IF (.NOT.ALLOCATED( ftot )) ALLOCATE( ftot(dummy, dummy1))
00478         READ(14) ftot
00479     ENDIF
00480
00481     IF(realcheck( 29 ) == 1) THEN
00482         READ(14) dummy, dummy1
00483         IF (.NOT.ALLOCATED( h )) ALLOCATE( h(dummy, dummy1))
00484         READ(14) h
00485     ENDIF
00486
00487     IF(realcheck( 30 ) == 1) THEN
00488         READ(14) dummy, dummy1
00489         IF (.NOT.ALLOCATED( h0 )) ALLOCATE( h0(dummy, dummy1))
00490         READ(14) h0
00491     ENDIF
00492
00493     IF(realcheck( 31 ) == 1) THEN
00494         READ(14) dummy
00495         IF (.NOT.ALLOCATED( h2vect )) ALLOCATE( h2vect(dummy))
00496         READ(14) h2vect
00497     ENDIF
00498
00499     IF(realcheck( 32 ) == 1) THEN
00500         READ(14) dummy
00501         IF (.NOT.ALLOCATED( hdiag )) ALLOCATE( hdiag(dummy))
00502         READ(14) hdiag
00503     ENDIF
00504
00505     IF(realcheck( 33 ) == 1) THEN
00506         READ(14) dummy, dummy1
00507         IF (.NOT.ALLOCATED( hdown )) ALLOCATE( hdown(dummy, dummy1))
00508         READ(14) hdown
00509     ENDIF
00510
00511     IF(realcheck( 34 ) == 1) THEN
00512         READ(14) dummy
00513         IF (.NOT.ALLOCATED( hed )) ALLOCATE( hed(dummy))
00514         READ(14) hed
00515     ENDIF
00516
00517     IF(realcheck( 35 ) == 1) THEN
00518         READ(14) dummy
00519         IF (.NOT.ALLOCATED( hef )) ALLOCATE( hef(dummy))
00520         READ(14) hef
00521     ENDIF
00522
00523     IF(realcheck( 36 ) == 1) THEN
00524         READ(14) dummy
00525         IF (.NOT.ALLOCATED( hep )) ALLOCATE( hep(dummy))
00526         READ(14) hep
00527     ENDIF
00528
00529     IF(realcheck( 37 ) == 1) THEN
00530         READ(14) dummy
00531         IF (.NOT.ALLOCATED( hes )) ALLOCATE( hes(dummy))
00532         READ(14) hes
00533     ENDIF
00534
00535     IF(realcheck( 38 ) == 1) THEN
00536         READ(14) dummy
00537         IF (.NOT.ALLOCATED( hjj )) ALLOCATE( hjj(dummy))
00538         READ(14) hjj
00539     ENDIF
00540
00541     IF(realcheck( 39 ) == 1) THEN
00542         READ(14) dummy
00543         IF (.NOT.ALLOCATED( hr0 )) ALLOCATE( hr0(dummy))
00544         READ(14) hr0
00545     ENDIF

```

```

00546
00547 IF(realcheck( 40 ) == 1)THEN
00548   READ(14) dummyn
00549   IF(.NOT.ALLOCATED( hubbardu )) ALLOCATE( hubbardu(dummyn))
00550   READ(14) hubbardu
00551 ENDIF
00552
00553 IF(realcheck( 41 ) == 1)THEN
00554   READ(14) dummyn ,dummyn1
00555   IF(.NOT.ALLOCATED( hup )) ALLOCATE( hup(dummyn,dummyn1))
00556   READ(14) hup
00557 ENDIF
00558
00559 IF(realcheck( 42 ) == 1)THEN
00560   READ(14) dummyn ,dummyn1
00561   IF(.NOT.ALLOCATED( kevals )) ALLOCATE( kevals(dummyn,dummyn1))
00562   READ(14) kevals
00563 ENDIF
00564
00565 IF(realcheck( 43 ) == 1)THEN
00566   READ(14) dummyn
00567   IF(.NOT.ALLOCATED( lcnshift )) ALLOCATE( lcnshift(dummyn))
00568   READ(14) lcnshift
00569 ENDIF
00570
00571 IF(realcheck( 44 ) == 1)THEN
00572   READ(14) dummyn
00573   IF(.NOT.ALLOCATED( mass )) ALLOCATE( mass(dummyn))
00574   READ(14) mass
00575 ENDIF
00576
00577 IF(realcheck( 45 ) == 1)THEN
00578   READ(14) dummyn
00579   IF(.NOT.ALLOCATED( mycharge )) ALLOCATE( mycharge(dummyn))
00580   READ(14) mycharge
00581 ENDIF
00582
00583 IF(realcheck( 46 ) == 1)THEN
00584   READ(14) dummyn ,dummyn1
00585   IF(.NOT.ALLOCATED( nonotmp )) ALLOCATE( nonotmp(dummyn,dummyn1))
00586   READ(14) nonotmp
00587 ENDIF
00588
00589 IF(realcheck( 47 ) == 1)THEN
00590   READ(14) dummyn
00591   IF(.NOT.ALLOCATED( nono_evals )) ALLOCATE( nono_evals(dummyn))
00592   READ(14) nono_evals
00593 ENDIF
00594
00595 IF(realcheck( 48 ) == 1)THEN
00596   READ(14) dummyn
00597   IF(.NOT.ALLOCATED( nono_work )) ALLOCATE( nono_work(dummyn))
00598   READ(14) nono_work
00599 ENDIF
00600
00601 IF(realcheck( 49 ) == 1)THEN
00602   READ(14) dummyn
00603   IF(.NOT.ALLOCATED( olddeltaqs )) ALLOCATE( olddeltaqs(dummyn))
00604   READ(14) olddeltaqs
00605 ENDIF
00606
00607 IF(realcheck( 50 ) == 1)THEN
00608   READ(14) dummyn
00609   IF(.NOT.ALLOCATED( olddeltaspin )) ALLOCATE( olddeltaspin(dummyn))
00610   READ(14) olddeltaspin
00611 ENDIF
00612
00613 IF(realcheck( 51 ) == 1)THEN
00614   READ(14) dummyn ,dummyn1
00615   IF(.NOT.ALLOCATED( orthoh )) ALLOCATE( orthoh(dummyn,dummyn1))
00616   READ(14) orthoh
00617 ENDIF
00618
00619 IF(realcheck( 52 ) == 1)THEN
00620   READ(14) dummyn ,dummyn1
00621   IF(.NOT.ALLOCATED( orthohdown )) ALLOCATE( orthohdown(dummyn,dummyn1))
00622   READ(14) orthohdown
00623 ENDIF
00624
00625 IF(realcheck( 53 ) == 1)THEN
00626   READ(14) dummyn ,dummyn1
00627   IF(.NOT.ALLOCATED( orthohup )) ALLOCATE( orthohup(dummyn,dummyn1))
00628   READ(14) orthohup
00629 ENDIF
00630
00631 IF(realcheck( 54 ) == 1)THEN
00632   READ(14) dummyn ,dummyn1

```

```

00633         IF (.NOT.ALLOCATED( orthorho )) ALLOCATE( orthorho(dummysn,dummysn1))
00634         READ(14) orthorho
00635     ENDIF
00636
00637     IF(realcheck( 55 ) == 1)THEN
00638         READ(14) dummysn ,dummysn1
00639         IF (.NOT.ALLOCATED( overl )) ALLOCATE( overl(dummysn,dummysn1))
00640         READ(14) overl
00641     ENDIF
00642
00643     IF(realcheck( 56 ) == 1)THEN
00644         READ(14) dummysn ,dummysn1
00645         IF (.NOT.ALLOCATED( pair )) ALLOCATE( pair(dummysn,dummysn1))
00646         READ(14) pair
00647     ENDIF
00648
00649     IF(realcheck( 57 ) == 1)THEN
00650         READ(14) dummysn
00651         IF (.NOT.ALLOCATED( phist )) ALLOCATE( phist(dummysn))
00652         READ(14) phist
00653     ENDIF
00654
00655     IF(realcheck( 58 ) == 1)THEN
00656         READ(14) dummysn
00657         IF (.NOT.ALLOCATED( phistx )) ALLOCATE( phistx(dummysn))
00658         READ(14) phistx
00659     ENDIF
00660
00661     IF(realcheck( 59 ) == 1)THEN
00662         READ(14) dummysn
00663         IF (.NOT.ALLOCATED( phisty )) ALLOCATE( phisty(dummysn))
00664         READ(14) phisty
00665     ENDIF
00666
00667     IF(realcheck( 60 ) == 1)THEN
00668         READ(14) dummysn
00669         IF (.NOT.ALLOCATED( phistz )) ALLOCATE( phistz(dummysn))
00670         READ(14) phistz
00671     ENDIF
00672
00673     IF(realcheck( 61 ) == 1)THEN
00674         READ(14) dummysn ,dummysn1
00675         IF (.NOT.ALLOCATED( pnk )) ALLOCATE( pnk(dummysn,dummysn1))
00676         READ(14) pnk
00677     ENDIF
00678
00679     IF(realcheck( 62 ) == 1)THEN
00680         READ(14) dummysn ,dummysn1
00681         IF (.NOT.ALLOCATED( potcoef )) ALLOCATE( potcoef(dummysn,dummysn1))
00682         READ(14) potcoef
00683     ENDIF
00684
00685     IF(realcheck( 63 ) == 1)THEN
00686         READ(14) dummysn ,dummysn1
00687         IF (.NOT.ALLOCATED( ppr )) ALLOCATE( ppr(dummysn,dummysn1))
00688         READ(14) ppr
00689     ENDIF
00690
00691     IF(realcheck( 64 ) == 1)THEN
00692         READ(14) dummysn ,dummysn1
00693         IF (.NOT.ALLOCATED( ppspl )) ALLOCATE( ppspl(dummysn,dummysn1))
00694         READ(14) ppspl
00695     ENDIF
00696
00697     IF(realcheck( 65 ) == 1)THEN
00698         READ(14) dummysn ,dummysn1
00699         IF (.NOT.ALLOCATED( ppval )) ALLOCATE( ppval(dummysn,dummysn1))
00700         READ(14) ppval
00701     ENDIF
00702
00703     IF(realcheck( 66 ) == 1)THEN
00704         READ(14) dummysn
00705         IF (.NOT.ALLOCATED( qlist )) ALLOCATE( qlist(dummysn))
00706         READ(14) qlist
00707     ENDIF
00708
00709     IF(realcheck( 67 ) == 1)THEN
00710         READ(14) dummysn
00711         IF (.NOT.ALLOCATED( respchi )) ALLOCATE( respchi(dummysn))
00712         READ(14) respchi
00713     ENDIF
00714
00715     IF(realcheck( 68 ) == 1)THEN
00716         READ(14) dummysn ,dummysn1
00717         IF (.NOT.ALLOCATED( rhodown )) ALLOCATE( rhodown(dummysn,dummysn1))
00718         READ(14) rhodown
00719     ENDIF

```

```

00720
00721 IF (realcheck( 69 ) == 1) THEN
00722   READ (14) dummy_n
00723   IF (.NOT.ALLOCATED( rhodownzero )) ALLOCATE( rhodownzero(dummy_n) )
00724   READ (14) rhodownzero
00725 ENDIF
00726
00727 IF (realcheck( 70 ) == 1) THEN
00728   READ (14) dummy_n , dummy_n1
00729   IF (.NOT.ALLOCATED( rhoup )) ALLOCATE( rhoup(dummy_n, dummy_n1) )
00730   READ (14) rhoup
00731 ENDIF
00732
00733 IF (realcheck( 71 ) == 1) THEN
00734   READ (14) dummy_n
00735   IF (.NOT.ALLOCATED( rhoupzero )) ALLOCATE( rhoupzero(dummy_n) )
00736   READ (14) rhoupzero
00737 ENDIF
00738
00739 IF (realcheck( 72 ) == 1) THEN
00740   READ (14) dummy_n , dummy_n1
00741   IF (.NOT.ALLOCATED( sh2 )) ALLOCATE( sh2(dummy_n, dummy_n1) )
00742   READ (14) sh2
00743 ENDIF
00744
00745 IF (realcheck( 73 ) == 1) THEN
00746   READ (14) dummy_n
00747   IF (.NOT.ALLOCATED( sinlist )) ALLOCATE( sinlist(dummy_n) )
00748   READ (14) sinlist
00749 ENDIF
00750
00751 IF (realcheck( 74 ) == 1) THEN
00752   READ (14) dummy_n , dummy_n1
00753   IF (.NOT.ALLOCATED( smat )) ALLOCATE( smat(dummy_n, dummy_n1) )
00754   READ (14) smat
00755 ENDIF
00756
00757 IF (realcheck( 75 ) == 1) THEN
00758   READ (14) dummy_n
00759   IF (.NOT.ALLOCATED( spinlist )) ALLOCATE( spinlist(dummy_n) )
00760   READ (14) spinlist
00761 ENDIF
00762
00763 IF (realcheck( 76 ) == 1) THEN
00764   READ (14) dummy_n , dummy_n1
00765   IF (.NOT.ALLOCATED( spintmp )) ALLOCATE( spintmp(dummy_n, dummy_n1) )
00766   READ (14) spintmp
00767 ENDIF
00768
00769 IF (realcheck( 77 ) == 1) THEN
00770   READ (14) dummy_n , dummy_n1
00771   IF (.NOT.ALLOCATED( spin_pnk )) ALLOCATE( spin_pnk(dummy_n, dummy_n1) )
00772   READ (14) spin_pnk
00773 ENDIF
00774
00775 IF (realcheck( 78 ) == 1) THEN
00776   READ (14) dummy_n
00777   IF (.NOT.ALLOCATED( thist )) ALLOCATE( thist(dummy_n) )
00778   READ (14) thist
00779 ENDIF
00780
00781 IF (realcheck( 79 ) == 1) THEN
00782   READ (14) dummy_n
00783   IF (.NOT.ALLOCATED( tmpbodiag )) ALLOCATE( tmpbodiag(dummy_n) )
00784   READ (14) tmpbodiag
00785 ENDIF
00786
00787 IF (realcheck( 80 ) == 1) THEN
00788   READ (14) dummy_n
00789   IF (.NOT.ALLOCATED( tmprhodown )) ALLOCATE( tmprhodown(dummy_n) )
00790   READ (14) tmprhodown
00791 ENDIF
00792
00793 IF (realcheck( 81 ) == 1) THEN
00794   READ (14) dummy_n
00795   IF (.NOT.ALLOCATED( tmprhoup )) ALLOCATE( tmprhoup(dummy_n) )
00796   READ (14) tmprhoup
00797 ENDIF
00798
00799 IF (realcheck( 82 ) == 1) THEN
00800   READ (14) dummy_n , dummy_n1
00801   IF (.NOT.ALLOCATED( twoxx2 )) ALLOCATE( twoxx2(dummy_n, dummy_n1) )
00802   READ (14) twoxx2
00803 ENDIF
00804
00805 IF (realcheck( 83 ) == 1) THEN
00806   READ (14) dummy_n , dummy_n1

```

```

00807         IF (.NOT.ALLOCATED( umat )) ALLOCATE( umat (dummyn,dummyn1))
00808         READ(14) umat
00809     ENDIF
00810
00811     IF(realcheck( 84 ) == 1) THEN
00812         READ(14) dummyn
00813         IF (.NOT.ALLOCATED( upevals )) ALLOCATE( upevals (dummyn))
00814         READ(14) upevals
00815     ENDIF
00816
00817     IF(realcheck( 85 ) == 1) THEN
00818         READ(14) dummyn ,dummyn1
00819         IF (.NOT.ALLOCATED( upevecs )) ALLOCATE( upevecs (dummyn,dummyn1))
00820         READ(14) upevecs
00821     ENDIF
00822
00823     IF(realcheck( 86 ) == 1) THEN
00824         READ(14) dummyn ,dummyn1
00825         IF (.NOT.ALLOCATED( v )) ALLOCATE( v (dummyn,dummyn1))
00826         READ(14) v
00827     ENDIF
00828
00829     IF(realcheck( 87 ) == 1) THEN
00830         READ(14) dummyn
00831         IF (.NOT.ALLOCATED( vhist )) ALLOCATE( vhist (dummyn))
00832         READ(14) vhist
00833     ENDIF
00834
00835     IF(realcheck( 88 ) == 1) THEN
00836         READ(14) dummyn
00837         IF (.NOT.ALLOCATED( wdd )) ALLOCATE( wdd (dummyn))
00838         READ(14) wdd
00839     ENDIF
00840
00841     IF(realcheck( 89 ) == 1) THEN
00842         READ(14) dummyn
00843         IF (.NOT.ALLOCATED( wff )) ALLOCATE( wff (dummyn))
00844         READ(14) wff
00845     ENDIF
00846
00847     IF(realcheck( 90 ) == 1) THEN
00848         READ(14) dummyn
00849         IF (.NOT.ALLOCATED( work )) ALLOCATE( work (dummyn))
00850         READ(14) work
00851     ENDIF
00852
00853     IF(realcheck( 91 ) == 1) THEN
00854         READ(14) dummyn
00855         IF (.NOT.ALLOCATED( wpp )) ALLOCATE( wpp (dummyn))
00856         READ(14) wpp
00857     ENDIF
00858
00859     IF(realcheck( 92 ) == 1) THEN
00860         READ(14) dummyn
00861         IF (.NOT.ALLOCATED( wss )) ALLOCATE( wss (dummyn))
00862         READ(14) wss
00863     ENDIF
00864
00865     IF(realcheck( 93 ) == 1) THEN
00866         READ(14) dummyn ,dummyn1
00867         IF (.NOT.ALLOCATED( x2 )) ALLOCATE( x2 (dummyn,dummyn1))
00868         READ(14) x2
00869     ENDIF
00870
00871     IF(realcheck( 94 ) == 1) THEN
00872         READ(14) dummyn ,dummyn1
00873         IF (.NOT.ALLOCATED( x2down )) ALLOCATE( x2down (dummyn,dummyn1))
00874         READ(14) x2down
00875     ENDIF
00876
00877     IF(realcheck( 95 ) == 1) THEN
00878         READ(14) dummyn ,dummyn1
00879         IF (.NOT.ALLOCATED( x2hrho )) ALLOCATE( x2hrho (dummyn,dummyn1))
00880         READ(14) x2hrho
00881     ENDIF
00882
00883     IF(realcheck( 96 ) == 1) THEN
00884         READ(14) dummyn ,dummyn1
00885         IF (.NOT.ALLOCATED( x2up )) ALLOCATE( x2up (dummyn,dummyn1))
00886         READ(14) x2up
00887     ENDIF
00888
00889     IF(realcheck( 97 ) == 1) THEN
00890         READ(14) dummyn ,dummyn1
00891         IF (.NOT.ALLOCATED( xmat )) ALLOCATE( xmat (dummyn,dummyn1))
00892         READ(14) xmat
00893     ENDIF

```



```
00894
00895  IF (realcheck( 98 ) == 1) THEN
00896    READ(14) dummy
00897    IF (.NOT.ALLOCATED( zheevd_rwork )) ALLOCATE( zheevd_rwork(dummy) )
00898    READ(14) zheevd_rwork
00899  ENDIF
00900
00901  CLOSE(14)
00902
00903 END SUBROUTINE readrestartlib
```

8.351 readtb.f90 File Reference

Functions/Subroutines

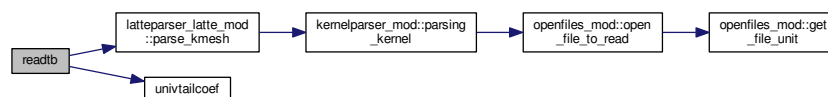
- subroutine [readtb](#)

8.351.1 Function/Subroutine Documentation

8.351.1.1 subroutine [readtb](#) ()

Definition at line 23 of file [readtb.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE readtb
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE univarray
00027   USE mdarray
00028   USE nonoarray
00029   USE spinarray
00030   USE kspacearray
00031   USE latteparser_latte_mod
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, J, K
00036   CHARACTER(LEN=20) :: HD
00037   REAL(LATTEPREC) :: TAILPARAMS(6)
00038   IF (existerror) RETURN
00039
00040   OPEN(unit=22,status="OLD", file=trim(parampath)//"/electrons.dat")
00041
00042   READ(22,*) hd, noelem
00043
00044   IF (.NOT.ALLOCATED(wss)) THEN
00045     ALLOCATE(wss(noelem),wpp(noelem),wdd(noelem),wff(
noelem))
00046   ENDIF
00047
00048   ALLOCATE(ele(noelem), basis(noelem), atocc(noelem),
hes(noelem), &
00049     hep(noelem), hed(noelem), hef(noelem), mass(
noelem), &
00050     hubbardu(noelem))
00051
00052   READ(22,*) hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd
00053
00054   DO i = 1, noelem
00055     READ(22,*) ele(i), basis(i), atocc(i), hes(i), hep(i),
hed(i), hef(i), &
00056       mass(i), hubbardu(i), wss(i), wpp(i), wdd(i), wff(i)
00057   ENDDO
00058
00059   CLOSE(22)
00060
00061   IF (basistype .EQ. "ORTHO") THEN
00062     OPEN(unit=11,status="OLD", file=trim(parampath)//"/bondints.ortho")
00063   ELSE
00064     OPEN(unit=11,status="OLD", file=trim(parampath)//"/bondints.nonortho")
00065   ENDIF
00066
00067   READ(11,*) hd, noint
00068
00069   ALLOCATE(ele1(noint), ele2(noint), btype(noint),&
bond(14, noint))
00070
00071
00072
00073   IF (basistype .EQ. "ORTHO") THEN
00074
00075     READ(11,*) hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd
00076
00077     DO i = 1, noint
00078
00079       READ(11,*) ele1(i), ele2(i), btype(i), (bond(j,i), j = 1, 8)
00080
00081     ENDDO
00082
00083   ELSE
00084
00085     ALLOCATE( overl(14,noint) )
00086
00087     READ(11,*) hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, hd, &
hd, hd, hd, hd, hd, hd, hd, hd
00088
00089

```

```

00090      DO i = 1, noint
00091
00092          READ(11,*) ele1(i), ele2(i), btype(i), (bond(j,i), j = 1, 8), &
00093              (overl(k,i), k = 1, 8)
00094
00095      ENDDO
00096
00097  ENDIF
00098
00099  CLOSE(11)
00100
00101  ! If we're doing k-space integration, let's read in the k point mesh
00102  IF (kon .EQ. 1) THEN
00103
00104      IF (latteinexists) THEN
00105          CALL parse_kmesh("latte.in")
00106      ELSE
00107          OPEN(unit=11, status="OLD", file=trim(parampath)//"/kmesh.in")
00108          READ(11,*) nkx, nky, nkz
00109          READ(11,*) kshift(1), kshift(2), kshift(3)
00110          CLOSE (11)
00111      ENDIF
00112
00113      nktot = nkx*nky*nkz
00114
00115  ENDIF
00116
00117  DO i = 1, noint
00118
00119      CALL univtailcoef(bond(:,i))
00120
00121      IF (basistype .EQ. "NONORTHO") CALL univtailcoef(overl(:,i))
00122
00123  ENDDO
00124
00125  RETURN
00126
00127 END SUBROUTINE readtb

```

8.353 relaxcommon.f90 File Reference

Modules

- module [relaxcommon](#)

Variables

- real(latteprec), dimension(:,,:), allocatable [relaxcommon::d1](#)
- real(latteprec) [relaxcommon::mxrlx](#)
- real(latteprec), dimension(:,,:), allocatable [relaxcommon::oldd](#)
- real(latteprec), dimension(:,,:), allocatable [relaxcommon::oldf](#)
- real(latteprec) [relaxcommon::prevf](#)
- real(latteprec) [relaxcommon::relconst](#)
- character(len=2) [relaxcommon::reltype](#)
- real(latteprec) [relaxcommon::rlxftol](#)

8.354 relaxcommon.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !

```

```

00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE relaxcommon
00023
00024 USE myprecision
00025
00026 IMPLICIT NONE
00027 SAVE
00028
00029 CHARACTER(LEN=2) :: reltype
00030 REAL(LATTEPREC) :: mxrlx, rlxfcol
00031 REAL(LATTEPREC) :: prevf, relconst
00032 ! REAL(LATTEPREC) :: PREVE
00033
00034 ! These are for the conjugated gradient relaxation
00035
00036 REAL(LATTEPREC), ALLOCATABLE :: oldf(:, :), dl(:, :), oldd(:, :)
00037
00038 END MODULE relaxcommon

```

8.355 resetprohd.f90 File Reference

Functions/Subroutines

- subroutine [resetprohd](#)

8.355.1 Function/Subroutine Documentation

8.355.1.1 subroutine resetprohd ()

Definition at line 23 of file [resetprohd.f90](#).


```

00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE resetprohd
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE kspacearray
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   !
00032   ! Let's try keeping Sum_i H_ii D_ii constant = 0
00033   !
00034
00035   INTEGER :: I, K, J, NUMORB, INDEX
00036   REAL(LATTEPREC) :: MYSUM
00037   COMPLEX(LATTEPREC) :: ZMYSUM
00038   IF (existererror) RETURN
00039
00040   mysum = zero
00041   zmysum = cmplx(zero)
00042
00043   IF (kon .EQ. 0) THEN ! Real-space
00044
00045     DO i = 1, hdim
00046       mysum = mysum + h(i,i)*bo(i,i)
00047     ENDDO
00048
00049     mysum = mysum/float(hdim)
00050
00051     DO i = 1, hdim
00052       h(i,i) = h(i,i) - mysum
00053     ENDDO
00054
00055   ELSE ! k-space
00056
00057     DO k = 1, nktot
00058       index = 0
00059       DO i = 1, nats
00060
00061         SELECT CASE(basis(elempointer(i)))
00062
00063           CASE("s")
00064             numorb = 1
00065           CASE("p")
00066             numorb = 3
00067           CASE("d")
00068             numorb = 5
00069           CASE("f")
00070             numorb = 7
00071           CASE("sp")
00072             numorb = 4
00073           CASE("sd")
00074             numorb = 6
00075           CASE("sf")
00076             numorb = 8
00077           CASE("pd")
00078             numorb = 8
00079           CASE("pf")
00080             numorb = 10
00081           CASE("df")
00082             numorb = 12
00083           CASE("spd")
00084             numorb = 9
00085           CASE("spf")
00086             numorb = 11
00087           CASE("sdf")
00088             numorb = 13
00089           CASE("pdf")
00090             numorb = 15
00091           CASE("spdf")
00092             numorb = 16
00093         END SELECT
00094
00095       DO j = 1, numorb

```

```

00096             index = index + 1
00097             zmysum = zmysum + lcnshift(i)*kbo(index,index,k)
00098             ENDDO
00099         ENDDO
00100     ENDDO
00101
00102     zmysum = zmysum/REAL(hdim*nktot)
00103
00104     !      PRINT*, REAL(ZMYSUM)
00105
00106     DO i = 1, nats
00107
00108         lcnshift(i) = lcnshift(i) - REAL(zmysum)
00109
00110     ENDDO
00111
00112 ENDIF
00113
00114 RETURN
00115
00116 END SUBROUTINE resetprohd
00117

```

8.357 restartarray.f90 File Reference

Modules

- module [restartarray](#)

Variables

- real(latteprec), dimension(:), allocatable [restartarray::tmpbodiag](#)
- integer [restartarray::tmphdim](#)
- real(latteprec), dimension(:), allocatable [restartarray::tmprhdown](#)
- real(latteprec), dimension(:), allocatable [restartarray::tmprhoup](#)

8.358 restartarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE restartarray
00023
00024     USE myprecision
00025
00026     IMPLICIT NONE
00027     SAVE
00028
00029     INTEGER :: tmphdim
00030     REAL(LATTEPREC), ALLOCATABLE :: tmprhoup(:), tmprhdown(:),
00031         tmpbodiag(:)
00032
00033 END MODULE restartarray

```


8.359 rhozero.f90 File Reference

Functions/Subroutines

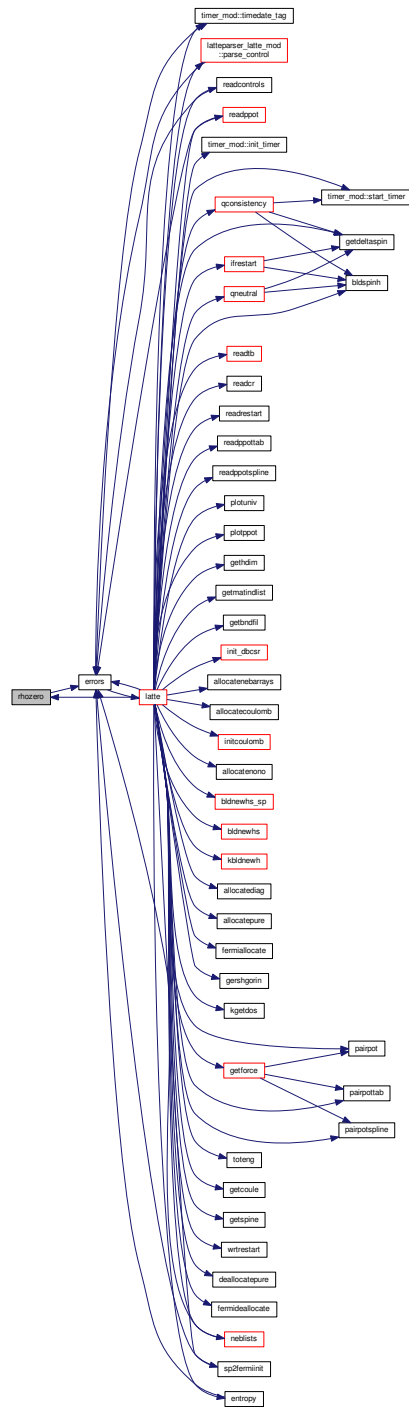
- subroutine [rhozero](#)

8.359.1 Function/Subroutine Documentation

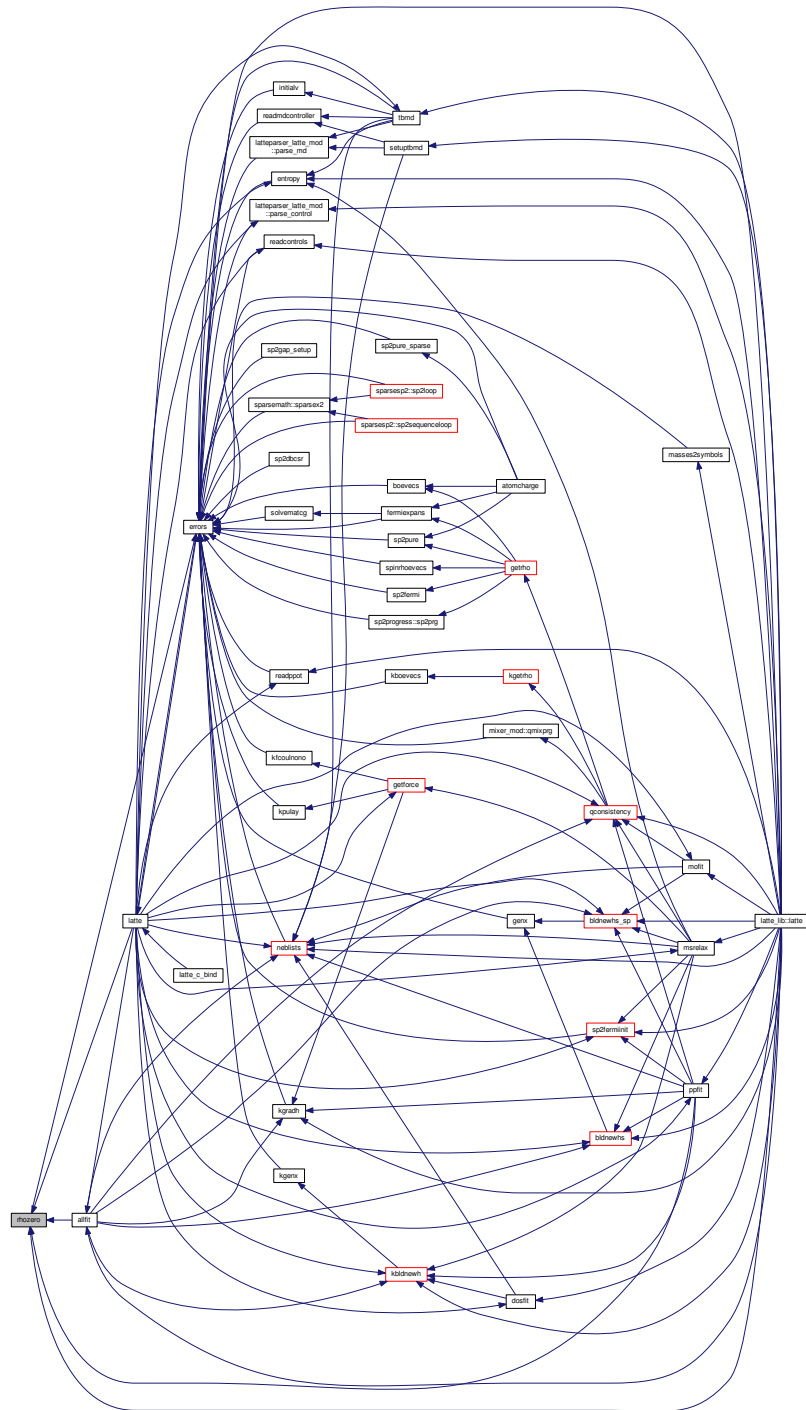
8.359.1.1 subroutine rhozero ()

Definition at line 23 of file [rhozero.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.360 rhozero.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE rhozero
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE kspacearray
00028   USE restartarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, K, SUBI, INDEX
00034   REAL(LATTEPREC) :: STOT, PTOT, NUMPERORB
00035   REAL(LATTEPREC) :: SUP, SDOWN, PUP, PDOWN, DUP, DDOWN, FUP, FDOWN
00036   REAL(LATTEPREC), PARAMETER :: SPINMAXS = one, spinmaxp = three
00037   REAL(LATTEPREC), PARAMETER :: SPINMAXD = five, spinmaxf = seven
00038   IF (existerror) RETURN
00039
00040   espin_zero = zero
00041
00042   IF (spinon .EQ. 0) THEN
00043
00044     ! No spins
00045
00046     ALLOCATE(bozero(hdim))
00047
00048     bozero = zero
00049
00050     index = 0
00051
00052     DO i = 1, nats
00053
00054       SELECT CASE(basis(elempointer(i)))
00055
00056       CASE("s")
00057
00058         bozero(index + 1) = atocc(elempointer(i))
00059         index = index + 1
00060
00061       CASE("p")
00062
00063         ! Distribute the electrons evenly between the orbitals
00064
00065         numperorb = atocc(elempointer(i))/three
00066         bozero(index + 1) = numperorb
00067         bozero(index + 2) = numperorb
00068         bozero(index + 3) = numperorb
00069
00070         index = index + 3
00071
00072       CASE("d")
00073
00074         numperorb = atocc(elempointer(i))/five
00075         bozero(index + 1) = numperorb
00076         bozero(index + 2) = numperorb
00077         bozero(index + 3) = numperorb
00078         bozero(index + 4) = numperorb
00079         bozero(index + 5) = numperorb
00080
00081         index = index + 5
00082
00083       CASE("f")
00084
00085         numperorb = atocc(elempointer(i))/seven
00086         bozero(index + 1) = numperorb
00087         bozero(index + 2) = numperorb
00088         bozero(index + 3) = numperorb
00089         bozero(index + 4) = numperorb
00090         bozero(index + 5) = numperorb
00091         bozero(index + 6) = numperorb
00092         bozero(index + 7) = numperorb
00093

```

```

00094
00095         index = index + 7
00096
00097     CASE("sp")
00098
00099         ! First fill s, then p
00100
00101         IF (atocc(elempointer(i)) .LE. two) THEN
00102
00103             bozero(index + 1) = atocc(elempointer(i))
00104
00105             index = index + 1
00106
00107             bozero(index + 1) = zero
00108             bozero(index + 2) = zero
00109             bozero(index + 3) = zero
00110
00111             index = index + 3
00112
00113         ELSE
00114
00115             bozero(index + 1) = two
00116
00117             index = index + 1
00118
00119             numperorb = (atocc(elempointer(i)) - two)/three
00120
00121             bozero(index + 1) = numperorb
00122             bozero(index + 2) = numperorb
00123             bozero(index + 3) = numperorb
00124
00125             index = index + 3
00126
00127         ENDIF
00128
00129     CASE("sd")
00130
00131         bozero(index + 1) = two
00132
00133         index = index + 1
00134
00135         numperorb = (atocc(elempointer(i)) - two)/five
00136
00137         bozero(index + 1) = numperorb
00138         bozero(index + 2) = numperorb
00139         bozero(index + 3) = numperorb
00140         bozero(index + 4) = numperorb
00141         bozero(index + 5) = numperorb
00142
00143         index = index + 5
00144
00145     CASE("sf")
00146
00147         ! Caution - f electron system!
00148
00149         bozero(index + 1) = two
00150
00151         index = index + 1
00152
00153         numperorb = (atocc(elempointer(i)) - two)/seven
00154
00155         bozero(index + 1) = numperorb
00156         bozero(index + 2) = numperorb
00157         bozero(index + 3) = numperorb
00158         bozero(index + 4) = numperorb
00159         bozero(index + 5) = numperorb
00160         bozero(index + 6) = numperorb
00161         bozero(index + 7) = numperorb
00162
00163         index = index + 7
00164
00165     CASE("pd")
00166
00167         ! Fill d, then p
00168
00169         numperorb = (atocc(elempointer(i)) - ten)/three
00170
00171         bozero(index + 1) = numperorb
00172         bozero(index + 2) = numperorb
00173         bozero(index + 3) = numperorb
00174
00175         index = index + 3
00176
00177         bozero(index + 1) = two
00178         bozero(index + 2) = two
00179         bozero(index + 3) = two
00180         bozero(index + 4) = two

```

```

00181         bozero(index + 5) = two
00182
00183         index = index + 5
00184
00185     CASE("pf")
00186
00187         ! Fill f, then p, I guess
00188
00189         numberorb = (atocc(elempointer(i)) - fourteen)/
three
00190
00191         bozero(index + 1) = numberorb
00192         bozero(index + 2) = numberorb
00193         bozero(index + 3) = numberorb
00194
00195         index = index + 3
00196
00197         bozero(index + 1) = two
00198         bozero(index + 2) = two
00199         bozero(index + 3) = two
00200         bozero(index + 4) = two
00201         bozero(index + 5) = two
00202         bozero(index + 6) = two
00203         bozero(index + 7) = two
00204
00205         index = index + 7
00206
00207
00208     CASE("df")
00209
00210         ! For the light actinides we have 1 in the d, and the rest in the f
00211         ! (except Th, where we have 2d's and zero f's)
00212
00213         IF (atocc(elempointer(i)) .EQ. 2) THEN
00214
00215             numberorb = two/five
00216
00217             bozero(index + 1) = numberorb
00218             bozero(index + 2) = numberorb
00219             bozero(index + 3) = numberorb
00220             bozero(index + 4) = numberorb
00221             bozero(index + 5) = numberorb
00222
00223             index = index + 5
00224
00225             bozero(index + 1) = zero
00226             bozero(index + 2) = zero
00227             bozero(index + 3) = zero
00228             bozero(index + 4) = zero
00229             bozero(index + 5) = zero
00230             bozero(index + 6) = zero
00231             bozero(index + 7) = zero
00232
00233             index = index + 7
00234
00235         ELSE
00236
00237             numberorb = one/five
00238
00239             bozero(index + 1) = numberorb
00240             bozero(index + 2) = numberorb
00241             bozero(index + 3) = numberorb
00242             bozero(index + 4) = numberorb
00243             bozero(index + 5) = numberorb
00244
00245             index = index + 5
00246
00247             numberorb = (atocc(elempointer(i)) - one)/seven
00248
00249             bozero(index + 1) = numberorb
00250             bozero(index + 2) = numberorb
00251             bozero(index + 3) = numberorb
00252             bozero(index + 4) = numberorb
00253             bozero(index + 5) = numberorb
00254             bozero(index + 6) = numberorb
00255             bozero(index + 7) = numberorb
00256
00257             index = index + 7
00258
00259         ENDIF
00260
00261     CASE("spd")
00262
00263         ! s, d, then p
00264
00265         IF (atocc(elempointer(i)) .LE. two) THEN
00266

```

```

00267         bozero(index + 1) = atocc(elempointer(i))
00268
00269         index = index + 1
00270
00271         bozero(index + 1) = zero
00272         bozero(index + 2) = zero
00273         bozero(index + 3) = zero
00274
00275         index = index + 3
00276
00277         bozero(index + 1) = zero
00278         bozero(index + 2) = zero
00279         bozero(index + 3) = zero
00280         bozero(index + 4) = zero
00281         bozero(index + 5) = zero
00282
00283         index = index + 5
00284
00285     ELSE IF (atocc(elempointer(i)) .GT. two .AND. &
00286             atocc(elempointer(i)) .LE. twelve) THEN
00287
00288         bozero(index + 1) = two
00289
00290         index = index + 1
00291
00292         numperorb = (atocc(elempointer(i)) - two)/five
00293
00294         bozero(index + 1) = zero
00295         bozero(index + 2) = zero
00296         bozero(index + 3) = zero
00297
00298         index = index + 3
00299
00300         bozero(index + 1) = numperorb
00301         bozero(index + 2) = numperorb
00302         bozero(index + 3) = numperorb
00303         bozero(index + 4) = numperorb
00304         bozero(index + 5) = numperorb
00305
00306         index = index + 5
00307
00308     ELSE
00309
00310         bozero(index + 1) = two
00311
00312         index = index + 1
00313
00314         numperorb = (atocc(elempointer(i)) - twelve)/
three
00315
00316         bozero(index + 1) = numperorb
00317         bozero(index + 2) = numperorb
00318         bozero(index + 3) = numperorb
00319
00320         index = index + 3
00321
00322         bozero(index + 1) = two
00323         bozero(index + 2) = two
00324         bozero(index + 3) = two
00325         bozero(index + 4) = two
00326         bozero(index + 5) = two
00327
00328         index = index + 5
00329
00330     ENDIF
00331
00332     CASE("spf")
00333
00334         ! s, f, then p
00335
00336         IF (atocc(elempointer(i)) .LE. two) THEN
00337
00338             bozero(index + 1) = atocc(elempointer(i))
00339
00340             index = index + 1
00341
00342             bozero(index + 1) = zero
00343             bozero(index + 2) = zero
00344             bozero(index + 3) = zero
00345
00346             index = index + 3
00347
00348             bozero(index + 1) = zero
00349             bozero(index + 2) = zero
00350             bozero(index + 3) = zero
00351             bozero(index + 4) = zero
00352             bozero(index + 5) = zero

```

```

00353         bozero(index + 6) = zero
00354         bozero(index + 7) = zero
00355
00356         index = index + 7
00357
00358     ELSE IF (atocc(elempointer(i)) .GT. two .AND. &
00359             atocc(elempointer(i)) .LE. sixteen) THEN
00360
00361         bozero(index + 1) = two
00362
00363         index = index + 1
00364
00365         numberorb = (atocc(elempointer(i)) - two)/seven
00366
00367         bozero(index + 1) = zero
00368         bozero(index + 2) = zero
00369         bozero(index + 3) = zero
00370
00371         index = index + 3
00372
00373         bozero(index + 1) = numberorb
00374         bozero(index + 2) = numberorb
00375         bozero(index + 3) = numberorb
00376         bozero(index + 4) = numberorb
00377         bozero(index + 5) = numberorb
00378         bozero(index + 6) = numberorb
00379         bozero(index + 7) = numberorb
00380
00381         index = index + 7
00382
00383     ELSE
00384
00385         bozero(index + 1) = two
00386
00387         index = index + 1
00388
00389         numberorb = (atocc(elempointer(i)) - sixteen)/
three
00390
00391         bozero(index + 1) = numberorb
00392         bozero(index + 2) = numberorb
00393         bozero(index + 3) = numberorb
00394
00395         index = index + 3
00396
00397         bozero(index + 1) = two
00398         bozero(index + 2) = two
00399         bozero(index + 3) = two
00400         bozero(index + 4) = two
00401         bozero(index + 5) = two
00402         bozero(index + 6) = two
00403         bozero(index + 7) = two
00404
00405         index = index + 7
00406
00407     ENDIF
00408
00409     CASE("sdf")
00410
00411         ! Let's build this for the light actinides
00412
00413         ! 7s has 2 electrons, 6d2, 5f0 for Th, and 6d1, 5f? for the others
00414
00415     IF (atocc(elempointer(i)) .GE. three .AND. &
00416         atocc(elempointer(i)) .LE. four) THEN
00417
00418         bozero(index + 1) = two
00419
00420         index = index + 1
00421
00422         numberorb = (atocc(elempointer(i)) - two)/five
00423
00424         bozero(index + 1) = numberorb
00425         bozero(index + 2) = numberorb
00426         bozero(index + 3) = numberorb
00427         bozero(index + 4) = numberorb
00428         bozero(index + 5) = numberorb
00429
00430         index = index + 5
00431
00432         bozero(index + 1) = zero
00433         bozero(index + 2) = zero
00434         bozero(index + 3) = zero
00435         bozero(index + 4) = zero
00436         bozero(index + 5) = zero
00437         bozero(index + 6) = zero
00438         bozero(index + 7) = zero

```



```

00439         index = index + 7
00440
00441     ELSEIF (atocc(elempointer(i)) .GT. four) THEN
00442
00443         bozero(index + 1) = two
00444
00445         index = index + 1
00446
00447         !               NUMPERORB = ONE/FIVE
00448         numperorb = zero
00449
00450         bozero(index + 1) = numperorb
00451         bozero(index + 2) = numperorb
00452         bozero(index + 3) = numperorb
00453         bozero(index + 4) = numperorb
00454         bozero(index + 5) = numperorb
00455
00456         index = index + 5
00457
00458         !               NUMPERORB = (ATOC(ELEMPONTER(I)) - THREE)/SEVEN
00459         numperorb = (atocc(elempointer(i)) - two)/seven
00460
00461         bozero(index + 1) = numperorb
00462         bozero(index + 2) = numperorb
00463         bozero(index + 3) = numperorb
00464         bozero(index + 4) = numperorb
00465         bozero(index + 5) = numperorb
00466         bozero(index + 6) = numperorb
00467         bozero(index + 7) = numperorb
00468
00469         index = index + 7
00470
00471     ELSE
00472
00473         CALL errors("rhozero","Check the number of electrons &
00474             & you're using with the sdf basis")
00475
00476     ENDIF
00477
00478     CASE("pdf")
00479
00480         ! I suppose we fill the f, then the d, and finally the p
00481
00482         IF (atocc(elempointer(i)) .LE. fourteen) THEN
00483
00484             numperorb = atocc(elempointer(i))/seven
00485
00486             bozero(index + 1) = numperorb
00487             bozero(index + 2) = numperorb
00488             bozero(index + 3) = numperorb
00489             bozero(index + 4) = numperorb
00490             bozero(index + 5) = numperorb
00491             bozero(index + 6) = numperorb
00492             bozero(index + 7) = numperorb
00493
00494             index = index + 7
00495
00496             bozero(index + 1) = zero
00497             bozero(index + 2) = zero
00498             bozero(index + 3) = zero
00499             bozero(index + 4) = zero
00500             bozero(index + 5) = zero
00501
00502             index = index + 5
00503
00504             bozero(index + 1) = zero
00505             bozero(index + 2) = zero
00506             bozero(index + 3) = zero
00507
00508             index = index + 3
00509
00510         ELSEIF (atocc(elempointer(i)) .GT. fourteen .AND. &
00511             atocc(elempointer(i)) .LE. twentyfour) THEN
00512
00513             bozero(index + 1) = two
00514             bozero(index + 2) = two
00515             bozero(index + 3) = two
00516             bozero(index + 4) = two
00517             bozero(index + 5) = two
00518             bozero(index + 6) = two
00519             bozero(index + 7) = two
00520
00521             index = index + 7
00522
00523             numperorb = (atocc(elempointer(i)) - fourteen)/
00524         five

```

```

00525
00526         bozero(index + 1) = numberorb
00527         bozero(index + 2) = numberorb
00528         bozero(index + 3) = numberorb
00529         bozero(index + 4) = numberorb
00530         bozero(index + 5) = numberorb
00531
00532         index = index + 5
00533
00534         bozero(index + 1) = zero
00535         bozero(index + 2) = zero
00536         bozero(index + 3) = zero
00537
00538         index = index + 3
00539
00540     ELSE
00541
00542         bozero(index + 1) = two
00543         bozero(index + 2) = two
00544         bozero(index + 3) = two
00545         bozero(index + 4) = two
00546         bozero(index + 5) = two
00547         bozero(index + 6) = two
00548         bozero(index + 7) = two
00549
00550         index = index + 7
00551
00552         bozero(index + 1) = two
00553         bozero(index + 2) = two
00554         bozero(index + 3) = two
00555         bozero(index + 4) = two
00556         bozero(index + 5) = two
00557
00558         index = index + 5
00559
00560         numberorb = (atocc(elempointer(i)) - twentyfour)/
three
00561
00562         bozero(index + 1) = numberorb
00563         bozero(index + 2) = numberorb
00564         bozero(index + 3) = numberorb
00565
00566         index = index + 3
00567
00568     ENDIF
00569
00570     CASE("spdf")
00571
00572         ! s, then f, then d, then p...
00573
00574         IF (atocc(elempointer(i)) .LE. two) THEN
00575
00576             numberorb = atocc(elempointer(i))
00577
00578             bozero(index + 1) = numberorb
00579
00580             bozero(index + 1) = zero
00581             bozero(index + 2) = zero
00582             bozero(index + 3) = zero
00583             bozero(index + 4) = zero
00584             bozero(index + 5) = zero
00585             bozero(index + 6) = zero
00586             bozero(index + 7) = zero
00587
00588             index = index + 7
00589
00590             bozero(index + 1) = zero
00591             bozero(index + 2) = zero
00592             bozero(index + 3) = zero
00593             bozero(index + 4) = zero
00594             bozero(index + 5) = zero
00595
00596             index = index + 5
00597
00598             bozero(index + 1) = zero
00599             bozero(index + 2) = zero
00600             bozero(index + 3) = zero
00601
00602             index = index + 3
00603
00604         ELSEIF (atocc(elempointer(i)) .GT. two .AND. &
00605             atocc(elempointer(i)) .LE. sixteen) THEN
00606
00607             bozero(index + 1) = two
00608
00609             numberorb = (atocc(elempointer(i)) - two)/seven
00610

```

```

00611         bozero(index + 1) = numberorb
00612         bozero(index + 2) = numberorb
00613         bozero(index + 3) = numberorb
00614         bozero(index + 4) = numberorb
00615         bozero(index + 5) = numberorb
00616         bozero(index + 6) = numberorb
00617         bozero(index + 7) = numberorb
00618
00619         index = index + 7
00620
00621         bozero(index + 1) = zero
00622         bozero(index + 2) = zero
00623         bozero(index + 3) = zero
00624         bozero(index + 4) = zero
00625         bozero(index + 5) = zero
00626
00627         index = index + 5
00628
00629         bozero(index + 1) = zero
00630         bozero(index + 2) = zero
00631         bozero(index + 3) = zero
00632
00633         index = index + 3
00634
00635         ELSEIF (atocc(elempointer(i)) .GT. sixteen .AND. &
00636                atocc(elempointer(i)) .LE. twentysix) THEN
00637
00638             bozero(index + 1) = two
00639
00640             index = index + 1
00641
00642             bozero(index + 1) = two
00643             bozero(index + 2) = two
00644             bozero(index + 3) = two
00645             bozero(index + 4) = two
00646             bozero(index + 5) = two
00647             bozero(index + 6) = two
00648             bozero(index + 7) = two
00649
00650             index = index + 7
00651
00652             numberorb = (atocc(elempointer(i)) - sixteen)/
five
00653
00654             bozero(index + 1) = numberorb
00655             bozero(index + 2) = numberorb
00656             bozero(index + 3) = numberorb
00657             bozero(index + 4) = numberorb
00658             bozero(index + 5) = numberorb
00659
00660             index = index + 5
00661
00662             bozero(index + 1) = zero
00663             bozero(index + 2) = zero
00664             bozero(index + 3) = zero
00665
00666             index = index + 3
00667
00668         ELSE
00669
00670             bozero(index + 1) = two
00671
00672             index = index + 1
00673
00674             bozero(index + 1) = two
00675             bozero(index + 2) = two
00676             bozero(index + 3) = two
00677             bozero(index + 4) = two
00678             bozero(index + 5) = two
00679             bozero(index + 6) = two
00680             bozero(index + 7) = two
00681
00682             index = index + 7
00683
00684             bozero(index + 1) = two
00685             bozero(index + 2) = two
00686             bozero(index + 3) = two
00687             bozero(index + 4) = two
00688             bozero(index + 5) = two
00689
00690             index = index + 5
00691
00692             numberorb = (atocc(elempointer(i)) - twentysix)/
three
00693
00694             bozero(index + 1) = numberorb
00695             bozero(index + 2) = numberorb

```

```

00696         bozero(index + 3) = numperorb
00697
00698         index = index + 3
00699
00700     ENDIF
00701
00702 END SELECT
00703
00704 ENDDO
00705
00706
00707 IF (kon .EQ. 0) THEN ! Real-space
00708
00709     IF (restart .EQ. 0) THEN
00710
00711         !
00712         ! Initialize the diagonal elements of BO to those for free atoms
00713         !
00714
00715         DO i = 1, hdim
00716             bo(i,i) = bozero(i)
00717         ENDDO
00718
00719     ELSEIF (restart .EQ. 1) THEN
00720
00721         DO i = 1, hdim
00722             bo(i,i) = tmpbodiag(i)
00723         ENDDO
00724
00725         DEALLOCATE(tmpbodiag)
00726
00727     ENDIF
00728
00729 ELSE ! k-space
00730
00731     IF (restart .EQ. 0) THEN
00732
00733         !
00734         ! Initialize the diagonal elements of BO to those for free atoms
00735         !
00736
00737         DO k = 1, nktot
00738             DO i = 1, hdim
00739                 kbo(i,i,k) = cmplx(bozero(i))
00740             ENDDO
00741         ENDDO
00742
00743     ELSEIF (restart .EQ. 1) THEN
00744
00745         DO k = 1, nktot
00746             DO i = 1, hdim
00747                 kbo(i,i,k) = cmplx(tmpbodiag(i))
00748             ENDDO
00749         ENDDO
00750
00751         DEALLOCATE(tmpbodiag)
00752
00753     ENDIF
00754
00755 ENDIF
00756
00757 ELSEIF (spinon .EQ. 1) THEN
00758
00759     ! With spins
00760
00761     ALLOCATE(rhoupzero(hdim), rhodownzero(hdim))
00762
00763     rhoupzero = zero
00764     rhodownzero = zero
00765
00766     index = 0
00767
00768     DO i = 1, nats
00769
00770         SELECT CASE(basis(elempointer(i)))
00771
00772         CASE("s")
00773
00774             IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
00775
00776                 rhoupzero(index+1) = atocc(elempointer(i))
00777                 rhodownzero(index+1) = zero
00778
00779                 index = index + 1
00780
00781             ENDIF
00782
00783         ENDSELECT
00784     ENDDO

```

```

00783         ELSE
00784
00785             rhoupzero(index+1) = one
00786             rhodownzero(index+1) = atocc(elempointer(i)) - spinmaxs
00787
00788             index = index + 1
00789
00790         ENDIF
00791
00792     CASE("p")
00793
00794         IF (atocc(elempointer(i)) .LE. spinmaxp) THEN
00795
00796             pup = atocc(elempointer(i))/spinmaxp
00797             pdown = zero
00798
00799         ELSE
00800
00801             pup = one
00802             pdown = (atocc(elempointer(i)) - spinmaxp)/spinmaxp
00803
00804         ENDIF
00805
00806         rhoupzero(index + 1) = pup
00807         rhoupzero(index + 2) = pup
00808         rhoupzero(index + 3) = pup
00809
00810         rhodownzero(index + 1) = pdown
00811         rhodownzero(index + 2) = pdown
00812         rhodownzero(index + 3) = pdown
00813
00814         index = index + 3
00815
00816     CASE("d")
00817
00818         IF (atocc(elempointer(i)) .LE. spinmaxd) THEN
00819
00820             dup = atocc(elempointer(i))/spinmaxd
00821             ddown = zero
00822
00823         ELSE
00824
00825             dup = one
00826             ddown = (atocc(elempointer(i)) - spinmaxd)/spinmaxd
00827
00828         ENDIF
00829
00830         rhoupzero(index + 1) = dup
00831         rhoupzero(index + 2) = dup
00832         rhoupzero(index + 3) = dup
00833         rhoupzero(index + 4) = dup
00834         rhoupzero(index + 5) = dup
00835
00836         rhodownzero(index + 1) = ddown
00837         rhodownzero(index + 2) = ddown
00838         rhodownzero(index + 3) = ddown
00839         rhodownzero(index + 4) = ddown
00840         rhodownzero(index + 5) = ddown
00841
00842         index = index + 5
00843
00844     CASE("f")
00845
00846         IF (atocc(elempointer(i)) .LE. spinmaxf) THEN
00847
00848             fup = atocc(elempointer(i))/spinmaxf
00849             fdown = zero
00850
00851         ELSE
00852
00853             fup = one
00854             fdown = (atocc(elempointer(i)) - spinmaxf)/spinmaxf
00855
00856         ENDIF
00857
00858         rhoupzero(index + 1) = fup
00859         rhoupzero(index + 2) = fup
00860         rhoupzero(index + 3) = fup
00861         rhoupzero(index + 4) = fup
00862         rhoupzero(index + 5) = fup
00863         rhoupzero(index + 6) = fup
00864         rhoupzero(index + 7) = fup
00865
00866         rhodownzero(index + 1) = fdown
00867         rhodownzero(index + 2) = fdown
00868
00869

```

```

00870         rhodownzero(index + 3) = fdown
00871         rhodownzero(index + 4) = fdown
00872         rhodownzero(index + 5) = fdown
00873         rhodownzero(index + 6) = fdown
00874         rhodownzero(index + 7) = fdown
00875
00876         index = index + 7
00877
00878     CASE("sp")
00879
00880         IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
00881
00882             sup = atocc(elempointer(i))
00883             sdown = zero
00884             pup = zero
00885             pdown = zero
00886
00887         ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
00888             atocc(elempointer(i)) .LE. two*spinmaxs) THEN
00889
00890             sup = one
00891             sdown = atocc(elempointer(i)) - spinmaxs
00892             pup = zero
00893             pdown = zero
00894
00895         ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
00896             atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxp) THEN
00897
00898             sup = one
00899             sdown = one
00900             pup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxp
00901             pdown = zero
00902
00903         ELSE
00904
00905             sup = one
00906             sdown = one
00907             pup = one
00908             pdown = (atocc(elempointer(i)) - &
00909                 (two*spinmaxs + spinmaxp))/spinmaxp
00910
00911         ENDIF
00912
00913         rhoupzero(index + 1) = sup
00914         rhoupzero(index + 2) = pup
00915         rhoupzero(index + 3) = pup
00916         rhoupzero(index + 4) = pup
00917
00918         rhodownzero(index + 1) = sdown
00919         rhodownzero(index + 2) = pdown
00920         rhodownzero(index + 3) = pdown
00921         rhodownzero(index + 4) = pdown
00922
00923         index = index + 4
00924
00925     CASE("sd")
00926
00927         IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
00928
00929             sup = atocc(elempointer(i))
00930             sdown = zero
00931             dup = zero
00932             ddown = zero
00933
00934         ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
00935             atocc(elempointer(i)) .LE. two*spinmaxs) THEN
00936
00937             sup = one
00938             sdown = atocc(elempointer(i)) - spinmaxs
00939             dup = zero
00940             ddown = zero
00941
00942         ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
00943             atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxd) THEN
00944
00945             sup = one
00946             sdown = one
00947             dup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxd
00948             ddown = zero
00949
00950         ELSE
00951
00952             sup = one
00953             sdown = one
00954             dup = one
00955             ddown = one
00956

```

```

00957         ddown = (atocc(elempointer(i)) - &
00958                 (two*spinmaxs + spinmaxd)/spinmaxd)
00959
00960     ENDIF
00961
00962     rhoupzero(index + 1) = sup
00963     rhoupzero(index + 2) = dup
00964     rhoupzero(index + 3) = dup
00965     rhoupzero(index + 4) = dup
00966     rhoupzero(index + 5) = dup
00967     rhoupzero(index + 6) = dup
00968
00969
00970     rhodownzero(index + 1) = sdown
00971     rhodownzero(index + 2) = ddown
00972     rhodownzero(index + 3) = ddown
00973     rhodownzero(index + 4) = ddown
00974     rhodownzero(index + 5) = ddown
00975     rhodownzero(index + 6) = ddown
00976
00977     index = index + 6
00978
00979     CASE("sf")
00980
00981     IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
00982
00983         sup = atocc(elempointer(i))
00984         sdown = zero
00985         fup = zero
00986         fdown = zero
00987
00988     ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
00989            atocc(elempointer(i)) .LE. two*spinmaxs) THEN
00990
00991         sup = one
00992         sdown = atocc(elempointer(i)) - one
00993         fup = zero
00994         fdown = zero
00995
00996     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
00997            atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxf) THEN
00998
00999
01000         sup = one
01001         sdown = one
01002         fup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxf
01003         fdown = zero
01004
01005     ELSE
01006
01007         sup = one
01008         sdown = one
01009         fup = one
01010         fdown = (atocc(elempointer(i)) - &
01011                 (two*spinmaxs + spinmaxf)/spinmaxf)
01012
01013     ENDIF
01014
01015     rhoupzero(index + 1) = sup
01016     rhoupzero(index + 2) = fup
01017     rhoupzero(index + 3) = fup
01018     rhoupzero(index + 4) = fup
01019     rhoupzero(index + 5) = fup
01020     rhoupzero(index + 6) = fup
01021     rhoupzero(index + 7) = fup
01022     rhoupzero(index + 8) = fup
01023
01024     rhodownzero(index + 1) = sdown
01025     rhodownzero(index + 2) = fdown
01026     rhodownzero(index + 3) = fdown
01027     rhodownzero(index + 4) = fdown
01028     rhodownzero(index + 5) = fdown
01029     rhodownzero(index + 6) = fdown
01030     rhodownzero(index + 7) = fdown
01031     rhodownzero(index + 8) = fdown
01032
01033     index = index + 8
01034
01035     CASE("pd")
01036
01037     ! We'll fill the d, then the p
01038
01039     IF (atocc(elempointer(i)) .LE. spinmaxd) THEN
01040
01041         dup = atocc(elempointer(i))/spinmaxd
01042         ddown = zero
01043         pup = zero

```

```

01044         pdown = zero
01045
01046     ELSEIF (atocc(elempointer(i)) .GT. spinmaxd .AND. &
01047             atocc(elempointer(i)) .LE. two*spinmaxd) THEN
01048
01049         dup = one
01050         ddown = (atocc(elempointer(i)) - spinmaxd)/spinmaxd
01051         pup = zero
01052         pdown = zero
01053
01054     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxd .AND. &
01055             atocc(elempointer(i)) .LE. two*spinmaxd + spinmaxp) THEN
01056
01057
01058         dup = one
01059         ddown = one
01060         pup = (atocc(elempointer(i)) - two*spinmaxd)/spinmaxp
01061         pdown = zero
01062
01063     ELSE
01064
01065         dup = one
01066         ddown = one
01067         pup = one
01068         pdown = (atocc(elempointer(i)) - &
01069                 (two*spinmaxd + spinmaxp))/spinmaxp
01070
01071     ENDIF
01072
01073     rhoupzero(index + 1) = pup
01074     rhoupzero(index + 2) = pup
01075     rhoupzero(index + 3) = pup
01076     rhoupzero(index + 4) = dup
01077     rhoupzero(index + 5) = dup
01078     rhoupzero(index + 6) = dup
01079     rhoupzero(index + 7) = dup
01080     rhoupzero(index + 8) = dup
01081
01082     rhodownzero(index + 1) = pdown
01083     rhodownzero(index + 2) = pdown
01084     rhodownzero(index + 3) = pdown
01085     rhodownzero(index + 4) = ddown
01086     rhodownzero(index + 5) = ddown
01087     rhodownzero(index + 6) = ddown
01088     rhodownzero(index + 7) = ddown
01089     rhodownzero(index + 8) = ddown
01090
01091     index = index + 8
01092
01093     CASE("pf")
01094
01095         ! First the f, then the p
01096
01097         IF (atocc(elempointer(i)) .LE. spinmaxf) THEN
01098
01099             fup = atocc(elempointer(i))/spinmaxf
01100             fdown = zero
01101             pup = zero
01102             pdown = zero
01103
01104         ELSEIF (atocc(elempointer(i)) .GT. spinmaxf .AND. &
01105                 atocc(elempointer(i)) .LE. two*spinmaxf) THEN
01106
01107             fup = one
01108             fdown = (atocc(elempointer(i)) - spinmaxf)/spinmaxf
01109             pup = zero
01110             pdown = zero
01111
01112         ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxf .AND. &
01113                 atocc(elempointer(i)) .LE. two*spinmaxf + spinmaxp) THEN
01114
01115
01116             fup = one
01117             fdown = one
01118             pup = (atocc(elempointer(i)) - two*spinmaxf)/spinmaxp
01119             pdown = zero
01120
01121         ELSE
01122
01123             fup = one
01124             fdown = one
01125             pup = one
01126             pdown = (atocc(elempointer(i)) - &
01127                     (two*spinmaxf + spinmaxp))/spinmaxp
01128
01129         ENDIF
01130

```



```

01131         rhoupzero(index + 1) = pup
01132         rhoupzero(index + 2) = pup
01133         rhoupzero(index + 3) = pup
01134         rhoupzero(index + 4) = fup
01135         rhoupzero(index + 5) = fup
01136         rhoupzero(index + 6) = fup
01137         rhoupzero(index + 7) = fup
01138         rhoupzero(index + 8) = fup
01139         rhoupzero(index + 9) = fup
01140         rhoupzero(index + 10) = fup
01141
01142         rhodownzero(index + 1) = pdown
01143         rhodownzero(index + 2) = pdown
01144         rhodownzero(index + 3) = pdown
01145         rhodownzero(index + 4) = fdown
01146         rhodownzero(index + 5) = fdown
01147         rhodownzero(index + 6) = fdown
01148         rhodownzero(index + 7) = fdown
01149         rhodownzero(index + 8) = fdown
01150         rhodownzero(index + 9) = fdown
01151         rhodownzero(index + 10) = fdown
01152
01153         index = index + 10
01154
01155
01156     CASE("df")
01157
01158         ! Let's imagine these are the light actinides
01159
01160         ! Thorium
01161
01162         IF (atocc(elempointer(i)) .GT. four) THEN
01163             CALL errors("rhozero","The df basis is limited to 4 electrons (U)&
01164                 & at the moment. Modify the source to go beyond U")
01165         ENDIF
01166
01167         IF (atocc(elempointer(i)) .LE. two) THEN
01168
01169             dup = atocc(elempointer(i))/spinmaxd
01170             ddown = zero
01171             fup = zero
01172             fdown = zero
01173
01174         ELSE
01175
01176             dup = one/spinmaxd
01177             ddown = zero
01178             fup = (atocc(elempointer(i)) - one)/spinmaxf
01179             fdown = zero
01180
01181         ENDIF
01182
01183         rhoupzero(index + 1) = dup
01184         rhoupzero(index + 2) = dup
01185         rhoupzero(index + 3) = dup
01186         rhoupzero(index + 4) = dup
01187         rhoupzero(index + 5) = dup
01188         rhoupzero(index + 6) = fup
01189         rhoupzero(index + 7) = fup
01190         rhoupzero(index + 8) = fup
01191         rhoupzero(index + 9) = fup
01192         rhoupzero(index + 10) = fup
01193         rhoupzero(index + 11) = fup
01194         rhoupzero(index + 12) = fup
01195
01196         rhodownzero(index + 1) = ddown
01197         rhodownzero(index + 2) = ddown
01198         rhodownzero(index + 3) = ddown
01199         rhodownzero(index + 4) = ddown
01200         rhodownzero(index + 5) = ddown
01201         rhodownzero(index + 6) = fdown
01202         rhodownzero(index + 7) = fdown
01203         rhodownzero(index + 8) = fdown
01204         rhodownzero(index + 9) = fdown
01205         rhodownzero(index + 10) = fdown
01206         rhodownzero(index + 11) = fdown
01207         rhodownzero(index + 12) = fdown
01208
01209         index = index + 12
01210
01211     CASE("spd")
01212
01213         ! s, then d, then p...
01214
01215         IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
01216
01217             sup = atocc(elempointer(i))

```

```

01218         sdown = zero
01219         dup = zero
01220         ddown = zero
01221         pup = zero
01222         pdown = zero
01223
01224     ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
01225             atocc(elempointer(i)) .LE. two*spinmaxs) THEN
01226
01227         sup = one
01228         sdown = atocc(elempointer(i)) - spinmaxs
01229         dup = zero
01230         ddown = zero
01231         pup = zero
01232         pdown = zero
01233
01234     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
01235             atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxd) THEN
01236
01237         sup = one
01238         sdown = one
01239         dup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxd
01240         ddown = zero
01241         pup = zero
01242         pdown = zero
01243
01244     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs + spinmaxd .AND. &
01245             atocc(elempointer(i)) .LE. two*spinmaxs + two*spinmaxd) THEN
01246
01247         sup = one
01248         sdown = one
01249         dup = one
01250         ddown = (atocc(elempointer(i)) - &
01251                 (two*spinmaxs + spinmaxd))/spinmaxd
01252         pup = zero
01253         pdown = zero
01254
01255     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs +
01256             two*spinmaxd &
01257             .AND. atocc(elempointer(i)) .LE. &
01258                 two*spinmaxs + two*spinmaxd + spinmaxp ) THEN
01259
01260         sup = one
01261         sdown = one
01262         dup = one
01263         ddown = one
01264         pup = (atocc(elempointer(i)) - &
01265                 (two*spinmaxs + two*spinmaxd))/spinmaxp
01266         pdown = zero
01267
01268     ELSE
01269
01270         sup = one
01271         sdown = one
01272         dup = one
01273         ddown = one
01274         pup = one
01275         pdown = (atocc(elempointer(i)) - &
01276                 (two*spinmaxs + two*spinmaxd + spinmaxp))/spinmaxp
01277
01278     ENDIF
01279
01280     rhoupzero(index + 1) = sup
01281     rhoupzero(index + 2) = pup
01282     rhoupzero(index + 3) = pup
01283     rhoupzero(index + 4) = pup
01284     rhoupzero(index + 5) = dup
01285     rhoupzero(index + 6) = dup
01286     rhoupzero(index + 7) = dup
01287     rhoupzero(index + 8) = dup
01288     rhoupzero(index + 9) = dup
01289
01290     rhodownzero(index + 1) = sdown
01291     rhodownzero(index + 2) = pdown
01292     rhodownzero(index + 3) = pdown
01293     rhodownzero(index + 4) = pdown
01294     rhodownzero(index + 5) = ddown
01295     rhodownzero(index + 6) = ddown
01296     rhodownzero(index + 7) = ddown
01297     rhodownzero(index + 8) = ddown
01298     rhodownzero(index + 9) = ddown
01299
01300     index = index + 9
01301
01302     CASE("spf")
01303

```

```

01304
01305     ! s, then f, then p
01306
01307     IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
01308
01309         sup = atocc(elempointer(i))
01310         sdown = zero
01311         fup = zero
01312         fdown = zero
01313         pup = zero
01314         pdown = zero
01315
01316     ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
01317             atocc(elempointer(i)) .LE. two*spinmaxs) THEN
01318
01319         sup = one
01320         sdown = atocc(elempointer(i)) - spinmaxs
01321         fup = zero
01322         fdown = zero
01323         pup = zero
01324         pdown = zero
01325
01326     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
01327             atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxf) THEN
01328
01329         sup = one
01330         sdown = one
01331         fup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxf
01332         fdown = zero
01333         pup = zero
01334         pdown = zero
01335
01336     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs + spinmaxf .AND. &
01337             atocc(elempointer(i)) .LE. two*spinmaxs + two*spinmaxf) THEN
01338
01339         sup = one
01340         sdown = one
01341         fup = one
01342         fdown = (atocc(elempointer(i)) - &
01343                 (two*spinmaxs + spinmaxf))/spinmaxf
01344         pup = zero
01345         pdown = zero
01346
01347     ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs +
01348             two*spinmaxf &
01349             .AND. atocc(elempointer(i)) .LE. &
01350             two*spinmaxs + two*spinmaxf + spinmaxp) THEN
01351
01352         sup = one
01353         sdown = one
01354         fup = one
01355         fdown = one
01356         pup = (atocc(elempointer(i)) - &
01357                 (two*spinmaxs + two*spinmaxf))/spinmaxp
01358         pdown = zero
01359
01360     ELSE
01361
01362         sup = one
01363         sdown = one
01364         fup = one
01365         fdown = one
01366         pup = one
01367         pdown = (atocc(elempointer(i)) - &
01368                 (two*spinmaxs + two*spinmaxf + spinmaxp))/spinmaxp
01369
01370     ENDIF
01371
01372     rhoupzero(index + 1) = sup
01373     rhoupzero(index + 2) = pup
01374     rhoupzero(index + 3) = pup
01375     rhoupzero(index + 4) = pup
01376     rhoupzero(index + 5) = fup
01377     rhoupzero(index + 6) = fup
01378     rhoupzero(index + 7) = fup
01379     rhoupzero(index + 8) = fup
01380     rhoupzero(index + 9) = fup
01381     rhoupzero(index + 10) = fup
01382     rhoupzero(index + 11) = fup
01383
01384     rhodownzero(index + 1) = sdown
01385     rhodownzero(index + 2) = pdown
01386     rhodownzero(index + 3) = pdown
01387     rhodownzero(index + 4) = pdown
01388     rhodownzero(index + 5) = fdown
01389     rhodownzero(index + 6) = fdown

```

```

01390         rhodownzero(index + 7) = fdown
01391         rhodownzero(index + 8) = fdown
01392         rhodownzero(index + 9) = fdown
01393         rhodownzero(index + 10) = fdown
01394         rhodownzero(index + 11) = fdown
01395
01396         index = index + 11
01397
01398     CASE("sdf")
01399
01400         ! Let's do the light actinides explicitly
01401
01402         ! Thorium
01403
01404         IF (atocc(elempointer(i)) .LT. spinmaxs) THEN
01405
01406             sup = atocc(elempointer(i))
01407             sdown = zero
01408             fup = zero
01409             fdown = zero
01410             dup = zero
01411             ddown = zero
01412
01413         ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
01414             atocc(elempointer(i)) .LE. two*spinmaxs) THEN
01415
01416             sup = one
01417             sdown = atocc(elempointer(i)) - spinmaxs
01418             fup = zero
01419             fdown = zero
01420             dup = zero
01421             ddown = zero
01422
01423         ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
01424             atocc(elempointer(i)) .LE. two*spinmaxs + two) THEN
01425
01426             sup = one
01427             sdown = one
01428             fup = zero
01429             fdown = zero
01430             dup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxd
01431             ddown = zero
01432
01433         ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs +
01434             two) THEN
01435
01436             ! Having one electron in the 6d isn't working...
01437             ! Let's go with 4 in the 5 f instead
01438
01439             !             SUP = ONE
01440             !             SDOWN = ONE
01441             !             FUP = (ATOCC(ELEMPOINTER(I)) - ONE)/SPINMAXF
01442             !             FDOWN = ZERO
01443             !             DUP = ONE/SPINMAXD
01444             !             DDOWN = ZERO
01445
01446             sup = one
01447             sdown = one
01448             fup = (atocc(elempointer(i)) - two)/spinmaxf
01449             fdown = zero
01450             dup = zero
01451             ddown = zero
01452
01453         ENDIF
01454
01455         rhoupzero(index + 1) = sup
01456         rhoupzero(index + 2) = dup
01457         rhoupzero(index + 3) = dup
01458         rhoupzero(index + 4) = dup
01459         rhoupzero(index + 5) = dup
01460         rhoupzero(index + 6) = dup
01461         rhoupzero(index + 7) = fup
01462         rhoupzero(index + 8) = fup
01463         rhoupzero(index + 9) = fup
01464         rhoupzero(index + 10) = fup
01465         rhoupzero(index + 11) = fup
01466         rhoupzero(index + 12) = fup
01467         rhoupzero(index + 13) = fup
01468
01469         rhodownzero(index + 1) = sdown
01470         rhodownzero(index + 2) = ddown
01471         rhodownzero(index + 3) = ddown
01472         rhodownzero(index + 4) = ddown
01473         rhodownzero(index + 5) = ddown
01474         rhodownzero(index + 6) = ddown
01475         rhodownzero(index + 7) = fdown
01476         rhodownzero(index + 8) = fdown

```

```

01476         rhodownzero(index + 9) = fdown
01477         rhodownzero(index + 10) = fdown
01478         rhodownzero(index + 11) = fdown
01479         rhodownzero(index + 12) = fdown
01480         rhodownzero(index + 13) = fdown
01481
01482         index = index + 13
01483
01484         CASE("pdf")
01485
01486             ! f, then d, then p
01487
01488             IF (atocc(elempointer(i)) .LT. spinmaxf) THEN
01489
01490                 fup = atocc(elempointer(i))/spinmaxf
01491                 fdown = zero
01492                 dup = zero
01493                 ddown = zero
01494                 pup = zero
01495                 pdown = zero
01496
01497             ELSEIF (atocc(elempointer(i)) .GT. spinmaxf .AND. &
01498                    atocc(elempointer(i)) .LE. two*spinmaxf) THEN
01499
01500                 fup = one
01501                 fdown = (atocc(elempointer(i)) - spinmaxf)/spinmaxf
01502                 dup = zero
01503                 ddown = zero
01504                 pup = zero
01505                 pdown = zero
01506
01507             ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxf .AND. &
01508                    atocc(elempointer(i)) .LE. two*spinmaxf + spinmaxd) THEN
01509
01510                 fup = one
01511                 fdown = one
01512                 dup = (atocc(elempointer(i)) - two*spinmaxf)/spinmaxd
01513                 ddown = zero
01514                 pup = zero
01515                 pdown = zero
01516
01517             ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxf + spinmaxd .AND. &
01518                    atocc(elempointer(i)) .LE. two*spinmaxf + two*spinmaxd) THEN
01519
01520                 fup = one
01521                 fdown = one
01522                 dup = one
01523                 ddown = (atocc(elempointer(i)) - &
01524                        (two*spinmaxf + spinmaxd))/spinmaxd
01525                 pup = zero
01526                 pdown = zero
01527
01528             ELSEIF (atocc(elempointer(i)) .GT. &
01529                    two*spinmaxf + two*spinmaxd .AND. &
01530                    atocc(elempointer(i)) .LE. &
01531                    two*spinmaxf + two*spinmaxd + spinmaxp) THEN
01532
01533                 fup = one
01534                 fdown = one
01535                 dup = one
01536                 ddown = one
01537                 pup = (atocc(elempointer(i)) - &
01538                        (two*spinmaxf + two*spinmaxd))/spinmaxp
01539                 pdown = zero
01540
01541             ELSE
01542
01543                 fup = one
01544                 fdown = one
01545                 dup = one
01546                 ddown = one
01547                 pup = one
01548                 pdown = (atocc(elempointer(i)) - &
01549                        (two*spinmaxf + two*spinmaxd + spinmaxp))/spinmaxp
01550
01551             ENDIF
01552
01553         rhoupzero(index + 1) = pup
01554         rhoupzero(index + 2) = pup
01555         rhoupzero(index + 3) = pup
01556         rhoupzero(index + 4) = dup
01557         rhoupzero(index + 5) = dup
01558         rhoupzero(index + 6) = dup
01559         rhoupzero(index + 7) = dup
01560         rhoupzero(index + 8) = dup
01561         rhoupzero(index + 9) = fup
01562         rhoupzero(index + 10) = fup

```

```

01563      rhoupzero(index + 11) = fup
01564      rhoupzero(index + 12) = fup
01565      rhoupzero(index + 13) = fup
01566      rhoupzero(index + 14) = fup
01567      rhoupzero(index + 15) = fup
01568
01569      rhodownzero(index + 1) = pdown
01570      rhodownzero(index + 2) = pdown
01571      rhodownzero(index + 3) = pdown
01572      rhodownzero(index + 4) = ddown
01573      rhodownzero(index + 5) = ddown
01574      rhodownzero(index + 6) = ddown
01575      rhodownzero(index + 7) = ddown
01576      rhodownzero(index + 8) = ddown
01577      rhodownzero(index + 9) = fdown
01578      rhodownzero(index + 10) = fdown
01579      rhodownzero(index + 11) = fdown
01580      rhodownzero(index + 12) = fdown
01581      rhodownzero(index + 13) = fdown
01582      rhodownzero(index + 14) = fdown
01583      rhodownzero(index + 15) = fdown
01584
01585      index = index + 15
01586
01587      CASE("spdf")
01588
01589      ! s, f, d, then p
01590
01591      IF (atocc(elempointer(i)) .LE. spinmaxs) THEN
01592
01593          sup = atocc(elempointer(i))
01594          sdown = zero
01595          fup = zero
01596          fdown = zero
01597          dup = zero
01598          ddown = zero
01599          pup = zero
01600          pdown = zero
01601
01602      ELSEIF (atocc(elempointer(i)) .GT. spinmaxs .AND. &
01603              atocc(elempointer(i)) .LE. two*spinmaxs) THEN
01604
01605          sup = one
01606          sdown = (atocc(elempointer(i)) - spinmaxf)
01607          fup = zero
01608          fdown = zero
01609          dup = zero
01610          ddown = zero
01611          pup = zero
01612          pdown = zero
01613
01614      ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs .AND. &
01615              atocc(elempointer(i)) .LE. two*spinmaxs + spinmaxf) THEN
01616
01617          sup = one
01618          sdown = one
01619          fup = (atocc(elempointer(i)) - two*spinmaxs)/spinmaxf
01620          fdown = zero
01621          dup = zero
01622          ddown = zero
01623          pup = zero
01624          pdown = zero
01625
01626      ELSEIF (atocc(elempointer(i)) .GT. two*spinmaxs + spinmaxf .AND. &
01627              atocc(elempointer(i)) .LE. two*spinmaxs + two*spinmaxf) THEN
01628
01629          sup = one
01630          sdown = one
01631          fup = one
01632          fdown = (atocc(elempointer(i)) - &
01633                  (two*spinmaxs + spinmaxf))/spinmaxf
01634          dup = zero
01635          ddown = zero
01636          pup = zero
01637          pdown = zero
01638
01639      ELSEIF (atocc(elempointer(i)) .GT. &
01640              two*spinmaxs + two*spinmaxf .AND. &
01641              atocc(elempointer(i)) .LE. &
01642              two*spinmaxs + two*spinmaxf + spinmaxd) THEN
01643
01644          sup = one
01645          sdown = one
01646          fup = one
01647          fdown = one
01648          dup = (atocc(elempointer(i)) - &

```

```

01650         (two*spinmaxs + two*spinmaxf))/spinmaxd
01651         ddown = zero
01652         pup = zero
01653         pdown = zero
01654
01655     ELSEIF (atocc(elempointer(i)) .GT. &
01656            two*spinmaxs + two*spinmaxf + spinmaxd.AND. &
01657            atocc(elempointer(i)) .LE. &
01658            two*spinmaxs + two*spinmaxf + two*spinmaxd) THEN
01659
01660         sup = one
01661         sdown = one
01662         fup = one
01663         fdown = one
01664         dup = one
01665         ddown = (atocc(elempointer(i)) - &
01666            (two*spinmaxs + two*spinmaxf + spinmaxd))/spinmaxd
01667         pup = zero
01668         pdown = zero
01669
01670     ELSEIF (atocc(elempointer(i)) .GT. &
01671            two*spinmaxs + two*spinmaxf + two*spinmaxd .AND. &
01672            atocc(elempointer(i)) .LE. &
01673            two*spinmaxs + two*spinmaxf + two*spinmaxd + spinmaxp) THEN
01674
01675         sup = one
01676         sdown = one
01677         fup = one
01678         fdown = one
01679         dup = one
01680         ddown = one
01681         pup = (atocc(elempointer(i)) - &
01682            (two*spinmaxs + two*spinmaxf + two*spinmaxd))/spinmaxp
01683         pdown = zero
01684
01685     ELSE
01686
01687         sup = one
01688         sdown = one
01689         fup = one
01690         fdown = one
01691         dup = one
01692         ddown = one
01693         pup = one
01694         pdown = (atocc(elempointer(i)) - &
01695            (two*spinmaxs + two*spinmaxf + &
01696            two*spinmaxd + spinmaxp))/spinmaxp
01697
01698     ENDIF
01699
01700     rhoupzero(index + 1) = sup
01701     rhoupzero(index + 2) = pup
01702     rhoupzero(index + 3) = pup
01703     rhoupzero(index + 4) = pup
01704     rhoupzero(index + 5) = dup
01705     rhoupzero(index + 6) = dup
01706     rhoupzero(index + 7) = dup
01707     rhoupzero(index + 8) = dup
01708     rhoupzero(index + 9) = dup
01709     rhoupzero(index + 10) = fup
01710     rhoupzero(index + 11) = fup
01711     rhoupzero(index + 12) = fup
01712     rhoupzero(index + 13) = fup
01713     rhoupzero(index + 14) = fup
01714     rhoupzero(index + 15) = fup
01715     rhoupzero(index + 16) = fup
01716
01717     rhodownzero(index + 1) = sdown
01718     rhodownzero(index + 2) = pdown
01719     rhodownzero(index + 3) = pdown
01720     rhodownzero(index + 4) = pdown
01721     rhodownzero(index + 5) = ddown
01722     rhodownzero(index + 6) = ddown
01723     rhodownzero(index + 7) = ddown
01724     rhodownzero(index + 8) = ddown
01725     rhodownzero(index + 9) = ddown
01726     rhodownzero(index + 10) = fdown
01727     rhodownzero(index + 11) = fdown
01728     rhodownzero(index + 12) = fdown
01729     rhodownzero(index + 13) = fdown
01730     rhodownzero(index + 14) = fdown
01731     rhodownzero(index + 15) = fdown
01732     rhodownzero(index + 16) = fdown
01733
01734     index = index + 16
01735
01736 END SELECT

```

```

01737      ENDDO
01738
01739      IF (restart .EQ. 0) THEN
01740
01741          !
01742          ! Let's initialize our self-consistent spin densities as those for
01743          ! free atoms
01744          !
01745
01746          DO i = 1, hdim
01747
01748              rhoup(i,i) = rhoupzero(i)
01749              rhodown(i,i) = rhodownzero(i)
01750
01751          ENDDO
01752
01753      ELSEIF (restart .EQ. 1) THEN
01754
01755          DO i = 1, hdim
01756
01757              rhoup(i,i) = tmprhoup(i)
01758              rhodown(i,i) = tmprhodown(i)
01759
01760          ENDDO
01761
01762          DEALLOCATE(tmprhoup, tmprhodown)
01763
01764      ENDIF
01765
01766      !      print*, "a"
01767      !      CALL GETDELTASPIN
01768      !      print*, "b"
01769      !      CALL GETSPINE
01770      !      print*, "c"
01771      !      ESPIN_ZERO = ESPIN
01772
01773  ENDIF
01774
01775  RETURN
01776
01777 END SUBROUTINE rhozero

```

8.361 setuparray.f90 File Reference

Modules

- module [setuparray](#)

Variables

- character(len=2), dimension(:), allocatable [setuparray::atele](#)
- real(latteprec), dimension(:), allocatable [setuparray::atocc](#)
- character(len=4), dimension(:), allocatable [setuparray::basis](#)
- real(latteprec), dimension(:,:), allocatable [setuparray::bo](#)
- real(latteprec), dimension(:), allocatable [setuparray::bozero](#)
- character(len=3), dimension(:), allocatable [setuparray::btype](#)
- real(latteprec), dimension(:), allocatable [setuparray::coulombv](#)
- real(latteprec), dimension(:,:), allocatable [setuparray::cr](#)
- real(latteprec), dimension(:), allocatable [setuparray::deltaq](#)
- character(len=2), dimension(:), allocatable [setuparray::ele](#)
- character(len=2), dimension(:), allocatable [setuparray::ele1](#)
- character(len=2), dimension(:), allocatable [setuparray::ele2](#)
- integer, dimension(:), allocatable [setuparray::elempointer](#)
- real(latteprec), dimension(:,:), allocatable [setuparray::f](#)
- real(latteprec), dimension(:,:), allocatable [setuparray::fcoul](#)
- real(latteprec), dimension(:,:), allocatable [setuparray::fpp](#)

- real(latteprec), dimension(:, :), allocatable setuparray::fpul
- real(latteprec), dimension(:, :), allocatable setuparray::fscoul
- real(latteprec), dimension(:, :), allocatable setuparray::fsspin
- real(latteprec), dimension(:, :), allocatable setuparray::ftot
- real(latteprec), dimension(:, :), allocatable setuparray::h
- real(latteprec), dimension(:, :), allocatable setuparray::h0
- real(latteprec), dimension(:, :), allocatable setuparray::hdiag
- real(latteprec), dimension(:, :), allocatable setuparray::hed
- real(latteprec), dimension(:, :), allocatable setuparray::hef
- real(latteprec), dimension(:, :), allocatable setuparray::hep
- real(latteprec), dimension(:, :), allocatable setuparray::hes
- real(latteprec), dimension(:, :), allocatable setuparray::hr0
- real(latteprec), dimension(:, :), allocatable setuparray::hubbardu
- real(latteprec), dimension(:, :), allocatable setuparray::lcnshift
- integer, dimension(:, :), allocatable setuparray::matindlist
- real(latteprec), dimension(:, :), allocatable setuparray::mycharge
- real(latteprec), dimension(:, :), allocatable setuparray::orthorho
- real(latteprec), dimension(:, :), allocatable setuparray::qlist
- real(latteprec), dimension(:, :), allocatable setuparray::respchi
- integer, dimension(:, :), allocatable setuparray::spinindlist

8.362 setuparray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE setuparray
00023
00024     USE myprecision
00025
00026     IMPLICIT NONE
00027     SAVE
00028
00029     INTEGER, ALLOCATABLE :: elempointer(:)
00030     INTEGER, ALLOCATABLE :: matindlist(:), spinindlist(:)
00031     REAL(LATTEPREC), ALLOCATABLE :: cr(:, :)
00032     REAL(LATTEPREC), ALLOCATABLE :: hr0(:)
00033     REAL(LATTEPREC), ALLOCATABLE :: hes(:), hep(:), hed(:), hef(:),
00034     atocc(:)
00035     REAL(LATTEPREC), ALLOCATABLE :: h(:, :), bo(:, :), bozero(:), h0(:, :),
00036     hdiag(:)
00037     REAL(LATTEPREC), ALLOCATABLE :: orthorho(:, :)
00038     REAL(LATTEPREC), ALLOCATABLE :: f(:, :), fpp(:, :), ftot(:, :), fcoul(:, :),
00039     fpul(:, :), fscoul(:, :), fsspin(:, :),
00040     deltaq(:), mycharge(:), qlist(:)
00041     REAL(LATTEPREC), ALLOCATABLE :: lcnshift(:)
00042     REAL(LATTEPREC), ALLOCATABLE :: hubbardu(:)
00043     REAL(LATTEPREC), ALLOCATABLE :: respchi(:)
00044     CHARACTER(LEN=2), ALLOCATABLE :: ele1(:), ele2(:), atele(:)
00045     CHARACTER(LEN=3), ALLOCATABLE :: btype(:)
00046     CHARACTER(LEN=4), ALLOCATABLE :: basis(:)
00047

```

```

00046  ! More Coulomb related data
00047
00048  REAL(LATTEPREC), ALLOCATABLE :: coulombv(:)
00049
00050 END MODULE setuparray

```

8.363 setuptbmd.f90 File Reference

Functions/Subroutines

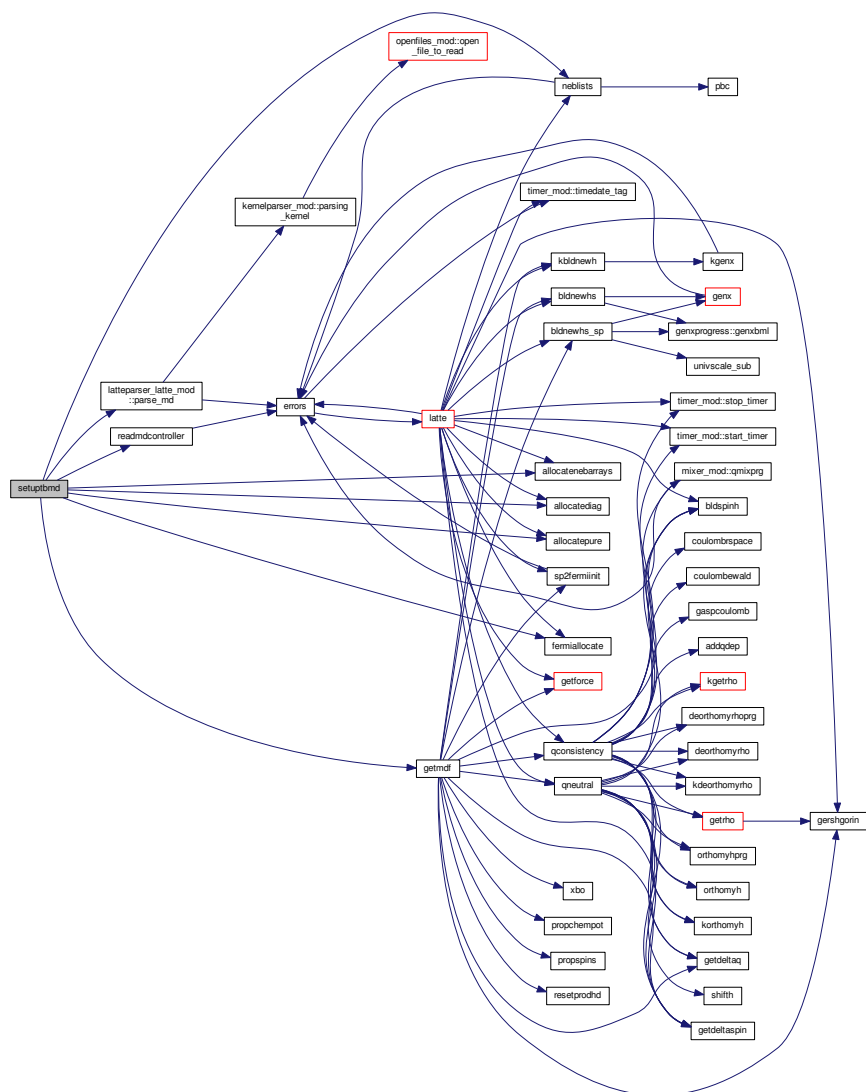
- subroutine [setuptbmd](#) (NEWSYSTEM)

8.363.1 Function/Subroutine Documentation

8.363.1.1 subroutine setuptbmd (integer NEWSYSTEM)

Definition at line 23 of file [setuptbmd.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.364 setuptbmd.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE setuptbmd(NEWSYSTEM)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE mdarray
00028   USE neblastarray
00029   USE coulombarray
00030   USE spinarray
00031   USE virialarray
00032   USE nonoarray
00033   USE myprecision
00034   USE latteparser_latte_mod
00035
00036   IMPLICIT NONE
00037
00038   INTEGER :: I
00039   INTEGER :: ITER
00040   INTEGER :: CURRITER, TOTSCF
00041   INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00042   REAL(LATTEPREC) :: THETIME, NEWSPIN, NEWECOUL
00043   REAL(LATTEPREC) :: RN, MYVOL
00044   INTEGER :: FLAGAND, NEWSYSTEM
00045
00046   IF (existerror) RETURN
00047
00048   !
00049   ! Read MDcontroller to determine what kind of MD simulation to do
00050   !
00051
00052   IF(newsystem == 1 .OR. (.NOT.libinit))THEN
00053
00054     IF (latteinexists) THEN
00055       CALL parse_md("latte.in")
00056     ELSE
00057       CALL readmdcontroller
00058     ENDIF
00059
00060     !
00061     ! Allocate stuff for building the neighbor lists, then build them
00062     !
  
```

```

00063      CALL allocatenebararrays
00064
00065      CALL flush(6)
00066
00067      ENDIF
00068
00069      CALL neblists(0)
00070
00071      !
00072      ! Allocate things depending on which method we're using
00073      ! to get the bond-order
00074      !
00075
00076      IF (.NOT.libinit) THEN
00077          IF (control .EQ. 1) THEN
00078              CALL allocatediag
00079          ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00080              CALL allocatepure
00081          ELSEIF (control .EQ. 3) THEN
00082              CALL fermiallocate
00083          ENDIF
00084      ENDIF
00085
00086      !
00087      IF (verbose >= 1) WRITE(*,*) "Getting MD forces ..."
00088      IF (restart .EQ. 0) CALL getmdf(0,1)
00089
00090      cumdt = zero
00091
00092      totscf = 0
00093
00094      RETURN
00095
00096 END SUBROUTINE setupbmd

```

8.365 shiftH.f90 File Reference

Functions/Subroutines

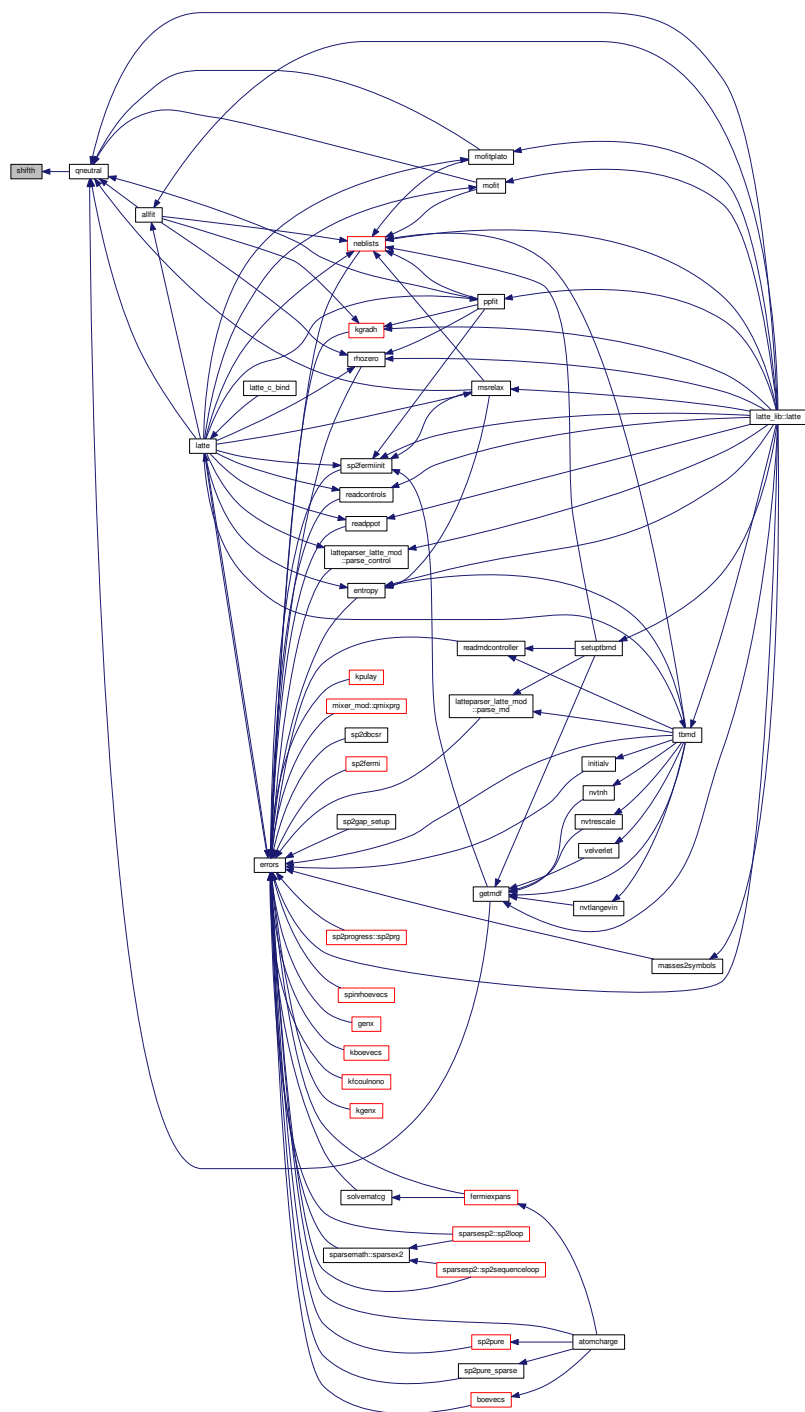
- subroutine [shifth](#) (CHI)

8.365.1 Function/Subroutine Documentation

8.365.1.1 subroutine shifth (real(latteprec) CHI)

Definition at line 23 of file [shiftH.f90](#).

Here is the caller graph for this function:



8.366 shiftH.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE    !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE shift(CHI)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE coulombarray
00027   USE nonoarray
00028   USE kspacearray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K
00034   INTEGER :: INDEX, NUMORB
00035   REAL(LATTEPREC) :: ES, EP, ED, EF
00036   REAL(LATTEPREC) :: HMOD, CHI
00037   COMPLEX(LATTEPREC) :: ZHMOD
00038   IF (existerror) RETURN
00039
00040
00041   index = 0
00042
00043   IF (kon .EQ. 0) THEN
00044
00045     IF ( basistype .EQ. "ORTHO") THEN
00046
00047       DO i = 1, nats
00048
00049         ! Using a constant
00050
00051         lcnshift(i) = lcnshift(i) + chi * deltaq(i)
00052
00053         SELECT CASE(basis(elempointer(i)))
00054
00055           CASE("s")
00056             numorb = 1
00057           CASE("p")
00058             numorb = 3
00059           CASE("d")
00060             numorb = 5
00061           CASE("f")
00062             numorb = 7
00063           CASE("sp")
00064             numorb = 4
00065           CASE("sd")
00066             numorb = 6
00067           CASE("sf")
00068             numorb = 8
00069           CASE("pd")
00070             numorb = 8
00071           CASE("pf")
00072             numorb = 10
00073           CASE("df")
00074             numorb = 12
00075           CASE("spd")
00076             numorb = 9
00077           CASE("spf")
00078             numorb = 11
00079           CASE("sdf")
00080             numorb = 13
00081           CASE("pdf")
00082             numorb = 15
00083           CASE("spdf")
00084             numorb = 16
00085         END SELECT
00086
00087         DO j = 1, numorb
00088           index = index + 1
00089           h(index, index) = hdiag(index) + lcnshift(i)
00090         ENDDO
00091       ENDDO
00092     ENDIF
00093   ENDIF

```

```

00094      ELSEIF ( basistype .EQ. "NONORTHO" ) THEN
00095
00096      !
00097      ! When we have a non-orthogonal basis, the electrostatic
00098      ! potential enters as SH_1
00099      !
00100
00101      DO i = 1, nats
00102
00103          lcnshift(i) = lcnshift(i) + chi * deltag(i)
00104
00105          SELECT CASE(basis(elempointer(i)))
00106
00107              CASE("s")
00108                  numorb = 1
00109              CASE("p")
00110                  numorb = 3
00111              CASE("d")
00112                  numorb = 5
00113              CASE("f")
00114                  numorb = 7
00115              CASE("sp")
00116                  numorb = 4
00117              CASE("sd")
00118                  numorb = 6
00119              CASE("sf")
00120                  numorb = 8
00121              CASE("pd")
00122                  numorb = 8
00123              CASE("pf")
00124                  numorb = 10
00125              CASE("df")
00126                  numorb = 12
00127              CASE("spd")
00128                  numorb = 9
00129              CASE("spf")
00130                  numorb = 11
00131              CASE("sdf")
00132                  numorb = 13
00133              CASE("pdf")
00134                  numorb = 15
00135              CASE("spdf")
00136                  numorb = 16
00137          END SELECT
00138
00139          DO j = 1, numorb
00140
00141              index = index + 1
00142              h(index, index) = hdiag(index) + lcnshift(i)
00143
00144          ENDDO
00145
00146      ENDDO
00147
00148  ENDIF
00149
00150  ELSE
00151
00152      ! IF (KON .EQ. 1) THEN
00153      ! k-space - we have to add the potential to all NKTOT Hamiltonians
00154
00155      ! Orthogonal basis only at the moment
00156
00157      IF ( basistype .EQ. "ORTHO" ) THEN
00158
00159          DO i = 1, nats
00160
00161              ! IF (CONTROL .EQ. 1) THEN
00162
00163              !Using the response function calculated in GETRESPF
00164
00165              ! LCNSHIFT(I) = LCNSHIFT(I) + RESPCHI(I)*DELTAQ(I)
00166
00167              ! ELSE
00168
00169              ! Using a constant
00170
00171              lcnshift(i) = lcnshift(i) + chi * deltag(i)
00172
00173              ! ENDIf
00174
00175          ENDDO
00176
00177          DO k = 1, nktot
00178
00179              index = 0
00180

```

```

00181         DO i = 1, nats
00182
00183             SELECT CASE(basis(elempointer(i)))
00184
00185                 CASE("s")
00186                     numorb = 1
00187                 CASE("p")
00188                     numorb = 3
00189                 CASE("d")
00190                     numorb = 5
00191                 CASE("f")
00192                     numorb = 7
00193                 CASE("sp")
00194                     numorb = 4
00195                 CASE("sd")
00196                     numorb = 6
00197                 CASE("sf")
00198                     numorb = 8
00199                 CASE("pd")
00200                     numorb = 8
00201                 CASE("pf")
00202                     numorb = 10
00203                 CASE("df")
00204                     numorb = 12
00205                 CASE("spd")
00206                     numorb = 9
00207                 CASE("spf")
00208                     numorb = 11
00209                 CASE("sdf")
00210                     numorb = 13
00211                 CASE("pdf")
00212                     numorb = 15
00213                 CASE("spdf")
00214                     numorb = 16
00215             END SELECT
00216
00217             DO j = 1, numorb
00218                 index = index + 1
00219                 hk(index, index, k) = hkdiag(index, k) + cmplx(lcnshift(i))
00220             ENDDO
00221
00222         ENDDO
00223
00224     ENDDO
00225
00226     ENDIF
00227 ENDIF
00228
00229 IF (debugon .EQ. 1) THEN
00230
00231     OPEN(unit=31, status="UNKNOWN", file="myH.dat")
00232
00233     print*, "Caution - the H0+H1 matrix is being written to file"
00234
00235     IF (kon .EQ. 0) THEN
00236
00237         DO i = 1, hdim
00238             WRITE(31,10) (h(i,j), j = 1, hdim)
00239         ENDDO
00240
00241     ELSE
00242
00243         DO k = 1, nktot
00244             WRITE(31,*) k
00245             DO i = 1, hdim
00246                 WRITE(31,10) (hk(i,j,k), j = 1, hdim)
00247             ENDDO
00248         ENDDO
00249
00250     ENDIF
00251
00252     CLOSE(31)
00253
00254 ENDIF
00255
00256 ! PRINT*, LCNSHIFT(1)
00257
00258
00259 10 FORMAT(100g18.9)
00260 11 FORMAT(i5,100g18.9)
00261
00262     RETURN
00263
00264 END SUBROUTINE shifth
00265
00266

```


8.367 shockcomp.f90 File Reference

Functions/Subroutines

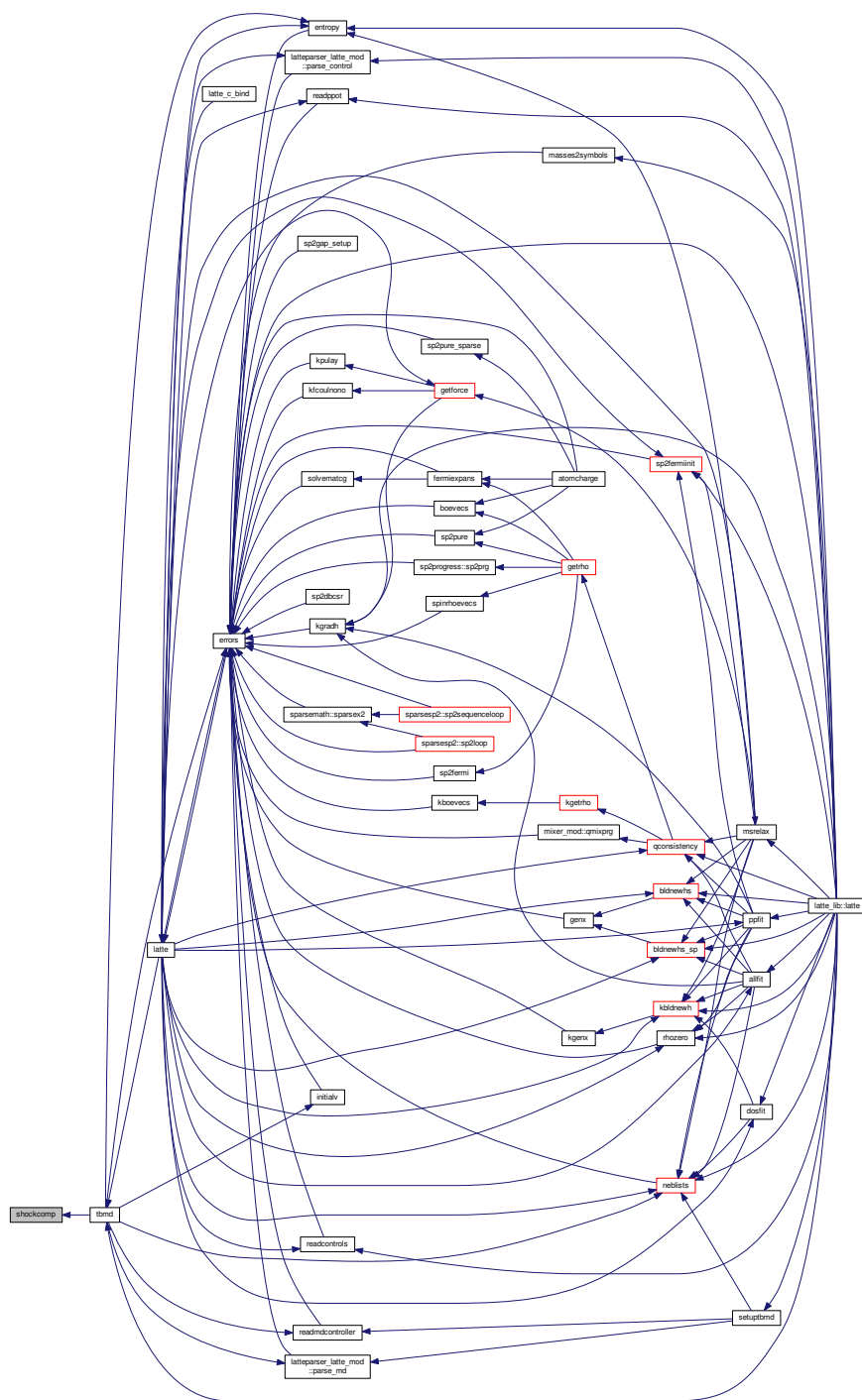
- subroutine [shockcomp](#)

8.367.1 Function/Subroutine Documentation

8.367.1.1 subroutine shockcomp ()

Definition at line 2 of file [shockcomp.f90](#).

Here is the caller graph for this function:



8.368 shockcomp.f90

```
00001 SUBROUTINE shockcomp
00002
00003     USE constants_mod
00004     USE mdarray
00005
00006     IMPLICIT NONE
```

```
00007  IF (existerror) RETURN
00008
00009  ! If the box has 90 degree angles, then its length in the three
00010  ! directions is BOX(1,1), BOX(2,2), AND BOX(3,3)...
00011
00012  box(shockdir, shockdir) = box(shockdir, shockdir) -
    uparticle*dt
00013
00014  ! BOXDIMS(SHOCKDIR) = BOX(2,SHOCKDIR) - BOX(1,SHOCKDIR)
00015
00016  END SUBROUTINE shockcomp
```

8.369 shutdown_dbcsr.f90 File Reference

Functions/Subroutines

- subroutine [shutdown_dbcsr](#)

8.369.1 Function/Subroutine Documentation

8.369.1.1 subroutine shutdown_dbcsr ()

Definition at line 23 of file [shutdown_dbcsr.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE shutdown_dbcsr
00023
00024   USE dbcsr_var_mod
00025   USE dbcsr_config
00026   USE dbcsr_types
00027   USE dbcsr_methods
00028   USE dbcsr_error_handling
00029   USE array_types,          ONLY: array_data,&
00030       array_ild_obj,&
00031       array_new,&
00032       array_nullify,&
00033       array_release,&
00034       array_size
00035   USE dbcsr_io
00036   USE dbcsr_operations
00037   USE dbcsr_ptr_util
00038   USE dbcsr_transformations
00039   USE dbcsr_util
00040   USE dbcsr_work_operations
00041   USE dbcsr_message_passing
00042
00043   USE dbcsr_block_access
00044   USE dbcsr_iterator_operations,  ONLY: dbcsr_iterator_blocks_left,&
00045       dbcsr_iterator_next_block,&
00046       dbcsr_iterator_start,&
00047       dbcsr_iterator_stop
00048
00049   USE dbcsr_dist_operations,      ONLY: create_bl_distribution,&
00050       dbcsr_get_stored_coordinates
00051
00052
00053   CALL dbcsr_distribution_release(dist_a)
00054   CALL dbcsr_mp_release(mp_env)
00055   CALL array_release(row_dist_a)
00056   CALL array_release(col_dist_a)
00057   CALL array_release(row_blk_sizes)
00058   CALL array_release(col_blk_sizes)
00059
00060
00061   CALL mp_world_finalize()
00062
00063
00064 END SUBROUTINE shutdown_dbcsr

```

8.371 slmmp.f90 File Reference

Functions/Subroutines

- `real(latteprec)` function `slmmp` (`L`, `M`, `MP`, `ALPHA`, `COSBETA`)

8.371.1 Function/Subroutine Documentation

8.371.1.1 `real(latteprec)` function `slmmp` (`integer L`, `integer M`, `integer MP`, `real(latteprec) ALPHA`, `real(latteprec) COSBETA`)

Definition at line 23 of file `slmmp.f90`.

8.372 slmmp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION slmmp(L, M, MP, ALPHA, COSBETA)
00023
00024 ! build S function defined in eqn. (7) of PRB 72 165107
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 REAL(LATTEPREC) :: SLMMP, COSBETA, ALPHA
00031 REAL(LATTEPREC), EXTERNAL :: AM, WIGNERD
00032 INTEGER :: L, M, MP
00033
00034 slmmp = am(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00035     WIGNERD(l, abs(m), mp, cosbeta) + WIGNERD(l, abs(m), -mp, cosbeta))
00036
00037 RETURN
00038
00039 END FUNCTION slmmp

```

8.373 solvematcg.f90 File Reference

Functions/Subroutines

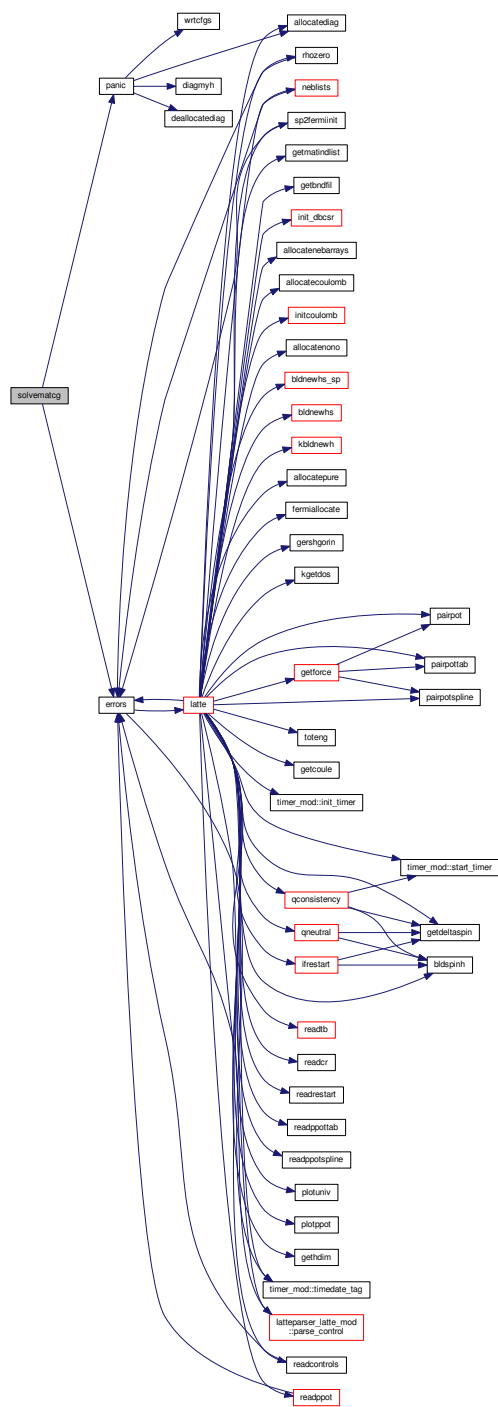
- subroutine [solvematcg](#)

8.373.1 Function/Subroutine Documentation

8.373.1.1 subroutine solvematcg ()

Definition at line 23 of file [solvematcg.f90](#).

Here is the call graph for this function:




```

00012 !
00013 ! Additionally, this program is free software; you can redistribute it
00014 ! and/or modify it under the terms of the GNU General Public License as
00015 ! published by the Free Software Foundation; version 2.0 of the License.
00016 ! Accordingly, this program is distributed in the hope that it will be
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
00019 ! Public License for more details.
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE solvemateg
00023
00024   USE constants_mod
00025   USE fermicommon
00026   USE setuparray
00027   USE spinarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, ITER, BREAKLOOP
00033   REAL(LATTEPREC) :: ERROR2
00034   REAL(LATTEPREC) :: ROVEC, POVEC, R1VEC
00035   REAL(LATTEPREC) :: XALPHA, XBETA
00036   IF (existerror) RETURN
00037
00038   IF (spinon .EQ. 0) THEN
00039
00040     ! Let's try storing X2 in P0...
00041
00042     #ifdef DOUBLEPREC
00043       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00044         bo, hdim, bo, hdim, 0.0d0, p0, hdim)
00045     #elif defined(SINGLEPREC)
00046       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00047         bo, hdim, bo, hdim, 0.0, p0, hdim)
00048     #endif
00049
00050     a = two*(p0 - bo)
00051
00052     DO i = 1, hdim
00053       a(i,i) = a(i,i) + one
00054     ENDDO
00055
00056     #ifdef DOUBLEPREC
00057       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00058         a, hdim, bo, hdim, 0.0d0, r0, hdim)
00059     #elif defined(SINGLEPREC)
00060       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00061         a, hdim, bo, hdim, 0.0, r0, hdim)
00062     #endif
00063
00064     r0 = r0 - p0
00065     p0 = -r0
00066
00067     iter = 0
00068
00069     breakloop = 0
00070
00071     DO WHILE (breakloop .EQ. 0)
00072
00073       iter = iter + 1
00074
00075       IF (iter .EQ. 50) THEN
00076         CALL panic
00077         CALL errors("solvemateg", "SOLVEMATEG NOT CONVERGING")
00078       ENDIF
00079
00080     #ifdef DOUBLEPREC
00081       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00082         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00083     #elif defined(SINGLEPREC)
00084       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00085         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00086     #endif
00087
00088     error2 = zero
00089
00090     !$OMP PARALLEL DO DEFAULT(NONE) &
00091     !$OMP SHARED (TMPMAT, R0, P0, BO, HDIM) &
00092     !$OMP PRIVATE (I, J, ROVEC, POVEC, R1VEC, XALPHA, XBETA) &
00093     !$OMP REDUCTION(+: ERROR2)
00094
00095     DO i = 1, hdim
00096
00097       r0vec = zero

```

```

00099      p0vec = zero
00100      r1vec = zero
00101
00102      DO j = 1, hdim
00103
00104          p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00105          r0vec = r0vec + r0(j,i)*r0(j,i)
00106
00107      ENDDO
00108
00109      xalpha = r0vec/p0vec
00110
00111      DO j = 1, hdim
00112
00113          ! New density matrix
00114
00115          bo(j,i) = bo(j,i) + p0(j,i)*xalpha
00116
00117          ! Calculating R1
00118
00119          r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00120
00121          r1vec = r1vec + r0(j,i)*r0(j,i)
00122
00123      ENDDO
00124
00125      error2 = error2 + r1vec
00126
00127      xbeta = r1vec/r0vec
00128
00129      DO j = 1, hdim
00130
00131          p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00132
00133      ENDDO
00134
00135      ENDDO
00136
00137      !$OMP END PARALLEL DO
00138      !$OMP BARRIER
00139
00140      IF (error2 .LT. cgtol2) THEN
00141          breakloop = 1
00142      ENDIF
00143
00144      !          PRINT*, ITER, ERROR2
00145
00146      ENDDO
00147
00148      ELSE
00149
00150          ! This is the spin-polarized version
00151
00152      #ifdef DOUBLEPREC
00153          CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00154                  rhoup, hdim, rhoup, hdim, 0.0d0, p0, hdim)
00155      #elif defined(SINGLEPREC)
00156          CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00157                  rhoup, hdim, rhoup, hdim, 0.0, p0, hdim)
00158      #endif
00159
00160      a = two*(p0 - rhoup)
00161
00162      DO i = 1, hdim
00163          a(i,i) = a(i,i) + one
00164      ENDDO
00165
00166      #ifdef DOUBLEPREC
00167          CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00168                  a, hdim, rhoup, hdim, 0.0d0, tmpmat, hdim)
00169      #elif defined(SINGLEPREC)
00170          CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00171                  a, hdim, rhoup, hdim, 0.0, tmpmat, hdim)
00172      #endif
00173
00174      r0 = tmpmat - p0
00175      p0 = -r0
00176
00177      iter = 0
00178
00179      breakloop = 0
00180
00181      DO WHILE (breakloop .EQ. 0)
00182
00183          iter = iter + 1
00184
00185          IF (iter .EQ. 50) THEN

```

```

00186         CALL panic
00187         CALL errors("solvemateg",' ("SOLVEMATEG NOT CONVERGING: SPIN UP")')
00188     ENDIF
00189
00190
00191     !      PRINT*, ITER, ERROR2
00192
00193 #ifdef DOUBLEPREC
00194     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00195         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00196 #elif defined(SINGLEPREC)
00197     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00198         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00199 #endif
00200
00201     error2 = zero
00202
00203     !$OMP PARALLEL DO DEFAULT(NONE) &
00204     !$OMP SHARED (TMPMAT, R0, P0, RHOU, HDIM) &
00205     !$OMP PRIVATE (I, J, R0VEC, P0VEC, R1VEC, XALPHA, XBETA) &
00206     !$OMP REDUCTION(+: ERROR2)
00207
00208     DO i = 1, hdim
00209
00210         r0vec = zero
00211         p0vec = zero
00212         r1vec = zero
00213
00214         DO j = 1, hdim
00215
00216             p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00217             r0vec = r0vec + r0(j,i)*r0(j,i)
00218
00219         ENDDO
00220
00221         xalpha = r0vec/p0vec
00222
00223         DO j = 1, hdim
00224
00225             ! New density matrix
00226
00227             rhoup(j,i) = rhoup(j,i) + p0(j,i)*xalpha
00228
00229             ! Calculating R1
00230
00231             r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00232
00233             r1vec = r1vec + r0(j,i)*r0(j,i)
00234
00235         ENDDO
00236
00237         error2 = error2 + r1vec
00238
00239         xbeta = r1vec/r0vec
00240
00241         DO j = 1, hdim
00242
00243             p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00244
00245         ENDDO
00246
00247     ENDDO
00248
00249     !$OMP END PARALLEL DO
00250     !$OMP BARRIER
00251
00252     !      PRINT*, "UP ", ITER, ERROR2
00253
00254     IF (iter .GT. 3 .AND. error2 .LT. cgtol2) THEN
00255         breakloop = 1
00256     ENDIF
00257
00258 ENDDO
00259
00260 #ifdef DOUBLEPREC
00261     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00262         rhodown, hdim, rhodown, hdim, 0.0d0, p0, hdim)
00263 #elif defined(SINGLEPREC)
00264     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00265         rhodown, hdim, rhodown, hdim, 0.0, p0, hdim)
00266 #endif
00267
00268     a = two*(p0 - rhodown)
00269
00270     DO i = 1, hdim
00271         a(i,i) = a(i,i) + one
00272     ENDDO

```

```

00273
00274 #ifdef DOUBLEPREC
00275     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00276         a, hdim, rhodown, hdim, 0.0d0, tmpmat, hdim)
00277 #elif defined(SINGLEPREC)
00278     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00279         a, hdim, rhodown, hdim, 0.0, tmpmat, hdim)
00280 #endif
00281
00282     r0 = tmpmat - p0
00283     p0 = -r0
00284
00285     iter = 0
00286
00287     breakloop = 0
00288
00289     DO WHILE (breakloop .EQ. 0)
00290
00291         iter = iter + 1
00292
00293         IF (iter .EQ. 50) THEN
00294             CALL panic
00295             CALL errors("solvematcg", ' ("SOLVEMATCG NOT CONVERGING: SPIN DOWN")')
00296         ENDIF
00297
00298         ! PRINT*, ITER, ERROR2
00299
00300 #ifdef DOUBLEPREC
00301     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00302         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00303 #elif defined(SINGLEPREC)
00304     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00305         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00306 #endif
00307
00308     error2 = zero
00309
00310     !$OMP PARALLEL DO DEFAULT(NONE) &
00311     !$OMP SHARED (TMPMAT, R0, P0, RHODOWN, HDIM) &
00312     !$OMP PRIVATE (I, J, R0VEC, P0VEC, R1VEC, XALPHA, XBETA) &
00313     !$OMP REDUCTION(+: ERROR2)
00314
00315     DO i = 1, hdim
00316
00317         r0vec = zero
00318         p0vec = zero
00319         r1vec = zero
00320
00321         DO j = 1, hdim
00322
00323             p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00324             r0vec = r0vec + r0(j,i)*r0(j,i)
00325
00326         ENDDO
00327
00328         xalpha = r0vec/p0vec
00329
00330         DO j = 1, hdim
00331
00332             ! New density matrix
00333
00334             rhodown(j,i) = rhodown(j,i) + p0(j,i)*xalpha
00335
00336             ! Calculating R1
00337
00338             r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00339
00340             r1vec = r1vec + r0(j,i)*r0(j,i)
00341
00342         ENDDO
00343
00344         error2 = error2 + r1vec
00345
00346         xbeta = r1vec/r0vec
00347
00348         DO j = 1, hdim
00349
00350             p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00351
00352         ENDDO
00353     ENDDO
00354
00355     !$OMP END PARALLEL DO
00356     !$OMP BARRIER
00357
00358     ! PRINT*, "DOWN ", ITER, ERROR2
00359

```

```
00360
00361      IF (iter .GT. 3 .AND. error2 .LT. cgtol2) THEN
00362          breakloop = 1
00363      ENDIF
00364
00365      ENDDO
00366
00367  ENDIF
00368
00369  RETURN
00370
00371 END SUBROUTINE solvematcg
```

8.375 solvematcg_sparse.f90 File Reference

Functions/Subroutines

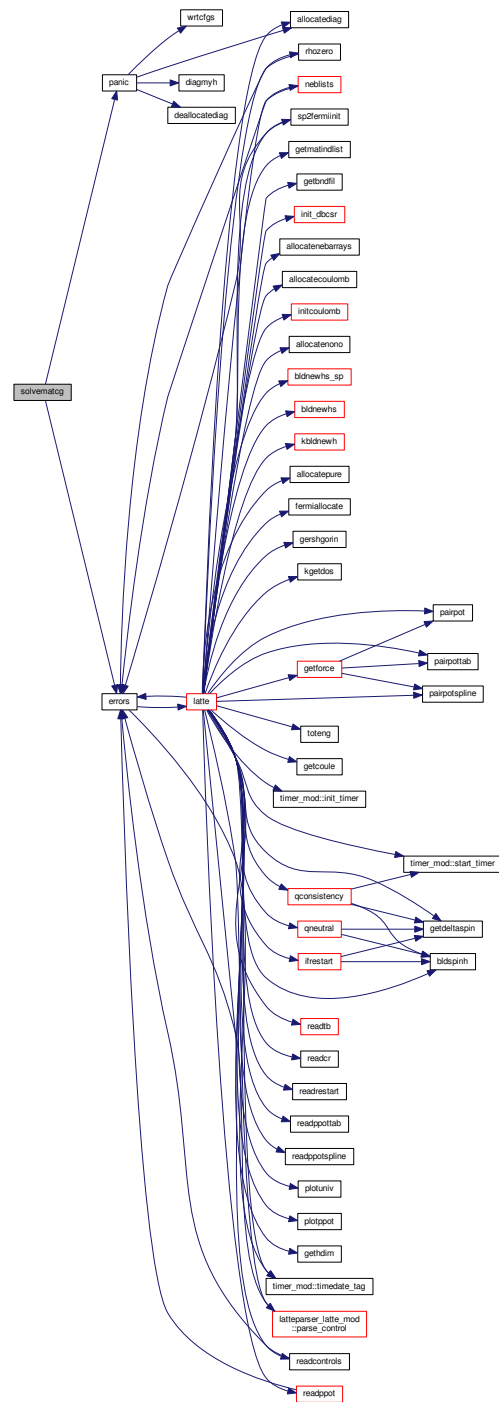
- subroutine [solvematcg](#)

8.375.1 Function/Subroutine Documentation

8.375.1.1 subroutine [solvematcg](#) ()

Definition at line 23 of file [solvematcg_sparse.f90](#).

Here is the call graph for this function:



8.376 solvematacg_sparse.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of      !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE solvemateg
00023
00024   USE constants_mod
00025   USE fermicommon
00026   USE setuparray
00027   USE spinarray
00028   USE myprecision
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, ITER, BREAKLOOP
00033   REAL(LATTEPREC) :: ERROR2
00034   REAL(LATTEPREC) :: R0VEC, P0VEC, R1VEC, XALPHA, XBETA
00035
00036   IF (spinon .EQ. 0) THEN
00037
00038     #ifdef DOUBLEPREC
00039       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00040         bo, hdim, bo, hdim, 0.0d0, x2, hdim)
00041     #elif defined(SINGLEPREC)
00042       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00043         bo, hdim, bo, hdim, 0.0, x2, hdim)
00044     #endif
00045
00046     a = two*(x2 - bo)
00047
00048     DO i = 1, hdim
00049       a(i,i) = a(i,i) + one
00050     ENDDO
00051
00052     #ifdef DOUBLEPREC
00053       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00054         a, hdim, bo, hdim, 0.0d0, tmpmat, hdim)
00055     #elif defined(SINGLEPREC)
00056       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00057         a, hdim, bo, hdim, 0.0, tmpmat, hdim)
00058     #endif
00059
00060     r0 = tmpmat - x2
00061     p0 = minusone*r0
00062
00063     iter = 0
00064
00065     breakloop = 0
00066
00067     DO WHILE (breakloop .EQ. 0)
00068
00069       iter = iter + 1
00070
00071       IF (iter .EQ. 50) THEN
00072         CALL panic
00073         CALL errors("solvemateg_sparse",'("SOLVEMATEG NOT CONVERGING")')
00074       ENDIF
00075
00076     #ifdef DOUBLEPREC
00077       CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00078         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00079     #elif defined(SINGLEPREC)
00080       CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00081         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00082     #endif
00083
00084     error2 = zero
00085
00086     !$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(GUIDED) &
00087     !$OMP SHARED (TMPMAT, R0, P0, BO,HDIM) &
00088     !$OMP PRIVATE(I, J, R0VEC, P0VEC, R1VEC, XALPHA, XBETA) &
00089     !$OMP REDUCTION(+ : ERROR2)
00090
00091     DO i = 1, hdim
00092
00093

```

```

00094         r0vec = zero
00095         p0vec = zero
00096         r1vec = zero
00097
00098         DO j = 1, hdim
00099
00100             p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00101             r0vec = r0vec + r0(j,i)*r0(j,i)
00102
00103         ENDDO
00104
00105         xalpha = r0vec/p0vec
00106
00107         DO j = 1, hdim
00108
00109             ! New density matrix
00110
00111             bo(j,i) = bo(j,i) + p0(j,i)*xalpha
00112
00113             ! Calculating R1
00114
00115             r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00116
00117             r1vec = r1vec + r0(j,i)*r0(j,i)
00118
00119         ENDDO
00120
00121         error2 = error2 + r1vec
00122
00123         xbeta = r1vec/r0vec
00124
00125         DO j = 1, hdim
00126
00127             p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00128
00129         ENDDO
00130
00131     ENDDO
00132
00133     !$OMP END PARALLEL DO
00134
00135     IF (error2 .LT. cgtol2) THEN
00136         breakloop = 1
00137     ENDIF
00138
00139     !          PRINT*, ITER, ERROR2
00140
00141     ENDDO
00142
00143     ELSE
00144
00145         ! This is the spin-polarized version
00146
00147     #ifdef DOUBLEPREC
00148         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00149             rhoup, hdim, rhoup, hdim, 0.0d0, x2, hdim)
00150     #elif defined(SINGLEPREC)
00151         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00152             rhoup, hdim, rhoup, hdim, 0.0, x2, hdim)
00153     #endif
00154
00155         a = two*(x2 - rhoup)
00156
00157         DO i = 1, hdim
00158             a(i,i) = a(i,i) + one
00159         ENDDO
00160
00161     #ifdef DOUBLEPREC
00162         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00163             a, hdim, rhoup, hdim, 0.0d0, tmpmat, hdim)
00164     #elif defined(SINGLEPREC)
00165         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00166             a, hdim, rhoup, hdim, 0.0, tmpmat, hdim)
00167     #endif
00168
00169         r0 = tmpmat - x2
00170         p0 = minusone*r0
00171
00172         iter = 0
00173
00174         breakloop = 0
00175
00176         DO WHILE (breakloop .EQ. 0)
00177
00178             iter = iter + 1
00179
00180             IF (iter .EQ. 50) THEN

```



```

00181         CALL panic
00182         CALL errors("solvemateg_sparse",' ("SOLVEMATEG NOT CONVERGING: SPIN UP"))
00183     ENDIF
00184
00185
00186     !      PRINT*, ITER, ERROR2
00187
00188 #ifdef DOUBLEPREC
00189     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00190         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00191 #elif defined(SINGLEPREC)
00192     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00193         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00194 #endif
00195
00196     error2 = zero
00197
00198     !$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(GUIDED) &
00199     !$OMP SHARED (TMPMAT, R0, P0, RHOU, HDIM) &
00200     !$OMP PRIVATE (I, J, R0VEC, P0VEC, R1VEC, XALPHA, XBETA) &
00201     !$OMP REDUCTION(+: ERROR2)
00202
00203     DO i = 1, hdim
00204
00205         r0vec = zero
00206         p0vec = zero
00207         r1vec = zero
00208
00209         DO j = 1, hdim
00210
00211             p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00212             r0vec = r0vec + r0(j,i)*r0(j,i)
00213
00214         ENDDO
00215
00216         xalpha = r0vec/p0vec
00217
00218         DO j = 1, hdim
00219
00220             ! New density matrix
00221
00222             rhoup(j,i) = rhoup(j,i) + p0(j,i)*xalpha
00223
00224             ! Calculating R1
00225
00226             r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00227
00228             r1vec = r1vec + r0(j,i)*r0(j,i)
00229
00230         ENDDO
00231
00232         error2 = error2 + r1vec
00233
00234         xbeta = r1vec/r0vec
00235
00236         DO j = 1, hdim
00237
00238             p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00239
00240         ENDDO
00241
00242     ENDDO
00243
00244     !$OMP END PARALLEL DO
00245
00246     !      PRINT*, "UP ", ITER, ERROR2
00247
00248     IF (iter .GT. 3 .AND. error2 .LT. cgtol2) THEN
00249         breakloop = 1
00250     ENDIF
00251
00252 ENDDO
00253
00254 #ifdef DOUBLEPREC
00255     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00256         rhodown, hdim, rhodown, hdim, 0.0d0, x2, hdim)
00257 #elif defined(SINGLEPREC)
00258     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00259         rhodown, hdim, rhodown, hdim, 0.0, x2, hdim)
00260 #endif
00261
00262     a = two*(x2 - rhodown)
00263
00264     DO i = 1, hdim
00265         a(i,i) = a(i,i) + one
00266     ENDDO
00267

```

```

00268 #ifdef DOUBLEPREC
00269     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00270         a, hdim, rhodown, hdim, 0.0d0, tmpmat, hdim)
00271 #elif defined(SINGLEPREC)
00272     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00273         a, hdim, rhodown, hdim, 0.0, tmpmat, hdim)
00274 #endif
00275
00276     r0 = tmpmat - x2
00277     p0 = minusone*r0
00278
00279     iter = 0
00280
00281     breakloop = 0
00282
00283     DO WHILE (breakloop .EQ. 0)
00284
00285         iter = iter + 1
00286
00287         IF (iter .EQ. 50) THEN
00288             CALL panic
00289             CALL errors("solvemateg_sparse", '("SOLVEMATCG NOT CONVERGING: SPIN DOWN")')
00290         ENDIF
00291
00292         ! PRINT*, ITER, ERROR2
00293
00294 #ifdef DOUBLEPREC
00295     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00296         a, hdim, p0, hdim, 0.0d0, tmpmat, hdim)
00297 #elif defined(SINGLEPREC)
00298     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00299         a, hdim, p0, hdim, 0.0, tmpmat, hdim)
00300 #endif
00301
00302     error2 = zero
00303
00304     !$OMP PARALLEL DO DEFAULT(NONE) SCHEDULE(GUIDED) &
00305     !$OMP SHARED (TMPMAT, R0, P0, RHODOWN, HDIM) &
00306     !$OMP PRIVATE(I, J, R0VEC, P0VEC, R1VEC, XALPHA, XBETA) &
00307     !$OMP REDUCTION(+: ERROR2)
00308
00309     DO i = 1, hdim
00310
00311         r0vec = zero
00312         p0vec = zero
00313         r1vec = zero
00314
00315         DO j = 1, hdim
00316
00317             p0vec = p0vec + p0(j,i)*tmpmat(j,i)
00318             r0vec = r0vec + r0(j,i)*r0(j,i)
00319
00320         ENDDO
00321
00322         xalpha = r0vec/p0vec
00323
00324         DO j = 1, hdim
00325
00326             ! New density matrix
00327
00328             rhodown(j,i) = rhodown(j,i) + p0(j,i)*xalpha
00329
00330             ! Calculating R1
00331
00332             r0(j,i) = r0(j,i) + tmpmat(j,i)*xalpha
00333
00334             r1vec = r1vec + r0(j,i)*r0(j,i)
00335
00336         ENDDO
00337
00338         error2 = error2 + r1vec
00339
00340         xbeta = r1vec/r0vec
00341
00342         DO j = 1, hdim
00343
00344             p0(j,i) = p0(j,i)*xbeta - r0(j,i)
00345
00346         ENDDO
00347
00348     ENDDO
00349
00350     !$OMP END PARALLEL DO
00351
00352     ! PRINT*, "DOWN ", ITER, ERROR2
00353
00354     IF (iter .GT. 3 .AND. error2 .LT. cgto12) THEN

```

```
00355         breakloop = 1
00356     ENDIF
00357
00358     ENDDO
00359
00360 ENDIF
00361
00362 RETURN
00363
00364 END SUBROUTINE solvemacg
```

8.377 solvematlpack.f90 File Reference

Functions/Subroutines

- subroutine [solvematlpack](#)

8.377.1 Function/Subroutine Documentation

8.377.1.1 subroutine [solvematlpack](#) ()

Definition at line 23 of file [solvematlpack.f90](#).


```

00011 ! with the version available from LANL.
00012 !
00013 ! Additionally, this program is free software; you can redistribute it
00014 ! and/or modify it under the terms of the GNU General Public License as
00015 ! published by the Free Software Foundation; version 2.0 of the License.
00016 ! Accordingly, this program is distributed in the hope that it will be
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
00019 ! Public License for more details.
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE solvematlpack
00023
00024     USE constants_mod
00025     USE fermicommon
00026     USE setuparray
00027     USE spinarray
00028     USE myprecision
00029
00030     IMPLICIT NONE
00031
00032     INTEGER :: I
00033     INTEGER :: INFO
00034     REAL(LATTEPREC), ALLOCATABLE :: IPIV(:)
00035     IF (existerror) RETURN
00036
00037     ALLOCATE(ipiv(hdim))
00038
00039     IF (spinon .EQ. 0) THEN
00040
00041 #ifdef DOUBLEPREC
00042     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00043             bo, hdim, bo, hdim, 0.0d0, x2, hdim)
00044 #elif defined(SINGLEPREC)
00045     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00046             bo, hdim, bo, hdim, 0.0, x2, hdim)
00047 #endif
00048
00049     a = two*(x2 - bo)
00050
00051     DO i = 1, hdim
00052         a(i,i) = a(i,i) + one
00053     ENDDO
00054
00055     bo = x2
00056
00057 #ifdef DOUBLEPREC
00058     CALL dgesv(hdim, hdim, a, hdim, ipiv, bo, hdim, info)
00059 #elif defined(SINGLEPREC)
00060     CALL sgesv(hdim, hdim, a, hdim, ipiv, bo, hdim, info)
00061 #endif
00062
00063     ELSE
00064
00065 #ifdef DOUBLEPREC
00066     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00067             rhoup, hdim, rhoup, hdim, 0.0d0, x2, hdim)
00068 #elif defined(SINGLEPREC)
00069     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00070             rhoup, hdim, rhoup, hdim, 0.0, x2, hdim)
00071 #endif
00072
00073     a = two*(x2 - rhoup)
00074
00075     DO i = 1, hdim
00076         a(i,i) = a(i,i) + one
00077     ENDDO
00078
00079     rhoup = x2
00080
00081 #ifdef DOUBLEPREC
00082     CALL dgesv(hdim, hdim, a, hdim, ipiv, rhoup, hdim, info)
00083 #elif defined(SINGLEPREC)
00084     CALL sgesv(hdim, hdim, a, hdim, ipiv, rhoup, hdim, info)
00085 #endif
00086
00087
00088 #ifdef DOUBLEPREC
00089     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00090             rhodown, hdim, rhodown, hdim, 0.0d0, x2, hdim)
00091 #elif defined(SINGLEPREC)
00092     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00093             rhodown, hdim, rhodown, hdim, 0.0, x2, hdim)
00094 #endif
00095
00096     a = two*(x2 - rhodown)
00097

```

```

00098      DO i = 1, hdim
00099          a(i,i) = a(i,i) + one
00100      ENDDO
00101
00102      rhodown = x2
00103
00104      #ifdef DOUBLEPREC
00105          CALL dgesv(hdim, hdim, a, hdim, ipiv, rhodown, hdim, info)
00106      #elif defined(SINGLEPREC)
00107          CALL sgesv(hdim, hdim, a, hdim, ipiv, rhodown, hdim, info)
00108      #endif
00109
00110      ENDIF
00111
00112      DEALLOCATE(ipiv)
00113
00114      RETURN
00115
00116 END SUBROUTINE solvematlpack
00117
00118
00119
00120
00121

```

8.379 sp2dbcsr.f90 File Reference

Functions/Subroutines

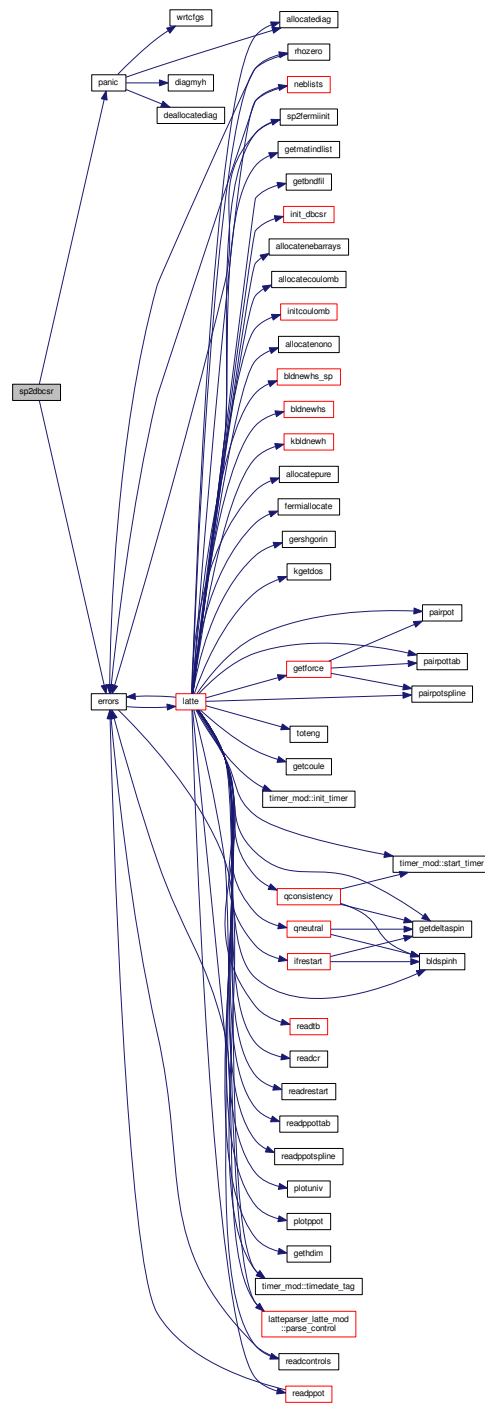
- subroutine [sp2dbcsr](#)

8.379.1 Function/Subroutine Documentation

8.379.1.1 subroutine [sp2dbcsr](#) ()

Definition at line 23 of file [sp2dbcsr.f90](#).

Here is the call graph for this function:



8.380 sp2dbcsr.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2dbcsr
00023
00024 !
00025 ! This subroutine implements Niklasson's SP2 density matrix purification
00026 ! algorithm.
00027 !
00028
00029 USE constants_mod
00030 USE setuparray
00031 USE purearray
00032 USE spinarray
00033 USE nonoarray
00034 USE myprecision
00035
00036 IMPLICIT NONE
00037
00038
00039 INTEGER :: I, J, ITER
00040 INTEGER :: BREAKLOOP
00041 INTEGER :: PUR
00042 REAL(LATTEPREC) :: TRX, OCC, TRX2, LIMDIFF
00043 REAL(LATTEPREC) :: GERSHFACT
00044 REAL(LATTEPREC) :: IDEMPERR, TRXOLD
00045 REAL(LATTEPREC) :: IDEMPERR1, IDEMPERR2
00046 REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0d-14
00047
00048
00049 idemperr = zero
00050 idemperr1 = zero
00051 idemperr2 = zero
00052
00053 !
00054 ! We're also using Niklasson's scheme to determine convergence
00055 !
00056
00057 occ = bndfil*float(hdim)
00058
00059 ! Using intrinsics is probably better than coding this ourselves
00060
00061 IF (basistype .EQ. "ORTHO") THEN
00062   bo = -h/maxminusmin
00063 ELSE
00064   bo = -orthoh/maxminusmin
00065 ENENDIF
00066
00067 gershfact = maxeval/maxminusmin
00068
00069 trx = zero
00070
00071 DO i = 1, hdim
00072   bo(i,i) = gershfact + bo(i,i)
00073   trx = trx + bo(i,i)
00074
00075 ENDDO
00076
00077 iter = 0
00078
00079 breakloop = 0
00080
00081 DO WHILE ( breakloop .EQ. 0 .AND. iter .LT. 100 )
00082
00083   iter = iter + 1
00084
00085   x2 = bo
00086
00087   CALL dgemm('N', 'N', hdim, hdim, hdim, minusone, &
00088     bo, hdim, bo, hdim, one, x2, hdim)
00089
00090
00091   trx2 = zero
00092
00093

```



```

00094      DO i = 1, hdim
00095          trx2 = trx2 + x2(i,i)
00096      ENDDO
00097
00098
00099      limdiff = abs(trx - trx2 - occ) - abs(trx + trx2 - occ)
00100
00101      IF ( limdiff .GE. idemtol ) THEN
00102
00103          bo = bo + x2
00104
00105          trx = trx + trx2
00106
00107      ELSEIF ( limdiff .LT. -idemtol ) THEN
00108
00109          bo = bo - x2
00110
00111          trx = trx - trx2
00112
00113      ENDIF
00114
00115      idemperr2 = idemperr1
00116      idemperr1 = idemperr
00117      idemperr = abs(trx2)
00118
00119      !          PRINT*, ITER, IDEMPERR, ABS(TRX - OCC)
00120      !      WRITE(*,10) ITER, IDEMPERR, IDEMPERR2 - IDEMPERR
00121 10  FORMAT(i4, 2g30.18)
00122      IF (sp2conv .EQ. "REL" .AND. iter .GE. minsp2iter &
00123          .AND. (idemperr2 .LE. idemperr .OR. &
00124              idemperr .LT. idemtol)) breakloop = 1
00125
00126      ! IF (ITER .EQ. 30) BREAKLOOP=1
00127      IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LT. idemtol) breakloop = 1
00128
00129      ENDDO
00130
00131      IF (iter .EQ. 100) THEN
00132          CALL panic
00133          CALL errors("sp2dbcsr", "SP2 purification is not converging: STOP!")
00134      ENDIF
00135
00136      bo = two*bo
00137
00138      RETURN
00139
00140      END SUBROUTINE sp2dbcsr

```

8.381 sp2fermi.f90 File Reference

Functions/Subroutines

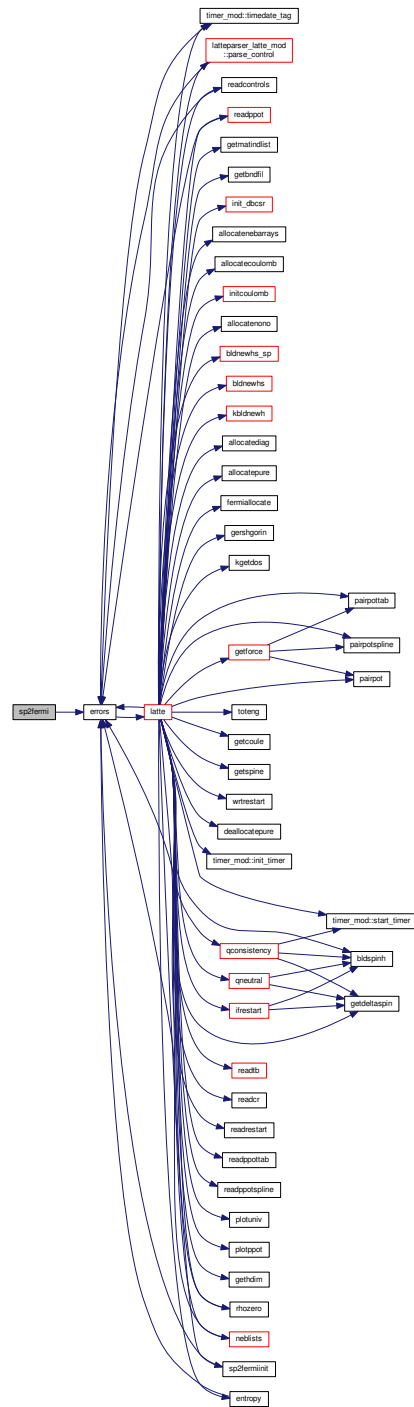
- subroutine [sp2fermi](#)

8.381.1 Function/Subroutine Documentation

8.381.1.1 subroutine sp2fermi ()

Definition at line 23 of file [sp2fermi.f90](#).

Here is the call graph for this function:



[illegible]

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2fermi
00023
00024 !
00025 ! This is an implementation of Niklasson's SP2 algorithm for the
00026 ! Fermi operator (i.e., finite temperature SP2)
00027 !
00028 ! GERSHGORIN and SP2FERMIINIT must be run first to initialize
00029 !
00030
00031 USE constants_mod
00032 USE setuparray
00033 USE purearray
00034 USE spinarray
00035 USE myprecision
00036
00037 IMPLICIT NONE
00038
00039 #ifdef GPUON
00040
00041 CALL sp2fermi_gpu(bndfil, hdim, spinon, bo, rhoup,
00042 rhodown, maxeval, &
00043 h, hup, hdown, maxminusmin, chempot, norecs,
00044 signlist, beta0, &
00045 breaktol, latteprec)
00046
00047 #elif defined(GPUOFF)
00048
00049 INTEGER :: I, J, II
00050 INTEGER :: ITER, BREAKLOOP
00051 REAL(LATTEPREC) :: OCC, GERSHFACT
00052 REAL(LATTEPREC) :: TRX, TRX1, TRXOMX, TRX2
00053 REAL(LATTEPREC) :: LAMBDA
00054 REAL(LATTEPREC) :: MAXSHIFT = one
00055 REAL(LATTEPREC) :: OCCERROR
00056 REAL(LATTEPREC) :: PREVEERROR, PREVEERROR2, PREVEERROR3
00057 REAL(LATTEPREC) :: GEMM_ALPHA, GEMM_BETA
00058 IF (existerror) RETURN
00059
00060 iter = 0
00061
00062 breakloop = 0
00063
00064 preveerror = zero
00065 preveerror2 = zero
00066 preveerror3 = zero
00067
00068 IF (spinon .EQ. 0) THEN
00069
00070 occ = bndfil*float(hdim)
00071
00072 DO WHILE (breakloop .EQ. 0 .AND. iter .LT. 50)
00073
00074 iter = iter + 1
00075
00076 bo = -h/maxminusmin
00077
00078 gershfact = (maxeval - chempot)/maxminusmin
00079
00080 DO i = 1, hdim
00081
00082 bo(i,i) = gershfact + bo(i,i)
00083
00084 ENDDO
00085
00086 DO ii = 1, norecs
00087
00088 x2 = bo
00089
00090 IF (signlist(ii) .EQ. 1) THEN
00091

```

```

00092 #ifdef DOUBLEPREC
00093     CALL dgemm('N', 'N', hdim, hdim, hdim, -1.0d0, &
00094             x2, hdim, x2, hdim, 2.0d0, bo, hdim)
00095 #elif defined(SINGLEPREC)
00096     CALL sgemm('N', 'N', hdim, hdim, hdim, -1.0, &
00097             x2, hdim, x2, hdim, 2.0, bo, hdim)
00098 #endif
00099
00100     ELSE
00101
00102 #ifdef DOUBLEPREC
00103     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00104             x2, hdim, x2, hdim, 0.0d0, bo, hdim)
00105 #elif defined(SINGLEPREC)
00106     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00107             x2, hdim, x2, hdim, 0.0, bo, hdim)
00108 #endif
00109
00110     ENDIF
00111
00112     ENDDO
00113
00114     trx = zero
00115     trx2 = zero
00116
00117     !
00118     ! Now we're getting TrX1 where X1 = beta0*(X-X^2)
00119     !
00120
00121     DO i = 1, hdim
00122
00123         trx = trx + bo(i,i)
00124
00125         DO j = 1, hdim
00126
00127             trx2 = trx2 + bo(j,i)*bo(j,i)
00128
00129         ENDDO
00130
00131     ENDDO
00132
00133
00134
00135     trxomx = trx - trx2
00136
00137     !PRINT*, TRX, TRX2
00138
00139     lambda = (occ - trx)/(beta0*trxomx)
00140
00141     !
00142     ! New CHEMPOT
00143     !
00144
00145     IF (abs(lambda) .GT. maxshift) lambda = sign(maxshift, lambda)
00146
00147     chempot = chempot + lambda
00148
00149     preverror3 = preverror2
00150     preverror2 = preverror
00151     preverror = occerror
00152     occerror = abs(occ - trx)
00153
00154 #ifdef DOUBLEPREC
00155
00156     IF (occerror .LT. breaktol) THEN
00157         breakloop = 1
00158     ENDIF
00159
00160 #elif defined(SINGLEPREC)
00161
00162     IF (occerror .EQ. preverror .OR. &
00163         occerror .EQ. preverror2 .OR. &
00164         occerror .EQ. preverror3 .OR. iter .EQ. 10 ) THEN
00165
00166         breakloop = 1
00167
00168     ENDIF
00169
00170 #endif
00171
00172     ENDDO
00173
00174     IF (iter .EQ. 100) THEN
00175         CALL errors("sp2fermi", "SP2FERMI is not converging: STOP!")
00176     ENDIF
00177
00178     !

```

```

00179      ! If you forget the following you'll spend about a day
00180      ! trying to find the bug in every other subroutine...
00181      !
00182
00183      !      BO = TWO*BO
00184
00185      ! Update occupancy
00186
00187      ! bo = 2 x ( bo + lambda*(beta0*(bo * (I - bo)))) <- we put this into
00188      ! GEMM-form
00189
00190      gemm_alpha = -two*lambda*beta0
00191      gemm_beta = two*(one + lambda*beta0)
00192
00193      x2 = bo
00194
00195      #ifdef DOUBLEPREC
00196      CALL dgemm('N', 'N', hdim, hdim, hdim, gemm_alpha, &
00197              x2, hdim, x2, hdim, gemm_beta, bo, hdim)
00198      #elif defined(SINGLEPREC)
00199      CALL sgemm('N', 'N', hdim, hdim, hdim, gemm_alpha, &
00200              x2, hdim, x2, hdim, gemm_beta, bo, hdim)
00201      #endif
00202
00203      ELSEIF (spinon .EQ. 1) THEN
00204
00205          DO WHILE ( breakloop .EQ. 0 )
00206
00207              iter = iter + 1
00208
00209              IF (iter .EQ. 50) THEN
00210                  CALL errors("sp2fermi", "SP2FERMI is not converging: STOP")
00211                  ENDIF
00212
00213              DO i = 1, hdim
00214                  DO j = i, hdim
00215
00216                      IF (i .EQ. j) THEN
00217
00218                          rhoup(i,i) = (maxeval - hup(i,i) - chempot)/
00219                          rhodown(i,i) = (maxeval - hdown(i,i) -
00220                          chempot) / &
00221                                  maxminusmin
00222
00223                      ELSE
00224
00225                          rhoup(j,i) = (zero - hup(j,i))/maxminusmin
00226                          rhoup(i,j) = rhoup(j,i)
00227
00228                          rhodown(j,i) = (zero - hdown(j,i))/maxminusmin
00229                          rhodown(i,j) = rhodown(j,i)
00230
00231                      ENDIF
00232
00233                  ENDDO
00234              ENDDO
00235
00236              DO ii = 1, norecs
00237
00238                  !
00239                  ! X*X
00240                  !
00241
00242              #ifdef DOUBLEPREC
00243              CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00244                      rhoup, hdim, rhoup, hdim, 0.0d0, x2up,
00245                      hdim)
00246              CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00247                      rhodown, hdim, rhodown, hdim, 0.0d0,
00248                      x2down, hdim)
00249              #elif defined(SINGLEPREC)
00250              CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00251                      rhoup, hdim, rhoup, hdim, 0.0, x2up, hdim)
00252              CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00253                      rhodown, hdim, rhodown, hdim, 0.0, x2down,
00254                      hdim)
00255              #endif
00256
00257              rhoup = rhoup + signlist(ii)*(rhoup - x2up)
00258              rhodown = rhodown + signlist(ii)*(rhodown -

```

```

x2down)
00261
00262      ENDDO
00263
00264      trx = zero
00265      trx1 = zero
00266
00267      !
00268      ! Now we're getting TrX1 where X1 = beta0*(X-X^2)
00269      !
00270
00271      DO i = 1, hdim
00272          DO j = i, hdim
00273
00274              IF (i .EQ. j) THEN
00275
00276                  trx1 = trx1 + rhoup(i,i)*(one - rhoup(i,i)) + &
00277                      rhodown(i,i)*(one - rhodown(i,i))
00278
00279              ELSE
00280
00281                  trx1 = trx1 - two*(rhoup(j,i)*rhoup(j,i) + &
00282                      rhodown(j,i)*rhodown(j,i))
00283
00284              ENDIF
00285
00286          ENDDO
00287
00288          trx = trx + rhoup(i,i) + rhodown(i,i)
00289
00290      ENDDO
00291
00292      trx1 = beta0*trx1
00293
00294      lambda = (totne - trx)/trx1
00295
00296      !
00297      ! New CHEMPOT
00298      !
00299
00300      IF (abs(lambda) .GT. maxshift) THEN
00301          lambda = sign(maxshift, lambda)
00302      ENDIF
00303
00304      chempot = chempot + lambda
00305
00306      preverror3 = preverror2
00307      preverror2 = preverror
00308      preverror = occerror
00309      occerror = abs(totne - trx)
00310
00311      !
00312      ! Convergence tests for double and single precision runs
00313      !
00314
00315      !          PRINT*, ITER, OCCERROR, PREVERROR, PREVERROR2, PREVERROR3
00316
00317      #ifdef DOUBLEPREC
00318
00319          IF (occerror .LT. breaktol) THEN
00320              breakloop = 1
00321          ENDIF
00322
00323      #elif defined(SINGLEPREC)
00324
00325          IF ( (iter .GE. 2) .AND. (occerror .EQ. preverror .OR. &
00326              occerror .EQ. preverror2 .OR. &
00327              occerror .EQ. preverror3 .OR. &
00328              iter .EQ. 10) ) THEN
00329
00330              breakloop = 1
00331
00332          ENDIF
00333
00334      #endif
00335
00336      ENDDO
00337
00338      ENDIF
00339
00340      #endif
00341
00342      RETURN
00343
00344      END SUBROUTINE sp2fermi
00345
00346

```

```
00347 ! NEWBETA = (TRXPLUS - TRXMINUS) / (TWO*DELTA*TRXOMX)
00348
00349 ! PRINT*, "SP2FERMI: BETA = ", NEWBETA
00350 ! PRINT*, "SP2FERMI: KBT = ", ONE/NEWBETA
```

8.383 sp2fermi_init.f90 File Reference

Functions/Subroutines

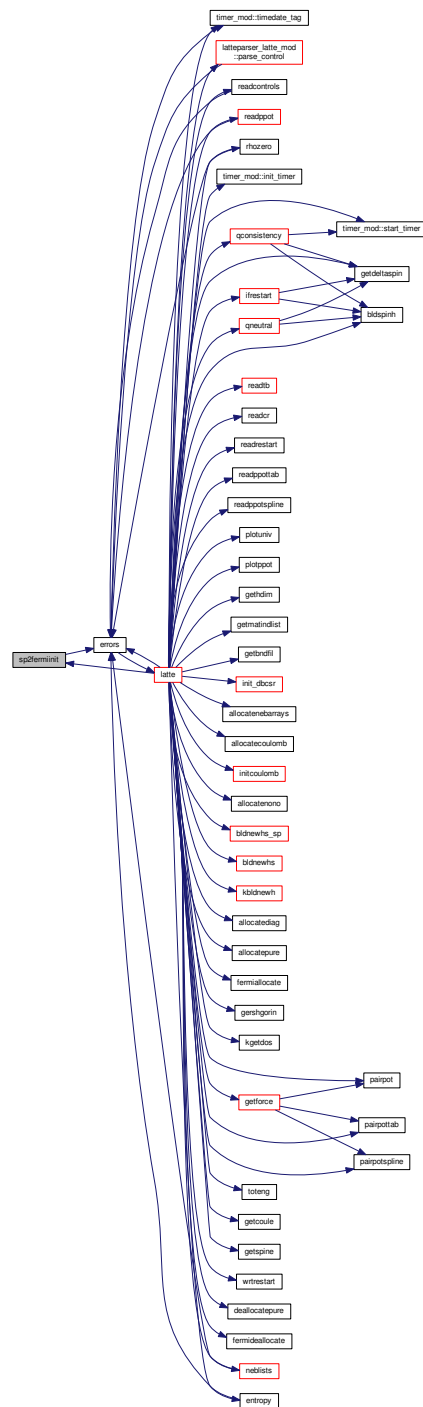
- subroutine [sp2fermiinit](#)

8.383.1 Function/Subroutine Documentation

8.383.1.1 subroutine sp2fermiinit ()

Definition at line 23 of file [sp2fermi_init.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2fermiinit
00023
00024 !
00025 ! This is an implementation of Niklasson's SP2 algorithm for the
00026 ! Fermi operator (i.e., finite temperature SP2)
00027 !
00028
00029 USE constants_mod
00030 USE setuparray
00031 USE purearray
00032 USE spinarray
00033 USE myprecision
00034
00035 IMPLICIT NONE
00036
00037 INTEGER :: I, J, II, ITER
00038 INTEGER :: BREAKLOOP, MYSIGN
00039 REAL(LATTEPREC) :: TRX, OCC
00040 REAL(LATTEPREC) :: TRX2, TRXOMX, TRX1
00041 REAL(LATTEPREC) :: LAMBDA
00042 REAL(LATTEPREC) :: MAXSHIFT = one
00043 REAL(LATTEPREC) :: OCCERROR = zero
00044 REAL(LATTEPREC), ALLOCATABLE :: X0X1(:,,:), X1X0(:,,:), X1(:,,:)
00045 REAL(LATTEPREC), ALLOCATABLE :: X0X1UP(:,,:), X0X1DOWN(:,,:)
00046 REAL(LATTEPREC), ALLOCATABLE :: X1X0UP(:,,:), X1X0DOWN(:,,:)
00047 REAL(LATTEPREC), ALLOCATABLE :: X1UP(:,,:), X1DOWN(:,,:)
00048 REAL(LATTEPREC) :: PREVEERROR, PREVEERROR2, PREVEERROR3
00049 IF (existerror) RETURN
00050 ! REAL(LATTEPREC) :: DELTA = 0.01D0, TRXPLUS, TRXMINUS, NEWBETA, NEWCHEMPOT
00051
00052 !
00053 ! We'll have spin and non-spin dependent versions separate
00054 !
00055
00056 iter = 0
00057
00058 breakloop = 0
00059
00060 preveerror = zero
00061 preveerror2 = zero
00062 preveerror3 = zero
00063
00064 IF (spinon .EQ. 0) THEN
00065
00066     ALLOCATE( x0x1(hdim, hdim), x1x0(hdim, hdim), x1(hdim,
00067 hdim) )
00068
00069     occ = bndfil*float(hdim)
00070
00071     DO WHILE ( breakloop .EQ. 0 )
00072
00073         iter = iter + 1
00074
00075         IF (iter .EQ. 100) THEN
00076             CALL errors("sp2fermi_init","SP2FERMIINIT is not converging: STOP!")
00077             ENDIF
00078
00079         x1 = zero
00080
00081         DO i = 1, hdim
00082             DO j = 1, hdim
00083
00084                 IF (i .EQ. j) THEN
00085
00086                     bo(i,i) = (maxeval - h(i,i) - chempot)/
00087 maxminusmin
00088
00089                     ELSE
00090
00091                         bo(j,i) = (zero - h(j,i))/maxminusmin
00092                         bo(i,j) = bo(j,i)
00093
00094                     ENDIF
00095
00096             END DO
00097         END DO
00098     END DO
00099
00100 
```

```

00092             ENDIF
00093
00094         ENDDO
00095
00096         x1(i,i) = minusone/maxminusmin
00097
00098     ENDDO
00099
00100     DO ii = 1, norecs
00101         !
00102         ! Density matrix squared
00103         !
00104         !
00105
00106     #ifdef DOUBLEPREC
00107         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00108             bo, hdim, bo, hdim, 0.0d0, x2, hdim)
00109     #elif defined(SINGLEPREC)
00110         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00111             bo, hdim, bo, hdim, 0.0, x2, hdim)
00112     #endif
00113
00114     trx = zero
00115     trx2 = zero
00116     DO i = 1, hdim
00117         trx = trx + bo(i,i)
00118         trx2 = trx2 + x2(i,i)
00119     ENDDO
00120
00121     IF ( abs(trx2 - occ) .LT. abs(two*trx - trx2 - occ) ) THEN
00122
00123         mysign = -1
00124
00125     ELSE
00126
00127         mysign = 1
00128
00129     ENDIF
00130
00131     signlist(ii) = mysign
00132
00133     !
00134     ! Density matrix X reponse
00135     !
00136
00137     #ifdef DOUBLEPREC
00138         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00139             bo, hdim, x1, hdim, 0.0d0, x0x1, hdim)
00140     #elif defined(SINGLEPREC)
00141         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00142             bo, hdim, x1, hdim, 0.0, x0x1, hdim)
00143     #endif
00144
00145     #ifdef DOUBLEPREC
00146         CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00147             x1, hdim, bo, hdim, 0.0d0, x1x0, hdim)
00148     #elif defined(SINGLEPREC)
00149         CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00150             x1, hdim, bo, hdim, 0.0, x1x0, hdim)
00151     #endif
00152
00153     !
00154     ! Update response
00155     !
00156
00157     x1 = x1 + float(mysign)*(x1 - x0x1 - x1x0)
00158
00159     !
00160     ! Update density matrix
00161     !
00162
00163     bo = bo + float(mysign)*(bo - x2)
00164
00165     ENDDO
00166
00167     trx = zero
00168     trxomx = zero
00169     trx1 = zero
00170
00171     DO i = 1, hdim
00172         DO j = 1, hdim
00173
00174             IF (i .EQ. j) THEN
00175
00176                 trxomx = trxomx + bo(i,i)*(one - bo(i,i))
00177
00178             
```

```

00179         ELSE
00180
00181             trxomx = trxomx - two*bo(j,i)*bo(j,i)
00182
00183         ENDIF
00184
00185     ENDDO
00186
00187     trx1 = trx1 + x1(i,i)
00188     trx = trx + bo(i,i)
00189
00190     ENDDO
00191
00192     beta0 = trx1/trxomx
00193
00194     kbt = abs(one/beta0)
00195
00196     lambda = (occ - trx)/trx1
00197
00198     IF (abs(lambda) .GT. maxshift) THEN
00199         lambda = sign(maxshift, lambda)
00200     ENDIF
00201
00202     chempot = chempot + lambda
00203
00204     preverror3 = preverror2
00205     preverror2 = preverror
00206     preverror = occerror
00207
00208     occerror = abs(occ - trx)
00209
00210     ! How we figure if we've reached convergence. An absolute
00211     ! tolerance works well in double precision, but in single
00212     ! precision we need to look for noise when we're near
00213     ! self-consistency
00214
00215     #ifdef DOUBLEPREC
00216
00217         IF (occerror .LT. breaktol) THEN
00218
00219             breakloop = 1
00220
00221         ENDIF
00222
00223     #elif defined(SINGLEPREC)
00224
00225         IF (occerror .EQ. preverror .OR. &
00226             occerror .EQ. preverror2 .OR. &
00227             occerror .EQ. preverror3 .OR. iter .EQ. 25 ) THEN
00228
00229             breakloop = 1
00230
00231         ENDIF
00232
00233     #endif
00234
00235     ENDDO
00236
00237     !
00238     ! Little bit of fine tuning...
00239     !
00240
00241     bo = two*(bo + lambda*x1)
00242
00243     DEALLOCATE( x0x1, x1x0 , x1 )
00244
00245     ELSE
00246
00247         ALLOCATE(x0x1up(hdim, hdim), x0x1down(hdim, hdim))
00248         ALLOCATE(x1x0up(hdim, hdim), x1x0down(hdim, hdim))
00249         ALLOCATE(x1up(hdim, hdim), x1down(hdim, hdim))
00250
00251         DO WHILE (breakloop .EQ. 0)
00252
00253             iter = iter + 1
00254
00255             IF (iter .EQ. 100) THEN
00256                 CALL errors("sp2fermi_init","SP2FERMIINIT is not converging: STOP!")
00257             ENDIF
00258
00259             x1up = zero
00260
00261             DO i = 1, hdim
00262                 DO j = i, hdim
00263
00264                     IF (i .EQ. j) THEN
00265

```

```

00266             rhoup(i,i) = (maxeval - hup(i,i) - chempot)/
maxminusmin
00267             rhodown(i,i) = (maxeval - hdown(i,i) -
chempot)/maxminusmin
00268
00269             ELSE
00270
00271             rhoup(j,i) = (zero - hup(j,i))/maxminusmin
00272             rhoup(i,j) = rhoup(j,i)
00273
00274             rhodown(j,i) = (zero - hdown(j,i))/maxminusmin
00275             rhodown(i,j) = rhodown(j,i)
00276
00277             ENDIF
00278
00279             ENDDO
00280
00281             x1up(i,i) = minusone/maxminusmin
00282
00283             ENDDO
00284
00285             x1down = x1up
00286
00287             DO ii = 1, norecs
00288
00289             !
00290             ! Density matrix squared
00291             !
00292
00293             #ifdef DOUBLEPREC
00294             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00295             rhoup, hdim, rhoup, hdim, 0.0d0, x2up,
hdim)
00296             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00297             rhodown, hdim, rhodown, hdim, 0.0d0,
x2down, hdim)
00298             #elif defined(SINGLEPREC)
00299             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00300             rhoup, hdim, rhoup, hdim, 0.0, x2up, hdim)
00301             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00302             rhodown, hdim, rhodown, hdim, 0.0, x2down,
hdim)
00303             #endif
00304
00305             trx = zero
00306             trx2 = zero
00307             DO i = 1, hdim
00308                 trx = trx + rhoup(i,i) + rhodown(i,i)
00309                 trx2 = trx2 + x2up(i,i) + x2down(i,i)
00310             ENDDO
00311
00312             IF (abs(trx2 - totne) .LT. abs(two*trx - trx2 - totne)) THEN
00313
00314                 mysign = -1
00315
00316             ELSE
00317
00318                 mysign = 1
00319
00320             ENDIF
00321
00322             signlist(ii) = mysign
00323
00324             #ifdef DOUBLEPREC
00325
00326             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00327             rhoup, hdim, x1up, hdim, 0.0d0, x0x1up, hdim)
00328             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00329             rhodown, hdim, x1down, hdim, 0.0d0, x0x1down,
hdim)
00330
00331             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00332             x1up, hdim, rhoup, hdim, 0.0d0, x1x0up, hdim)
00333             CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00334             x1down, hdim, rhodown, hdim, 0.0d0, x1x0down,
hdim)
00335
00336             #elif defined(SINGLEPREC)
00337
00338             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00339             rhoup, hdim, x1up, hdim, 0.0, x0x1up, hdim)
00340             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00341             rhodown, hdim, x1down, hdim, 0.0, x0x1down, hdim)
00342
00343             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00344             x1up, hdim, rhoup, hdim, 0.0, x1x0up, hdim)
00345             CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &

```

```

00346         xldown, hdim, rhodown, hdim, 0.0, xlx0down, hdim)
00347
00348 #endif
00349
00350         !
00351         ! Update response
00352         !
00353
00354
00355         xilup = xilup + mysign*(xilup - x0xilup - xlx0up)
00356         xldown = xldown + mysign*(xldown - x0xldown - xlx0down)
00357
00358         !
00359         ! Update density matrix
00360         !
00361
00362         rhoup = rhoup + mysign*(rhoup - x2up)
00363         rhodown = rhodown + mysign*(rhodown - x2down)
00364
00365     ENDDO
00366
00367     trx = zero
00368     trxomx = zero
00369     trx1 = zero
00370
00371     DO i = 1, hdim
00372         DO j = i, hdim
00373
00374             IF (i .EQ. j) THEN
00375
00376                 trxomx = trxomx + rhoup(i,i)*(one - rhoup(i,i)) + &
00377                     rhodown(i,i)*(one - rhodown(i,i))
00378
00379             ELSE
00380
00381                 trxomx = trxomx - two*(rhoup(j,i)*rhoup(j,i) + &
00382                     rhodown(j,i)*rhodown(j,i))
00383
00384             ENDIF
00385
00386         ENDDO
00387
00388         trx1 = trx1 + xilup(i,i) + xldown(i,i)
00389         trx = trx + rhoup(i,i) + rhodown(i,i)
00390
00391     ENDDO
00392
00393     beta0 = trx1/trxomx
00394
00395     kbt = abs(one/beta0)
00396
00397     lambda = (totne - trx)/trx1
00398
00399     IF (abs(lambda) .GT. maxshift) THEN
00400         lambda = sign(maxshift, lambda)
00401     ENDIF
00402
00403     chempot = chempot + lambda
00404
00405     preverror3 = preverror2
00406     preverror2 = preverror
00407     preverror = occerror
00408
00409     occerror = abs(totne - trx)
00410
00411     ! How we figure if we've reached convergence. An absolute
00412     ! tolerance works well in double precision, but in single
00413     ! precision we need to look for noise when we're near
00414     ! self-consistency
00415
00416 #ifdef DOUBLEPREC
00417     IF (occerror .LT. breaktol) THEN
00418
00419         breakloop = 1
00420
00421     ENDIF
00422
00423 #elif defined(SINGLEPREC)
00424     IF (occerror .EQ. preverror .OR. &
00425         occerror .EQ. preverror2 .OR. &
00426         occerror .EQ. preverror3 .OR. iter .EQ. 25 ) THEN
00427
00428         breakloop = 1
00429
00430     ENDIF
00431
00432

```

```

00433
00434 #endif
00435
00436     ENDDO
00437
00438     !
00439     ! Little bit of fine tuning...
00440     !
00441
00442     rhoup = rhoup + lambda*xlup
00443     rhodown = rhodown + lambda*xldown
00444
00445     DEALLOCATE( x0xlup, x0xldown, x1x0up, x1x0down, xlup, xldown )
00446
00447 ENDIF
00448
00449 WRITE(6,'("# SP2FERMI: kBT in eV = ", F14.8)') kbt
00450
00451 RETURN
00452
00453 END SUBROUTINE sp2fermiinit
00454
00455
00456 !
00457 ! Now find the correct beta for calculating the corresponding entropy
00458 !
00459
00460 ! ALLOCATE(X(HDIM, HDIM))
00461
00462 ! X = HALF*BO
00463
00464 ! TRXOMX = ZERO
00465
00466 ! DO I = 1, HDIM
00467 !   DO J = I, HDIM
00468
00469 !       IF (I .EQ. J) THEN
00470
00471 !           TRXOMX = TRXOMX + X(I,I)*(ONE - X(I,I))
00472
00473 !       ELSE
00474
00475 !           TRXOMX = TRXOMX - TWO*X(J,I)*X(J,I)
00476
00477 !       ENDIF
00478
00479 !   ENDDO
00480 ! ENDDO
00481
00482 ! NEWCHEMPOT = CHEMPOT + DELTA
00483
00484 ! DO I = 1, HDIM
00485 !   DO J = I, HDIM
00486
00487 !       IF (I .EQ. J) THEN
00488
00489 !           X(I,I) = (MAXEVAL - H(I,I) - NEWCHEMPOT)/MAXMINUSMIN
00490
00491 !       ELSE
00492
00493 !           X(J,I) = (ZERO - H(J,I))/MAXMINUSMIN
00494 !           X(I,J) = X(J,I)
00495
00496 !       ENDIF
00497
00498 !   ENDDO
00499 ! ENDDO
00500
00501 ! DO II = 1, NORECS
00502
00503 !
00504 ! BO^2
00505 !
00506
00507 ! IF (LATTEPREC .EQ. KIND(0.0D0)) THEN
00508 !     CALL DGEMM('N', 'N', HDIM, HDIM, HDIM, 1.0D0, &
00509 !         X, HDIM, X, HDIM, 0.0D0, X2, HDIM)
00510 ! ELSE
00511 !     CALL SGEMM('N', 'N', HDIM, HDIM, HDIM, 1.0, &
00512 !         X, HDIM, X, HDIM, 0.0, X2, HDIM)
00513 ! ENDIF
00514
00515 !
00516 ! We purify using the sequence of operations defined
00517 ! in SP2FERMIINIT
00518 !
00519

```



```

00520 !      X = X + SIGNLIST(II)*(X - X2)
00521
00522 !  ENDDO
00523
00524 !  TRXPLUS = ZERO
00525 !  DO I = 1, HDIM
00526 !      TRXPLUS = TRXPLUS + X(I,I)
00527 !  ENDDO
00528
00529 !  NEWCHEMPOT = CHEMPOT - DELTA
00530
00531 !  DO I = 1, HDIM
00532 !      DO J = I, HDIM
00533 !
00534 !          IF (I .EQ. J) THEN
00535 !
00536 !              X(I,I) = (MAXEVAL - H(I,I) - NEWCHEMPOT)/MAXMINUSMIN
00537 !
00538 !          ELSE
00539 !
00540 !              X(J,I) = (ZERO - H(J,I))/MAXMINUSMIN
00541 !              X(I,J) = X(J,I)
00542 !
00543 !          ENDIF
00544 !
00545 !      ENDDO
00546 !  ENDDO
00547
00548 !  DO II = 1, NORECS
00549
00550 !
00551 !  BO^2
00552 !
00553 !
00554 !      IF (LATTEPREC .EQ. KIND(0.0D0)) THEN
00555 !          CALL DGEMM('N', 'N', HDIM, HDIM, HDIM, 1.0D0, &
00556 !              X, HDIM, X, HDIM, 0.0D0, X2, HDIM)
00557 !      ELSE
00558 !          CALL SGEMM('N', 'N', HDIM, HDIM, HDIM, 1.0, &
00559 !              X, HDIM, X, HDIM, 0.0, X2, HDIM)
00560 !      ENDIF
00561
00562 !
00563 !  We purify using the sequence of operations defined
00564 !  in SP2FERMIINIT
00565 !
00566 !
00567 !      X = X + SIGNLIST(II)*(X - X2)
00568 !
00569 !  ENDDO
00570
00571 !  TRXMINUS = ZERO
00572 !  DO I = 1, HDIM
00573 !      TRXMINUS = TRXMINUS + X(I,I)
00574 !  ENDDO
00575
00576 !  DEALLOCATE(X)
00577
00578 !  NEWBETA = (TRXPLUS - TRXMINUS)/(TWO*DELTA*TRXOMX)
00579
00580 !  PRINT*, "SP2FERMI: BETA = ", NEWBETA
00581 !  PRINT*, "SP2FERMI: KBT = ", ONE/NEWBETA

```

8.385 sp2gap.f90 File Reference

Functions/Subroutines

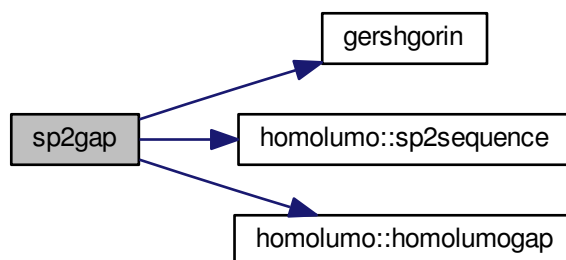
- subroutine [sp2gap](#)

8.385.1 Function/Subroutine Documentation

8.385.1.1 subroutine sp2gap ()

Definition at line 23 of file [sp2gap.f90](#).

Here is the call graph for this function:



8.386 sp2gap.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2gap
00023
00024 !
00025 ! This subroutine the HOMO-LUMO gap-based version of Niklasson's SP2
00026 ! method
00027 !
00028
00029
00030 USE constants_mod
00031 USE setuparray
00032 USE purearray
00033 USE spinarray
00034 USE nonoarray
00035 USE homolumo
00036 USE myprecision
00037
00038 IMPLICIT NONE
00039
00040 INTEGER :: I, J, ITER
00041 REAL(LATTEPREC) :: TRX, TRX2, TRXT, GERSHFACT
00042 IF (existerror) RETURN
00043
00044 ! Estimate the largest and smallest eigenvalues
00045
00046 CALL gershgorin
00047
00048 CALL sp2sequence
00049
00050 IF (basistype .EQ. "ORTHO") THEN
00051   bo = -h/maxminusmin
00052 ELSE
00053   bo = -orthoh/maxminusmin

```

```

00054     ENDIF
00055
00056     gershfact =  maxeval/maxminusmin
00057
00058     trx = zero
00059     DO i = 1, hdim
00060         bo(i,i) = gershfact + bo(i,i)
00061         trx = trx + bo(i,i)
00062     ENDDO
00063
00064     iter = 0
00065
00066     DO WHILE (iter .LT. nr_sp2_iter )
00067
00068         iter = iter + 1
00069
00070     #ifdef DOUBLEPREC
00071         CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00072             bo, hdim, bo, hdim, zero, x2, hdim)
00073     #elif defined(SINGLEPREC)
00074         CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00075             bo, hdim, bo, hdim, zero, x2, hdim)
00076     #endif
00077
00078     trx2 = zero
00079     DO i = 1, hdim
00080         trx2 = trx2 + x2(i,i)
00081     ENDDO
00082
00083     trxt = zero
00084
00085     DO i = 1, hdim
00086         DO j = 1, hdim
00087
00088             trxt = trxt + (bo(j,i) - x2(j,i))*(bo(j,i) - x2(j,i))
00089
00090         ENDDO
00091     ENDDO
00092
00093     IF (pp(iter) .EQ. 0) THEN
00094
00095         trx = two*trx - trx2
00096
00097         bo = two*bo - x2
00098
00099     ELSE
00100
00101         trx = trx2
00102
00103         bo = x2
00104
00105     ENDIF
00106
00107     vv(iter) = sqrt(trxt)
00108
00109     ENDDO
00110
00111     bo = two*bo
00112
00113     CALL homolumogap(iter)
00114
00115     RETURN
00116
00117 END SUBROUTINE sp2gap

```

8.387 sp2gap_setup.f90 File Reference

Functions/Subroutines

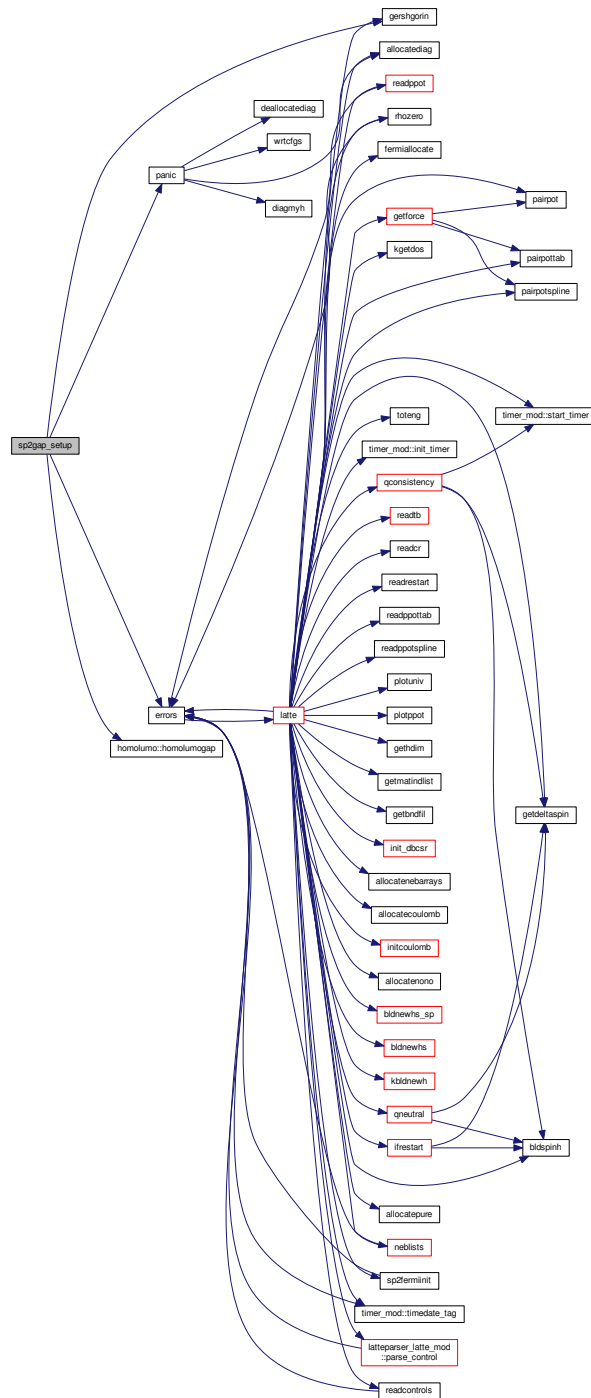
- subroutine [sp2gap_setup](#)

8.387.1 Function/Subroutine Documentation

8.387.1.1 subroutine sp2gap_setup ()

Definition at line 23 of file [sp2gap_setup.f90](#).

Here is the call graph for this function:



8.388 sp2gap_setup.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2gap_setup
00023
00024 !
00025 ! This subroutine the HOMO-LUMO gap-based version of Niklasson's SP2
00026 ! method
00027 !
00028
00029
00030 USE constants_mod
00031 USE setuparray
00032 USE purearray
00033 USE spinarray
00034 USE nonoarray
00035 USE homolumo
00036 USE myprecision
00037
00038 IMPLICIT NONE
00039
00040 INTEGER :: I, J, ITER
00041 INTEGER :: BREAKLOOP
00042 REAL(LATTEPREC) :: TRX, OCC, TRX2, TRXT
00043 REAL(LATTEPREC) :: GERSHFACT
00044 REAL(LATTEPREC) :: IDEMPERR, TRXOLD
00045 REAL(LATTEPREC) :: IDEMPERR1, IDEMPERR2
00046 REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0d-14
00047 IF (existerror) RETURN
00048
00049 ! Estimate the largest and smallest eigenvalues
00050
00051 CALL gershgorin
00052
00053 idemperr = zero
00054 idemperr1 = zero
00055 idemperr2 = zero
00056
00057 !
00058 ! We're also using Niklasson's scheme to determine convergence
00059 !
00060
00061 occ = bndfil*REAL(hdim)
00062
00063 IF (spinon .EQ. 0) THEN
00064
00065     ! Using intrinsics is probably better than coding this ourselves
00066
00067     ! Build the starting guess
00068
00069     IF (basistype .EQ. "ORTHO") THEN
00070         bo = -h/maxminusmin
00071     ELSE
00072         bo = -orthoh/maxminusmin
00073     ENDIF
00074
00075     gershfact = maxeval/maxminusmin
00076
00077     trx = zero
00078     DO i = 1, hdim
00079         bo(i,i) = gershfact + bo(i,i)
00080     TRX = TRX + bo(i,i)
00081     ENDDO
00082
00083     iter = 0
00084
00085     breakloop = 0
00086
00087     ! Compute X^2
00088
00089
00090 #ifdef DOUBLEPREC
00091     CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00092         bo, hdim, bo, hdim, zero, x2, hdim)
00093 #elif defined(SINGLEPREC)

```

```

00094      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00095                bo, hdim, bo, hdim, zero, x2, hdim)
00096 #endif
00097
00098      DO WHILE ( breakloop .EQ. 0 .AND. iter .LT. 100 )
00099
00100          iter = iter + 1
00101
00102          trx2 = zero
00103
00104          DO i = 1, hdim
00105              trx2 = trx2 + x2(i,i)
00106          ENDDO
00107
00108          trxold = trx
00109
00110          trxt = zero
00111
00112          DO i = 1, hdim
00113              DO j = 1, hdim
00114
00115                  ! Both BO amd X2 are symmetric
00116
00117                  trxt = trxt + (bo(j,i) - x2(j,i))*(bo(j,i) - x2(j,i))
00118
00119              ENDDO
00120          ENDDO
00121
00122          IF ( abs(trx2 - occ) .LT. abs(two*trx - trx2 - occ) ) THEN
00123
00124              bo = x2
00125
00126              trx = trx2
00127
00128              pp(iter) = 1
00129
00130          ELSE
00131
00132              bo = two*bo - x2
00133
00134              trx = two*trx - trx2
00135
00136              pp(iter) = 0
00137
00138          ENDIF
00139
00140          ! Compute square again
00141
00142 #ifdef DOUBLEPREC
00143      CALL dgemm('N', 'N', hdim, hdim, hdim, one, &
00144                bo, hdim, bo, hdim, zero, x2, hdim)
00145 #elif defined(SINGLEPREC)
00146      CALL sgemm('N', 'N', hdim, hdim, hdim, one, &
00147                bo, hdim, bo, hdim, zero, x2, hdim)
00148 #endif
00149
00150          vv(iter) = sqrt(trxt)
00151
00152          idemperr2 = idemperr1
00153          idemperr1 = idemperr
00154          idemperr = abs(trx - trxold)
00155
00156          IF (sp2conv .EQ. "REL" .AND. iter .GE. minsp2iter &
00157                .AND. (idemperr2 .LE. idemperr .OR. &
00158                      idemperr .LT. idemtol)) breakloop = 1
00159
00160          IF (sp2conv .EQ. "ABS" .AND. &
00161                abs(trx - trxold) .LT. idemtol) breakloop = 1
00162
00163      ENDDO
00164
00165      IF (iter .EQ. 100) THEN
00166          CALL panic
00167          CALL errors("sp2gap_setup","SP2 purification is not converging: STOP!")
00168      ENDIF
00169
00170      bo = two*bo
00171
00172      CALL homolumogap(iter)
00173
00174  ENDIF
00175
00176  RETURN
00177
00178 END SUBROUTINE sp2gap_setup

```

8.389 sp2progress.f90 File Reference

Modules

- module [sp2progress](#)

To apply the sp2 method using the progress library.

Functions/Subroutines

- subroutine, public [sp2progress::sp2prg](#) ()

This routine implements the sp2 technique with the routines of the progress library.

Variables

- type(sp2data_type), public [sp2progress::sp2d](#)
- logical, public [sp2progress::sp2init](#) = .FALSE.

8.390 sp2progress.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! COPYRIGHT 2010.  LOS ALAMOS NATIONAL SECURITY, LLC. THIS MATERIAL WAS      !
00003 ! PRODUCED UNDER U.S. GOVERNMENT CONTRACT DE-AC52-06NA25396 FOR LOS ALAMOS !
00004 ! NATIONAL LABORATORY (LANL), WHICH IS OPERATED BY LOS ALAMOS NATIONAL      !
00005 ! SECURITY, LLC FOR THE U.S. DEPARTMENT OF ENERGY. THE U.S. GOVERNMENT HAS !
00006 ! RIGHTS TO USE, REPRODUCE, AND DISTRIBUTE THIS SOFTWARE.  NEITHER THE      !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,        !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS           !
00009 ! SOFTWARE.  IF SOFTWARE IS MODIFIED TO PRODUCE DERIVATIVE WORKS, SUCH      !
00010 ! MODIFIED SOFTWARE SHOULD BE CLEARLY MARKED, SO AS NOT TO CONFUSE IT       !
00011 ! WITH THE VERSION AVAILABLE FROM LANL.                                     !
00012 !                                                                            !
00013 ! ADDITIONALLY, THIS PROGRAM IS FREE SOFTWARE; YOU CAN REDISTRIBUTE IT      !
00014 ! AND/OR MODIFY IT UNDER THE TERMS OF THE GNU GENERAL PUBLIC LICENSE AS     !
00015 ! PUBLISHED BY THE FREE SOFTWARE FOUNDATION; VERSION 2.0 OF THE LICENSE.    !
00016 ! ACCORDINGLY, THIS PROGRAM IS DISTRIBUTED IN THE HOPE THAT IT WILL BE      !
00017 ! USEFUL, BUT WITHOUT ANY WARRANTY; WITHOUT EVEN THE IMPLIED WARRANTY OF    !
00018 ! MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.  SEE THE GNU GENERAL !
00019 ! PUBLIC LICENSE FOR MORE DETAILS.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00027 MODULE sp2progress
00028
00029 #ifdef PROGRESSON
00030   USE bml
00031   USE prg_sp2_mod
00032   USE prg_sp2parser_mod
00033
00034   USE setuparray
00035   USE marray
00036   USE purearray
00037   USE nonoarray
00038   USE constants_mod
00039   USE nonoarray
00040   USE myprecision
00041 #endif
00042
00043   PRIVATE
00044
00045   PUBLIC :: sp2prg
00046
00047 #ifdef PROGRESSON
00048   LOGICAL, PUBLIC :: sp2init = .false. !COUTER TO KEEP TRACK OF THE TIMES
00049   ZMAT IS COMPUTED.
00049   TYPE(bml_matrix_t) :: orthoh_bml, orthox_bml
00050   TYPE(sp2data_type), PUBLIC :: sp2d
00051 #endif
00052
00053 CONTAINS

```

```

00054
00058     SUBROUTINE sp2prg()
00059         IMPLICIT NONE
00060
00061     #ifdef PROGRESSON
00062
00063         IF(verbose .GE. 1) WRITE(*,*) "In SP2PRG ..."
00064
00065         ! sp2 is the "SP2DATA_TYPE".
00066         IF(.NOT.sp2init)THEN
00067             CALL prg_parse_sp2(sp2d,"latte.in")
00068             sp2init = .true.
00069         ENDIF
00070
00071
00072         IF(sp2d%MDIM < 0)sp2d%MDIM = hdim
00073
00074         !! Convert Hamiltonian to bml format
00075         !! H should be in orthogonal form, ORTHOH
00076         CALL bml_zero_matrix(bml_matrix_ellpack, bml_element_real, &
00077             latteprec, hdim, sp2d%MDIM, orthoh_bml)
00078         CALL bml_zero_matrix(bml_matrix_ellpack, bml_element_real, &
00079             latteprec, hdim, sp2d%MDIM, orthox_bml)
00080         CALL bml_import_from_dense(bml_matrix_ellpack, &
00081             orthoh, orthoh_bml, zero, sp2d%MDIM)
00082
00083         !! Perform SP2 from progress
00084         IF(sp2d%FLAVOR.EQ."Basic")THEN
00085             CALL prg_sp2_basic(orthoh_bml, orthox_bml, sp2d%THRESHOLD, bndfil,
00086                 sp2d%MINSP2ITER, sp2d%MAXSP2ITER &
00087                 , sp2d%SP2CONV, sp2d%SP2TOL, sp2d%VERBOSE)
00088             ELSEIF(sp2d%FLAVOR.EQ."Alg1") THEN
00089                 CALL prg_sp2_alg1(orthoh_bml, orthox_bml, sp2d%THRESHOLD, bndfil,
00090                     sp2d%MINSP2ITER, sp2d%MAXSP2ITER &
00091                     , sp2d%SP2CONV, sp2d%SP2TOL, sp2d%VERBOSE)
00092             ELSEIF(sp2d%FLAVOR.EQ."Alg2") THEN
00093                 CALL prg_sp2_alg2(orthoh_bml, orthox_bml, sp2d%THRESHOLD, bndfil,
00094                     sp2d%MINSP2ITER, sp2d%MAXSP2ITER &
00095                     , sp2d%SP2CONV, sp2d%SP2TOL, sp2d%VERBOSE)
00096             ELSE
00097                 CALL errors("sp2progress", "No valid SP2 flavor")
00098             ENDIF
00099
00100         ! CALL BML_PRINT_MATRIX("ORTHOP", ORTHOX_BML, 1, 10, 1, 10)
00101
00102         !! Convert orthogonal density bml matrix back to dense
00103         !! BO will be converted to nonorthogonal after this call
00104         CALL bml_export_to_dense(orthox_bml, bo)
00105
00106         CALL bml_deallocate(orthoh_bml)
00107         CALL bml_deallocate(orthox_bml)
00108
00109     #endif
00110
00111     END SUBROUTINE sp2prg
00112
00113 END MODULE sp2progress

```

8.391 sp2pure.f90 File Reference

Functions/Subroutines

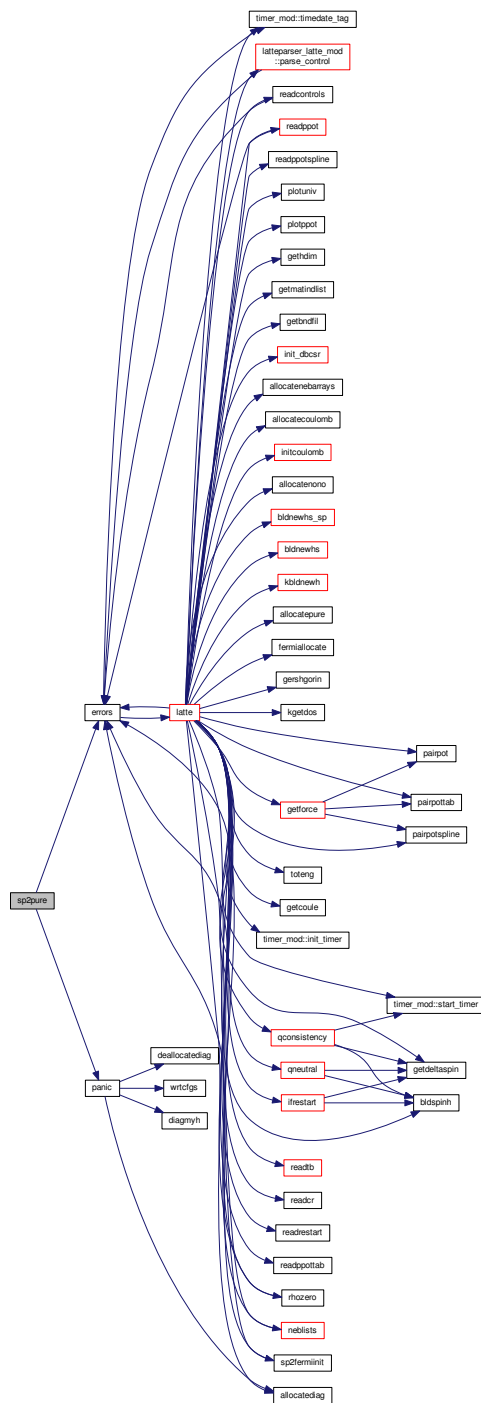
- subroutine [sp2pure](#)

8.391.1 Function/Subroutine Documentation

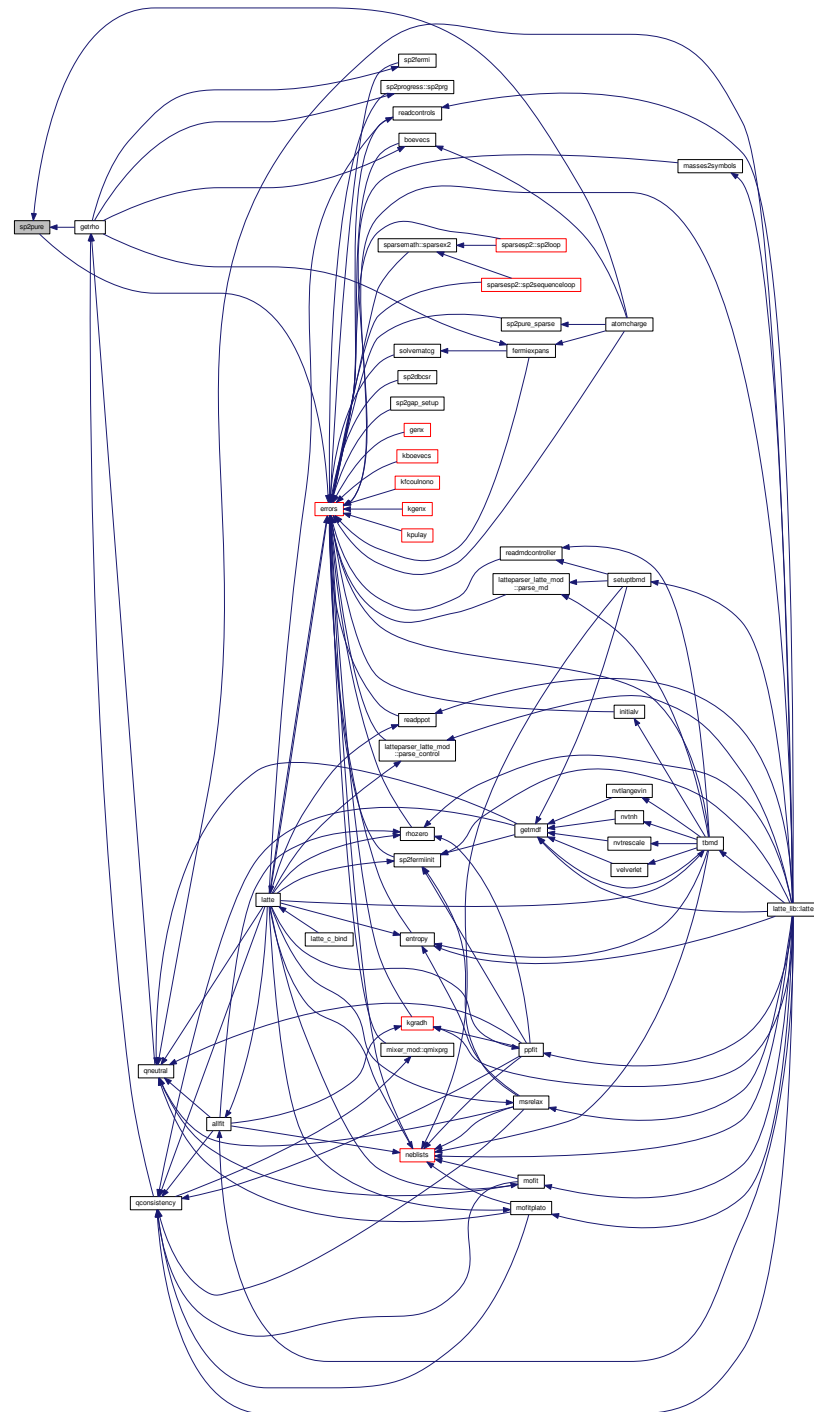
8.391.1.1 subroutine sp2pure ()

Definition at line 23 of file [sp2pure.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.392 sp2pure.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2pure
00023
00024 !
00025 ! This subroutine implements Niklasson's SP2 density matrix purification
00026 ! algorithm.
00027 !
00028
00029
00030 USE constants_mod
00031 USE setuparray
00032 USE purearray
00033 USE spinarray
00034 USE nonoarray
00035 USE myprecision
00036
00037 IMPLICIT NONE
00038
00039 #ifdef GPUON
00040
00041   INTEGER :: sp2convint
00042
00043   IF (sp2conv .EQ. "REL") THEN
00044     sp2convint = 0
00045   ELSEIF (sp2conv .EQ. "ABS") THEN
00046     sp2convint = 1
00047   ELSE
00048     CALL errors("sp2pure",'("Convergence criterion for SP2 not defined")')
00049   ENDIF
00050
00051   IF (basistype .EQ. "ORTHO") THEN
00052
00053     CALL sp2purify(bndfil, hdim, spinon, bo, rhoup,
00054 rhodown, maxeval, &
00055 h, hup, hdown, maxminusmin, minsp2iter, sp2convint,
00056 latteprec)
00057
00058     CALL sp2purify(bndfil, hdim, spinon, bo, rhoup,
00059 rhodown, maxeval, &
00060 orthoh, orthohup, orthohdown, maxminusmin,
00061 minsp2iter, &
00062 sp2convint, latteprec)
00063
00064   ENDIF
00065
00066 ! call sp2evolution(bndfil, hdim, spinon, bo, rhoup, rhodown, maxeval, &
00067 ! h, hup, hdown, maxminusmin, minsp2iter, sp2convint, latteprec)
00068
00069 #elif defined(GPUOFF)
00070
00071   INTEGER :: I, J, ITER
00072   INTEGER :: BREAKLOOP
00073   INTEGER :: PUR
00074   REAL(LATTEPREC) :: TRX, OCC, TRX2, LIMDIFF
00075   REAL(LATTEPREC) :: GERSHFACT
00076   REAL(LATTEPREC) :: IDEMPERR, TRXOLD
00077   REAL(LATTEPREC) :: IDEMPERR1, IDEMPERR2
00078 #ifdef DOUBLEPREC
00079   REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0d-14
00080 #elif defined(SINGLEPREC)
00081   REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0e-14
00082   IF (existerror) RETURN
00083 #endif
00084
00085   idemperr = zero
00086   idemperr1 = zero
00087   idemperr2 = zero
00088
00089 ! We're also using Niklasson's scheme to determine convergence
00090 !

```

```

00090
00091   occ = bndfil*float(hdim)
00092
00093
00094   IF (spinon .EQ. 0) THEN
00095
00096       ! Using intrinsics is probably better than coding this ourselves
00097
00098       ! Build the starting guess
00099
00100       IF (basistype .EQ. "ORTHO") THEN
00101           bo = -h/maxminusmin
00102       ELSE
00103           bo = -orthoh/maxminusmin
00104       ENDIF
00105
00106       gershfact = maxeval/maxminusmin
00107
00108       trx = zero
00109
00110       DO i = 1, hdim
00111
00112           bo(i,i) = gershfact + bo(i,i)
00113       trx = trx + bo(i,i)
00114
00115       ENDDO
00116
00117
00118       iter = 0
00119
00120       breakloop = 0
00121
00122       DO WHILE ( breakloop .EQ. 0 .AND. iter .LT. 100 )
00123
00124           iter = iter + 1
00125
00126           x2 = bo
00127
00128           #ifdef DOUBLEPREC
00129
00130
00131               CALL dgemm('N', 'N', hdim, hdim, hdim, minusone, &
00132                   bo, hdim, bo, hdim, one, x2, hdim)
00133
00134           #elif defined(SINGLEPREC)
00135
00136               CALL sgemm('N', 'N', hdim, hdim, hdim, minusone, &
00137                   bo, hdim, bo, hdim, one, x2, hdim)
00138
00139           #endif
00140
00141           trx2 = zero
00142
00143           DO i = 1, hdim
00144               trx2 = trx2 + x2(i,i)
00145           ENDDO
00146
00147
00148           limdiff = abs(trx - trx2 - occ) - abs(trx + trx2 - occ)
00149
00150           IF ( limdiff .GE. idemtol ) THEN
00151
00152               bo = bo + x2
00153
00154               trx = trx + trx2
00155
00156           ELSEIF ( limdiff .LT. -idemtol ) THEN
00157
00158               bo = bo - x2
00159
00160               trx = trx - trx2
00161
00162           ENDIF
00163
00164           idemperr2 = idemperr1
00165           idemperr1 = idemperr
00166           idemperr = abs(trx2)
00167
00168           ! PRINT*, ITER, IDEMPERR, ABS(TRX - OCC)
00169           ! WRITE(*,10) ITER, IDEMPERR, IDEMPERR2 - IDEMPERR
00170 10    FORMAT(i4, 2g30.18)
00171       IF (sp2conv .EQ. "REL" .AND. iter .GE. minsp2iter &
00172           .AND. (idemperr2 .LE. idemperr .OR. &
00173               idemperr .LT. idemtol)) breakloop = 1
00174
00175       ! IF (ITER .EQ. 30) BREAKLOOP=1
00176       IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LT. idemtol) breakloop = 1

```

```

00177
00178      ENDDO
00179
00180      !      PRINT*, "TRX = ", TRX
00181
00182      IF (iter .EQ. 100) THEN
00183          CALL panic
00184          CALL errors("sp2pure", "SP2 purification is not converging: STOP!")
00185      ENDIF
00186
00187      bo = two*bo
00188
00189      ELSE
00190
00191          !
00192          ! Start by remapping the two H matrices such that
00193          ! all eigenvalues are [0:1]. We have the Gersgorin bounds
00194          ! for both Hup and Hdown
00195          !
00196
00197          IF (basistype .EQ. "ORTHO") THEN
00198              rhoup = -hup/maxminusmin
00199              rhodown = -hdown/maxminusmin
00200          ELSE
00201              rhoup = -orthohup/maxminusmin
00202              rhodown = -orthohdown/maxminusmin
00203          ENDIF
00204
00205          gershfact = maxeval/maxminusmin
00206
00207          trx = zero
00208          DO i = 1, hdim
00209              rhoup(i,i) = gershfact + rhoup(i,i)
00210              rhodown(i,i) = gershfact + rhodown(i,i)
00211              trx = trx + rhoup(i,i) + rhodown(i,i)
00212          ENDDO
00213
00214          !      PRINT*, "TRX = ", TRX, "TOTNE = ", TOTNE
00215
00216          iter = 0
00217
00218          breakloop = 0
00219
00220          DO WHILE ( breakloop .EQ. 0 .AND. iter .LT. 100 )
00221
00222              iter = iter + 1
00223
00224              x2up = rhoup
00225              x2down = rhodown
00226
00227              #ifdef DOUBLEPREC
00228
00229                  CALL dgemm('N', 'N', hdim, hdim, hdim, minusone, &
00230                      rhoup, hdim, rhoup, hdim, one, x2up, hdim)
00231                  CALL dgemm('N', 'N', hdim, hdim, hdim, minusone, &
00232                      rhodown, hdim, rhodown, hdim, one, x2down,
00233                      hdim)
00234              #elif defined(SINGLEPREC)
00235
00236                  CALL sgemm('N', 'N', hdim, hdim, hdim, minusone, &
00237                      rhoup, hdim, rhoup, hdim, one, x2up, hdim)
00238                  CALL sgemm('N', 'N', hdim, hdim, hdim, minusone, &
00239                      rhodown, hdim, rhodown, hdim, one, x2down,
00240                      hdim)
00241              #endif
00242
00243              trx2 = zero
00244
00245              DO i = 1, hdim
00246                  trx2 = trx2 + x2up(i,i) + x2down(i,i)
00247              ENDDO
00248
00249              trxold = trx
00250
00251              limdiff = abs(trx - trx2 - totne) - abs(trx + trx2 - totne)
00252
00253              IF (limdiff .GE. idemtoll) THEN
00254
00255                  !$OMP PARALLEL DO DEFAULT(NONE) &
00256                  !$OMP SHARED(RHOUP, RHODOWN, X2UP, X2DOWN, HDIM) &
00257                  !$OMP PRIVATE(I,J)
00258
00259                  DO i = 1, hdim
00260                      DO j = 1, hdim
00261

```

```

00262             rhoup(j,i) = rhoup(j,i) + x2up(j,i)
00263             rhodown(j,i) = rhodown(j,i) + x2down(j,i)
00264
00265             ENDDO
00266         ENDDO
00267
00268         !$OMP END PARALLEL DO
00269         !$OMP BARRIER
00270
00271         trx = trx + trx2
00272
00273     ELSEIF ( limdiff .LT. -idemtol ) THEN
00274
00275
00276         !$OMP PARALLEL DO DEFAULT(NONE) &
00277         !$OMP SHARED(RHoup, RHODOWN, X2UP, X2DOWN, HDIM) &
00278         !$OMP PRIVATE(I,J)
00279
00280         DO i = 1, hdim
00281             DO j = 1, hdim
00282
00283                 rhoup(j,i) = rhoup(j,i) - x2up(j,i)
00284                 rhodown(j,i) = rhodown(j,i) - x2down(j,i)
00285
00286             ENDDO
00287         ENDDO
00288
00289         !$OMP END PARALLEL DO
00290         !$OMP BARRIER
00291
00292         trx = trx - trx2
00293
00294     ENDIF
00295
00296     idemperr2 = idemperr1
00297     idemperr1 = idemperr
00298     idemperr = abs(trx2)
00299
00300     !          PRINT*, ITER, IDEMPERR, LIMDIFF, TRX - TOTNE
00301
00302     IF (sp2conv .EQ. "REL" .AND. iter .GE. minsp2iter &
00303         .AND. (idemperr2 .LE. idemperr .OR. &
00304             idemperr .LT. idemtol) ) breakloop = 1
00305
00306     IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LE. idemtol) breakloop = 1
00307
00308     ENDDO
00309
00310     IF (iter .EQ. 100) THEN
00311         CALL panic
00312         CALL errors("sp2pure", "SP2 purification not converging: STOP!")
00313     ENDIF
00314
00315 ENDIF
00316
00317 #endif
00318
00319 RETURN
00320
00321 END SUBROUTINE sp2pure

```

8.393 sp2pure_sparse.f90 File Reference

Functions/Subroutines

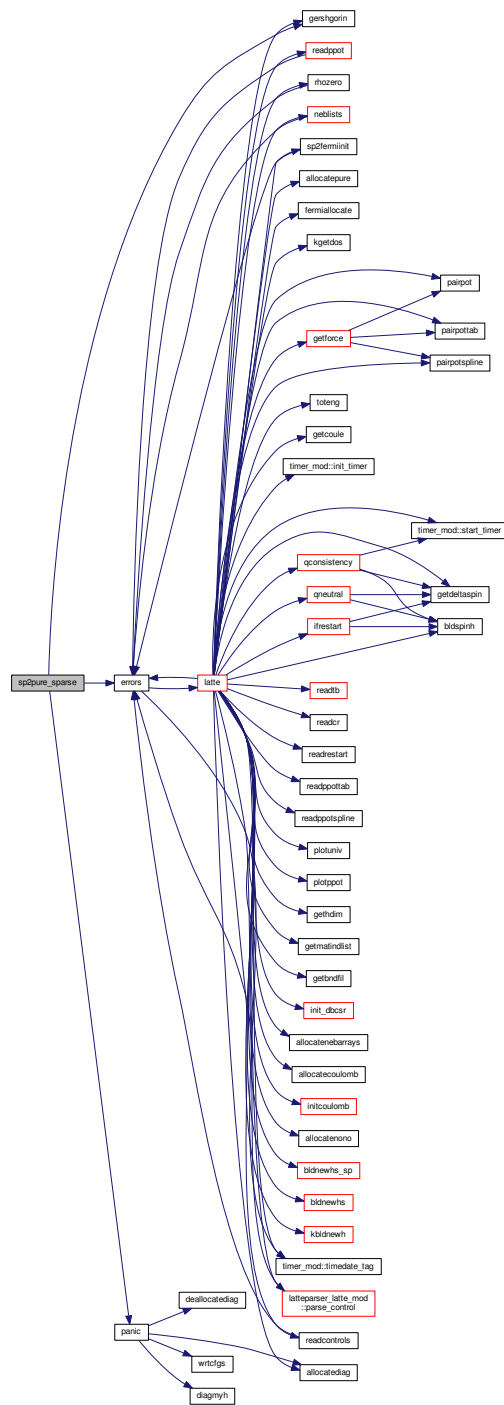
- subroutine [sp2pure_sparse](#)

8.393.1 Function/Subroutine Documentation

8.393.1.1 subroutine sp2pure_sparse ()

Definition at line 23 of file [sp2pure_sparse.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.394 sp2pure_sparse.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2pure_sparse
00023
00024 !
00025 ! This subroutine implements Niklasson's SP2 density matrix purification
00026 ! algorithm.
00027 !
00028
00029 #ifdef DBCSR_ON
00030 !*****
00031 !use these dbcsr mod files
00032 USE dbcsr_var_mod
00033 USE dbcsr_types
00034 USE dbcsr_methods
00035 USE dbcsr_error_handling
00036 USE array_types, ONLY: array_data,&
00037     array_ild_obj,&
00038     array_new,&
00039     array_nullify,&
00040     array_release,&
00041     array_size
00042 USE dbcsr_io
00043 USE dbcsr_operations
00044 USE dbcsr_ptr_util
00045 USE dbcsr_transformations
00046 USE dbcsr_util
00047 USE dbcsr_work_operations
00048 USE dbcsr_message_passing
00049
00050 USE dbcsr_block_access
00051 USE dbcsr_iterator_operations, ONLY: dbcsr_iterator_blocks_left,&
00052     dbcsr_iterator_next_block,&
00053     dbcsr_iterator_start,&
00054     dbcsr_iterator_stop
00055
00056 USE dbcsr_dist_operations, ! ONLY: create_bl_distribution,&
00057     ! dbcsr_get_stored_coordinates
00058 !*****
00059 #endif
00060 USE constants_mod
00061 USE setuparray
  
```



```

00062 USE sparsearray
00063 USE nonoarray
00064 USE myprecision
00065
00066 IMPLICIT NONE
00067
00068 INTEGER :: I, J, K, ITERZ, a, b
00069 INTEGER :: BREAKLOOP
00070 INTEGER :: IP, JP, KP, NNZX2, PREVNZX2
00071 INTEGER :: NEWROW
00072 INTEGER :: COUNT, KEEP, PREVKEEP
00073 INTEGER :: NEWXB
00074
00075 !
00076 ! Naming: CX = ColIndX  RX = RowIndX  VX = ValX
00077 !
00078
00079 INTEGER, ALLOCATABLE :: CX(:), CXTMP(:)
00080 REAL(LATTEPREC) :: TRX, OCC
00081 REAL(LATTEPREC) :: TRX2, TR2XX2, TRXOLD
00082 REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0e-14
00083 REAL(LATTEPREC), ALLOCATABLE :: VX(:), VXTMP(:)
00084 REAL(LATTEPREC) :: LIMDIFF, ABSHAM
00085 REAL(LATTEPREC) :: CIRCLE(2), RADIUS, GERSHFACT
00086 REAL(LATTEPREC), ALLOCATABLE :: BLOCK_2D(:,:)
00087 REAL(LATTEPREC) :: FROB
00088
00089 !
00090 ! We're also using Niklasson's scheme to determine convergence
00091 !
00092
00093 REAL(LATTEPREC) :: IDEMPERR = zero, idemperr1 = zero, idemperr2 = zero
00094 IF (existerror) RETURN
00095
00096 occ = bndfil*float(hdim)
00097
00098 trx = zero
00099
00100 ! build the sparse H matrix and compute the bounds using
00101 ! Gershgorin circles
00102
00103 CALL gershgorin
00104
00105 #ifndef DBCSR_ON
00106
00107 IF (spinon .EQ. 1) THEN
00108     CALL errors("sp2pure_sparse", "Sparse SP2 with DBSCR for &
00109         & spin-polarized systems hasn't been implemented just yet")
00110 ENDIF
00111
00112
00113 !
00114 ! We have to put things into an array which has been padded
00115 ! to handle the block size
00116 !
00117
00118 bo_padded = zero
00119
00120 IF (basistype .EQ. "ORTHO") THEN
00121
00122     DO i = 1, hdim
00123         DO j = 1, hdim
00124             bo_padded(j,i) = -h(j,i)/maxminusmin
00125         ENDDO
00126     ENDDO
00127
00128 ELSE
00129
00130     DO i = 1, hdim
00131         DO j = 1, hdim
00132             bo_padded(j,i) = -orthoh(j,i)/maxminusmin
00133         ENDDO
00134     ENDDO
00135
00136 ENDIF
00137
00138 gershfact = maxeval/maxminusmin
00139
00140
00141 !important for quick convergence
00142
00143 trx = zero
00144
00145 DO i = 1, hdim
00146
00147     bo_padded(i,i) = gershfact + bo_padded(i,i)
00148     trx = trx + bo_padded(i,i)

```

```

00149
00150 ENDDO
00151
00152 !initiallizes matrix a
00153
00154 CALL dbcsr_init(matrix_a)
00155 CALL dbcsr_init(matrix_b)
00156
00157 !create the dbcsr matrix, a double precision non symmetric matrix
00158 !with nblkrows_total x nblkcols_total blocks and
00159 !sizes "sum(row_blk_sizes)" x "sum(col_blk_sizes)", distributed as
00160 !specified by the dist_a object
00161
00162 CALL dbcsr_create (matrix_a,"Matrix A ",&
00163   dist_a, dbcsr_type_no_symmetry,&
00164   row_blk_sizes,&
00165   col_blk_sizes,&
00166   data_type=dbcsr_type_real_8,&
00167   error=error)
00168
00169 CALL dbcsr_create (matrix_b,"Matrix B ",&
00170   dist_a, dbcsr_type_no_symmetry,&
00171   row_blk_sizes,&
00172   col_blk_sizes,&
00173   data_type=dbcsr_type_real_8,&
00174   error=error)
00175
00176 !puts values into matrix_a
00177
00178 ALLOCATE(block_2d(blksz,blksz))
00179
00180 DO j=1, dbcsr_nblkcols_total(matrix_a)
00181   DO i=1, dbcsr_nblkrows_total(matrix_a)
00182
00183     frob = zero
00184     DO a = 1, blksz
00185       DO b = 1, blksz
00186
00187         block_2d(b,a)=bo_padded(blksz*(i-1)+b,blksz*(j-1)+a)
00188         frob = frob + block_2d(b,a)*block_2d(b,a)
00189
00190       ENDDO
00191     ENDDO
00192
00193     tr = .false.
00194
00195     IF (frob .GT. 1.0d-12) THEN
00196
00197       CALL dbcsr_get_stored_coordinates (matrix_a, i, j, tr,
00198   proc_holds_blk)
00199
00200       IF (proc_holds_blk .EQ. dbcsr_mp_mynode(dbcsr_distribution_mp(dbcsr_distribution(
00201   matrix_a)))) THEN
00202
00203         !call to put a block into the matrix
00204
00205         CALL dbcsr_put_block(matrix_a, i, j, block_2d)
00206
00207       ENDIF
00208     ENDDO
00209   ENDDO
00210
00211 !refinallizes matrix a
00212
00213 CALL dbcsr_finalize(matrix_a, error=error)
00214 CALL dbcsr_finalize(matrix_b, error=error)
00215
00216 ! Now throw away all the BLKSZ*BLKSZ blocks that have nothing in them
00217
00218 ! OCC = dbcsr_get_occupation(matrix_a)
00219 ! PRINT*, "OCC 1 = ", OCC
00220
00221 ! CALL dbcsr_filter(matrix_a, eps=1.0d-12, error=error)
00222
00223 ! OCC = dbcsr_get_occupation(matrix_a)
00224 ! PRINT*, "OCC 2 = ", OCC
00225
00226 iterz = 0
00227
00228 breakloop = 0
00229
00230 DO WHILE ( breakloop .EQ. 0 .AND. iterz .LT. 100 )
00231
00232   iterz = iterz + 1
00233

```

```

00234
00235     CALL dbcsr_copy(matrix_b, matrix_a, error=error)
00236
00237     IF (thresholdon .EQ. 1) THEN
00238
00239     ! Since we threshold based on the Frobenius norm, we need to multiply
00240     ! the numerical threshold by the block size in order to maintain a
00241     ! connection with our element-by-element algorithm
00242     ! X_f = sqrt(sum a_ij^2) -> sqrt(blksz ^2 * eps^2) = blksz*eps
00243
00244     CALL dbcsr_multiply('n','n', minusone, matrix_a,
matrix_a, one, &
00245         matrix_b, filter_eps=float(blksz)*numthresh,
error=error)
00246
00247     ELSE
00248
00249     CALL dbcsr_multiply('n','n', minusone, matrix_a,
matrix_a, one, &
00250         matrix_b, error=error)
00251
00252     ENDIF
00253
00254     trx2 = zero
00255
00256     CALL dbcsr_trace(matrix_b, trx2, error=error)
00257
00258     limdiff = abs(trx - trx2 - occ) - abs(trx + trx2 - occ)
00259
00260     IF ( limdiff .GE. idemtoll ) THEN
00261
00262         !X <- X + X_temp
00263         CALL dbcsr_add(matrix_a, matrix_b, one, one, error=
error)
00264
00265         trx = trx + trx2
00266
00267     ELSEIF ( limdiff .LT. -idemtol ) THEN
00268
00269         !X <- X - X_temp
00270         CALL dbcsr_add(matrix_a, matrix_b, one, minusone,
error=error)
00271
00272         trx = trx - trx2
00273
00274     ENDIF
00275
00276     ! Filtering, if desired
00277
00278     ! IF (THRESHOLDON .EQ. 1) CALL dbcsr_filter(matrix_a, &
! eps=NUMTHRESH, error=error)
00280
00281     idemperr2 = idemperr1
00282     idemperr1 = idemperr
00283     idemperr = abs(trx2)
00284
00285
00286     ! WRITE(*,10) ITERZ, IDEMPERR, IDEMPERR2 - IDEMPERR
00287 10 FORMAT(i4, 2g30.18)
00288     IF (sp2conv .EQ. "REL" .AND. iterz .GE. minsp2iter &
.AND. (idemperr2 .LE. idemperr .OR. &
idemperr .LT. idemtoll)) breakloop = 1
00291
00292     ! IF (ITERZ .EQ. 30) BREAKLOOP=1
00293     IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LT. idemtoll) breakloop = 1
00294
00295     ENDDO
00296
00297     IF (iterz .EQ. 100) THEN
00298         CALL panic
00299         CALL errors("sp2pure_sparse", "SP2 purification is not converging: STOP!")
00300     ENDIF
00301
00302
00303     !shares data with all procs
00304     CALL dbcsr_replicate_all(matrix_a, error)
00305
00306     ! BO = ZERO
00307
00308
00309     DO j=1, dbcsr_nblkcols_total(matrix_a)
00310         DO i=1, dbcsr_nblkrows_total(matrix_a)
00311
00312             tr = .false.
00313
00314             CALL dbcsr_get_stored_coordinates (matrix_a, i, j, tr,
proc_holds_blk)

```

```

00315
00316         !call to put a block into the matrix
00317
00318         CALL dbcsr_get_block(matrix_a, i, j, block_2d, tr, found)
00319
00320         DO a = 1, blkksz
00321             DO b = 1, blkksz
00322
00323                 IF(found) THEN
00324                     bo_padded(blkksz*(i-1)+b,blkksz*(j-1)+a) = two*block_2d(b,a)
00325                 ENDIF
00326
00327             ENDDO
00328         ENDDO
00329
00330
00331         ENDDO
00332     ENDDO
00333     DEALLOCATE(block_2d)
00334
00335
00336     CALL dbcsr_release(matrix_a)
00337     CALL dbcsr_release(matrix_b)
00338
00339     DO i = 1, hdim
00340         DO j = 1, hdim
00341             bo(j,i) = bo_padded(j,i)
00342         ENDDO
00343     ENDDO
00344
00345
00346 #elif defined(DBCSR_OFF)
00347
00348
00349     ALLOCATE( vx(nnz), cx(nnz) )
00350
00351     count = 0
00352
00353     ! Put the normalized H into compressed sparse row format
00354
00355     IF (basistype .EQ. "ORTHO") THEN
00356
00357         DO i = 1, hdim
00358
00359             newrow = 0
00360
00361             DO j = 1, hdim
00362
00363                 IF ( abs( h(j,i) ) .GT. hthresh ) THEN
00364
00365                     count = count + 1
00366
00367                     IF (i .EQ. j) THEN
00368
00369                         vx( count ) = (maxeval - h(j,i))/maxminusmin
00370
00371                     ELSE
00372
00373                         vx( count ) = -h(j,i)/maxminusmin
00374
00375                     ENDIF
00376
00377                     cx( count ) = j
00378
00379                     IF ( newrow .EQ. 0 ) THEN
00380                         rx( i ) = count
00381                         newrow = 1
00382                     ENDIF
00383
00384                 ENDIF
00385
00386             ENDDO
00387         ENDDO
00388
00389     ELSE
00390
00391         DO i = 1, hdim
00392
00393             newrow = 0
00394
00395             DO j = 1, hdim
00396
00397                 IF ( abs( orthoh(j,i) ) .GT. hthresh ) THEN
00398
00399                     count = count + 1
00400
00401                     IF (i .EQ. j) THEN

```

```

00402
00403         vx( count ) = (maxeval - orthoh(j,i))/maxminusmin
00404
00405     ELSE
00406
00407         vx( count ) = -orthoh(j,i)/maxminusmin
00408
00409     ENDIF
00410
00411     cx( count ) = j
00412
00413     IF ( newrow .EQ. 0 ) THEN
00414         rx( i ) = count
00415         newrow = 1
00416     ENDIF
00417
00418     ENDIF
00419
00420     ENDDO
00421     ENDDO
00422
00423 ENDIF
00424
00425 rx(hdim + 1) = count + 1
00426
00427 iterz = 0
00428 breakloop = 0
00429
00430 prevnnzx2 = 0
00431 prevkeep = 0
00432
00433 DO WHILE ( breakloop .EQ. 0 .AND. iterz .LT. 100 )
00434
00435     iterz = iterz + 1
00436
00437     !
00438     ! Sparse BO * BO
00439     !
00440
00441     ! First we need to determine the number of non-zeros in X*X
00442     ! so we can allocate storage
00443
00444
00445     IF (iterz .LE. fillinstop) THEN
00446
00447         ip = 0
00448
00449         DO i = 1, hdim
00450
00451             xb = 0
00452
00453             DO jp = rx( i ), rx( i + 1 ) - 1
00454
00455                 j = cx( jp )
00456
00457                 DO kp = rx( j ), rx( j + 1 ) - 1
00458
00459                     k = cx( kp )
00460
00461                     IF ( xb( k ) .NE. i ) THEN
00462
00463                         ip = ip + 1
00464                         xb( k ) = i
00465
00466                     ENDIF
00467                 ENDDO
00468             ENDDO
00469         ENDDO
00470
00471         nnzx2 = ip
00472
00473     ENDIF
00474
00475     ! Allocate a bit more storage just to be safe
00476
00477     IF (iterz .EQ. fillinstop) nnzx2 = int(1.1*nnzx2)
00478
00479     !     print*, ITERZ, NNZX2, PREVNZX2, FLOAT(NNZX2)/FLOAT((HDIM*HDIM))
00480
00481     IF (iterz .EQ. 1) THEN
00482
00483         ALLOCATE( vxtmp( nnzx2 ), cxtmp( nnzx2 ))
00484
00485     ELSEIF (iterz .GT. 1 .AND. nnzx2 .GT. prevnnzx2) THEN
00486
00487         ! We will not enter for ITERZ > FILLINSTOP because
00488         ! of the condition PREVNZX2 = NNZX2

```

```

00489
00490     DEALLOCATE( vxtmp, cxtmp )
00491     ALLOCATE( vxtmp( nnzx2 ), cxtmp( nnzx2 ) )
00492
00493     prevnnzx2 = nnzx2
00494
00495     ENDIF
00496
00497     ip = 1
00498     xb = 0
00499
00500     tr2xx2 = zero
00501     trx2 = zero
00502
00503     DO i = 1, hdim
00504
00505         rxtmp(i) = ip
00506
00507         DO jp = rx( i ), rx( i + 1 ) - 1
00508
00509             j = cx( jp )
00510
00511             ! Computing these traces in this loop rather than separately
00512
00513             IF ( j .EQ. i ) tr2xx2 = tr2xx2 + vx(jp)
00514
00515             DO kp = rx( j ), rx( j + 1 ) - 1
00516
00517                 k = cx( kp )
00518
00519                 IF ( xb( k ) .NE. i ) THEN
00520
00521                     cxtmp( ip ) = k
00522                     ip = ip + 1
00523                     xb( k ) = i
00524                     work( k ) = vx( jp ) * vx( kp )
00525
00526                 ELSE
00527
00528                     work( k ) = work( k ) + vx( jp ) * vx( kp )
00529
00530                 ENDIF
00531
00532             ENDDO
00533
00534         ENDDO
00535
00536         DO j = rxtmp( i ), ip - 1
00537
00538             vxtmp( j ) = work( cxtmp( j ) )
00539
00540             ! same here
00541
00542             IF ( cxtmp(j) .EQ. i ) trx2 = trx2 + vxtmp( j )
00543
00544         ENDDO
00545
00546     ENDDO
00547
00548     rxtmp( hdim + 1 ) = ip
00549
00550     tr2xx2 = two*tr2xx2 - trx2
00551
00552     trxold = trx
00553
00554     limdiff = abs(trx2 - occ) - abs(tr2xx2 - occ)
00555
00556     IF ( limdiff .LT. -idemtol ) THEN
00557
00558         !
00559         ! X <= X^2
00560         !
00561
00562         trx = trx2
00563
00564     ELSEIF ( limdiff .GT. idemtol ) THEN
00565
00566         trx = tr2xx2
00567
00568         !
00569         ! X <= 2X - X^2
00570         !
00571
00572     DO i = 1, hdim
00573
00574         DO ip = rxtmp( i ), rxtmp( i + 1 ) - 1
00575

```

```

00576         work( cxtmp( ip ) ) = -vxtmp( ip )
00577
00578     ENDDO
00579
00580     DO ip = rx( i ), rx( i + 1 ) - 1
00581
00582         work( cx( ip ) ) = work( cx( ip ) ) + two*vx( ip )
00583
00584     ENDDO
00585
00586     DO ip = rxtmp( i ), rxtmp( i + 1 ) - 1
00587
00588         vxtmp( ip ) = work( cxtmp( ip ) )
00589
00590     ENDDO
00591
00592     ENDDO
00593
00594     ENDIF
00595
00596     IF (abs(limdiff) .GT. idemtol) THEN
00597
00598         IF (thresholdon .EQ. 1) THEN
00599
00600
00601             IF (iterz .LT. fillinstop) THEN
00602
00603                 keep = 0
00604                 DO i = 1, nnzx2
00605                     IF ( abs( vxtmp( i ) ) .GT. numthresh ) keep = keep + 1
00606                 ENDDO
00607
00608                 IF ( keep .GT. prevkeep ) THEN
00609                     DEALLOCATE( vx, cx )
00610                     ALLOCATE( vx( keep ), cx( keep ) )
00611                     prevkeep = keep
00612                 ENDIF
00613
00614             ELSEIF (iterz .EQ. fillinstop) THEN
00615
00616                 DEALLOCATE( vx, cx )
00617                 ALLOCATE( vx( nnzx2 ), cx( nnzx2 ) )
00618
00619             ENDIF
00620
00621             count = 0
00622             DO i = 1, hdim
00623                 newrow = 0
00624                 DO j = rxtmp( i ), rxtmp( i + 1 ) - 1
00625
00626                     ! Throwing away small elements while copying
00627
00628                     IF ( abs( vxtmp( j ) ) .GT. numthresh ) THEN
00629
00630                         count = count + 1
00631                         vx( count ) = vxtmp( j )
00632                         cx( count ) = cxtmp( j )
00633
00634                         IF ( newrow .EQ. 0 ) THEN
00635                             rx( i ) = count
00636                             newrow = 1
00637                         ENDIF
00638
00639                     ENDIF
00640                 ENDDO
00641             ENDDO
00642
00643             rx( hdim + 1 ) = count + 1
00644
00645         ELSE
00646
00647             IF (iterz .LE. fillinstop .AND. rx(hdim + 1) .NE. nnzx2 ) THEN
00648
00649                 DEALLOCATE( vx, cx )
00650                 ALLOCATE( vx(nnzx2), cx(nnzx2) )
00651
00652             ENDIF
00653
00654             vx = vxtmp
00655             cx = cxtmp
00656             rx = rxtmp
00657
00658         ENDIF
00659
00660     ENDIF
00661
00662     idemperr2 = idemperr1

```

```

00663      idemperr1 = idemperr
00664      idemperr = abs(trx - trxold)
00665
00666      !      PRINT*, IDEMPERR, ABS(TRX2 - OCC) - ABS(TR2XX2 - OCC)
00667      !      PRINT*, ITERZ, IDEMPERR0 - IDEMPERR2, ABS(TRX - OCC)
00668      !PRINT*, ITERZ, ABS(TRX - OCC)
00669
00670      IF (sp2conv .EQ. "REL" .AND. iterz .GE. minsp2iter &
00671          .AND. (idemperr .GE. idemperr2) ) breakloop = 1
00672
00673      IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LE. idemtol) breakloop = 1
00674
00675      ENDDO
00676
00677      IF (iterz .EQ. 100) THEN
00678          CALL panic
00679          CALL errors("sp2pure_sparse", "Sparse SP2 purification is not converging: STOP!")
00680      ENDIF
00681
00682      bo = zero
00683
00684      DO i = 1, hdim
00685          DO j = rx( i ), rx( i + 1 ) - 1
00686
00687              bo( i, cx(j) ) = two*vx( j )
00688
00689          ENDDO
00690      ENDDO
00691
00692      DEALLOCATE(vx, vxtmp, cx, cxtmp)
00693
00694      #endif
00695
00696      RETURN
00697
00698  END SUBROUTINE sp2pure_sparse

```

8.395 sp2pure_sparse_parallel.f90 File Reference

Functions/Subroutines

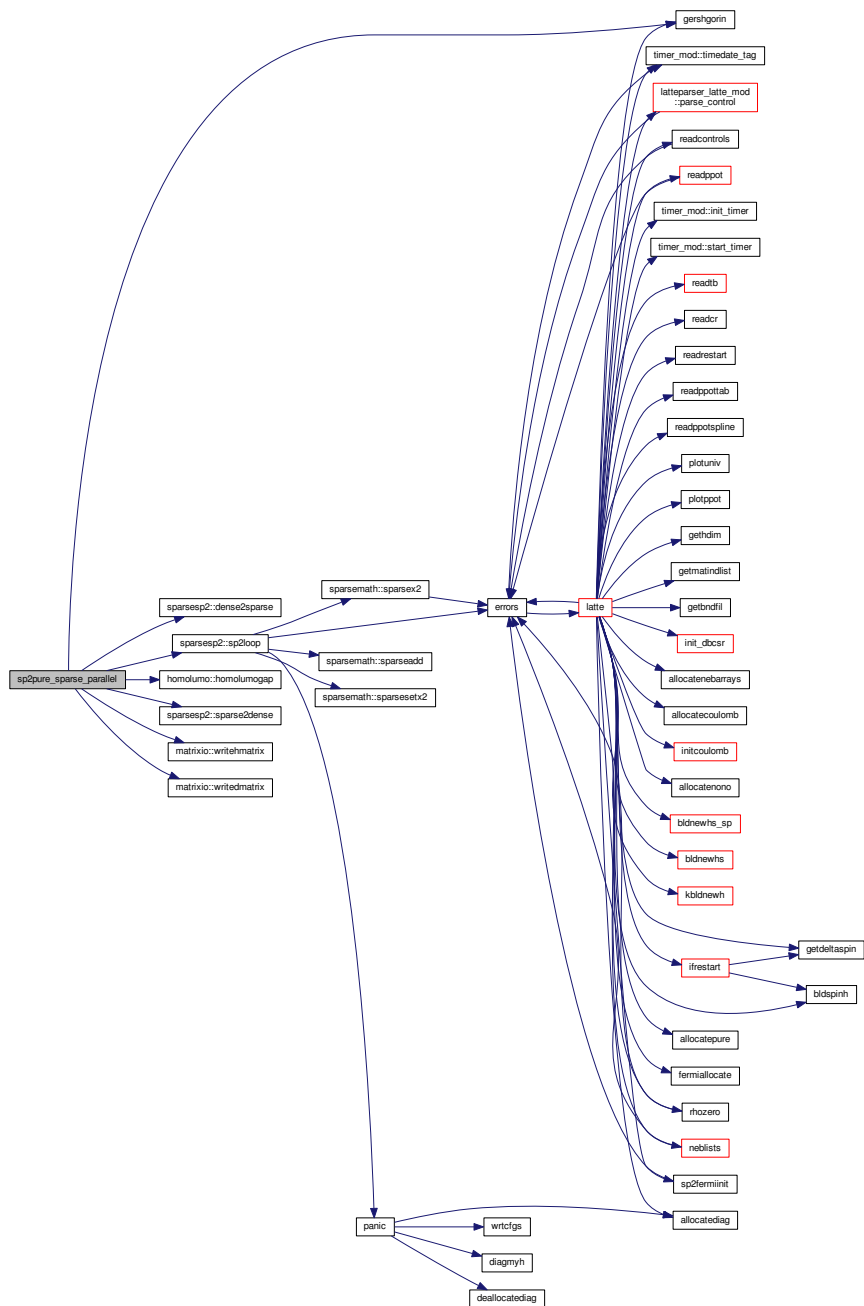
- subroutine [sp2pure_sparse_parallel](#) (MDITER)

8.395.1 Function/Subroutine Documentation

8.395.1.1 subroutine sp2pure_sparse_parallel (integer MDITER)

Definition at line 23 of file [sp2pure_sparse_parallel.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2pure_sparse_parallel(MDITER)
00023
00024 !
00025 ! PARALLEL SHARED MEMORY OPENMP VERSION
00026 ! This subroutine implements Niklasson's SP2 density matrix purification
00027 ! algorithm.
00028 !
00029
00030 USE constants_mod
00031 USE timer_mod
00032 USE setuparray
00033 USE purearray
00034 USE sparsearray
00035 USE nonoarray
00036 USE myprecision
00037 USE matrixio
00038 USE sparsesp2
00039 USE homolumo
00040
00041 IMPLICIT NONE
00042
00043 INTEGER :: M, ITERZ, MDITER
00044 INTEGER :: NNZSTART, NNZEND
00045 INTEGER, ALLOCATABLE :: IIA(:), JJA(:, :)
00046
00047 REAL(LATTEPREC), ALLOCATABLE :: AAN(:, :)
00048 IF (existerror) RETURN
00049
00050 ! Calculate M - max number of non-zeroes per row
00051 m = nnzstart(msparse, hdim)
00052
00053 ! build the sparse H matrix and compute the bounds using
00054 ! Gershgorin circles
00055 CALL gershgorin
00056
00057 ! Allocate for parallelism
00058 ALLOCATE(iaa(hdim))
00059 ALLOCATE(jja(m, hdim))
00060 ALLOCATE(aan(m, hdim))
00061
00062 ! Put the normalized H into compressed sparse row format
00063 ! TX = START_TIMER(SP2ALL_TIMER)
00064 ! TX = START_TIMER(SP2SPARSE_TIMER)
00065
00066 ! Convert dense H matrix to sparse
00067 ! TX = START_TIMER(DENSE2SPARSE_TIMER)
00068 CALL dense2sparse(h, hdim, iia, jja, aan)
00069 ! TX = STOP_TIMER(DENSE2SPARSE_TIMER)
00070
00071 ! Do SP2 algorithm
00072 ! TX = START_TIMER(DMBUILD_TIMER)
00073 CALL sp2loop(m, iterz, iia, jja, aan)
00074 ! TX = STOP_TIMER(DMBUILD_TIMER)
00075
00076 ! Calculate HOMO-LUMO gap estimate
00077 CALL homolumogap(iterz)
00078
00079 ! TX = STOP_TIMER(SP2SPARSE_TIMER)
00080
00081 ! Convert sparse to dense
00082 ! TX = START_TIMER(SPARSE2DENSE_TIMER)
00083 CALL sparse2dense(two, iia, jja, aan, bo, hdim)
00084 ! TX = STOP_TIMER(SPARSE2DENSE_TIMER)
00085
00086 ! TX = STOP_TIMER(SP2ALL_TIMER)
00087
00088 ! Reset number of non-zeroes
00089 msparse = nnzend(m, hdim)
00090
00091 ! Write out input and output matrix
00092 IF (debugon .GE. 2) THEN
00093   CALL writehmatrix(hdim, msparse, h, nr_sp2_iter,

```

```

pp)
00094   ENDIF
00095   IF (debugon .EQ. 3) THEN
00096     CALL writedmatrix(hdim, bo)
00097   ENDIF
00098
00099   ! Deallocate for sparse format
00100   DEALLOCATE(iaa)
00101   DEALLOCATE(jja)
00102   DEALLOCATE(aan)
00103
00104   RETURN
00105
00106 END SUBROUTINE sp2pure_sparse_parallel

```

8.397 sp2pure_sparse_parallel_simple.f90 File Reference

Functions/Subroutines

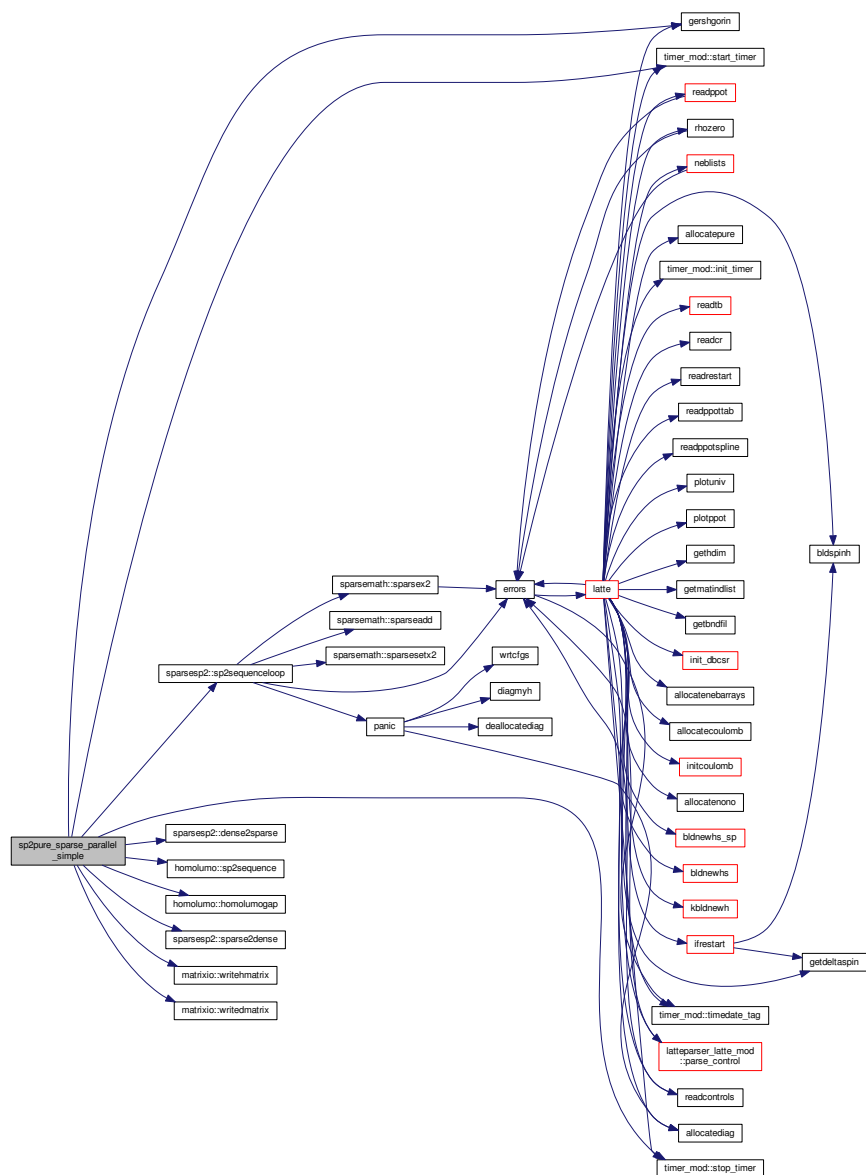
- subroutine [sp2pure_sparse_parallel_simple](#) (MDITER)

8.397.1 Function/Subroutine Documentation

8.397.1.1 subroutine [sp2pure_sparse_parallel_simple](#) (integer *MDITER*)

Definition at line 23 of file [sp2pure_sparse_parallel_simple.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2pure_sparse_parallel_simple(MDITER)
00023
00024 !
00025 ! PARALLEL SHARED MEMORY OPENMP VERSION
00026 ! This subroutine implements Niklasson's SP2 density matrix purification
00027 ! algorithm.
00028 !
00029
00030 USE constants_mod
00031 USE timer_mod
00032 USE setuparray
00033 USE purearray
00034 USE sparsearray
00035 USE nonoarray
00036 USE myprecision
00037 USE matrixio
00038 USE sparsesp2
00039 USE homolumo
00040
00041 IMPLICIT NONE
00042
00043 INTEGER :: M, ITERZ, MDITER
00044 INTEGER :: NNZSTART, NNZEND
00045 INTEGER, ALLOCATABLE :: IIA(:), JJA(:, :)
00046
00047 REAL(LATTEPREC), ALLOCATABLE :: AAN(:, :)
00048 IF (existerror) RETURN
00049
00050 ! Calculate M - max number of non-zeroes per row
00051 m = nnzstart(msparse, hdim)
00052
00053 ! build the sparse H matrix and compute the bounds using
00054 ! Gershgorin circles
00055 CALL gershgorin
00056
00057 ! Allocate for parallelism
00058 ALLOCATE(iaa(hdim))
00059 ALLOCATE(jja(m, hdim))
00060 ALLOCATE(aan(m, hdim))
00061
00062 ! Put the normalized H into compressed sparse row format
00063 tx = start_timer(sp2all_timer)
00064 tx = start_timer(sp2sparse_timer)
00065
00066 ! Convert dense H matrix to sparse
00067 tx = start_timer(dense2sparse_timer)
00068 CALL dense2sparse(h, hdim, iia, jja, aan)
00069 tx = stop_timer(dense2sparse_timer)
00070
00071 ! Calculate SP2 sequence of X^2 and 2X-X^2
00072 ! based on HOMO-LUMO gap
00073 CALL sp2sequence
00074
00075 ! Do SP2 algorithm using calculated sequence
00076 tx = start_timer(dmbuild_timer)
00077 CALL sp2sequenceloop(m, iterz, iia, jja, aan)
00078 tx = stop_timer(dmbuild_timer)
00079
00080 ! Calculate HOMO-LUMO gap estimate
00081 CALL homolumogap(iterz)
00082
00083 tx = stop_timer(sp2sparse_timer)
00084
00085 ! Convert sparse to dense
00086 tx = start_timer(sparse2dense_timer)
00087 CALL sparse2dense(two, iia, jja, aan, bo, hdim)
00088 tx = stop_timer(sparse2dense_timer)
00089
00090 tx = stop_timer(sp2all_timer)
00091
00092 ! Reset number of non-zeroes
00093 msparse = nnzend(m, hdim)

```

```

00094
00095 ! Write out input and output matrix
00096 IF (debugon .GE. 2) THEN
00097     CALL writehmatrix(hdim, msparse, h, nr_sp2_iter,
00098     pp)
00098 ENDIF
00099 IF (debugon .EQ. 3) THEN
00100     CALL writedmatrix(hdim, bo)
00101 ENDIF
00102
00103 ! Deallocate for sparse format
00104 DEALLOCATE(iia)
00105 DEALLOCATE(jja)
00106 DEALLOCATE(aan)
00107
00108 RETURN
00109
00110 END SUBROUTINE sp2pure_sparse_parallel_simple

```

8.399 sp2pure_subgraph_parallel.f90 File Reference

Functions/Subroutines

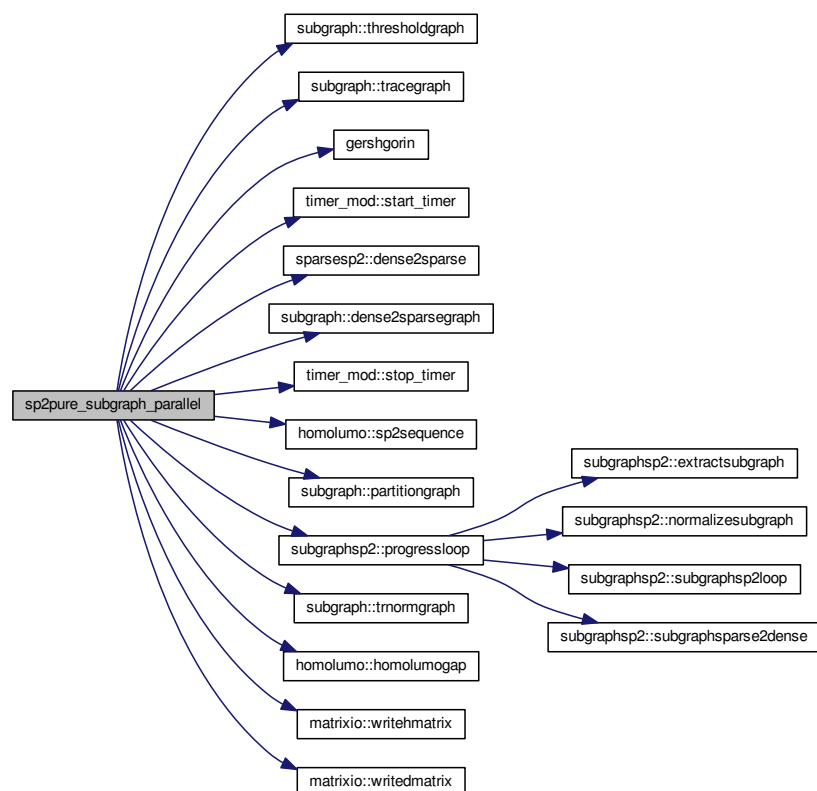
- subroutine [sp2pure_subgraph_parallel](#) (MDITER)

8.399.1 Function/Subroutine Documentation

8.399.1.1 subroutine [sp2pure_subgraph_parallel](#) (integer, intent(in) *MDITER*)

Definition at line 23 of file [sp2pure_subgraph_parallel.f90](#).

Here is the call graph for this function:



[illegible]

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National   !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE    !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2pure_subgraph_parallel(MDITER)
00023
00024 !
00025 ! PARALLEL SHARED MEMORY OPENMP VERSION
00026 ! This subroutine implements Niklasson's SP2 density matrix purification
00027 ! algorithm.
00028 !
00029
00030 USE constants_mod
00031 USE timer_mod
00032 USE setuparray
00033 USE purearray
00034 USE sparsearray
00035 USE nonoarray
00036 USE myprecision
00037 USE subgraph
00038 USE sparsesp2
00039 USE subgraphsp2
00040 USE xboarray
00041 USE homolumo
00042 USE matrixio
00043
00044 IMPLICIT NONE
00045
00046 INTEGER, INTENT(IN) :: MDITER
00047 INTEGER :: M, ITERZ
00048 INTEGER, ALLOCATABLE :: IIG(:), JJG(:,,:), IIH(:), JJH(:,,:)
00049 INTEGER :: NNZSTART, NNZEND
00050
00051 REAL(LATTEPREC) :: TRACE
00052 REAL(LATTEPREC), ALLOCATABLE :: GGN(:,,:), HHN(:,,:)
00053 IF (existerror) RETURN
00054
00055 ! First time through use current Hamiltonian and threshold
00056 IF (first_step.EQ.1) THEN
00057   WRITE(*,*) ' '
00058   WRITE(*,*) '# WARNING PARTITIONED SUBGRAPH TECHNIQUE NOT YET EVALUATED, EXPERIMENTAL VERISON !!!!! '
00059   WRITE(*,*) ' '
00060
00061   WRITE(*,*) '# SUBGRAPH FIRST '
00062   g = bo
00063   CALL thresholdgraph
00064   trace = tracegraph(hdim)
00065   !write(*,*) "graph trace = ", TRACE
00066   first_step = 0
00067 ENDIF
00068
00069 ! Calculate M
00070 m = nnzstart(msparse, hdim)
00071
00072 ! build the sparse H matrix and compute the bounds using
00073 ! Gershgorin circles
00074 CALL gershgorin
00075
00076 ! Allocate for parallelism
00077 ALLOCATE(iig(hdim), iih(hdim))
00078 ALLOCATE(jjg(m, hdim), jjh(m, hdim))
00079 ALLOCATE(ggn(m, hdim), hhn(m, hdim))
00080
00081 ! Put the normalized H into compressed sparse row format
00082 tx = start_timer(sp2all_timer)
00083 tx = start_timer(sp2sparse_timer)
00084
00085 ! Convert dense H matrix to sparse
00086 tx = start_timer(dense2sparse_timer)
00087 CALL dense2sparse(h, hdim, iih, jjh, hhn)
00088 CALL dense2sparsegraph(iig, jjg, ggn)
00089 tx = stop_timer(dense2sparse_timer)
00090
00091 ! Start timer for density matrix build
00092 tx = start_timer(dmbuild_timer)
00093

```

```

00094      ! Calculate SP2 sequence of X^2 and 2X-X^2
00095      CALL sp2sequence
00096
00097      ! Partition nodes into subgraphs
00098      CALL partitiongraph
00099
00100      ! Do progress loop
00101      CALL progressloop(iih, jjh, hhn, iig, jjg)
00102
00103      ! Density matrix becomes new graph for next iteration
00104      ! Threshold graph
00105      g = bo
00106      CALL thresholdgraph
00107
00108      ! Calculate norm for each iteration
00109      CALL trnormgraph(iterz)
00110
00111      ! End of density matrix build
00112      tx = stop_timer(dmbuild_timer)
00113
00114      ! Calculate HOMO-LUMO gap estimate
00115      CALL homolumogap(iterz)
00116
00117      tx = stop_timer(sp2sparse_timer)
00118      tx = stop_timer(sp2all_timer)
00119
00120      ! Reset number of non-zeroes
00121      msparse = nnzend(m, hdim)
00122
00123      ! Write out input and output matrix
00124      IF (debugon .GE. 2) THEN
00125          CALL writehmatrix(hdim, msparse, h, nr_sp2_iter,
00126                          pp)
00127      ENDIF
00128      IF (debugon .EQ. 3) THEN
00129          CALL writedmatrix(hdim, bo)
00130      ENDIF
00131
00132      ! Deallocate for sparse format
00133      DEALLOCATE(iig, iih)
00134      DEALLOCATE(jjg, jjh)
00135      DEALLOCATE(hhn, ggn)
00136
00137      DEALLOCATE(nr_of_nodes_in_part)
00138      DEALLOCATE(node_in_part)
00139      DEALLOCATE(vvx)
00140
00141      RETURN
00142  END SUBROUTINE sp2pure_subgraph_parallel

```

8.401 sp2T.f90 File Reference

Functions/Subroutines

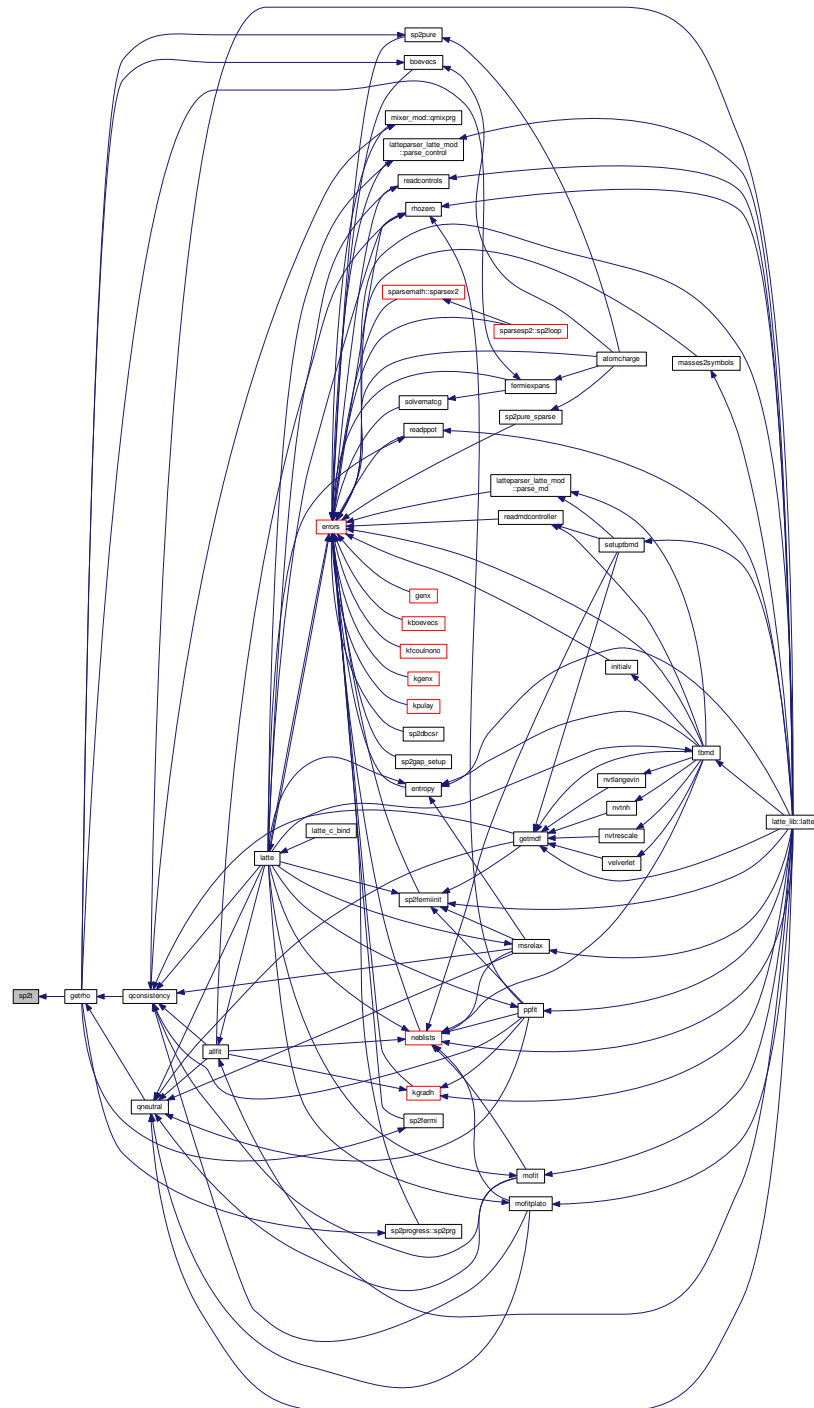
- subroutine [sp2t](#)

8.401.1 Function/Subroutine Documentation

8.401.1.1 subroutine [sp2t](#)()

Definition at line 23 of file [sp2T.f90](#).

Here is the caller graph for this function:



8.402 sp2T.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE sp2t
00023
00024 !
00025 ! This subroutine implements the finite temperature version of
00026 ! Niklasson's SP2 purification scheme. It will give us something that
00027 ! looks very similar to a Fermi-Dirac distribution (although it's
00028 ! not exactly the FD distribution).
00029 !
00030
00031 USE constants_mod
00032 USE setuparray
00033 USE purearray
00034 USE spinarray
00035 USE myprecision
00036
00037 IMPLICIT NONE
00038
00039 INTEGER :: I, J, ITER, II
00040 REAL(LATTEPREC) :: TRX2, TRX
00041 REAL(LATTEPREC) :: OCC, LAMBDA
00042 IF (existerror) RETURN
00043
00044 occ = bndfil*float(hdim)
00045
00046 IF (spinon .EQ. 0) THEN
00047
00048 !
00049 ! Rescale H such that eigenvalues are in the range [0:1] based
00050 ! on Gershgorin circles
00051 !
00052
00053 DO i = 1, hdim
00054   DO j = i, hdim
00055
00056     IF (i .EQ. j) THEN
00057
00058       bo(j,i) = (maxeval - h(j,i))/maxminusmin
00059
00060     ELSEIF (i .NE. j) THEN
00061
00062       bo(j,i) = (zero - h(j,i))/maxminusmin
00063       bo(i,j) = bo(j,i)
00064
00065     ENDIF
00066
00067   ENDDO
00068 ENDDO
00069
00070 iter = 0
00071
00072 DO ii = 1, norecs
00073
00074   iter = iter + 1
00075
00076   !
00077   ! Calculating X*X
00078   !
00079
00080 #ifdef DOUBLEPREC
00081   CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00082     bo, hdim, bo, hdim, 0.0d0, x2, hdim)
00083 #elif defined(SINGLEPREC)
00084   CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00085     bo, hdim, bo, hdim, 0.0, x2, hdim)
00086 #endif
00087
00088   trx2 = zero
00089   trx = zero
00090
00091   DO i = 1, hdim
00092
00093     trx2 = trx2 + x2(i,i)

```

```

00094         trx = trx + bo(i,i)
00095
00096     ENDDO
00097
00098     IF (abs(trx2 - occ) .LT. abs(two*trx - trx2 - occ)) THEN
00099
00100         bo = x2
00101
00102     ELSE
00103
00104         bo = two*bo - x2
00105
00106     ENDIF
00107
00108     ENDDO
00109
00110     !
00111     ! Now apply a shift such that we get the correct occupation after
00112     ! only NOREC applications of the purification algorithm
00113     !
00114
00115 #ifdef DOUBLEPREC
00116     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00117         bo, hdim, bo, hdim, 0.0d0, x2, hdim)
00118 #elif defined(SINGLEPREC)
00119     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00120         bo, hdim, bo, hdim, 0.0, x2, hdim)
00121 #endif
00122
00123     trx = zero
00124     trx2 = zero
00125
00126     DO i = 1, hdim
00127
00128         trx = trx + bo(i,i)
00129         trx2 = trx2 + x2(i,i)
00130
00131     ENDDO
00132
00133     lambda = (occ - trx)/(trx - trx2)
00134
00135     !
00136     ! Here we also double the density matrix to get the
00137     ! bond-order in non-spin polarized calculations
00138     !
00139
00140     bo = two*( bo + lambda*( bo - x2 ))
00141
00142     ELSE
00143
00144     !
00145     ! Remapping eigenvalues of H
00146     !
00147
00148     DO i = 1, hdim
00149         DO j = i, hdim
00150
00151             IF (i .EQ. j) THEN
00152
00153                 rhoup(j,i) = (maxeval - hup(j,i))/maxminusmin
00154
00155                 rhodown(j,i) = (maxeval - hdown(j,i)) / &
00156                     maxminusmin
00157
00158             ELSEIF (i .NE. j) THEN
00159
00160                 rhoup(j,i) = (zero - hup(j,i))/maxminusmin
00161                 rhoup(i,j) = rhoup(j,i)
00162
00163                 rhodown(j,i) = (zero - hdown(j,i))/maxminusmin
00164                 rhodown(i,j) = rhodown(j,i)
00165
00166             ENDIF
00167
00168         ENDDO
00169     ENDDO
00170
00171     iter = 0
00172
00173     DO ii = 1, norecs
00174
00175         iter = iter + 1
00176
00177         !
00178         ! Calculating Xup*Xup and Xdown*Xdown
00179         !
00180

```

```

00181 #ifdef DOUBLEPREC
00182
00183     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00184             rhoup, hdim, rhoup, hdim, 0.0d0, x2up, hdim)
00185     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00186             rhodown, hdim, rhodown, hdim, 0.0d0, x2down,
00187             hdim)
00188 #elif defined(SINGLEPREC)
00189
00190     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00191             rhoup, hdim, rhoup, hdim, 0.0, x2up, hdim)
00192     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00193             rhodown, hdim, rhodown, hdim, 0.0, x2down,
00194             hdim)
00195 #endif
00196
00197     trx = zero
00198     trx2 = zero
00199
00200     DO i = 1, hdim
00201
00202         trx = trx + rhoup(i,i) + rhodown(i,i)
00203         trx2 = trx2 + x2up(i,i) + x2down(i,i)
00204
00205     ENDDO
00206
00207     IF (abs(trx2 - totne) .LT. abs(two*trx - trx2 - totne)) THEN
00208
00209         rhoup = x2up
00210         rhodown = x2down
00211
00212     ELSE
00213
00214         rhoup = two*rhoup - x2up
00215         rhodown = two*rhodown - x2down
00216
00217     ENDIF
00218
00219     ENDDO
00220
00221     !
00222     ! Apply a shift such that we get the correct trace after NORECS
00223     ! SP2 recursions
00224     !
00225
00226 #ifdef DOUBLEPREC
00227
00228     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00229             rhoup, hdim, rhoup, hdim, 0.0d0, x2up, hdim)
00230     CALL dgemm('N', 'N', hdim, hdim, hdim, 1.0d0, &
00231             rhodown, hdim, rhodown, hdim, 0.0d0, x2down,
00232             hdim)
00233 #elif defined(SINGLEPREC)
00234
00235     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00236             rhoup, hdim, rhoup, hdim, 0.0, x2up, hdim)
00237     CALL sgemm('N', 'N', hdim, hdim, hdim, 1.0, &
00238             rhodown, hdim, rhodown, hdim, 0.0, x2down,
00239             hdim)
00240 #endif
00241
00242     trx = zero
00243     trx2 = zero
00244
00245     DO i = 1, hdim
00246
00247         trx = trx + rhoup(i,i) + rhodown(i,i)
00248         trx2 = trx2 + x2up(i,i) + x2down(i,i)
00249
00250     ENDDO
00251
00252     IF (abs(totne - trx) .GT. 1.0d-8) THEN
00253
00254         lambda = (totne - trx)/(trx - trx2)
00255
00256         ! print*, "TOTNE - TRX = ", TOTNE - TRX
00257         ! print*, "TRX - TRX2 = ", TRX - TRX2
00258
00259         rhoup = rhoup + lambda*(rhoup - x2up)
00260         rhodown = rhodown + lambda*(rhodown - x2down)
00261
00262     ENDIF
00263

```

```

00264      ENDIF
00265
00266      RETURN
00267
00268 END SUBROUTINE sp2t

```

8.403 sparsearray.f90 File Reference

Modules

- module [sparsearray](#)

Variables

- real(latteprec), dimension(:,:), allocatable [sparsearray::bo_padded](#)
- integer [sparsearray::fillinstop](#)
- real(latteprec), parameter [sparsearray::hthresh](#) = 1.0D-14
- integer [sparsearray::msparse](#)
- integer [sparsearray::nnz](#)
- integer [sparsearray::nnz_max](#)
- integer [sparsearray::nnz_pad](#) = 3
- real(latteprec) [sparsearray::numthresh](#)
- integer, dimension(:), allocatable [sparsearray::rx](#)
- integer, dimension(:), allocatable [sparsearray::rxtmp](#)
- integer [sparsearray::thresholdon](#)
- real(latteprec), dimension(:), allocatable [sparsearray::work](#)
- integer, dimension(:), allocatable [sparsearray::xb](#)

8.404 sparsearray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it       !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it     !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be     !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of   !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE sparsearray
00023
00024     USE myprecision
00025
00026     IMPLICIT NONE
00027     SAVE
00028
00029     INTEGER :: thresholdon, fillinstop, nnz, nnz_pad = 3,
nnz_max, msparse
00030     ! INTEGER :: NR_SP2_ITER
00031     ! INTEGER, ALLOCATABLE :: RX(:), RXTMP(:), XB(:), PP(:) ! Used for sparse sp2
00032     INTEGER, ALLOCATABLE :: rx(:), rxtmp(:), xb(:)
00033     ! REAL(LATTEPREC) :: NUMTHRESH, EHOMO, ELUMO
00034     REAL(LATTEPREC) :: numthresh
00035     ! REAL(LATTEPREC), ALLOCATABLE :: WORK(:), VV(:) ! Used for sparse sp2
00036     REAL(LATTEPREC), ALLOCATABLE :: work(:)
00037     REAL(LATTEPREC), PARAMETER :: hthresh = 1.0d-14
00038     REAL(LATTEPREC), ALLOCATABLE :: bo_padded(:,:)
00039
00040 END MODULE sparsearray

```


8.405 sparsemath.f90 File Reference

Modules

- module [sparsemath](#)

Functions/Subroutines

- subroutine [sparsemath::sparseadd](#) (TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
- subroutine [sparsemath::sparsesetx2](#) (TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
- subroutine [sparsemath::sparsex2](#) (TTRX, TTRX2, II, JJ, VAL, II2, JJ2, VAL2)

Variables

- integer, dimension(:), allocatable [sparsemath::ix](#)
- integer, dimension(:,:), allocatable [sparsemath::jjb](#)
- real(latteprec), dimension(:), allocatable [sparsemath::x](#)
- real(latteprec), dimension(:), allocatable [sparsemath::y](#)

8.406 sparsemath.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE sparsemath
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER, ALLOCATABLE :: ix(:), jjb(:, :)
00030
00031   REAL(LATTEPREC), ALLOCATABLE :: x(:), y(:)
00032
00033 CONTAINS
00034
00035   !
00036   ! sparsex2 - Sparse matrix multiply  X^2
00037   !
00038   SUBROUTINE sparsex2(TTRX, TTRX2, II, JJ, VAL, II2, JJ2, VAL2)
00039
00040     USE myprecision
00041     USE constants_mod
00042     USE sparsearray
00043     USE omp_lib
00044
00045     IMPLICIT NONE
00046     INTEGER, INTENT(IN) :: II(:), JJ(:, :)
```

```

00047     INTEGER, INTENT(INOUT) :: II2(:), JJ2(:, :)
00048     INTEGER :: I, J, K, L, LL, JP, KP
00049
00050     REAL(LATTEPREC) :: AX, XTMP
00051     REAL(LATTEPREC), INTENT(INOUT) :: TTRX, TTRX2
00052     REAL(LATTEPREC), INTENT(IN) :: VAL(:, :)
00053     REAL(LATTEPREC), INTENT(INOUT) :: VAL2(:, :)
00054
00055     x = zero
00056     y = zero
00057     ix = 0
00058     jjb = 0
00059
00060     !$OMP PARALLEL DO FIRSTPRIVATE(IX,X) PRIVATE(I,J,L,LL,XTMP,AX,K,KP,JP) &
00061     !$OMP REDUCTION(+:TTRX,TTRX2)
00062     DO i = 1, hdim
00063         l = 0
00064         DO jp = 1, ii(i)
00065             ax = val(jp,i)
00066             j = jj(jp,i)
00067             IF (j .EQ. i) THEN
00068                 ttrx = ttrx + ax
00069             ENDIF
00070             DO kp = 1, ii(j)
00071                 k = jj(kp,j)
00072                 IF (ix(k) .EQ. 0) THEN
00073                     l = l + 1
00074                     jj2(l,i) = k
00075                     ix(k) = i
00076                     x(k) = zero
00077                 ENDIF
00078                 x(k) = x(k) + ax*val(kp,j) ! TEMPORARY STORAGE VECTOR LENGTH FULL N
00079             ENDDO
00080         ENDDO
00081
00082         ! Check if over number of non-zeroes limit
00083         IF (l > msparse) THEN
00084             WRITE(6,*) "ERROR: Number of non-zeroes per row =", l, "> MSPARSE, Increase MSPARSE."
00085             CALL errors("sparsemath", "Number of non-zeroes per row > MSPARSE")
00086         ENDIF
00087
00088         ll = 0
00089         DO j = 1, l
00090             jp = jj2(j,i)
00091             xtmp = x(jp)
00092             IF (jp .EQ. i) THEN
00093                 ttrx2 = ttrx2 + xtmp
00094                 ll = ll + 1
00095                 val2(ll,i) = xtmp
00096                 jj2(ll,i) = jp
00097             ELSEIF (abs(xtmp) .GT. numthresh) THEN
00098                 ll = ll + 1
00099                 val2(ll,i) = xtmp
00100                 jj2(ll,i) = jp
00101             ENDIF
00102             ix(jp) = 0
00103             x(jp) = zero
00104         ENDDO
00105         ii2(i) = ll
00106     ENDDO
00107     !$OMP END PARALLEL DO
00108
00109 END SUBROUTINE sparsex2
00110
00111 !
00112 ! sparseadd - Sparse matrix add X = 2X - X^2
00113 !
00114 SUBROUTINE sparseadd(TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
00115
00116     USE myprecision
00117     USE constants_mod
00118     USE sparsearray
00119     USE omp_lib
00120
00121     IMPLICIT NONE
00122     INTEGER :: I, J, K, L, LL, JP, KP
00123     INTEGER, INTENT(IN) :: II2(:), JJ2(:, :)
00124     INTEGER, INTENT(INOUT) :: II(:), JJ(:, :)
00125
00126     REAL(LATTEPREC) :: XTMP
00127     REAL(LATTEPREC), INTENT(INOUT) :: TTRNORM
00128     REAL(LATTEPREC), INTENT(IN) :: VAL2(:, :)
00129     REAL(LATTEPREC), INTENT(INOUT) :: VAL(:, :)
00130
00131     x = zero
00132     y = zero
00133     ix = 0

```

```

00134     jjb = 0
00135
00136     ttrnorm = zero
00137
00138     !$OMP PARALLEL DO FIRSTPRIVATE(IX,X,Y) PRIVATE(I,L,LL,XTMP,K,JP) &
00139     !$OMP REDUCTION(+:TTRNORM)
00140
00141     DO i = 1, hdim
00142         l = 0
00143         DO jp = 1, ii(i)
00144             k = jj(jp,i)
00145             IF (ix(k) .EQ. 0) THEN
00146                 x(k) = zero
00147                 ix(k) = i
00148                 y(k) = zero
00149                 l = l + 1
00150                 jjb(l,i) = k
00151             ENDIF
00152             x(k) = x(k) + two*val(jp,i)
00153             y(k) = y(k) + val(jp,i)
00154         ENDDO
00155
00156         DO jp = 1, ii2(i)
00157             k = jj2(jp,i)
00158             IF (ix(k) .EQ. 0) THEN
00159                 x(k) = zero
00160                 ix(k) = i
00161                 y(k) = zero
00162                 l = l + 1
00163                 jjb(l,i) = k
00164             ENDIF
00165             x(k) = x(k) - val2(jp,i)
00166             y(k) = y(k) - val2(jp,i)
00167         ENDDO
00168
00169         ii(i) = l
00170         ll = 0
00171         DO jp = 1, l
00172             xtmp = x(jjb(jp,i))
00173             ttrnorm = ttrnorm + y(jjb(jp,i))*y(jjb(jp,i))
00174             IF (abs(xtmp) .GT. numthresh) THEN ! THIS THRESHOLDING COULD BE IGNORED!?
00175                 ll = ll + 1
00176                 val(ll,i) = xtmp
00177                 jj(ll,i) = jjb(jp,i)
00178             ENDIF
00179             x(jjb(jp,i)) = zero
00180             ix(jjb(jp,i)) = 0
00181             y(jjb(jp,i)) = zero
00182             jjb(jp,i) = 0
00183         ENDDO
00184         ii(i) = ll
00185     ENDDO
00186     !$OMP END PARALLEL DO
00187
00188     END SUBROUTINE sparseadd
00189
00190     !
00191     ! sparsesetx2 - Sparse matrix set to X^2
00192     !
00193     SUBROUTINE sparsesetx2(TTRNORM, II, JJ, VAL, II2, JJ2, VAL2)
00194
00195     USE myprecision
00196     USE constants_mod
00197     USE sparsearray
00198     USE omp_lib
00199
00200     IMPLICIT NONE
00201     INTEGER :: I, J, K, L, LL, JP, KP
00202     INTEGER, INTENT(INOUT) :: II(:), JJ(:, :)
00203     INTEGER, INTENT(IN) :: II2(:), JJ2(:, :)
00204
00205     REAL(LATTEPREC) :: XTMP
00206     REAL(LATTEPREC), INTENT(INOUT) :: TTRNORM
00207     REAL(LATTEPREC), INTENT(IN) :: VAL2(:, :)
00208     REAL(LATTEPREC), INTENT(INOUT) :: VAL(:, :)
00209
00210     x = zero
00211     y = zero
00212     ix = 0
00213     jjb = 0
00214
00215     ttrnorm = zero
00216
00217     !$OMP PARALLEL DO FIRSTPRIVATE(IX,X,Y) PRIVATE(I,L,LL,K,XTMP,JP) &
00218     !$OMP REDUCTION(+:TTRNORM)
00219     DO i = 1, hdim ! X = X^2
00220

```

```

00221      l = 0
00222      DO jp = 1, ii(i)
00223          k = jj(jp,i)
00224          IF (ix(k) .EQ. 0) THEN
00225              x(k) = zero
00226              ix(k) = i
00227              y(k) = zero
00228              l = l + 1
00229              jjb(l,i) = k
00230          ENDIF
00231          y(k) = y(k) + val(jp,i)
00232      ENDDO
00233
00234      DO jp = 1, ii2(i)
00235          k = jj2(jp,i)
00236          IF (ix(k) .EQ. 0) THEN
00237              x(k) = zero
00238              ix(k) = i
00239              y(k) = zero
00240              l = l + 1
00241              jjb(l,i) = k
00242          ENDIF
00243          x(k) = x(k) + val2(jp,i)
00244          y(k) = y(k) - val2(jp,i)
00245      ENDDO
00246
00247      ll = 0
00248      DO jp = 1, l
00249          xtmp = x(jjb(jp,i))
00250          ttrnorm = ttrnorm + y(jjb(jp,i))*y(jjb(jp,i))
00251          IF (abs(xtmp) .GT. numthresh) THEN ! THIS THRESHOLDING COULD BE IGNORED!?
00252              ll = ll + 1
00253              val(ll,i) = xtmp
00254              jj(ll,i) = jjb(jp,i)
00255          ENDIF
00256          x(jjb(jp,i)) = zero
00257          ix(jjb(jp,i)) = 0
00258          y(jjb(jp,i)) = zero
00259          jjb(jp,i) = 0
00260      ENDDO
00261      ii(i) = ll
00262  ENDDO
00263  !$OMP END PARALLEL DO
00264
00265  END SUBROUTINE sparsesetx2
00266
00267  END MODULE sparsemath

```

8.407 sparsesp2.f90 File Reference

Modules

- module [sparsesp2](#)

Functions/Subroutines

- subroutine [sparsesp2::dense2sparse](#) (HARRAY, HSIZE, II, JJ, VAL)
- subroutine [sparsesp2::sp2loop](#) (MSIZE, ITER, II, JJ, VAL)
- subroutine [sparsesp2::sp2sequenceloop](#) (MSIZE, ITER, II, JJ, VAL)
- subroutine [sparsesp2::sparse2dense](#) (SCALAR, II, JJ, VAL, DARRAY, HSIZE)

8.408 sparsesp2.f90

```

00001  !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002  ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003  ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004  ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005  ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006  ! rights to use, reproduce, and distribute this software.  NEITHER THE  !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.     !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE sparsesp2
00023
00024     IMPLICIT NONE
00025
00026 CONTAINS
00027
00028     SUBROUTINE dense2sparse(HARRAY, HSIZE, II, JJ, VAL)
00029
00030         USE myprecision
00031         USE constants_mod
00032         USE sparsearray
00033         USE purearray
00034         USE nonoarray
00035
00036         IMPLICIT NONE
00037         INTEGER, INTENT(IN) :: HSIZE
00038         INTEGER, INTENT(INOUT) :: II(:), JJ(:, :)
00039         INTEGER :: I, J, L, K, BANDWIDTH
00040
00041         REAL(LATTEPREC), INTENT(IN) :: HARRAY(:, :)
00042         REAL(LATTEPREC), INTENT(INOUT) :: VAL(:, :)
00043         REAL(LATTEPREC) :: XTMP
00044
00045         val = zero
00046         jj = 0
00047         ii = 0
00048
00049         bandwidth = 0
00050
00051         IF (basistype .EQ. "ORTHO") THEN
00052
00053             ! New compressed row format
00054
00055             !$OMP PARALLEL DO PRIVATE(I,J,K,L,XTMP) REDUCTION(MAX:BANDWIDTH)
00056             DO i = 1, hsize
00057                 l = 0
00058                 DO j = 1, hsize
00059                     xtmp = harray(j,i)
00060                     IF (j .EQ. i) THEN
00061                         l = l + 1
00062                         jj(l,i) = j
00063                         val(l,i) = (maxeval - xtmp)/maxminusmin
00064                     ELSEIF (abs(xtmp) .GE. hthresh) THEN
00065                         l = l + 1
00066                         jj(l,i) = j
00067                         val(l,i) = -xtmp/maxminusmin
00068                         bandwidth = max(bandwidth, abs(i-j))
00069                     ENDIF
00070                 ENDDO
00071                 ii(i) = l
00072             ENDDO
00073             !$OMP END PARALLEL DO
00074
00075         ELSE
00076
00077             !$OMP PARALLEL DO PRIVATE(I,J,L,XTMP) REDUCTION(MAX:BANDWIDTH)
00078             DO i = 1, hsize
00079                 l = 0
00080                 DO j = 1, hsize
00081                     xtmp = orthoh(j,i)
00082                     IF (j .EQ. i) THEN
00083                         l = l + 1
00084                         jj(l,i) = j
00085                         val(l,i) = (maxeval - xtmp)/maxminusmin
00086                     ELSEIF (abs(xtmp) .GE. hthresh) THEN
00087                         l = l + 1
00088                         jj(l,i) = j
00089                         val(l,i) = -xtmp/maxminusmin
00090                         bandwidth = max(bandwidth, abs(i-j))
00091                     ENDIF
00092                 ENDDO
00093                 ii(i) = l

```

```

00094         ENDDO
00095         !$OMP END PARALLEL DO
00096
00097     ENDIF
00098
00099 END SUBROUTINE dense2sparse
00100
00101
00102 SUBROUTINE sparse2dense(SCALAR, II, JJ, VAL, DARRAY, HSIZE)
00103
00104     USE myprecision
00105     USE constants_mod
00106
00107     IMPLICIT NONE
00108     INTEGER :: I,J
00109     INTEGER, INTENT(IN) :: HSIZE
00110     INTEGER, INTENT(IN) :: II(:), JJ(:, :)
00111
00112     REAL(LATTEPREC), INTENT(IN) :: SCALAR
00113     REAL(LATTEPREC), INTENT(IN) :: VAL(:, :)
00114     REAL(LATTEPREC), INTENT(INOUT) :: DARRAY(:, :)
00115
00116     darray = zero
00117
00118     DO i = 1, hsize
00119         DO j = 1, ii(i)
00120             darray(i, jj(j,i)) = scalar*val(j,i)
00121         ENDDO
00122     ENDDO
00123
00124 END SUBROUTINE sparse2dense
00125
00126 SUBROUTINE sp2loop(MSIZE, ITER, II, JJ, VAL)
00127
00128     USE myprecision
00129     USE constants_mod
00130     USE sparsearray
00131     USE purearray
00132     USE sparsemath
00133     USE omp_lib
00134     USE matrixio
00135
00136     IMPLICIT NONE
00137     INTEGER, INTENT(INOUT) :: MSIZE, ITER
00138     INTEGER, INTENT(INOUT) :: II(:), JJ(:, :)
00139     INTEGER :: BREAKLOOP, MAXM
00140     INTEGER, ALLOCATABLE :: IIC(:), JJC(:, :)
00141
00142     REAL(LATTEPREC) :: TRX, TRNORM, OCC
00143     REAL(LATTEPREC) :: TRX2, TR2XX2, TRXOLD
00144     REAL(LATTEPREC) :: LIMDIFF, IDEMPERR, IDEMPERR1, IDEMPERR2
00145     REAL(LATTEPREC), INTENT(INOUT) :: VAL(:, :)
00146     REAL(LATTEPREC), ALLOCATABLE :: CCN(:, :)
00147     REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0e-14
00148
00149     ! Allocate temporary arrays
00150     ALLOCATE(ix(hdim))
00151     ALLOCATE(x(hdim), y(hdim))
00152     ALLOCATE(iic(hdim))
00153     ALLOCATE(jjc(msize, hdim), jjb(msize, hdim))
00154     ALLOCATE(ccn(msize, hdim))
00155
00156     occ = bndfil*float(hdim)
00157
00158     idemperr2 = zero
00159     idemperr1 = zero
00160     idemperr = zero
00161
00162     maxm = 0
00163
00164     ix = 0
00165     x = 0
00166     y = 0
00167
00168     ccn = 0
00169     jjc = 0
00170     jjb = 0
00171     iic = 0
00172
00173     iter = 0
00174     breakloop = 0
00175
00176     DO WHILE ( breakloop .EQ. 0 .AND. iter .LT. 100 )
00177         iter = iter + 1
00178
00179         !
00180

```

```

00181      ! Sparse BO * BO
00182      !
00183
00184      trx = zero
00185      trx2 = zero
00186
00187      ! Matrix multiply X^2
00188      CALL sparsex2(trx, trx2, ii, jj, val, iic, jjc, ccn)
00189
00190      maxm = max(maxm, 2+maxval(iic))
00191
00192      tr2xx2 = two*trx - trx2
00193      trxold = trx
00194      limdiff = abs(trx2 - occ) - abs(tr2xx2 - occ)
00195
00196      IF ( limdiff .GT. idemtol ) THEN ! X <= 2X-X^2
00197
00198          trx = two * trx - trx2
00199
00200          pp(iter) = 0
00201
00202          ! Sparse matrix X = 2X - X^2
00203          CALL sparseadd(trnorm, ii, jj, val, iic, jjc, ccn)
00204
00205      ELSEIF ( limdiff .LT. -idemtol ) THEN ! X <= X^2
00206
00207          trx = trx2
00208          pp(iter) = 1;
00209
00210          ! Sparse matrix X = X^2
00211          CALL sparsesetx2(trnorm, ii, jj, val, iic, jjc, ccn)
00212
00213      ELSE
00214
00215          trx = trxold
00216          breakloop = 1
00217
00218      ENDIF
00219
00220      vv(iter) = sqrt(trnorm)
00221      !WRITE(*,*) 'IT = ', ITER, ' VV(', ITER, ',) = ', VV(ITER), PP(ITER)
00222
00223      idemperr2 = idemperr1
00224      idemperr1 = idemperr
00225      idemperr = abs(trx - trxold)
00226
00227      IF (sp2conv .EQ. "REL" .AND. iter .GE. minsp2iter &
00228          .AND. (idemperr .GE. idemperr2) ) breakloop = 1
00229
00230      IF (sp2conv .EQ. "ABS" .AND. abs(limdiff) .LE. idemtol) breakloop = 1
00231
00232      ! NOT converging
00233      IF (iter .EQ. 100) THEN
00234          CALL panic
00235          CALL errors("sparsesp2", "Sparse SP2 purification is not converging: STOP!")
00236      ENDIF
00237
00238      ENDDO
00239
00240      nr_sp2_iter = iter
00241
00242      ! Reset number of non-zeroes per row - MSPARSE
00243      msize = maxm
00244
00245      ! MSIZE = HDIM
00246      ! Deallocate temporary arrays
00247      DEALLOCATE(ix)
00248      DEALLOCATE(x, y)
00249      DEALLOCATE(iic)
00250      DEALLOCATE(jjc, jjb)
00251      DEALLOCATE(ccn)
00252
00253      END SUBROUTINE sp2loop
00254
00255
00256      SUBROUTINE sp2sequenceloop(MSIZE, ITER, II, JJ, VAL)
00257
00258          USE myprecision
00259          USE constants_mod
00260          USE purearray
00261          USE sparsearray
00262          USE sparsemath
00263          USE omp_lib
00264
00265          IMPLICIT NONE
00266          INTEGER, INTENT(INOUT) :: MSIZE, ITER
00267          INTEGER :: BREAKLOOP, MAXM

```

```

00268     INTEGER, INTENT(INOUT) :: II(:), JJ(:, :)
00269     INTEGER, ALLOCATABLE :: IIC(:), JJC(:, :)
00270
00271     REAL(LATTEPREC) :: TRX, TRNORM, OCC
00272     REAL(LATTEPREC) :: TRX2, TR2XX2, TRXOLD
00273     REAL(LATTEPREC) :: LIMDIFF, IDEMPERR, IDEMPERR1, IDEMPERR2
00274     REAL(LATTEPREC), INTENT(INOUT) :: VAL(:, :)
00275     REAL(LATTEPREC), ALLOCATABLE :: CCN(:, :)
00276     REAL(LATTEPREC), PARAMETER :: IDEMTOL = 1.0e-14
00277
00278     ! Allocate temporary arrays
00279     ALLOCATE(ix(hdim))
00280     ALLOCATE(x(hdim), y(hdim))
00281     ALLOCATE(iic(hdim))
00282     ALLOCATE(jjc(msize, hdim), jjb(msize, hdim))
00283     ALLOCATE(ccn(msize, hdim))
00284
00285     occ = bndfil*float(hdim)
00286
00287     idemperr2 = 0.0
00288     idemperr1 = 0.0
00289     idemperr = 0.0
00290
00291     maxm = 0
00292
00293     ix = 0
00294     x = zero
00295     y = zero
00296
00297     ccn = zero
00298     jjc = 0
00299     iic = 0
00300
00301     iter = 0
00302     breakloop = 0
00303     ! WRITE(*,*) ' NR_SP2_ITER = ', NR_SP2_ITER
00304
00305     DO WHILE (iter .LT. nr_sp2_iter )
00306
00307         iter = iter + 1
00308
00309         !
00310         ! Sparse BO * BO
00311         !
00312
00313         trx = zero
00314         trx2 = zero
00315
00316         ! Matrix multiply X^2
00317         CALL sparsex2(trx, trx2, ii, jj, val, iic, jjc, ccn)
00318
00319         maxm = max(maxm, 2*maxval(iic))
00320
00321
00322         tr2xx2 = two*trx - trx2
00323         trxold = trx
00324         limdiff = abs(trx2 - occ) - abs(tr2xx2 - occ)
00325
00326         IF (pp(iter).EQ.0) THEN ! X <= 2X-X^2
00327             trx = two * trx - trx2
00328
00329             ! Sparse matrix X = 2X - X^2
00330             CALL sparseadd(trnorm, ii, jj, val, iic, jjc, ccn)
00331
00332         ELSE ! X <= X^2
00333
00334             trx = trx2
00335
00336             ! Sparse matrix X = X^2
00337             CALL sparsesetx2(trnorm, ii, jj, val, iic, jjc, ccn)
00338
00339         ENDIF
00340
00341         vv(iter) = sqrt(trnorm)
00342         ! WRITE(*,*) ' IT = ', ITER, ' VV(', ITER, ',) = ', VV(ITER), PP(ITER)
00343
00344         ! NOT converging
00345         IF (iter .GE. 100) THEN
00346             CALL panic
00347             CALL errors("sparsesp2", "Sparse SP2 purification is not converging: STOP!")
00348         ENDIF
00349
00350     ENDDO
00351     nr_sp2_iter = iter
00352
00353     ! Reset number of non-zeroes per row - MSPARSE
00354

```



```

00355      ! PRINT*, MAXM, HDIM, 2*MAXVAL(IIC)
00356
00357      msize = maxm
00358      ! MSIZE = HDIM
00359      ! Deallocate temporary arrays
00360      DEALLOCATE(ix)
00361      DEALLOCATE(x, y)
00362      DEALLOCATE(iic)
00363      DEALLOCATE(jjc, jjb)
00364      DEALLOCATE(ccn)
00365
00366      END SUBROUTINE sp2sequenceloop
00367
00368      END MODULE sparsesp2

```

8.409 spinarray.f90 File Reference

Modules

- module [spinarray](#)

Variables

- integer [spinarray::deltadim](#)
- real(latteprec), dimension(:), allocatable [spinarray::deltaspin](#)
- real(latteprec), dimension(:), allocatable [spinarray::h2vect](#)
- real(latteprec), dimension(:, :), allocatable [spinarray::hdown](#)
- real(latteprec), dimension(:, :), allocatable [spinarray::hup](#)
- real(latteprec), dimension(:), allocatable [spinarray::olddeltaspin](#)
- real(latteprec), dimension(:, :), allocatable [spinarray::rhdown](#)
- real(latteprec), dimension(:), allocatable [spinarray::rhdownzero](#)
- real(latteprec), dimension(:, :), allocatable [spinarray::rhoup](#)
- real(latteprec), dimension(:), allocatable [spinarray::rhoupzero](#)
- real(latteprec), dimension(:), allocatable [spinarray::spinlist](#)
- real(latteprec), dimension(:), allocatable [spinarray::wdd](#)
- real(latteprec), dimension(:), allocatable [spinarray::wff](#)
- real(latteprec), dimension(:), allocatable [spinarray::wpp](#)
- real(latteprec), dimension(:), allocatable [spinarray::wss](#)

8.410 spinarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021

```

```

00022 MODULE spinarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER :: deltadim
00030   REAL(LATTEPREC), ALLOCATABLE :: wss(:), wpp(:), wdd(:), wff(:)
00031   REAL(LATTEPREC), ALLOCATABLE :: hup(:, :), hdown(:, :)
00032   REAL(LATTEPREC), ALLOCATABLE :: rhoup(:, :), rhodown(:, :)
00033   REAL(LATTEPREC), ALLOCATABLE :: rhoupzero(:), rhodownzero(:)
00034   REAL(LATTEPREC), ALLOCATABLE :: deltaspin(:), olddeltaspin(:)
00035   REAL(LATTEPREC), ALLOCATABLE :: spinlist(:), h2vect(:)
00036
00037 END MODULE spinarray

```

8.411 spinrhodirect.f90 File Reference

Functions/Subroutines

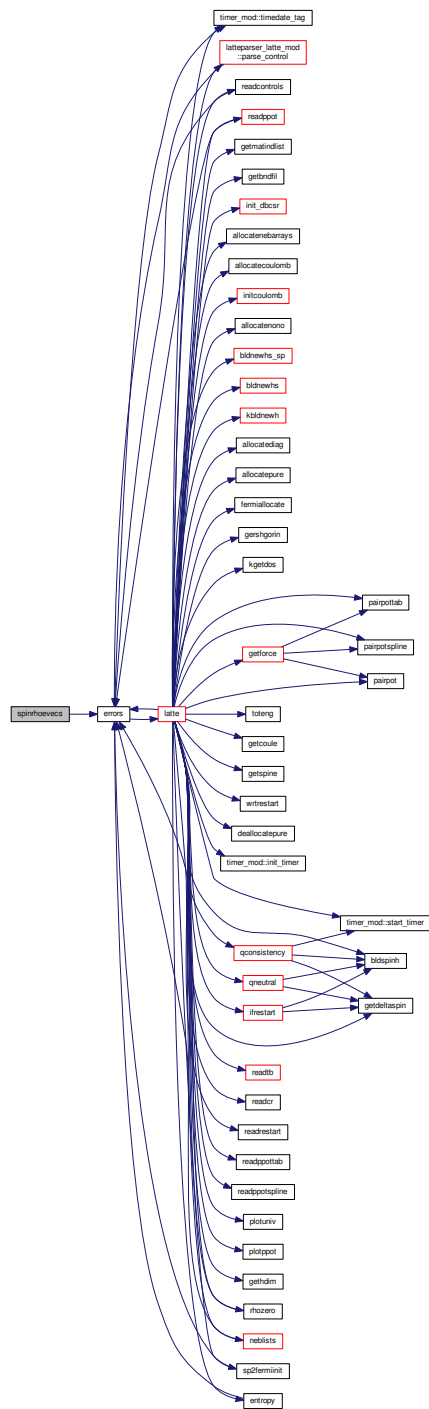
- subroutine [spinrhoevcs](#)

8.411.1 Function/Subroutine Documentation

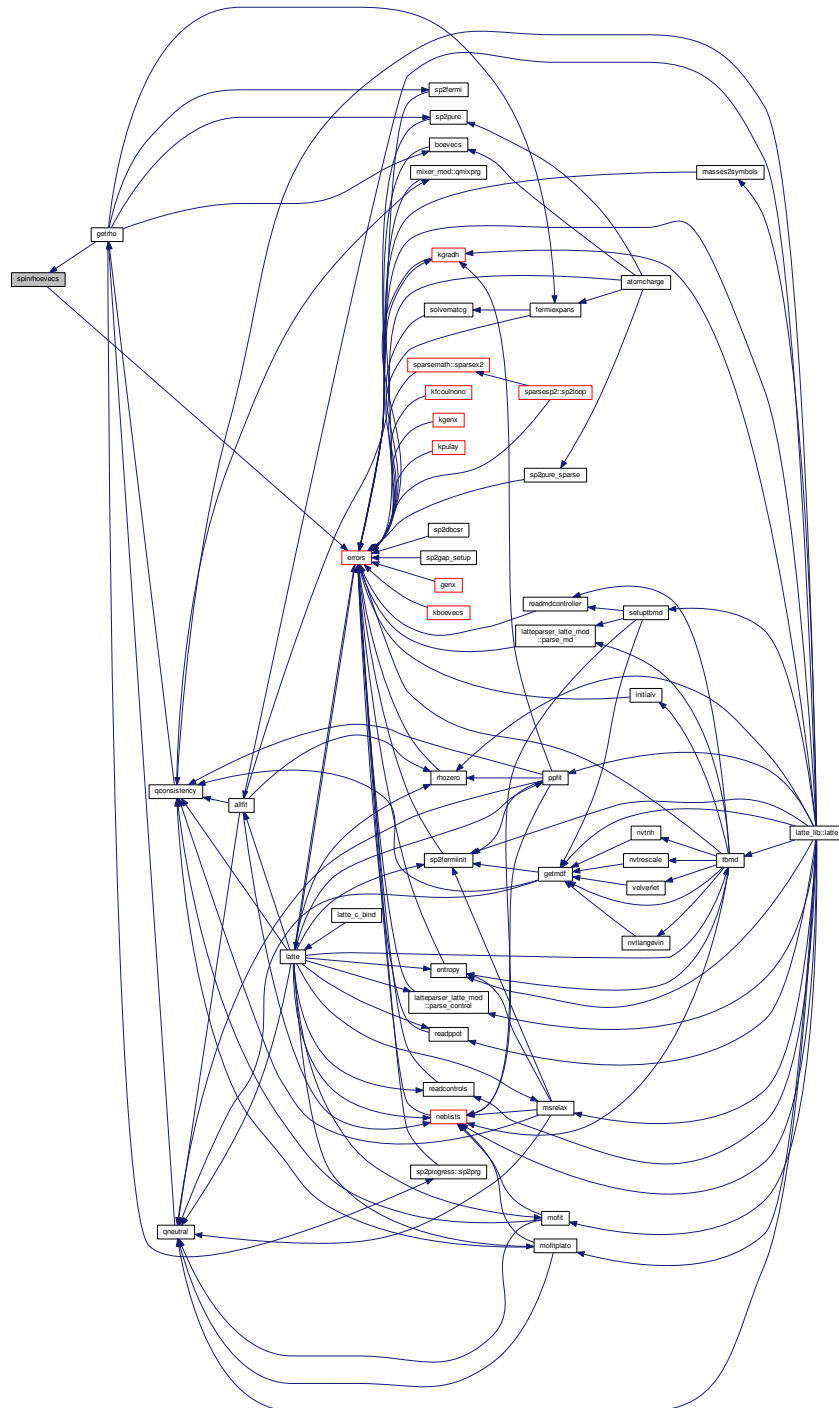
8.411.1.1 subroutine spinrhoevcs ()

Definition at line 23 of file [spinrhodirect.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.412 spinrhodirect.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE spinrhoevecs
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE diagarray
00027   USE spinarray
00028   USE mdarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K
00034   INTEGER :: ITER, BREAKLOOP
00035   INTEGER :: UPNE, DOWNNE
00036   REAL(LATTEPREC) :: OCCERROR , OCCUP, OCCDOWN
00037   REAL(LATTEPREC) :: EXPARG, FDIRAC
00038   REAL(LATTEPREC) :: SHIFTCP
00039   REAL(LATTEPREC) :: FDIRACARG, DFDIRAC
00040   REAL(LATTEPREC), PARAMETER :: MAXSHIFT = one
00041   REAL(LATTEPREC) :: HOMO, LUMO
00042   REAL(LATTEPREC) :: S, OCCLOGOCC_ELECTRONS, OCCLOGOCC_HOLES
00043   IF (existerror) RETURN
00044
00045   ! NUMLIMIT = EXP(-EXPTOL)
00046
00047   rhoup = zero
00048   rhodown = zero
00049
00050   iter = 0
00051
00052   occerror = 100000000.0
00053
00054   !
00055   ! For a finite electronic temperature
00056   !
00057
00058   IF (kbt .GT. 0.000001) THEN
00059
00060     DO WHILE ( abs(occerror) .GT. breaktol .AND. iter .LT. 100 )
00061
00062       iter = iter + 1
00063
00064       occup = zero
00065       occdown = zero
00066       dffdirac = zero
00067
00068       DO i = 1, hdim
00069
00070         fdiracarg = (upevals(i) - chempot)/kbt
00071
00072         fdiracarg = max(fdiracarg, -exptol)
00073         fdiracarg = min(fdiracarg, exptol)
00074
00075         exparg = exp(fdiracarg)
00076         fdirac = one/(one + exparg)
00077         occup = occup + fdirac
00078         dffdirac = dffdirac + exparg*fdirac*fdirac
00079
00080         fdiracarg = (downevals(i) - chempot)/kbt
00081
00082         fdiracarg = max(fdiracarg, -exptol)
00083         fdiracarg = min(fdiracarg, exptol)
00084
00085
00086         exparg = exp(fdiracarg)
00087         fdirac = one/(one + exparg)
00088         occdown = occdown + fdirac
00089         dffdirac = dffdirac + exparg*fdirac*fdirac
00090
00091       ENDDO
00092
00093       dffdirac = dffdirac/kbt

```

```

00094
00095     occerror = totne - occup - occdown
00096
00097     IF (abs(dfdirac) .LT. numlimit) dfdirac = sign(numlimit, dfdirac)
00098
00099     shiftcp = occerror/dfdirac
00100
00101     IF (abs(shiftcp) .GT. maxshift) shiftcp = sign(maxshift, shiftcp)
00102
00103     chempot = chempot + shiftcp
00104
00105 ENDDO
00106
00107 IF (iter .GT. 100) THEN
00108     CALL errors("spinrhodirect", "Newton-Raphson scheme to find the &
00109         & chemical potential has not converged")
00110 ENDIF
00111
00112
00113 ! Now we have the chemical potential we can build the density matrices
00114
00115 ! Entropy and HOMO-LUMO: only computed when we need them during MD
00116
00117 s = zero
00118
00119 IF (mdon .EQ. 0 .OR. &
00120     (mdon .EQ. 1 .AND. mod(entropyiter, wrtfreq) .EQ. 0 )) THEN
00121
00122     ! We will compute the HOMO-LUMO gap and the entropy in here too
00123
00124     ! Starting guesses
00125
00126     homo = min(upevals(1), downevals(1))
00127     lumo = max(upevals(hdim), downevals(hdim))
00128
00129     DO i = 1, hdim
00130
00131         ! Entropy
00132
00133         fdiracarg = (upevals(i) - chempot)/kbt
00134
00135         fdiracarg = max(fdiracarg, -exptol)
00136         fdiracarg = min(fdiracarg, exptol)
00137
00138         fdirac = one/(one + exp(fdiracarg))
00139
00140         occlogocc_electrons = fdirac * log(fdirac)
00141         occlogocc_holes = (one - fdirac) * log(one - fdirac)
00142
00143         s = s + occlogocc_electrons + occlogocc_holes
00144
00145         fdiracarg = (downevals(i) - chempot)/kbt
00146
00147         fdiracarg = max(fdiracarg, -exptol)
00148         fdiracarg = min(fdiracarg, exptol)
00149
00150         fdirac = one/(one + exp(fdiracarg))
00151
00152         occlogocc_electrons = fdirac * log(fdirac)
00153         occlogocc_holes = (one - fdirac) * log(one - fdirac)
00154
00155         s = s + occlogocc_electrons + occlogocc_holes
00156
00157         ! Homo-Lumo
00158
00159         IF (upevals(i) .LT. chempot .AND. &
00160             upevals(i) .GT. homo) homo = upevals(i)
00161         IF (downevals(i) .LT. chempot .AND. &
00162             downevals(i) .GT. lumo) lumo = downevals(i)
00163
00164         ! LUMO = smallest eigenvalue > chemical potential
00165
00166         IF (upevals(i) .GT. chempot .AND. &
00167             upevals(i) .LT. lumo) lumo = upevals(i)
00168         IF (downevals(i) .GT. chempot .AND. &
00169             downevals(i) .LT. lumo) lumo = downevals(i)
00170
00171
00172     ENDDO
00173
00174     egap = lumo - homo
00175
00176 ENDIF
00177
00178 ente = -kbt*s
00179
00180 ! Building density matrices

```

```

00181
00182     DO i = 1, hdim
00183
00184         fdiracarg = (upevals(i) - chempot)/kbt
00185
00186         fdiracarg = max(fdiracarg, -exptol)
00187         fdiracarg = min(fdiracarg, exptol)
00188
00189         fdirac = one/(one + exp(fdiracarg))
00190
00191         CALL dger(hdim, hdim, fdirac, upevecs(:,i), 1, upevecs(:,i), 1,
rhoup, hdim)
00192
00193         fdiracarg = (downevals(i) - chempot)/kbt
00194
00195         fdiracarg = max(fdiracarg, -exptol)
00196         fdiracarg = min(fdiracarg, exptol)
00197
00198         fdirac = one/(one + exp(fdiracarg))
00199
00200         CALL dger(hdim, hdim, fdirac, downevals(:,i), 1,
downevals(:,i), 1, rhodown, hdim)
00201
00202     ENDDO
00203
00204     ELSE ! For zero electronic temperature
00205
00206         ! Occupy the lowest eigenvalues
00207
00208         ! See whether we occupy spin up or down first
00209
00210         IF (upevals(1) .LT. downevals(1)) THEN
00211             upne = 1
00212             downne = 0
00213             chempot = upevals(1)
00214         ELSE
00215             upne = 0
00216             downne = 1
00217             chempot = downevals(1)
00218         ENDIF
00219
00220         ! Go through the eigenvalues and occupy if
00221         ! 1) its energy >= the previous occupied eigenvalue
00222         ! 2) its energy <= the energy of the unoccupied eigenvalue from the other
00223         ! spin channel
00224
00225         ! Stop when the total number of electrons reaches the target
00226         ! Works with 0 and 02.
00227
00228         breakloop = 0
00229         IF ( upne + downne .EQ. totne ) breakloop = 1
00230
00231         DO WHILE ( breakloop .EQ. 0 )
00232
00233             IF ( upevals(upne + 1) .GE. chempot .AND. &
00234                 upevals(upne + 1) .LE. downevals(downne + 1)) THEN
00235                 upne = upne + 1
00236                 chempot = upevals(upne)
00237             ENDIF
00238
00239             IF ( upne + downne .EQ. totne ) breakloop = 1
00240
00241             IF ( breakloop .EQ. 0 .AND. &
00242                 downevals(downne + 1) .GE. chempot .AND. &
00243                 downevals(downne + 1) .LE. upevals(upne + 1)) THEN
00244                 downne = downne + 1
00245                 chempot = downevals(downne)
00246             ENDIF
00247
00248             IF ( upne + downne .EQ. totne ) breakloop = 1
00249
00250         ENDDO
00251
00252         ! Here HOMO = CHEMPOT, so the energy of the
00253         ! LUMO = min(upevals(upne + 1), downevals(downne + 1))
00254
00255         lumo = min(upevals(upne + 1), downevals(downne + 1))
00256
00257         egap = lumo - chempot
00258
00259         DO i = 1, upne
00260
00261             CALL dger(hdim, hdim, one, upevecs(:,i), 1, upevecs(:,i), 1,
rhoup, hdim)
00262
00263         ENDDO
00264

```

```
00265      DO i = 1, downne
00266
00267          CALL dger(hdim, hdim, one, downvecs(:,i), 1,
downvecs(:,i), 1, rhodown, hdim)
00268
00269      ENDDO
00270
00271  ENDIF
00272
00273  RETURN
00274
00275 END SUBROUTINE spinrhovecs
```

8.413 stdescent.f90 File Reference

Functions/Subroutines

- subroutine [stdescent](#) (ITER, DELTAENERGY, DELTAF)

8.413.1 Function/Subroutine Documentation

8.413.1.1 subroutine `stdescent` (integer *ITER*, real(latteprec) *DELTAENERGY*, real(latteprec) *DELTAF*)

Definition at line 23 of file [stdescent.f90](#).


```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was made available
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos National
00004 ! National Laboratory (LANL), which is operated by Los Alamos National Security, LLC
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has the right
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE stdescent(ITER, DELTAENERGY, DELTAF)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE relaxcommon
00027   USE myprecision
00028
00029   IMPLICIT NONE
00030
00031   INTEGER :: I, J, ITER
00032   REAL(LATTEPREC) :: MAXF, DISP, MYSHIFT, DELTAENERGY, DELTAF
00033   REAL(LATTEPREC), PARAMETER :: MAXMCONST = 0.05d0, minmconst = 1.0d-3
00034   REAL(LATTEPREC), PARAMETER :: MAXSHIFT = 0.05d0
00035   IF (existerror) RETURN
00036
00037   IF (iter .EQ. 1) THEN
00038
00039       relconst = two*minmconst
00040
00041   ELSE
00042
00043       ! If the energy and maximum force have gone down, increase the step size
00044
00045       IF (deltaenergy .LE. zero .AND. deltaf .LT. zero) THEN
00046
00047           relconst = 1.01*relconst
00048
00049       ELSEIF (deltaenergy .GT. zero .OR. deltaf .GT. zero) THEN
00050
00051           ! If the energy is increasing - decrease the step size
00052
00053           relconst = 0.5*relconst
00054
00055       ENDIF
00056
00057   ENDIF
00058
00059   relconst = min(relconst, maxmconst)
00060   relconst = max(relconst, minmconst)
00061
00062   ! PREVF = MAXF
00063
00064   DO i = 1, nats
00065
00066       DO j = 1, 3
00067
00068           myshift = relconst*ftot(j,i)
00069
00070           IF (abs(myshift) .GT. maxshift) myshift = sign(maxshift, myshift)
00071
00072           cr(j,i) = cr(j,i) + myshift
00073
00074       ENDDO
00075
00076   ENDDO
00077
00078   RETURN
00079
00080 END SUBROUTINE stdescent

```

8.415 subgraph.f90 File Reference

Modules

- module [subgraph](#)

Functions/Subroutines

- subroutine `subgraph::dense2sparsegraph` (IIG, JJG, GGN)
- subroutine `subgraph::partitiongraph`
- subroutine `subgraph::thresholdgraph`
- real(latteprec) function `subgraph::tracegraph` (HDIM)
- subroutine `subgraph::trnormgraph` (ITERZ)

Variables

- integer `subgraph::first_step`
- real(latteprec), dimension(:,:), allocatable `subgraph::g`
- real(latteprec), parameter `subgraph::geps` = 1.0E-3
- integer, dimension(:,:), allocatable `subgraph::node_in_part`
- integer `subgraph::nr_nodes`
- integer, dimension(:), allocatable `subgraph::nr_of_nodes_in_part`
- integer `subgraph::nr_part`
- real(latteprec), dimension(:,:), allocatable `subgraph::vvx`

8.416 subgraph.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE subgraph
00023
00024   USE purearray
00025   USE myprecision
00026
00027   IMPLICIT NONE
00028   SAVE
00029
00030   INTEGER :: nr_nodes, nr_part, first_step
00031   INTEGER, ALLOCATABLE :: node_in_part(:,:), nr_of_nodes_in_part(:)
00032   REAL(LATTEPREC), ALLOCATABLE :: g(:,:)
00033   REAL(LATTEPREC), ALLOCATABLE :: vvx(:,:)
00034   REAL(LATTEPREC), PARAMETER :: geps = 1.0e-3
00035
00036 CONTAINS
00037
00038   SUBROUTINE thresholdgraph
00039
00040     USE myprecision
00041     USE constants_mod
00042     USE omp_lib
00043
00044     IMPLICIT NONE
00045     INTEGER :: I, J
00046
00047     !$OMP PARALLEL DO PRIVATE(I,J)
00048     DO j = 1, hdim
00049       DO i = 1, hdim

```

```

00050         IF (abs(g(i,j)).LT.geps) g(i,j) = zero
00051     ENDDO
00052 ENDDO
00053 !$OMP END PARALLEL DO
00054
00055 END SUBROUTINE thresholdgraph
00056
00057
00058 SUBROUTINE dense2sparsegraph(IIG, JJG, GGN)
00059
00060     USE myprecision
00061     USE constants_mod
00062
00063     IMPLICIT NONE
00064     INTEGER :: I, J, K, L
00065     INTEGER, INTENT(INOUT) :: IIG(:), JJG(:, :)
00066
00067     REAL(LATTEPREC) :: XTMP
00068     REAL(LATTEPREC), INTENT(INOUT) :: GGN(:, :)
00069
00070     !$OMP PARALLEL DO PRIVATE(I,J,K,L,XTMP)
00071     DO i = 1, hdim
00072         l = 0
00073         DO j = 1, hdim
00074             xtmp = g(j,i)
00075             IF (j .EQ. i) THEN
00076                 l = l + 1
00077                 jjg(l,i) = j
00078                 ggn(l,i) = xtmp
00079             ELSEIF (abs(xtmp) .GE. geps) THEN
00080                 l = l + 1
00081                 jjg(l,i) = j
00082                 ggn(l,i) = xtmp
00083             ENDIF
00084         ENDDO
00085         iig(i) = l
00086     ENDDO
00087     !$OMP END PARALLEL DO
00088
00089 END SUBROUTINE dense2sparsegraph
00090
00091
00092 SUBROUTINE partitiongraph
00093
00094     USE myprecision
00095     USE constants_mod
00096     USE sparsearray
00097     USE omp_lib
00098
00099     IMPLICIT NONE
00100     INTEGER :: IT, I, J
00101
00102     ! Partition graph
00103     ! Example liquid methane number of nodes = 8 = number of orbitals/molecule
00104     nr_nodes = 1 !!! Uniform partitioning for simplicity
00105     nr_part = hdim/nr_nodes
00106     !write(*,*) "number parts = ", NR_PART, " number nodes = ", NR_NODES
00107
00108     ALLOCATE(nr_of_nodes_in_part(nr_part))
00109     ALLOCATE(node_in_part(nr_part,nr_nodes))
00110     ALLOCATE(vvx(100,nr_part))
00111     vx = zero
00112
00113     !!! CREATE SIMPLE UNIFORM SUB-GRAPH PARTITIONING OF THE NODES
00114     it = 0
00115     DO i = 1, nr_part
00116         DO j = 1, nr_nodes
00117             it = it + 1
00118             node_in_part(i,j) = it
00119         ENDDO
00120         nr_of_nodes_in_part(i) = nr_nodes
00121     ENDDO
00122
00123 END SUBROUTINE partitiongraph
00124
00125
00126 SUBROUTINE trnormgraph(ITERZ)
00127
00128     USE myprecision
00129     USE constants_mod
00130     USE sparsearray
00131
00132     IMPLICIT NONE
00133     INTEGER :: I, J
00134     INTEGER, INTENT(INOUT) :: ITERZ
00135
00136     REAL(LATTEPREC) :: TRACE

```

```

00137
00138     vv = zero
00139
00140     !!! STORE SQRT(Tr[(Tr(X-X^2)^2]) FOR HOMO-LUMO ESTIMATE
00141     DO j = 1, nr_sp2_iter
00142         trace = zero
00143         DO i = 1, nr_part
00144             trace = trace + vv(x(j,i))
00145         ENDDO
00146         vv(j) = sqrt(trace)
00147     ENDDO
00148     iterz = nr_sp2_iter
00149
00150 END SUBROUTINE trnormgraph
00151
00152
00153 REAL(LATTEPREC) FUNCTION tracegraph(HDIM)
00154
00155     USE myprecision
00156
00157     INTEGER :: I
00158     INTEGER, INTENT(IN) :: HDIM
00159
00160     REAL(LATTEPREC) :: TRACE
00161
00162     trace = 0
00163     DO i = 1, hdim
00164         trace = trace + g(i,i)
00165     ENDDO
00166
00167     tracegraph = trace
00168
00169     RETURN
00170
00171 END FUNCTION tracegraph
00172
00173 END MODULE subgraph

```

8.417 subgraphsp2.f90 File Reference

Modules

- module [subgraphsp2](#)

Functions/Subroutines

- subroutine [subgraphsp2::extractsubgraph](#) (I, IIH, JJH, HHN, IIG, JJG, IX, JJN, JJP, LG, L, LL)
- subroutine [subgraphsp2::normalizesubgraph](#) (XS, JJN, L)
- subroutine [subgraphsp2::progressloop](#) (IIH, JJH, HHN, IIG, JJG)
- subroutine [subgraphsp2::subgraphsp2loop](#) (I, XS, JJP, L, LL)
- subroutine [subgraphsp2::subgraphsp2dense](#) (SCALAR, L, LL, JJN, JJP, DARRAY, XS)

Variables

- integer, dimension(:), allocatable [subgraphsp2::ix](#)
- integer, dimension(:), allocatable [subgraphsp2::jjn](#)
- integer, dimension(:), allocatable [subgraphsp2::jpp](#)
- integer, dimension(:), allocatable [subgraphsp2::lg](#)
- real(latteprec), dimension(:,:), allocatable [subgraphsp2::xs](#)

8.418 subgraphsp2.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE subgraphsp2
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   INTEGER, ALLOCATABLE :: ix(:), jjn(:), jjp(:), lg(:)
00030
00031   REAL(LATTEPREC), ALLOCATABLE :: xs(:, :)
00032
00033 CONTAINS
00034
00035   SUBROUTINE progressloop(IIH, JJH, HHN, IIG, JJG)
00036
00037     USE myprecision
00038     USE constants_mod
00039     USE sparsearray
00040     USE setuparray
00041     USE subgraph
00042     USE omp_lib
00043
00044     IMPLICIT NONE
00045     INTEGER :: I, L, LL
00046     INTEGER, INTENT(IN) :: IIH(:), JJH(:, :), IIG(:), JJG(:, :)
00047
00048     REAL(LATTEPREC) :: OCC
00049     REAL(LATTEPREC), INTENT(IN) :: HHN(:, :)
00050
00051     ALLOCATE(ix(hdim))
00052     ALLOCATE(jjn(hdim), jjp(hdim))
00053     ALLOCATE(lg(hdim))
00054
00055     occ = bndfil*float(hdim)
00056
00057     !!! Progress loop
00058     ix = 0
00059     jjn = 0
00060     jjp = 0
00061     lg = 0
00062
00063     !$OMP PARALLEL DO FIRSTPRIVATE(IX, JJP, JJN, LG) &
00064     !$OMP PRIVATE(I, L, LL, XS)
00065     DO i = 1, nr_part
00066
00067         ll = 0
00068         l = 0
00069
00070     !!! Extract small dense subgraph Hamiltonians
00071     CALL extractsubgraph(i, iih, jjh, hhn, iig, jjg, ix, jjn,
00072                          jjp, lg, l, ll)
00073
00074     ! Normalize small dense subgraph
00075     ALLOCATE(xs(l, l))
00076     CALL normalizesubgraph(xs, jjn, l)
00077
00078     ! Subgraph sp2 loop on each subgraph
00079     CALL subgraphsp2loop(i, xs, jjp, l, ll)
00080
00081     ! Collect each partial density matrix over subgraphs
00082     CALL subgraphsp2dense(two, l, ll, jjn, jjp,
00083                          bo, xs)
00084     DEALLOCATE(xs)

```

```

00083
00084      ENDDO
00085      !$OMP END PARALLEL DO
00086
00087      ! Deallocate arrays
00088      DEALLOCATE(ix)
00089      DEALLOCATE(jjn,jjp)
00090      DEALLOCATE(lg)
00091
00092      END SUBROUTINE progressloop
00093
00094      SUBROUTINE extractsubgraph(I, IIH, JJH, HHN, IIG, JJG, IX, JJN, JJP, LG, L, LL)
00095
00096      USE myprecision
00097      USE constants_mod
00098      USE subgraph
00099
00100      IMPLICIT NONE
00101
00102      INTEGER :: J, II, JP, K, L0, LS
00103      INTEGER, INTENT(IN) :: I
00104      INTEGER, INTENT(INOUT) :: L, LL
00105      INTEGER, INTENT(IN) :: IIH(:), JJH(:,:), IIG(:), JJG(:,:)
00106      INTEGER, INTENT(INOUT) :: IX(:), JJN(:), JJP(:), LG(:)
00107
00108      REAL(LATTEPREC) :: TRACE
00109      REAL(LATTEPREC), INTENT(IN) :: HHN(:,:)
00110
00111      ! Extract elements from graph
00112      DO j = 1, nr_of_nodes_in_part(i)
00113          ii = node_in_part(i,j)
00114
00115          ! NR_OF_NODES(:) = [2,3,3,1,4,...]
00116          ! NODE_PART(:, :) = [1,2; 3,4,5; 6,9,21; 12; 17,8,11,55; ...]
00117
00118          ! From graph
00119          DO jp = 1, iig(ii)
00120              k = jjg(jp,ii)
00121              IF (ix(k) .NE. i) THEN
00122                  ix(k) = i
00123                  l = l + 1
00124                  jjn(l) = k      ! VECTOR FOR SUBGRAPH EXTRACTION IN a
00125                  lg(k) = 1
00126              ENDIF
00127              IF (k.EQ.ii) THEN
00128                  ll = ll + 1
00129                  jjp(ll) = lg(k) ! ROW INDEX OF ESSENTIAL DM NODES IN SUBGRAPH PARTITIONING
00130              ENDIF
00131          ENDDO
00132      ENDDO
00133      !if (I .EQ. 1) write(*,*) "From graph 1: L LL = ", L, " ", LL
00134      !if (I .EQ. 2) write(*,*) "From graph 2: L LL = ", L, " ", LL
00135
00136      ! Add possible new elements in H (Here the same as before, so no change!)
00137      l0 = l
00138      DO j = 1, nr_of_nodes_in_part(i)
00139          ii = node_in_part(i,j)
00140
00141          ! NR_OF_NODES(:) = [2,3,3,1,4,...]
00142          ! NODE_PART(:, :) = [1,2; 3,4,5; 6,9,21; 12; 17,8,11,55; ...]
00143
00144          ! From Hamiltonian
00145          DO jp = 1, iih(ii)
00146              k = jjh(jp,ii)
00147              IF (ix(k) .NE. i) THEN
00148                  ix(k) = i
00149                  l = l + 1
00150                  jjn(l) = k      ! VECTOR FOR SUBGRAPH EXTRACTION IN a
00151              ENDIF
00152          ENDDO
00153      ENDDO
00154      !if(I .EQ. 1)write(*,*) "From H: L LL = ", L, " ", LL
00155
00156      ! Perform a "double jump" for possible extra elements
00157      ! based on graph
00158      ls = l
00159      DO j = 1,ls
00160          ii = jjn(j)
00161          DO jp = 1, iig(ii)
00162              k = jjg(jp,ii)
00163              IF (ix(k) .NE. i) THEN
00164                  ix(k) = i
00165                  l = l + 1
00166                  jjn(l) = k
00167              ENDIF
00168          ENDDO
00169      ENDDO

```

```

00170      !if(I.EQ.1)write(*,*) "From graph double jump: L LL = ", L, " ", LL
00171
00172      ! Example of automatic partitioning
00173      ! Includes 1st orbital or number of nodes plus halo
00174      IF (i.EQ.1) THEN
00175        WRITE(*,*) '# SUBGRAPH_1_SIZE: ', 1, ' x ', 1
00176      ENDIF
00177      IF (i.EQ.10) THEN
00178        WRITE(*,*) '# SUBGRAPH_10_SIZE: ', 1, ' x ', 1
00179      ENDIF
00180
00181      END SUBROUTINE extractsubgraph
00182
00183      SUBROUTINE normalizesubgraph(XS, JJN, L)
00184
00185        USE myprecision
00186        USE constants_mod
00187        USE setuparray
00188
00189        IMPLICIT NONE
00190        INTEGER :: J, JA, JB
00191        INTEGER, INTENT(IN) :: L
00192        INTEGER, INTENT(IN) :: JJN(:)
00193
00194        REAL(LATTEPREC), INTENT(INOUT) :: XS(:, :)
00195
00196        ! Get values for subgraph from dense H
00197        DO ja = 1, 1
00198          DO jb = 1, 1
00199            xs(ja,jb) = h(jjn(ja),jjn(jb)) ! (*)
00200          ENDDO
00201        ENDDO
00202
00203        xs = minusone * xs
00204        DO j = 1,1
00205          xs(j,j) = xs(j,j) + maxeval
00206        ENDDO
00207        xs = (one/(maxeval-mineval))*xs
00208
00209      END SUBROUTINE normalizesubgraph
00210
00211      SUBROUTINE subgraphsp2loop(I, XS, JJP, L, LL)
00212
00213        USE myprecision
00214        USE constants_mod
00215        USE purearray
00216        USE sparsearray
00217        USE subgraph
00218
00219        IMPLICIT NONE
00220        INTEGER :: IT, J, JA, JB
00221        INTEGER, INTENT(IN) :: I, L, LL
00222        INTEGER, INTENT(IN) :: JJP(:)
00223
00224        REAL(LATTEPREC) :: TRACE
00225        REAL(LATTEPREC), INTENT(INOUT) :: XS(:, :)
00226        REAL(LATTEPREC), ALLOCATABLE :: XST(:, :)
00227
00228        ! Subgraph sp2 loop on XS
00229        !
00230        ! Calculate X-X^2 for first iteration
00231        ! XST = -1.0 * XS^2 + 1.0 * XST
00232        ALLOCATE(xst(1,1))
00233        !! XST = XS - MATMUL(XS,XS)
00234        xst = xs
00235        CALL dgemm('N', 'N', 1, 1, 1, minusone, &
00236          xs, 1, xs, 1, one, xst, 1)
00237
00238        ! Calculate trace of (X-X^2)^2 for each iteration
00239        DO it = 1,nr_sp2_iter
00240          trace = zero
00241          DO j = 1,11
00242            ja = jjp(j)
00243            DO jb = 1,1
00244              trace = trace + xst(jb,ja)*xst(jb,ja) ! Tr[(X-X^2)^2]
00245            ENDDO
00246          !if (I.EQ.1) write(*,*)"IT trace = ", IT, " ", TRACE
00247          ENDDO
00248          vvx(it,i) = vvx(it,i) + trace ! ADD IN SHARED MEMORY VVX
00249
00250          ! Calculate X-X^2 for each iteration
00251          ! XS = XS +/- XST and XST = XS
00252          ! XST = -1.0 * XS^2 + 1.0 * XST
00253          xs = xs + (one-two*pp(it))*xst
00254          !! XST = XS - MATMUL(XS,XS)
00255          xst = xs
00256

```



```

00257      CALL dgemm('N', 'N', 1, 1, 1, minusone, &
00258                xs, 1, xs, 1, one, xst, 1)
00259      ENDDO
00260
00261      DEALLOCATE(xst)
00262
00263      END SUBROUTINE subgraphsp2loop
00264
00265      SUBROUTINE subgraphsparse2dense(SCALAR, L, LL, JJN, JJP, DARRAY, XS)
00266
00267      USE myprecision
00268
00269      IMPLICIT NONE
00270      INTEGER :: JA, JB
00271      INTEGER, INTENT(IN) :: L, LL
00272      INTEGER, INTENT(IN) :: JJN(:), JJP(:)
00273
00274      REAL(LATTEPREC), INTENT(IN) :: SCALAR
00275      REAL(LATTEPREC), INTENT(INOUT) :: XS(:, :)
00276      REAL(LATTEPREC), INTENT(INOUT) :: DARRAY(:, :)
00277
00278      DO ja = 1, ll ! Collect density matrix over nodes for partial density matrix
00279          ! WRITE(*,*) ' JJP = ', JJP(JA)
00280          DO jb = 1, 1
00281              darray(jjn(jb), jjn(jjp(ja))) = scalar*xs(jb, jjp(ja))
00282          ENDDO
00283      ENDDO
00284
00285      END SUBROUTINE subgraphsparse2dense
00286
00287      END MODULE subgraphsp2

```

8.419 summary.f90 File Reference

Functions/Subroutines

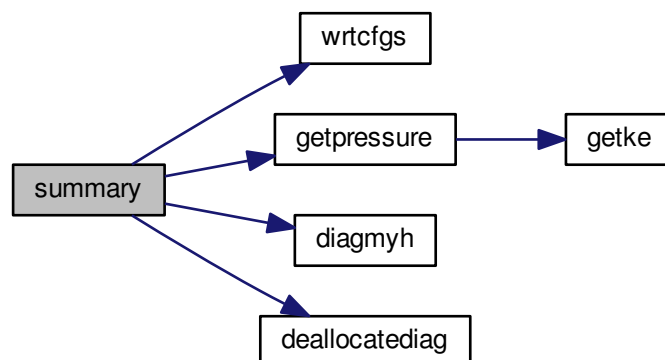
- subroutine [summary](#)

8.419.1 Function/Subroutine Documentation

8.419.1.1 subroutine [summary](#) ()

Definition at line 23 of file [summary.f90](#).

Here is the call graph for this function:




```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS            !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such        !
00010 ! modified software should be clearly marked, so as not to confuse it         !
00011 ! with the version available from LANL.                                       !
00012 !                                                                              !
00013 ! Additionally, this program is free software; you can redistribute it        !
00014 ! and/or modify it under the terms of the GNU General Public License as       !
00015 ! published by the Free Software Foundation; version 2.0 of the License.      !
00016 ! Accordingly, this program is distributed in the hope that it will be        !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of     !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General   !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE summary
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE purearray
00028   USE sparsearray
00029   USE coulombarray
00030   USE spinarray
00031   USE myprecision
00032   USE diagarray
00033   USE virialarray
00034   USE nonoarray
00035
00036   IMPLICIT NONE
00037
00038   INTEGER :: I, J, INDEX, NUMORB
00039   REAL(LATTEPREC) :: FDIRAC, SUMQ, ATOMSPIN
00040   IF (existerror) RETURN
00041
00042   IF ( verbose < 0 ) RETURN
00043
00044   CALL wrtcfgs(-999)
00045
00046   OPEN(unit=24, status="UNKNOWN", file="mylastLATTEcalc")
00047
00048   IF (mdon .EQ. 1) THEN
00049     WRITE(6, '( "# Molecular dynamics calculation" )')
00050     WRITE(24, '( "Molecular dynamics calculation" )')
00051     IF (qiter .EQ. 0) THEN
00052       WRITE(6, '( "# Using SCF-free Fast QMD" )')
00053       WRITE(24, '( "Using SCF-free Fast QMD" )')
00054     ENDIF
00055   ENDIF
00056
00057
00058   IF (electro .EQ. 1) THEN
00059     WRITE(6, '( "# Self-consistent charge transfer: on" )')
00060     WRITE(6, '( "# SCF tolerance = ", G8.3 )') elec_qtol
00061     WRITE(24, '( "Self-consistent charge transfer: on" )')
00062     WRITE(24, '( "SCF tolerance = ", G8.3 )') elec_qtol
00063     IF (elecsmeth .EQ. 0) THEN
00064       WRITE(6, '( "# Using Ewald summation" )')
00065       WRITE(6, '( "# Coulomb accuracy = ", G8.3 )') coulacc
00066       WRITE(6, '( "# Real space cut-off for Ewald sum = ", F6.2 )') coulcut
00067       WRITE(24, '( "Using Ewald summation" )')
00068       WRITE(24, '( "Coulomb accuracy = ", G8.3 )') coulacc
00069       WRITE(24, '( "Real space cut-off for Ewald sum = ", F6.2 )') coulcut
00070     ELSEIF (elecsmeth .EQ. 1) THEN
00071       WRITE(6, '( "# Real space electrostatics" )')
00072       WRITE(6, '( "# Cut-off tail applied between ", F6.2, F6.2 )') &
00073         coulrl, coulcut
00074       WRITE(24, '( "Real space electrostatics" )')
00075       WRITE(24, '( "Cut-off tail applied between ", F6.2, F6.2 )') &
00076         coulrl, coulcut
00077     ENDIF
00078   ELSE
00079     WRITE(6, '( "# Local charge neutrality: on" )')
00080     WRITE(6, '( "# SCF tolerance = ", G8.3 )') elec_qtol
00081     WRITE(24, '( "Local charge neutrality: on" )')
00082     WRITE(24, '( "SCF tolerance = ", G8.3 )') elec_qtol
00083   ENDIF
00084
00085   IF (spinon .EQ. 1) THEN
00086     WRITE(6, '( "# Spin-polarized calculation" )')
00087     WRITE(24, '( "Spin-polarized calculation" )')
00088   ENDIF
00089
00090   IF (control .EQ. 2 .AND. sparseon .EQ. 1) THEN
00091     WRITE(6, '( "# Sparse matrix calculations for O(N)" )')
00092     WRITE(24, '( "Sparse matrix calculations for O(N)" )')
00093     WRITE(6, '( "# Numerical threshold = ", G8.3 )') numthresh

```

```

00094     WRITE(24, '("Numerical threshold = ", G8.3)') numthresh
00095 ENDIF
00096
00097 IF (control .EQ. 1) THEN
00098     WRITE(6, '("# Diagonalization")')
00099     WRITE(24, '("Diagonalization")')
00100 ELSEIF (control .EQ. 2) THEN
00101     WRITE(6, '("# SP2 purification at zero temperature")')
00102     WRITE(24, '("SP2 purification at zero temperature")')
00103 ELSEIF (control .EQ. 3) THEN
00104     WRITE(6, '("# Recursive expansion of the Fermi operator")')
00105     WRITE(24, '("Recursive expansion of the Fermi operator")')
00106 ELSEIF (control .EQ. 4) THEN
00107     WRITE(6, '("# SP2 purification at finite temperature")')
00108     WRITE(24, '("SP2 purification at finite temperature")')
00109 ELSEIF (control .EQ. 5) THEN
00110     WRITE(6, '("# SP2/Fermi method at finite temperature")')
00111     WRITE(24, '("SP2/Fermi method at finite temperature")')
00112 ENDIF
00113
00114 ! CALL ALLOCATEDIAG
00115
00116 IF (ALLOCATED(diag_iwork) .EQV. .true.) THEN
00117     WRITE(6, '("# Using xSYEVD for diagonalization - potentially fast and potentially unreliable")')
00118     WRITE(24, '("# Using xSYEVD for diagonalization - potentially fast and potentially unreliable")')
00119 ELSEIF (ALLOCATED(diag_iwork) .EQV. .false.) THEN
00120     WRITE(6, '("# Using xSYEV for diagonalization")')
00121     WRITE(24, '("# Using xSYEV for diagonalization")')
00122 ENDIF
00123
00124
00125 IF (control .NE. 2) THEN
00126     WRITE(6, '("# KBT (in eV) = ", F16.8)') kbt
00127     WRITE(24, '("KBT (in eV) = ", F16.8)') kbt
00128 ENDIF
00129
00130 IF (control .EQ. 5) THEN
00131     WRITE(6, '("# Gershgorin: MAXEVAL, MINEVAL = ", 2F16.8)') maxeval,
mineval
00132     WRITE(24, '("Gershgorin: MAXEVAL, MINEVAL = ", 2F16.8)') maxeval,
mineval
00133 ENDIF
00134
00135 IF (latteprec .EQ. kind(0.0d0)) THEN
00136     WRITE(6, '("# Double precision arithmetic")')
00137     WRITE(24, '("Double precision arithmetic")')
00138 ELSEIF (latteprec .EQ. kind(0.0)) THEN
00139     WRITE(6, '("# Single precision arithmetic")')
00140     WRITE(24, '("Single precision arithmetic")')
00141 ENDIF
00142
00143 IF (entropykind .EQ. 0) THEN
00144     WRITE(6, '("# Entropy set = 0")')
00145     WRITE(24, '("Entropy set = 0")')
00146 ELSEIF (entropykind .EQ. 1) THEN
00147     WRITE(6, '("# Using exact ln form for entropy")')
00148     WRITE(24, '("Using exact ln form for entropy")')
00149 ELSEIF (entropykind .EQ. 2) THEN
00150     WRITE(6, '("# Using the close-to-exact expansion of exact entropy (2)")')
00151     WRITE(24, '("Using the close-to-exact expansion of exact entropy (2)")')
00152 ELSEIF (entropykind .EQ. 3) THEN
00153     WRITE(6, '("# Using 4th order approximation for entropy")')
00154     WRITE(24, '("Using 4th order approximation for entropy")')
00155 ELSEIF (entropykind .EQ. 4) THEN
00156     WRITE(6, '("# Using 8th order approximation for entropy")')
00157     WRITE(24, '("Using 8th order approximation for entropy")')
00158 ENDIF
00159
00160 WRITE(6, '("# Tr[ rho*H ] = ", F16.8)') trrhoh
00161 WRITE(6, '("# Pairwise energy = ", F16.8)') erep
00162 WRITE(24, '("Tr[ rho*H ] = ", F16.8)') trrhoh
00163 WRITE(24, '("Pairwise energy = ", F16.8)') erep
00164
00165 IF (electro .EQ. 1) THEN
00166     WRITE(6, '("# Coulombic + onsite E = ", F16.8)') ecoul
00167     WRITE(24, '("Coulombic + onsite E = ", F16.8)') ecoul
00168 ENDIF
00169
00170 IF (control .NE. 2) THEN
00171     WRITE(6, '("# Electron entropy TS = ", F16.8)') ente
00172     WRITE(24, '("Electron entropy TS = ", F16.8)') ente
00173 ENDIF
00174
00175 IF (control .EQ. 1 .OR. control .EQ. 3 .OR. control .EQ. 5) THEN
00176     WRITE(6, '("# Chemical potential = ", F16.8)') chempot
00177     WRITE(24, '("Chemical potential = ", F16.8)') chempot
00178 ENDIF

```

```

00179
00180 IF (spinon .EQ. 1) THEN
00181   WRITE(6, '( "# Self-consistent spin energy = ", F16.8)') espin
00182   WRITE(6, '( "# Free atom spin energy = ", F16.8)') espin_zero
00183   WRITE(24, '( "Self-consistent spin energy = ", F16.8)') espin
00184   WRITE(24, '( "Free atom spin energy = ", F16.8)') espin_zero
00185 ENDIF
00186
00187 CALL getpressure
00188
00189 WRITE(6, '( "# Pressure (GPa) = ", F16.8)') pressure
00190 WRITE(24, '( "# Pressure (GPa) = ", F16.8)') pressure
00191
00192 IF (spinon .EQ. 0) THEN
00193   WRITE(6, '( "# Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul
00194   WRITE(6, '( ""')
00195   WRITE(6, '( "# FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente
00196   WRITE(6, '( ""')
00197   WRITE(24, '( "Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul
00198   WRITE(24, '( ""')
00199   WRITE(24, '( "FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente
00200   WRITE(24, '( ""')
00201 ELSEIF (spinon .EQ. 1) THEN
00202   WRITE(6, '( "# Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul + &
00203     espin
00204   WRITE(6, '( ""')
00205   WRITE(6, '( "# FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente + &
00206     espin
00207   WRITE(6, '( ""')
00208   WRITE(24, '( "Total energy (zero K) = ", F16.8)') trrhoh + erep -
ecoul + &
00209     espin
00210   WRITE(24, '( ""')
00211   WRITE(24, '( "FREE ENERGY = ", F16.8)') trrhoh + erep - ecoul -
ente + &
00212     espin
00213   WRITE(24, '( ""')
00214 ENDIF
00215
00216 WRITE(6,60) "#checkP ", sysvol, trrhoh + erep - ecoul -
ente, -pressure/togpa
00217
00218 60 FORMAT(a7, 3f18.6)
00219
00220 ! Write the stress tensor
00221
00222 WRITE(6, '( ""')
00223 WRITE(6, '( "# Stress tensor (GPa)")')
00224 WRITE(6,251) "# ", strten(1), strten(4), strten(6)
00225 WRITE(6,251) "# ", strten(4), strten(2), strten(5)
00226 WRITE(6,251) "# ", strten(6), strten(5), strten(3)
00227 WRITE(6, '( ""')
00228 WRITE(24, '( ""')
00229 WRITE(24, '( "# Stress tensor (GPa)")')
00230 WRITE(24,251) "# ", strten(1), strten(4), strten(6)
00231 WRITE(24,251) "# ", strten(4), strten(2), strten(5)
00232 WRITE(24,251) "# ", strten(6), strten(5), strten(3)
00233 WRITE(24, '( ""')
00234
00235 250 FORMAT(3g20.9)
00236 251 FORMAT(a2,1x,3g20.9)
00237
00238 IF (kon .EQ. 0) THEN
00239
00240   IF (spinon .EQ. 0) THEN
00241
00242     ! Analyse the eigenvalues only if we've used diagonalization
00243     ! to compute the density matrix
00244
00245     IF (control .EQ. 1) THEN
00246
00247       IF (basistype .EQ. "ORTHO") THEN
00248
00249         WRITE(6, '( ""')
00250         WRITE(6, '( "# Eigenvalues")')
00251         WRITE(24, '( ""')
00252         WRITE(24, '( "Eigenvalues")')
00253
00254         IF (kbt .GT. 0.00000001) THEN
00255
00256           DO i = 1, hdim

```

```

00257
00258         fdirac = one/(one+exp((evals(i) - chempot)/
kbt))
00259
00260         IF (evals(i) .LE. chempot) THEN
00261
00262             WRITE(6, 54) "# ", i, evals(i), fdirac, "*"
00263             WRITE(24, 53) i, evals(i), fdirac, "*"
00264
00265         ELSE
00266
00267             WRITE(6, 54) "# ", i, evals(i), fdirac, "o"
00268             WRITE(24, 53) i, evals(i), fdirac, "o"
00269
00270         ENDIF
00271
00272     ENDDO
00273
00274 ELSE
00275
00276     DO i = 1, hdim
00277
00278         IF (evals(i) .LE. chempot) THEN
00279
00280             WRITE(6, 54) "# ", i, evals(i), one, "*"
00281             WRITE(24, 53) i, evals(i), one, "*"
00282
00283         ELSE
00284
00285             WRITE(6, 54) "# ", i, evals(i), zero, "o"
00286             WRITE(24, 53) i, evals(i), zero, "o"
00287
00288         ENDIF
00289
00290     ENDDO
00291
00292 ENDIF
00293
00294 WRITE(6, ' ("")')
00295 WRITE(6, ' ("# Band width = ", F12.6)') evals(hdim) - evals(1)
00296 WRITE(6, ' ("")')
00297 WRITE(24, ' ("")')
00298 WRITE(24, ' ("Band width = ", F12.6)') evals(hdim) - evals(1)
00299 WRITE(24, ' ("")')
00300
00301 WRITE(6, ' ("")')
00302 WRITE(6, ' ("# HOMO-LUMO = ", F12.6)') egap
00303 WRITE(6, ' ("")')
00304 WRITE(24, ' ("")')
00305 WRITE(24, ' ("HOMO-LUMO = ", F12.6)') egap
00306 WRITE(24, ' ("")')
00307
00308
00309 ELSE
00310
00311     ! Non-orthogonal basis - look at eigenvalues of H and XHX
00312
00313     WRITE(6, ' ("")')
00314     WRITE(6, ' ("# Eigenvalues XHX")')
00315     WRITE(24, ' ("")')
00316     WRITE(24, ' ("Eigenvalues XHX")')
00317
00318     IF (kbt .GT. 0.00000001) THEN
00319
00320         DO i = 1, hdim
00321
00322             fdirac = one/(one+exp((evals(i) - chempot)/
kbt))
00323
00324             IF (evals(i) .LE. chempot) THEN
00325
00326                 WRITE(6, 54) "# ", i, evals(i), fdirac, "*"
00327                 WRITE(24, 53) i, evals(i), fdirac, "*"
00328
00329             ELSE
00330
00331                 WRITE(6, 54) "# ", i, evals(i), fdirac, "o"
00332                 WRITE(24, 53) i, evals(i), fdirac, "o"
00333
00334             ENDIF
00335
00336         ENDDO
00337
00338         !
00339         ! WRITE(6, ' ("Eigenvectors of XHX")')
00340         ! WRITE(24, ' ("Eigenvectors of XHX")')
00341         ! Eigenvectors too

```

```

00342          !              DO I = 1, HDIM
00343
00344          !              WRITE(6,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00345          !              WRITE(24,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00346
00347          !              ENDDO
00348
00349      ELSE
00350
00351          DO i = 1, hdim
00352
00353              IF (evals(i) .LE. chempot) THEN
00354
00355                  WRITE(6, 54) "# ", i, evals(i), one, "*"
00356                  WRITE(24, 53) i, evals(i), one, "*"
00357
00358              ELSE
00359
00360                  WRITE(6, 54) "# ", i, evals(i), zero, "o"
00361                  WRITE(24, 53) i, evals(i), zero, "o"
00362
00363              ENDIF
00364
00365          ENDDO
00366
00367          !              WRITE(6,'("Eigenvectors of XHX")')
00368          !              WRITE(24,'("Eigenvectors of XHX")')
00369          ! Eigenvectors too
00370
00371          !              DO I = 1, HDIM
00372
00373          !              WRITE(6,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00374          !              WRITE(24,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00375
00376          !              ENDDO
00377
00378      ENDIF
00379
00380      orthoh = h
00381
00382      CALL diagmyh
00383
00384      WRITE(6,'(" ")')
00385      WRITE(6,'("# Eigenvalues H")')
00386      WRITE(24,'(" ")')
00387      WRITE(24,'("Eigenvalues H")')
00388
00389      IF (kbt .GT. 0.00000001) THEN
00390
00391          DO i = 1, hdim
00392
00393              fdirac = one/(one+exp((evals(i) - chempot)/
kbt))
00394
00395              IF (evals(i) .LE. chempot) THEN
00396
00397                  WRITE(6, 54) "# ", i, evals(i), fdirac, "*"
00398                  WRITE(24, 53) i, evals(i), fdirac, "*"
00399
00400              ELSE
00401
00402                  WRITE(6, 54) "# ", i, evals(i), fdirac, "o"
00403                  WRITE(24, 53) i, evals(i), fdirac, "o"
00404
00405              ENDIF
00406
00407          ENDDO
00408
00409          !              WRITE(6,'("Eigenvectors of H")')
00410          !              WRITE(24,'("Eigenvectors of H")')
00411          ! Eigenvectors too
00412
00413          !              DO I = 1, HDIM
00414
00415          !              WRITE(6,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00416          !              WRITE(24,'(100F12.6)') (EVECS(I,J), J = 1, HDIM)
00417
00418          !              ENDDO
00419
00420      ELSE
00421
00422          DO i = 1, hdim
00423
00424              IF (evals(i) .LE. chempot) THEN
00425
00426

```

```

00427             WRITE(6, 54) "# ", i, evals(i), one, "*"
00428             WRITE(24, 53) i, evals(i), one, "*"
00429
00430             ELSE
00431
00432             WRITE(6, 54) "# ", i, evals(i), zero, "o"
00433             WRITE(24, 53) i, evals(i), zero, "o"
00434
00435             ENDIF
00436
00437             ENDDO
00438
00439             !             WRITE(6, '("Eigenvectors of H")')
00440             !             WRITE(24, '("Eigenvectors of H")')
00441             ! Eigenvectors too
00442
00443             !             DO I = 1, HDIM
00444
00445             !             WRITE(6, ' (100F12.6)') (EVECS(I,J), J = 1, HDIM)
00446             !             WRITE(24, ' (100F12.6)') (EVECS(I,J), J = 1, HDIM)
00447
00448             !             ENDDO
00449
00450
00451             ENDIF
00452
00453             ENDIF
00454
00455             !             CALL GENDIAG
00456
00457
00458             CALL deallocateddiag
00459
00460             ENDIF
00461
00462 53             FORMAT(i6, 2x, f14.8, 1x, g18.12, 1x, a1)
00463 54             FORMAT(a2, i6, 2x, f14.8, 1x, g18.12, 1x, a1)
00464
00465             !             CALL DEALLOCATEDIAG
00466
00467             ELSEIF (spinon .EQ. 1 .AND. control .EQ. 1) THEN
00468
00469             WRITE(6, ' ("")')
00470             WRITE(6, ' ("# Eigenvalues")')
00471             WRITE(6, ' ("# Up : Down")')
00472             WRITE(24, ' ("")')
00473             WRITE(24, ' ("Eigenvalues")')
00474             WRITE(24, ' (" Up : Down")')
00475             DO i = 1, hdim
00476                 WRITE(6, 152) "# ", i, upevals(i), downevals(i)
00477                 WRITE(24, 52) i, upevals(i), downevals(i)
00478             ENDDO
00479
00480 52             FORMAT(i6, 2x, f14.8, 4x, f14.8)
00481 152            FORMAT(a2, i6, 2x, f14.8, 4x, f14.8)
00482
00483             CALL deallocateddiag
00484
00485             ENDIF
00486
00487             ENDIF
00488             ! IF (ELECTRO .EQ. 1) THEN
00489
00490             WRITE(6, ' ("# Partial charges")')
00491             WRITE(6, ' ("# Atom Type Charge (e)")')
00492             WRITE(24, ' ("Partial charges")')
00493             WRITE(24, ' (" Atom Type Charge (e)")')
00494             sumq = zero
00495             DO i = 1, nats
00496                 WRITE(6, 150) "# ", i, atele(i), -deltaq(i)
00497                 WRITE(24, 50) i, atele(i), -deltaq(i)
00498                 sumq = sumq + deltaq(i)
00499             ENDDO
00500
00501             WRITE(6, ' ("# Mulliken occupancies")')
00502             WRITE(6, ' ("# Atom Type Free atom Self-consistent")')
00503             WRITE(24, ' ("Mulliken occupancies")')
00504             WRITE(24, ' (" Atom Type Free atom Self-consistent")')
00505
00506             DO i = 1, nats
00507                 WRITE(6, 155) "# ", i, atele(i), atocc(elempointer(i)),
mycharge(i)
00508                 WRITE(24, 55) i, atele(i), atocc(elempointer(i)),
mycharge(i)
00509             ENDDO
00510
00511

```



```

00512 WRITE(6,'("# Sum of partial charges =", G16.8)') sumq
00513 WRITE(24,'(" Sum of partial charges =", G16.8)') sumq
00514
00515 50 FORMAT(i6, 4x, a2, 2x, f11.8)
00516 150 FORMAT(a2, i6, 4x, a2, 2x, f11.8)
00517
00518 55 FORMAT(i6, 4x, a2, 7x, f12.6, 5x, f12.6)
00519 155 FORMAT(a2, i6, 4x, a2, 7x, f12.6, 5x, f12.6)
00520
00521 ! ENDDO
00522
00523 IF (spinon .EQ. 1) THEN
00524
00525     WRITE(6,'("#")')
00526     WRITE(6,'("# Orbital spin densities")')
00527     WRITE(6,'("# Orbital index      Spin density")')
00528     WRITE(24,'("#")')
00529     WRITE(24,'("Orbital spin densities")')
00530     WRITE(24,'("Orbital index      Spin density")')
00531     DO i = 1, deltadim
00532         WRITE(6,151) "# ", i, deltaspin(i)
00533         WRITE(24,51) i, deltaspin(i)
00534     ENDDO
00535
00536     WRITE(6,'("#")')
00537     WRITE(6,'("# Per atom spin densities")')
00538     WRITE(6,'("# Atom Type      Spin density")')
00539     WRITE(24,'("#")')
00540     WRITE(24,'("# Per atom spin densities")')
00541     WRITE(24,'("# Atom Type      Spin density")')
00542
00543     index = 0
00544     DO i = 1, nats
00545
00546         SELECT CASE(basis(elempointer(i)))
00547
00548             CASE("s")
00549
00550                 numorb = 1
00551
00552             CASE("p")
00553
00554                 numorb = 1
00555
00556             CASE("d")
00557
00558                 numorb = 1
00559
00560             CASE("f")
00561
00562                 numorb = 1
00563
00564             CASE("sp")
00565
00566                 numorb = 2
00567
00568             CASE("sd")
00569
00570                 numorb = 2
00571
00572             CASE("sf")
00573
00574                 numorb = 2
00575
00576             CASE("pd")
00577
00578                 numorb = 2
00579
00580             CASE("pf")
00581
00582                 numorb = 2
00583
00584             CASE("df")
00585
00586                 numorb = 2
00587
00588             CASE("spd")
00589
00590                 numorb = 3
00591
00592             CASE("spf")
00593
00594                 numorb = 3
00595
00596             CASE("sdf")
00597
00598                 numorb = 3

```

```

00599
00600      CASE("pdf")
00601
00602          numorb = 3
00603
00604      CASE("spdf")
00605
00606          numorb = 4
00607
00608      END SELECT
00609
00610      atomspin = zero
00611      DO j = 1, numorb
00612          index = index + 1
00613          atomspin = atomspin + deltaspin(index)
00614      ENDDO
00615
00616      WRITE(6,156) "# ", i, atele(i), atomspin
00617      WRITE(24,56) i, atele(i), atomspin
00618
00619
00620      ENDDO
00621
00622
00623 156  FORMAT(a2,1x,i6,3x,a2,6x,f12.6)
00624 56   FORMAT(i6,1x,a2,1x,f12.6)
00625
00626
00627
00628 51   FORMAT(i6, 16x,f14.8)
00629 151  FORMAT(a2, i6, 16x,f14.8)
00630
00631      ENDIF
00632
00633      CLOSE(24)
00634
00635      ! CALL WRTCFGS(-999)
00636
00637
00638      RETURN
00639
00640 END SUBROUTINE summary

```

8.421 tabtest.f90 File Reference

Functions/Subroutines

- subroutine [tabtest](#)

8.421.1 Function/Subroutine Documentation

8.421.1.1 subroutine tabtest ()

Definition at line 23 of file [tabtest.f90](#).

8.422 tabtest.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !

```

```

00012 !
00013 ! Additionally, this program is free software; you can redistribute it
00014 ! and/or modify it under the terms of the GNU General Public License as
00015 ! published by the Free Software Foundation; version 2.0 of the License.
00016 ! Accordingly, this program is distributed in the hope that it will be
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
00019 ! Public License for more details.
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE tabtest
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE neblastarray
00028   USE virialarray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, NEWJ, J, K, PPSEL, BREAKLOOP
00034   INTEGER :: PBCI, PBCJ, PBCK, COUNT
00035   INTEGER :: KLO, KHI
00036   REAL(LATTEPREC) :: RCUT2, RCUT, MAGR, MAGR2
00037   REAL(LATTEPREC) :: FORCE(3), DC(3), RIJ(3)
00038   REAL(LATTEPREC) :: GRAD, TMPE
00039   REAL(LATTEPREC) :: A, B, DX
00040   IF (existerror) RETURN
00041
00042   OPEN(unit=47, status="UNKNOWN", file="tabtest.dat")
00043
00044   DO i = 1, 1001
00045
00046     magr = one + one*REAL(i-1)/1000
00047
00048     klo = 1
00049     khi = pptablenth(1)
00050
00051     DO WHILE (khi - klo .GT. 1)
00052
00053       k = (khi + klo)/2
00054
00055       IF (ppr(k,1) .GT. magr) THEN
00056         khi = k
00057       ELSE
00058         klo = k
00059       ENDIF
00060
00061     ENDDO
00062
00063     dx = ppr(khi,1) - ppr(klo,1)
00064
00065     a = (ppr(khi, 1) - magr)/dx
00066     b = (magr - ppr(klo, 1))/dx
00067
00068     tmpe = a*ppval(klo,1) + b*ppval(khi, 1) + &
00069           ((a*a*a - a)*ppspl(klo,1) + &
00070            (b*b*b - b)*ppspl(khi,1))*(dx/dx/six)
00071
00072     erep = erep + tmpe
00073
00074     grad = (ppval(khi,1) - ppval(klo,1))/dx + &
00075            ((one - three*a*a)*ppspl(klo,1) + &
00076             (three*b*b - one)*ppspl(khi,1))*(dx/six)
00077
00078     WRITE(47,*) magr, tmpe, grad
00079
00080   ENDDO
00081
00082   CLOSE(47)
00083
00084   RETURN
00085
00086 END SUBROUTINE tabtest
00087

```

8.423 tbmd.f90 File Reference

Functions/Subroutines

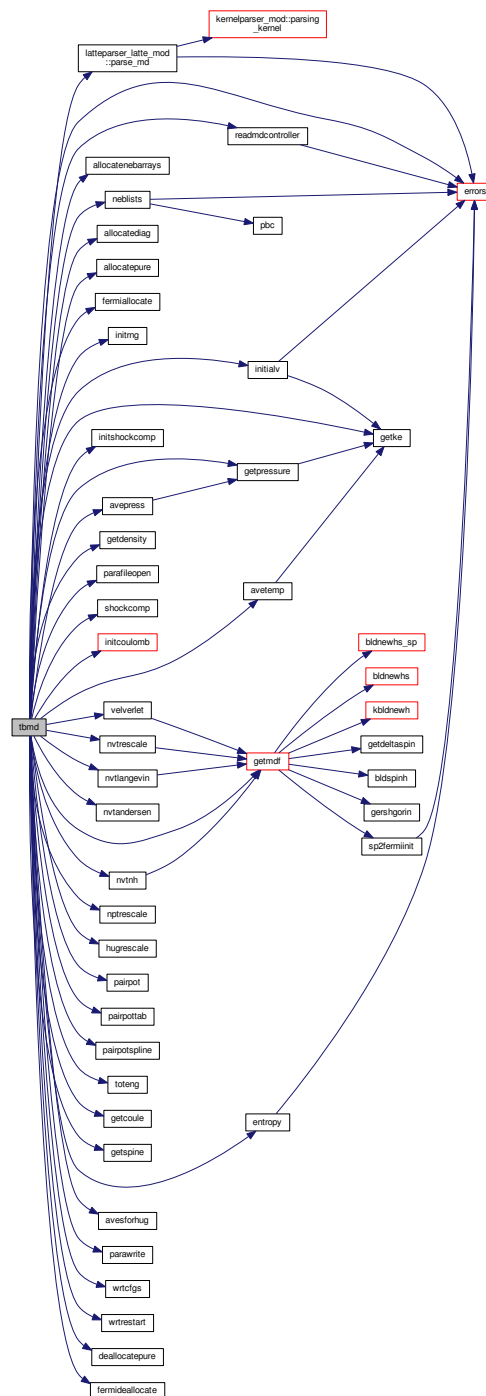
- subroutine [tbmd](#)

8.423.1 Function/Subroutine Documentation

8.423.1.1 subroutine tbmd ()

Definition at line 23 of file [tbmd.f90](#).

Here is the call graph for this function:



```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE

```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                            !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE tbmd
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE ppotarray
00027   USE mdarray
00028   USE neblistarray
00029   USE coulombarray
00030   USE spinarray
00031   USE virialarray
00032   USE nonoarray
00033   USE myprecision
00034   USE latteparser_latte_mod
00035
00036   IMPLICIT NONE
00037
00038   INTEGER :: I
00039   INTEGER :: ITER
00040   INTEGER :: CURRITER, TOTSCF
00041   INTEGER :: START_CLOCK, STOP_CLOCK, CLOCK_RATE, CLOCK_MAX
00042   REAL(LATTEPREC) :: THETIME, NEWESPIN, NEWECOUL
00043   REAL(LATTEPREC) :: RN, MYVOL
00044   INTEGER :: FLAGAND
00045
00046   IF (existerror) RETURN
00047
00048   !
00049   ! Read MDcontroller to determine what kind of MD simulation to do
00050   !
00051   IF (latteinexists) THEN
00052     CALL parse_md("latte.in")
00053   ELSE
00054     CALL readmdcontroller
00055   ENDIF
00056   !
00057   ! Allocate stuff for building the neighbor lists, then build them
00058   !
00059
00060   CALL allocatenebararrays
00061
00062   CALL neblists(0)
00063
00064   !
00065   ! Allocate things depending on which method we're using
00066   ! to get the bond-order
00067   !
00068
00069   IF (control .EQ. 1) THEN
00070     CALL allocatediag
00071   ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00072     CALL allocatepure
00073   ELSEIF (control .EQ. 3) THEN
00074     CALL fermiallocate
00075   ENDIF
00076
00077   !
00078   ! Set the seed for the random number generator
00079   ! Applied to initializing the velocities and the
00080   ! Langevin and Andersen thermostats.
00081   !
00082
00083   IF (seedinit .EQ. "RANDOM") CALL initrng
00084
00085
00086   IF (restart .EQ. 0) THEN
00087
00088     ALLOCATE (v(3,nats))
00089
00090     !
00091     ! Initialize velocities if TOINITTEMP = 1
00092     !
00093

```

```

00094      IF (toinittemp .EQ. 1) THEN
00095          CALL initialv
00096      ELSE
00097          WRITE(6,*) "Caution: you haven't initialized velocities"
00098      ENDIF
00099
00100  ENDIF
00101
00102      !
00103      ! If we're going to run with the Hugoniosat, some things need
00104      ! to be initialized
00105      !
00106
00107      IF (shockon .EQ. 1) CALL initshockcomp
00108
00109
00110      !
00111      ! Get forces - we need these at this point only if we're running
00112      ! NVE MD with the velocity verlet algorithm
00113      !
00114
00115      curriter = 0
00116
00117      IF (restart .EQ. 0) THEN
00118
00119          CALL getmdf(0,1)
00120
00121      ELSEIF (restart .EQ. 1) THEN
00122
00123          !
00124          ! If we've read from a restart file then we don't need to run
00125          ! qconsistency to full self-consistency at the first time step of
00126          ! the new run - we're already there...
00127          !
00128
00129          ! Yes we do until this is fixed
00130
00131          CALL getmdf(0,1)
00132
00133      ENDIF
00134
00135      IF (restart .EQ. 0) THEN
00136          iter = -1
00137      ELSE
00138          iter = contiter - 1
00139      ENDIF
00140
00141      IF (gethug .EQ. 1 .AND. nvton .NE. 0) THEN
00142          CALL errors("tbmd",.NE."Please don't have GETHUG = 1 and NVTON 0")
00143      ENDIF
00144
00145      IF (gethug .EQ. 1 .AND. npton .NE. 0) THEN
00146          CALL errors("tbmd",.NE."Please don't have GETHUG = 1 and NPTON 0")
00147      ENDIF
00148
00149      IF (nvton .EQ. 1 .OR. npton .EQ. 1) THEN
00150
00151          ALLOCATE(thist(aveper))
00152
00153          thist = ttargget
00154
00155      ENDIF
00156
00157      IF (npton .EQ. 1) THEN
00158
00159          CALL getpressure
00160
00161          IF (npttype .EQ. "ISO") THEN
00162
00163              ! Change box dims assuming a hydrostatic pressure
00164
00165              ALLOCATE( phist(aveper) )
00166              phist = pressure
00167
00168          ELSE ! Allow each box vector to change independently
00169
00170              ALLOCATE (phistx(aveper), phisty(aveper), phistz(
00171  aveper))
00172              phistx = pressure
00173              phisty = pressure
00174              phistz = pressure
00175
00176          ENDIF
00177
00178          CALL getdensity
00179

```

```

00180     ENDIF
00181
00182     IF (gethug .EQ. 1) THEN
00183
00184         ! Allocate the arrays for the averages
00185
00186         CALL getpressure
00187         CALL getke
00188
00189         myvol = abs(box(1,1)*(box(2,2)*box(3,3) - box(3,2)*box(2,3)) + &
00190             box(1,2)*(box(2,1)*box(3,3) - box(3,1)*box(2,3)) + &
00191             box(1,3)*(box(2,1)*box(3,2) - box(3,1)*box(2,2)))
00192
00193         ALLOCATE(phist(aveper/wrtfreq), thist(aveper/
00194 wrtfreq), ehist(aveper/wrtfreq))
00195         ALLOCATE(vhist(aveper/wrtfreq))
00196
00197         phist = pressure
00198         thist = temperature
00199         ehist = e0
00200         vhist = myvol
00201
00202     ENDIF
00203
00204     IF (parrep .EQ. 1) CALL parafilename
00205
00206     cumdt = zero
00207
00208     totscf = 0
00209
00210     ! OPEN(UNIT=30, STATUS="UNKNOWN", FILE="AMD_gap_Ef.dat")
00211
00212     WRITE(6,17) "#","Time (ps)", "Free energy (eV)", "T (K)", "Pressure (GPa)"
00213
00214 17 FORMAT(a1, 2x, a10, 6x, a16, 2x, a5, 3x, a14)
00215
00216     ! CALL SYSTEM_CLOCK(START_CLOCK, CLOCK_RATE, CLOCK_MAX)
00217
00218     curriter = 0
00219
00220     DO WHILE (curriter .LE. maxiter)
00221
00222         totscf = totscf + scfs_ii
00223
00224         curriter = curriter + 1
00225
00226         iter = iter + 1
00227         entropyiter = iter
00228
00229         IF (shockon .EQ. 1 .AND. iter .GE. shockstart .AND. &
00230             iter .LT. shockstop) THEN
00231
00232             CALL shockcomp
00233
00234             !
00235             ! Since we're changing the dimensions of the box, we
00236             ! should probably also adjust the reciprocal lattice vectors
00237             ! used in the Ewald sum. There's no harm in doing this
00238             ! every time step while the box size is changing
00239             !
00240             ! We may as well reinitialize the Coulomb stuff since
00241             ! it also optimizes the real-space cut-off based on the volume
00242             ! of the cell
00243             !
00244
00245             CALL initcoulomb
00246
00247         ENDIF
00248
00249         IF ((nvton .EQ. 0 .AND. npton .EQ. 0 .AND. gethug .EQ. 0) .OR. curriter .GT.
00250 thermrun) THEN
00251
00252             CALL velverlet(curriter)
00253
00254         ELSEIF (nvton .EQ. 1 .AND. curriter .LE. thermrun) THEN
00255
00256             ! Velocity rescaling thermostat
00257
00258             !
00259             ! To smooth things out, let's average the
00260             ! temperature over the previous AVEPER time steps
00261             !
00262
00263             CALL avetemp
00264
00265             IF (mod(iter, thermper) .EQ. 0 .AND. curriter .GT. 1) THEN

```



```

00265
00266         CALL nvtrescale
00267
00268     ELSE
00269
00270         CALL velverlet(curriter)
00271
00272     ENDIF
00273
00274     ELSEIF (nvton .EQ. 2 .AND. curriter .LE. thermrun) THEN
00275
00276         ! Langevin thermostat
00277
00278         CALL nvtlangevin(iter)
00279
00280     ELSEIF (nvton .EQ. 3 .AND. curriter .LE. thermrun) THEN
00281
00282         ! Andersen thermostat
00283
00284
00285         cumdt = cumdt + dt
00286
00287         CALL velverlet(curriter)
00288
00289         CALL nvtandersen
00290
00291     ELSEIF (nvton .EQ. 4 .AND. curriter .LE. thermrun) THEN
00292
00293         ! NOSE thermostat
00294
00295         CALL nvtanh
00296
00297     ELSEIF (npton .EQ. 1 .AND. curriter .LE. thermrun) THEN
00298
00299         ! Velocity rescaling thermostat
00300
00301         !
00302         ! To smooth things out, let's average the
00303         ! temperature over the previous AVEPER time steps
00304         !
00305
00306         CALL nvtlangevin(iter)
00307
00308         CALL avepress
00309
00310         IF (mod(iter, thermper) .EQ. 0 .AND. curriter .GT. 1) THEN
00311
00312             CALL nptrescale
00313             CALL getdensity
00314             CALL initcoulomb
00315
00316         ENDIF
00317
00318
00319     ELSEIF (gethug .EQ. 1 .AND. curriter .LE. thermrun) THEN
00320
00321         CALL nvtlangevin(iter)
00322
00323         IF (mod(iter, thermper) .EQ. 0 .AND. curriter .GT. 1) THEN
00324
00325             CALL hugrescale
00326             CALL getdensity
00327             CALL initcoulomb
00328
00329         ENDIF
00330
00331     ENDIF
00332
00333     IF (mod(iter,wrtfreq) .EQ. 0) THEN
00334
00335         ! The non-orthogonal density matrix is obtained from
00336         ! GETMDF (when the Pulay force is computed)
00337
00338         IF (ppoton .EQ. 1) THEN
00339             CALL pairpot
00340         ELSEIF (ppoton .EQ. 2) THEN
00341             CALL pairpottab
00342         ELSEIF (ppoton .EQ. 3) THEN
00343             CALL pairpotspline
00344         ENDIF
00345
00346         CALL getke
00347
00348         CALL toteng
00349
00350         ! For the 0 SCF MD the coulomb energy is calculated in GETMDF
00351

```

```

00352      IF (qiter .NE. 0) THEN
00353          ecoul = zero
00354          IF (electro .EQ. 1) CALL getcoule
00355      ENDIF
00356
00357      espin = zero
00358      IF (spinon .EQ. 1) CALL getspine
00359
00360      CALL getpressure
00361
00362      IF (control .NE. 1 .AND. control .NE. 2 .AND. kbt .GT. 0.000001 ) THEN
00363
00364          ! Only required when using the recursive expansion of the Fermi operator
00365
00366          ! 2/26/13
00367          ! The entropy is now calculated when we get the density
00368          ! matrix in the spin polarized case with diagonalization,
00369          ! as it should be...
00370
00371          CALL entropy
00372
00373      ENDIF
00374
00375      tote = trrhoh + erep + kee - ente - ecoul +
00376      espin
00377      !      write(*,*)"Ekin", KEE
00378      !      write(*,*)"Epot", TRRH OH + EREP - ENTE - ECOUL + ESPIN
00379      !      write(*,*)"components",TRRH OH, EREP, ENTE, ECOUL
00380
00381      IF (gethug .EQ. 1) CALL avesforhug(pressure,
00382      tote, temperature, sysvol)
00383
00384      IF (abs(tote) .GT. 10000000000.0) THEN
00385          CALL errors("tbmd","The calculation has diverged - check SCF parameters")
00386      ENDIF
00387
00388      !      write(70,*) ITER, CR(1,1)
00389
00389      thetime = REAL(iter)*DT/THOUSAND
00390
00391      IF (parrep .EQ. 0) THEN
00392
00393          IF (npton .EQ. 0 .AND. gethug .EQ. 0 .AND. nvton .NE. 0) THEN
00394
00395              IF (nvton .NE. 4) THEN
00396
00397                  WRITE(6,99)"Data", thetime, tote, temperature,
00398      pressure, egap, &
00399                      chempot !, TRRH OH, EREP, KEE, ECOUL, REAL(NUMSCF)
00400
00401              ELSE
00402
00402                  ! Special case for Nose Hoover
00403                  WRITE(6,99)"Data", thetime, tote, tote+consmot,
00404      temperature, pressure, &
00405                      egap, chempot !, STRTEN(1), STRTEN(2), STRTEN(3), STRTEN(4), &
00406                      !STRTEN(5), STRTEN(6)
00407
00408              ENDIF
00409
00410          ENDIF
00411
00412          IF (nvton .EQ. 0 .AND. npton .EQ. 0 .AND. gethug .EQ. 0) THEN
00413
00414              WRITE(6,99)"Data", thetime, tote, temperature,
00415      pressure, egap, &
00416                      chempot
00417
00418          ENDIF
00419
00420          IF (npton .NE. 0 .AND. nvton .EQ. 0 .AND. gethug .EQ. 0) THEN
00421
00422              WRITE(6,99)"Data", thetime, tote, temperature,
00423      pressure, &
00424                      egap, massden, box(1,1), box(2,2), box(3,3), &
00425                      sysvol
00426
00427          ENDIF
00428
00429          IF (gethug .EQ. 1 .AND. nvton .EQ. 0 .AND. npton .EQ. 0) THEN
00430
00431              WRITE(6,99)"Data", thetime, tote, temperature,
00432      pressure, &
00433                      egap, massden, hg, ttarget, box(1,1),
00434      box(2,2), box(3,3), &
00435                      sysvol

```

```

00431
00432         ENDIF
00433
00434         FLUSH(6)
00435
00436         ELSE
00437
00438             CALL parawrite(thetime)
00439
00440         ENDIF
00441
00442
00443 99         FORMAT(a4,20g18.9)
00444
00445 16         FORMAT(f12.5, f20.8, 1x, f9.1, 1x, f12.3, 1x, g18.9, 1x, g18.9)
00446
00447         ENDIF
00448
00449         IF (mod(iter, dumpfreq) .EQ. 0) CALL wrtcfgs(iter)
00450
00451         IF (mod(iter, rsfreq) .EQ. 0) CALL wrtrestart(iter)
00452
00453         IF (mod(iter, udneigh) .EQ. 0) CALL neblists(1)
00454
00455
00456     ENDDO
00457
00458     ! CALL SYSTEM_CLOCK(STOP_CLOCK, CLOCK_RATE, CLOCK_MAX)
00459     ! PRINT*, HDIM, FLOAT(STOP_CLOCK - START_CLOCK)/FLOAT(MAXITER*CLOCK_RATE)
00460
00461     IF (control .EQ. 1) THEN
00462         ! CALL DEALLOCATEDIAG
00463     ELSEIF (control .EQ. 2 .OR. control .EQ. 4 .OR. control .EQ. 5) THEN
00464         CALL deallocatepure
00465     ELSEIF (control .EQ. 3) THEN
00466         CALL fermideallocate
00467     ENDIF
00468
00469     IF (nvton .EQ. 1) DEALLOCATE(thist)
00470     IF (npton .EQ. 1) THEN
00471         IF (npttype .EQ. "ISO") THEN
00472             DEALLOCATE(thist, phist)
00473         ELSE
00474             DEALLOCATE(thist, phistx, phisty, phistz)
00475         ENDIF
00476     ENDIF
00477
00478     RETURN
00479
00480 END SUBROUTINE tbmd

```

8.425 timer_mod.f90 File Reference

Modules

- module [timer_mod](#)

Functions/Subroutines

- integer function [timer_mod::init_timer](#) ()
- integer function [timer_mod::shutdown_timer](#) ()
- integer function [timer_mod::start_timer](#) (ITIMER)
- integer function [timer_mod::stop_timer](#) (ITIMER)
- real(8) function [timer_mod::time_mls](#) ()
- subroutine [timer_mod::timedate_tag](#) (TAG)
- integer function [timer_mod::timer_results](#) ()

Variables

- integer `timer_mod::dense2sparse_timer`
- integer `timer_mod::dmbuild_timer`
- integer `timer_mod::latte_timer`
- integer `timer_mod::num_timers`
- integer `timer_mod::sp2all_timer`
- integer `timer_mod::sp2sparse_timer`
- integer `timer_mod::sparse2dense_timer`
- real, dimension(:), allocatable `timer_mod::tavg`
- integer `timer_mod::tclock_max`
- integer `timer_mod::tclock_rate`
- integer, dimension(:), allocatable `timer_mod::tcount`
- character(len=20), dimension(:), allocatable `timer_mod::tname`
- real, dimension(:), allocatable `timer_mod::tpercent`
- integer, dimension(:), allocatable `timer_mod::tstart`
- integer `timer_mod::tstart_clock`
- integer `timer_mod::tstop_clock`
- real, dimension(:), allocatable `timer_mod::tsum`
- integer, dimension(:), allocatable `timer_mod::ttotal`
- integer `timer_mod::tx`

8.426 timer_mod.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was  !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National  !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE  !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,  !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS  !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such  !
00010 ! modified software should be clearly marked, so as not to confuse it  !
00011 ! with the version available from LANL.  !
00012 !  !
00013 ! Additionally, this program is free software; you can redistribute it  !
00014 ! and/or modify it under the terms of the GNU General Public License as  !
00015 ! published by the Free Software Foundation; version 2.0 of the License.  !
00016 ! Accordingly, this program is distributed in the hope that it will be  !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of  !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.  !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE timer_mod
00023
00024   IMPLICIT NONE
00025   SAVE
00026
00027   INTEGER :: tstart_clock, tstop_clock, tclock_rate,
00028             tclock_max
00029   INTEGER :: tx, num_timers
00030   INTEGER :: latte_timer, dense2sparse_timer,
00031             dmbuild_timer, sparse2dense_timer
00032   INTEGER :: sp2all_timer, sp2sparse_timer
00033   INTEGER, ALLOCATABLE :: tstart(:), ttotal(:), tcount(:)
00034   REAL, ALLOCATABLE :: tavg(:), tsum(:), tpercent(:)
00035   CHARACTER(LEN=20), ALLOCATABLE :: tname(:)
00036
00037   CONTAINS
00038
00039   ! Initialize timers
00040   !
00041   FUNCTION init_timer()
00042
00043     INTEGER :: I, INIT_TIMER
00044
00045     num_timers = 6
00046
00047   END FUNCTION init_timer

```

```

00045     IF (.NOT.ALLOCATED(tstart)) ALLOCATE(tstart(num_timers),
ttotal(num_timers), tcount(num_timers))
00046     IF (.NOT.ALLOCATED(tname)) ALLOCATE(tname(num_timers))
00047     IF (.NOT.ALLOCATED(tavg)) THEN
00048         ALLOCATE(tavg(num_timers), tsum(num_timers),
tpercent(num_timers))
00049     ENDIF
00050
00051     ! Timer handles, names, and counters
00052     latte_timer = 1
00053     sp2all_timer = 2
00054     sp2sparse_timer = 3
00055     dense2sparse_timer = 4
00056     dmbuild_timer = 5
00057     sparse2dense_timer = 6
00058
00059     tname(latte_timer) = "LATTE"
00060     tname(sp2all_timer) = "Sp2All"
00061     tname(sp2sparse_timer) = " Sp2Sparse"
00062     tname(dense2sparse_timer) = " Dense2Sparse"
00063     tname(dmbuild_timer) = " DMBuild"
00064     tname(sparse2dense_timer) = " Sparse2Dense"
00065
00066     tttotal = 0
00067     tcount = 0
00068
00069     init_timer = num_timers
00070
00071     END FUNCTION init_timer
00072
00073     ! Done with timers
00074     !
00075     FUNCTION shutdown_timer()
00076
00077     INTEGER :: SHUTDOWN_TIMER
00078
00079     DEALLOCATE(tstart, tttotal, tcount)
00080     DEALLOCATE(tname)
00081
00082     shutdown_timer = num_timers
00083
00084     END FUNCTION shutdown_timer
00085
00086     ! Start Timing
00087     !
00088     FUNCTION start_timer(ITIMER)
00089
00090     INTEGER :: ITIMER, START_TIMER
00091
00092     CALL system_clock(tstart_clock, tclock_rate,
tclock_max)
00093     tstart(itimer) = tstart_clock
00094
00095     start_timer = tstart_clock
00096
00097     END FUNCTION start_timer
00098
00099     ! Stop timing
00100     !
00101     FUNCTION stop_timer(ITIMER)
00102
00103     INTEGER :: ITIMER, TDELTA, STOP_TIMER
00104
00105     CALL system_clock(tstop_clock, tclock_rate, tclock_max)
00106     tdelta = tstop_clock - tstart(itimer)
00107     tttotal(itimer) = tttotal(itimer) + tdelta
00108     tcount(itimer) = tcount(itimer) + 1
00109
00110     stop_timer = tstop_clock
00111
00112     END FUNCTION stop_timer
00113
00114     ! Print performance results
00115     !
00116     FUNCTION timer_results()
00117
00118     INTEGER :: I, TIMER_RESULTS
00119
00120     print *, ""
00121     WRITE(6,*) "Timer                # Calls  Avg/Call (s)      Total (s)      % Time"
00122     print *, ""
00123
00124     DO i = 1, num_timers
00125
00126         IF (tcount(i) .GT. 0) THEN
00127
00128             tavg(i) = (float(tttotal(i))/float(tclock_rate))/float(

```

```

        tcount(i))
00129         tsum(i) = float(tttotal(i))/float(tclock_rate)
00130         tpercent(i) = (tsum(i) / tsum(1)) * 100.0
00131
00132         WRITE(6,10) tname(i), tcount(i), tavg(i), tsum(i),
tpercent(i)
00133 10      FORMAT(a25, i4, 3g16.6)
00134         ENDIF
00135
00136         ENDDO
00137
00138         timer_results = num_timers
00139
00140     END FUNCTION timer_results
00141
00142     ! Print a tag time and date
00143     !
00144     SUBROUTINE timedate_tag(TAG)
00145     CHARACTER(LEN=*) :: TAG
00146     INTEGER :: VALUES(8)
00147
00148     CALL date_and_time(values=values)
00149
00150     WRITE(*,' (A2,1X,A,1X,I2,A1,I2,2X,A2,1X,I2,A1,I2,A1,I4)') " #",trim(tag),&
00151         & values(5),":",values(6),&
00152         & "on",values(2),"/",values(3),"/",values(1)
00153
00154     END SUBROUTINE timedate_tag
00155
00156     ! Get the actual time in mls
00157     !
00158     FUNCTION time_mls()
00159     REAL(8) :: TIME_MLS
00160     INTEGER :: TIMEVECTOR(8)
00161
00162     time_mls = 0.0d0
00163     CALL date_and_time(values=timevector)
00164     time_mls=timevector(5)*60.0d0*60.0d0*1000.0d0 + timevector(6)*60.0d0*1000.0d0 &
00165         & + timevector(7)*1000.0d0 + timevector(8)
00166
00167     END FUNCTION time_mls
00168
00169
00170 END MODULE timer_mod

```

8.427 tlmmp.f90 File Reference

Functions/Subroutines

- real(latteprec) function [tlmmp](#) (L, M, MP, ALPHA, COSBETA)

8.427.1 Function/Subroutine Documentation

8.427.1.1 real(latteprec) function [tlmmp](#) (integer *L*, integer *M*, integer *MP*, real(latteprec) *ALPHA*, real(latteprec) *COSBETA*)

Definition at line 23 of file [tlmmp.f90](#).

8.428 tlmmp.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```

```

00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                   !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION tlmmp(L, M, MP, ALPHA, COSBETA)
00023
00024 ! build T function defined in eqn. (8) of PRB 72 165107
00025
00026 USE myprecision
00027
00028 IMPLICIT NONE
00029
00030 REAL(LATTEPREC) :: TLMMP, COSBETA, ALPHA
00031 REAL(LATTEPREC), EXTERNAL :: BM, WIGNERD
00032 INTEGER :: L, M, MP
00033
00034 IF (m == 0) THEN
00035     tlmmp = zero
00036 ELSE
00037     tlmmp = bm(m, alpha) * (REAL((-1)**MP, LATTEPREC) * &
00038         WIGNERD(l, abs(m), mp, cosbeta) - WIGNERD(l, abs(m), -mp, cosbeta))
00039 ENDIF
00040
00041 RETURN
00042
00043 END FUNCTION tlmmp

```

8.429 toteng.f90 File Reference

Functions/Subroutines

- subroutine [toteng](#)

8.429.1 Function/Subroutine Documentation

8.429.1.1 subroutine toteng ()

Definition at line 23 of file [toteng.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE toteng
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE spinarray
00027   USE nonoarray
00028   USE kspacearray
00029   USE myprecision
00030
00031   IMPLICIT NONE
00032
00033   INTEGER :: I, J, K
00034   COMPLEX(LATTEPREC) :: ZTRRHOB
00035   IF (existererror) RETURN
00036
00037   trrhob = zero
00038   ztrrhob = cmplx(zero,zero)
00039
00040   ! IF (ELECTRO .EQ. 1) THEN
00041
00042   IF (spinon .EQ. 0) THEN
00043
00044     IF (kon .EQ. 0) THEN
00045
00046       DO i = 1, hdim
00047         DO j = 1, hdim
00048
00049           trrhob = trrhob + bo(j,i)*h(j,i)
00050
00051         ENDDO
00052
00053         trrhob = trrhob - bozero(i)*h(i,i)
00054
00055       ENDDO
00056
00057     ELSE
00058
00059       DO k = 1, nktot
00060
00061         DO i = 1, hdim
00062           DO j = 1, hdim
00063
00064             ztrrhob = ztrrhob + kbo(j,i,k)*(hk(i,j,k))
00065
00066           ENDDO
00067
00068           ztrrhob = ztrrhob - cmplx(bozero(i))*hk(i,i,k)
00069
00070         ENDDO
00071
00072       ENDDO
00073
00074       trrhob = REAL(ztrrhob)/REAL(nktot)
00075
00076     ENDIF
00077
00078   ELSE
00079
00080     !
00081     ! This is everything: tr(rhoup - rhoupzero)*Hup +
00082     ! tr(rhdown - rhdownzero)*Hdown
00083     !
00084     ! Hup = H(Slater-Koster) + H(electrostatic) + H(spin)
00085     ! Hdown = H(Slater-Koster) + H(electrostatic) - H(spin)
00086     !
00087     ! Thus, we're calculating Covalent (from SK) + electrostatic (from H1) +
00088     ! spin (from H2) and we just need to add on the entropy and pairwise
00089     ! bits to get the total energy
00090     !
00091
00092     DO i = 1, hdim
00093       DO j = 1, hdim

```

```

00094
00095      trrhoh = trrhoh + (rhoup(j,i) + rhodown(j,i))*
00096      h(j,i)
00097      ENDDO
00098
00099      trrhoh = trrhoh - (rhoupzero(i) + rhodownzero(i))*
00100      h(i,i)
00101      ENDDO
00102
00103  ENDIF
00104
00105  ! ELSE
00106
00107  !   IF (KON .EQ. 0) THEN
00108
00109  !       DO I = 1, HDIM
00110  !           DO J = 1, HDIM!
00111  !
00112  !               TRRHOh = TRRHOh + BO(J,I)*H(J,I)
00113  !
00114  !           ENDDO
00115  !       ENDDO
00116
00117  !   ELSE
00118
00119  !       DO K = 1, NKTOT
00120
00121  !           DO I = 1, HDIM
00122  !               DO J = 1, HDIM
00123  !
00124  !                   ZTRRHOh = ZTRRHOh + KBO(J,I,K)*HK(I,J,K)
00125  !
00126  !               ENDDO
00127  !           ENDDO
00128
00129  !       ENDDO
00130
00131  !       TRRHOh = REAL(ZTRRHOh)/REAL(NKTOT)
00132  !
00133  !   ENDIF
00134
00135  ! ENDIF
00136
00137  ! Check for something bad happening
00138
00139
00140  RETURN
00141
00142 END SUBROUTINE toteng

```

8.431 univarray.f90 File Reference

Modules

- module [univarray](#)

Variables

- real(latteprec), dimension(:,,:), allocatable [univarray::bond](#)
- real(latteprec), dimension(:,,:), allocatable [univarray::overl](#)
- real(latteprec), dimension(:,,:), allocatable [univarray::pair](#)

8.432 univarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !

```

```

00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE univarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   REAL(LATTEPREC), ALLOCATABLE :: bond(:, :), overl(:, :), pair(:, :)
00030
00031 END MODULE univarray
00032
00033

```

8.433 univscaling.f90 File Reference

Functions/Subroutines

- subroutine [univscale_sub](#) (R, A, X)

8.433.1 Function/Subroutine Documentation

8.433.1.1 subroutine [univscale_sub](#) ([real\(latteprec\)](#) *R*, [real\(latteprec\)](#), dimension(14) *A*, [real\(latteprec\)](#) *X*)

Definition at line 38 of file [univscaling.f90](#).


```

00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 ! 1 = A0
00023 ! 2 = B1
00024 ! 3 = B2
00025 ! 4 = B3
00026 ! 5 = B4
00027 ! 6 = B5
00028 ! 7 = R1
00029 ! 8 = RCUT
00030 ! 9 = TAIL1
00031 ! 10 = TAIL2
00032 ! 11 = TAIL3
00033 ! 12 = TAIL4
00034 ! 13 = TAIL5
00035 ! 14 = TAIL6
00036
00037 SUBROUTINE univscale_sub(R, A, X)
00038
00039 USE myprecision
00040
00041 IMPLICIT NONE
00042
00043 REAL(LATTEPREC) :: A(14), X, R, RMINUSR1, POLYNOM, RMOD
00044
00045 IF (r .LE. a(7)) THEN
00046
00047     rmod = r - a(6)
00048
00049     polynom = rmod*(a(2) + rmod*(a(3) + rmod*(a(4) + a(5)*rmod)))
00050
00051     x = exp(polynom)
00052
00053 ELSEIF (r .GT. a(7) .AND. r .LT. a(8)) THEN
00054
00055     rminusr1 = r - a(7)
00056
00057     x = a(9) + rminusr1*(a(10) + &
00058         rminusr1*(a(11) + rminusr1*(a(12) + &
00059             rminusr1*(a(13) + rminusr1*a(14))))))
00060
00061 ELSE
00062
00063     x = zero
00064
00065 END IF
00066
00067 x = a(1)*x
00068
00069 RETURN
00070
00071 END SUBROUTINE univscale_sub

```

8.435 univscaling_function.f90 File Reference

Functions/Subroutines

- real(latteprec) function [univscale](#) (I, J, L1, L2, MP, R, WHICHINT)

8.435.1 Function/Subroutine Documentation

8.435.1.1 real(latteprec) function [univscale](#) (integer *I*, integer *J*, integer *L1*, integer *L2*, integer *MP*, real(latteprec) *R*, character(len=1) *WHICHINT*)

Definition at line 38 of file [univscaling_function.f90](#).

8.436 univscaling_function.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 ! 1 = A0
00023 ! 2 = B1
00024 ! 3 = B2
00025 ! 4 = B3
00026 ! 5 = B4
00027 ! 6 = B5
00028 ! 7 = R1
00029 ! 8 = RCUT
00030 ! 9 = TAIL1
00031 ! 10 = TAIL2
00032 ! 11 = TAIL3
00033 ! 12 = TAIL4
00034 ! 13 = TAIL5
00035 ! 14 = TAIL6
00036
00037 FUNCTION univscale(I, J, L1, L2, MP, R, WHICHINT)
00038
00039     USE constants_mod
00040     USE setuparray
00041     USE univarray
00042     USE myprecision
00043
00044     IMPLICIT NONE
00045
00046     INTEGER :: I, J, L1, L2, IP1, IP2, MP, IC
00047     INTEGER :: BREAKLOOP
00048     REAL(LATTEPREC) :: UNIVSCALE
00049     REAL(LATTEPREC) :: A(14), R, RMINUSR1, RMOD
00050     CHARACTER(LEN=1) :: WHICHINT
00051     CHARACTER(LEN=3) :: IGLTYPE
00052     IF (existerror) RETURN
00053
00054     ! can't test directly on L values because basis strings always list
00055     ! lower L values first
00056
00057     IF (l1 .GT. 12) THEN
00058         ip1 = 12
00059         ip2 = 11
00060     ELSE
00061         ip1 = 11
00062         ip2 = 12
00063     ENDIF
00064
00065     ! build basis string from L and M values - pure hackery
00066
00067     SELECT CASE (ip1)
00068     CASE (0)
00069         igltype = "s"
00070     CASE (1)
00071         igltype = "p"
00072     CASE (2)
00073         igltype = "d"
00074     CASE (3)
00075         igltype = "f"
00076     END SELECT
00077
00078     SELECT CASE (ip2)
00079     CASE (0)
00080         igltype = trim(igltype)//"s"
00081     CASE (1)
00082         igltype = trim(igltype)//"p"
00083     CASE (2)
00084         igltype = trim(igltype)//"d"

```

```

00085     CASE(3)
00086         igltype = trim(igltype)//"f"
00087     END SELECT
00088
00089     SELECT CASE (mp)
00090     CASE(0)
00091         igltype = trim(igltype)//"s"
00092     CASE(1)
00093         igltype = trim(igltype)//"p"
00094     CASE(2)
00095         igltype = trim(igltype)//"d"
00096     CASE(3)
00097         igltype = trim(igltype)//"f"
00098     END SELECT
00099
00100     ! It makes a difference if our atoms are of the species or not...
00101
00102     ! Easier case first ATELE(I) = ATELE(J)
00103
00104     IF (atele(i) .EQ. atele(j)) THEN
00105
00106         breakloop = 0
00107         ic = 0
00108         DO WHILE (breakloop .EQ. 0)
00109
00110             ic = ic + 1
00111             IF (atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic) .AND. &
00112                 igltype .EQ. btype(ic)) THEN
00113
00114                 ! Now we've ID'ed our bond integral
00115
00116                 SELECT CASE(whichint)
00117                 CASE("H") ! We're doing the H matrix build
00118                     a = bond(:,ic)
00119                 CASE("S") ! We're doing the S matrix build
00120                     a = overl(:,ic)
00121                 END SELECT
00122
00123                 breakloop = 1
00124
00125             ENDIF
00126         ENDDO
00127
00128     ELSE
00129
00130         ! Elements are different - care must be taken with p-s, s-p etc.
00131
00132         IF (l1 .EQ. l2) THEN ! This is a special case sss, pps, ppp etc.
00133
00134             breakloop = 0
00135             ic = 0
00136             DO WHILE (breakloop .EQ. 0)
00137
00138                 ic = ic + 1
00139                 IF (((atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic)) .OR. &
00140                     (atele(i) .EQ. ele2(ic) .AND. atele(j) .EQ. ele1(ic))) .AND. &
00141                     igltype .EQ. btype(ic)) THEN
00142
00143                     ! Now we've ID'ed our bond integral
00144
00145                     SELECT CASE(whichint)
00146                     CASE("H") ! We're doing the H matrix build
00147                         a = bond(:,ic)
00148                     CASE("S") ! We're doing the S matrix build
00149                         a = overl(:,ic)
00150                     END SELECT
00151
00152                     breakloop = 1
00153
00154                 ENDIF
00155             ENDDO
00156
00157         ELSE ! L1 .NE. L2
00158
00159             IF (l1 .LT. l2) THEN
00160
00161                 DO ic = 1, noint
00162
00163                     IF ((atele(i) .EQ. ele1(ic) .AND. atele(j) .EQ. ele2(ic)) .AND. &
00164                         igltype .EQ. btype(ic)) THEN
00165
00166                         ! Now we've ID'ed our bond integral
00167
00168                         SELECT CASE(whichint)
00169                         CASE("H") ! We're doing the H matrix build
00170                             a = bond(:,ic)
00171                         CASE("S") ! We're doing the S matrix build
00172                             a = overl(:,ic)

```

```

00172             END SELECT
00173
00174             ENDIF
00175         ENDDO
00176
00177     ELSE
00178
00179         DO ic = 1, noint
00180
00181             IF ((atele(i) .EQ. ele2(ic) .AND. atele(j) .EQ. ele1(ic)) .AND. &
00182                 igltype .EQ. btype(ic)) THEN
00183
00184                 ! Now we've ID'ed our bond integral
00185
00186                 SELECT CASE(whichint)
00187                 CASE("H") ! We're doing the H matrix build
00188                     a = bond(:,ic)
00189                 CASE("S") ! We're doing the S matrix build
00190                     a = overl(:,ic)
00191                 END SELECT
00192
00193             ENDIF
00194         ENDDO
00195
00196     ENDIF
00197 ENDIF
00198
00199 ENDIF
00200
00201 IF (r .LE. a(7)) THEN
00202
00203     rmod = r - a(6)
00204
00205     univscale = exp(rmod*(a(2) + rmod*(a(3) + rmod*(a(4) + a(5)*rmod)))
00206
00207 ELSEIF (r .GT. a(7) .AND. r .LT. a(8)) THEN
00208
00209     rminusr1 = r - a(7)
00210
00211     univscale = a(9) + rminusr1*(a(10) + &
00212         rminusr1*(a(11) + rminusr1*(a(12) + &
00213             rminusr1*(a(13) + rminusr1*a(14))))))
00214
00215 ELSE
00216
00217     univscale = zero
00218
00219 END IF
00220
00221 univscale = a(1)*univscale
00222
00223 ! permutation symmetry
00224
00225 IF (l1 .GT. l2 .AND. mod(l1 + l2, 2) .NE. 0) univscale = -univscale
00226
00227 ! PRINT*, UNIVSCALE
00228
00229 RETURN
00230
00231 END FUNCTION univscale
00232

```

8.437 univtailcoef.f90 File Reference

Functions/Subroutines

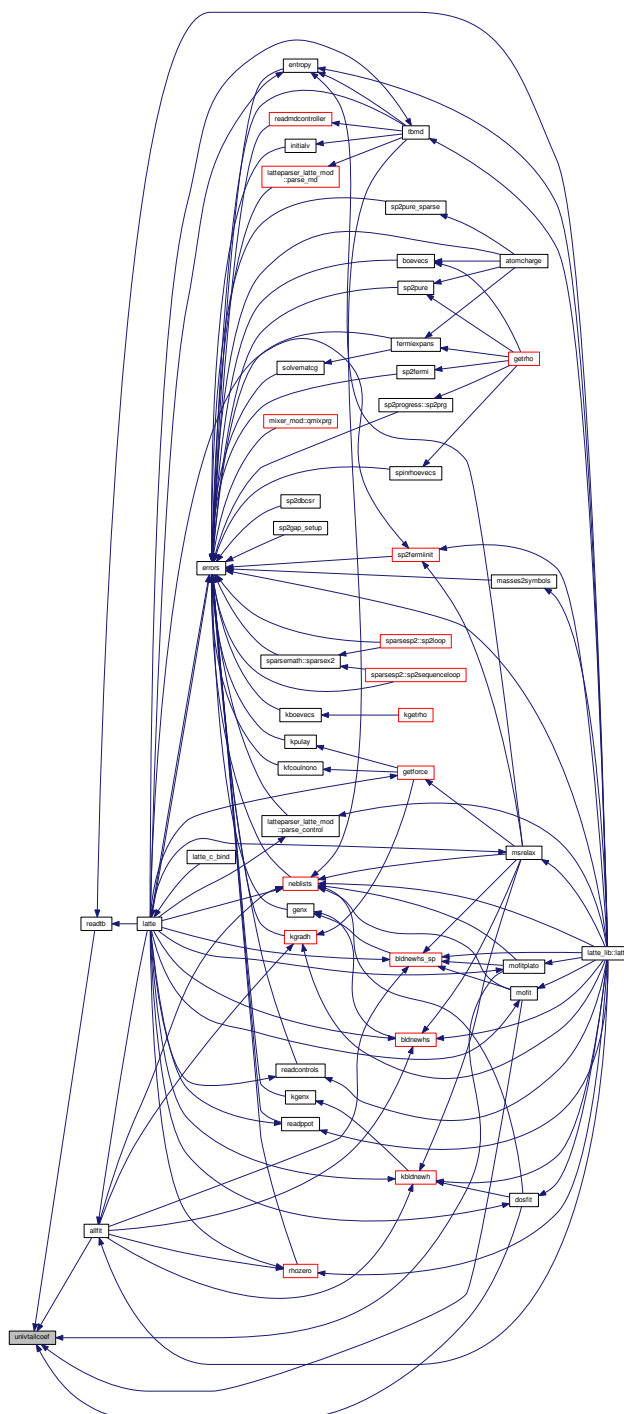
- subroutine [univtailcoef](#) (A)

8.437.1 Function/Subroutine Documentation

8.437.1.1 subroutine univtailcoef (real(latteprec), dimension(14) A)

Definition at line 23 of file [univtailcoef.f90](#).

Here is the caller graph for this function:



8.438 univtailcoef.f90

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was    !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE univtailcoef(A)
00023
00024   USE constants_mod
00025   USE myprecision
00026
00027   IMPLICIT NONE
00028
00029   INTEGER :: I
00030   REAL(LATTEPREC) :: A(14)
00031   REAL(LATTEPREC) :: R1, R1SQ, RCUT, RMOD
00032   REAL(LATTEPREC) :: SCL_R1, POLYNOM, DPOLY, DDPOLY
00033   REAL(LATTEPREC) :: DELTA, DELTA2, DELTA3, DELTA4
00034   IF (existerror) RETURN
00035
00036   IF ( abs(a(1)) .LT. 0.0000000000001 ) THEN
00037
00038     a(9) = zero
00039     a(10) = zero
00040     a(11) = zero
00041     a(12) = zero
00042     a(13) = zero
00043     a(14) = zero
00044
00045   ELSE
00046
00047     r1 = a(7)
00048     rcut = a(8)
00049
00050     r1sq = r1*r1
00051
00052     rmod = r1 - a(6)
00053
00054     polynom = rmod*(a(2) + rmod*(a(3) + rmod*(a(4) + a(5)*rmod)))
00055
00056     scl_r1 = exp(polynom)
00057
00058     !   CALL UNIVSCALE(R1, A, SCL_R1)
00059
00060     !   SCL_R1 = SCL_R1/A(1)
00061
00062     delta = rcut - r1
00063
00064     ! Now we're using a 6th order polynomial: fitted to value, first,
00065     ! and second derivatives at R1 and R_cut
00066
00067     a(9) = scl_r1
00068
00069     rmod = r1 - a(6)
00070
00071     dpoly = a(2) + two*a(3)*rmod + three*a(4)*rmod*rmod + &
00072             four*a(5)*rmod*rmod*rmod
00073
00074     a(10) = dpoly*scl_r1
00075
00076     ddpoly = two*a(3) + six*a(4)*rmod + twelve*a(5)*rmod*rmod
00077
00078     a(11) = (dpoly*dpoly + ddpoly)*scl_r1/two
00079
00080     delta2 = delta*delta
00081     delta3 = delta2*delta
00082     delta4 = delta3*delta
00083
00084     a(12) = (minusone/delta3)*(three*a(11)*delta2 + &
00085             six*a(10)*delta + ten*a(9))
00086
00087     a(13) = (one/delta4)*(three*a(11)*delta2 + &
00088             eight*a(10)*delta + fifteen*a(9))
00089
00090     a(14) = (minusone/(ten*delta3)) * &
00091             (six*a(13)*delta2 + three*a(12)*delta + a(11))
00092
00093   ENDIF

```

```
00094  
00095     RETURN  
00096  
00097 END SUBROUTINE univtailcoef
```

8.439 vdwtailcoef.f90 File Reference

Functions/Subroutines

- subroutine [vdwtailcoef](#)

8.439.1 Function/Subroutine Documentation

8.439.1.1 subroutine [vdwtailcoef](#) ()

Definition at line 23 of file [vdwtailcoef.f90](#).

```
00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE
```

```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE vdwtailcoef
00023
00024   USE constants_mod
00025   USE ppotarray
00026   USE myprecision
00027
00028   IMPLICIT NONE
00029
00030   INTEGER :: I, J, PPID
00031   REAL(LATTEPREC) :: A(17)
00032   REAL(LATTEPREC) :: DELTA, DELTA2, DELTA3, DELTA4
00033   REAL(LATTEPREC) :: X, Y, Z, MYR, R6
00034   REAL(LATTEPREC) :: R1, R1SQ, RCUT
00035   REAL(LATTEPREC) :: SCL_R1, POLY, DPOLY, DDPOLY, EXPTMP
00036   IF (existerror) RETURN
00037
00038   ! POTCOEF:
00039   ! 1  2  3  4  5  6  7  8  9  10  11 12 13 14 15 16
00040   ! A0 A1 A2 A3 A4 A5 A6 C R1 RCUT B1 B2 B3 B4 B5 B6
00041
00042   ! PHI = A0*EXP(A1*X + A2*X^2 + A3*X^3 + A4*X^4) + A5*EXP(A6*X) - C/X^6
00043
00044   ! Now lets' try this:
00045
00046   ! Phi A0*EXP(A1*(X - A5) + A2*(X - A5)^2 + A3*(X-A5)^3 + A4*(X-A5)^4)
00047
00048
00049
00050   !
00051   ! The cut-offs and joining functions look like this:
00052   !
00053   ! t(R) = B1 + B2*(R - R1) + B3*(R - R1)^2 + B4*(R - R1)^3 +
00054   !       B5*(R - R1)^4 + B6*(R - R1)^5
00055   !
00056
00057   DO ppid = 1, nopps
00058
00059       ! Join first : need values for pair potential at R1
00060
00061       r1 = potcoef(9,ppid) - potcoef(6,ppid)
00062       rcut = potcoef(10,ppid)
00063       r1sq = r1*r1
00064       ! R6 = R1SQ * R1SQ * R1SQ
00065
00066       ! CALL UNIVSCALE(R1, POTCOEF(:,PPID), SCL_R1)
00067
00068       poly = r1*(potcoef(2,ppid) + r1*(potcoef(3,ppid) + &
00069         r1*(potcoef(4,ppid) + r1*potcoef(5,ppid))))
00070
00071       scl_r1 = potcoef(1,ppid)*exp(poly)
00072
00073       ! EXPTMP = POTCOEF(6,PPID)*EXP(POTCOEF(7,PPID)*(R1 - POTCOEF(8,PPID)))
00074
00075       exptmp = zero
00076
00077       potcoef(11,ppid) = scl_r1 + exptmp ! - POTCOEF(8,PPID)/R6
00078
00079       dpoly = potcoef(2,ppid) + two*potcoef(3,ppid)*r1 + &
00080         three*potcoef(4,ppid)*r1sq + &
00081         four*potcoef(5,ppid)*r1*r1sq
00082       !- POTCOEF(6,PPID)/R1SQ
00083
00084       potcoef(12,ppid) = dpoly*scl_r1 + potcoef(7,ppid)*exptmp !+ &
00085       ! SIX*POTCOEF(8,PPID)/(R1*R6)
00086
00087       ddpoly = two*potcoef(3,ppid) + six*potcoef(4,ppid)*r1 + &
00088         twelve*potcoef(5,ppid)*r1sq
00089       !+ TWO*POTCOEF(6,PPID)/(R1*R1SQ)
00090
00091       potcoef(13,ppid) = half*( dpoly*dpoly + ddpoly)*scl_r1 + &
00092       potcoef(7,ppid)*potcoef(7,ppid)*exptmp ! - &
00093       !SIX*SEVEN*POTCOEF(8,PPID)/(R1SQ*R6) )

```

```

00094
00095     ! At the end of the join function:
00096
00097     r1 = potcoef(9,ppid)
00098     delta = rcut - r1
00099     delta2 = delta*delta
00100     delta3 = delta2*delta
00101     delta4 = delta3*delta
00102
00103     potcoef(14,ppid) = (minusone/delta3)*(three*potcoef(13,ppid)*delta2 + &
00104         six*potcoef(12,ppid)*delta + ten*potcoef(11,ppid))
00105
00106     potcoef(15,ppid) = (one/delta4)*(three*potcoef(13,ppid)*delta2 + &
00107         eight*potcoef(12,ppid)*delta + fifteen*potcoef(11,ppid))
00108
00109     potcoef(16,ppid) = (minusone/(ten*delta3))*(six*
00110 potcoef(15,ppid)*delta2 + &
00111         three*potcoef(14,ppid)*delta + potcoef(13,ppid))
00112
00112     ENDDO
00113
00114     RETURN
00115
00116 END SUBROUTINE vdwtailcoef
00117

```

8.441 velverlet.f90 File Reference

Functions/Subroutines

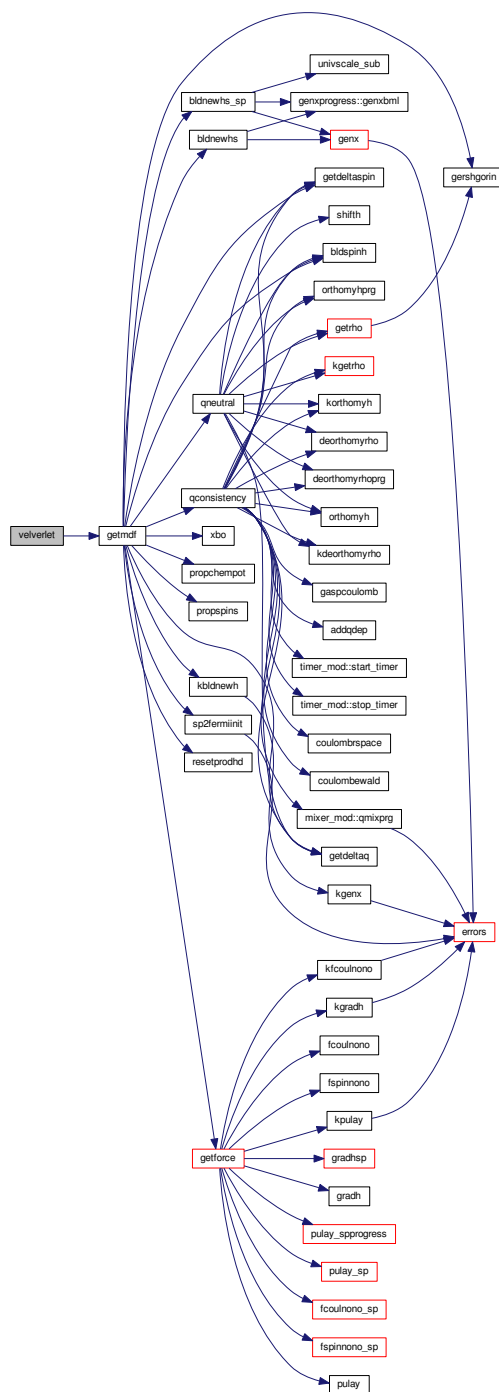
- subroutine [velverlet](#) (CURRITER)

8.441.1 Function/Subroutine Documentation

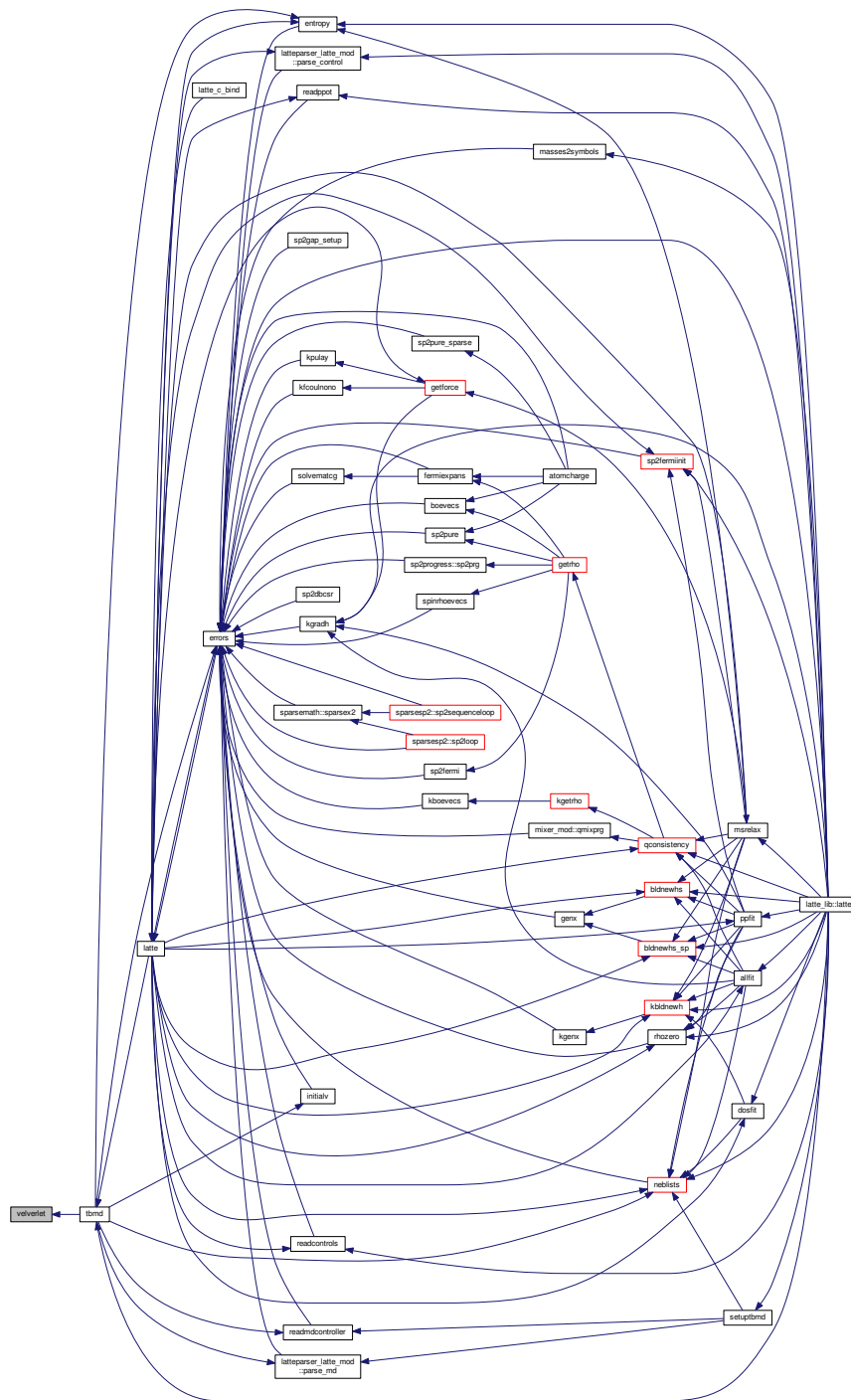
8.441.1.1 subroutine [velverlet](#) (integer *CURRITER*)

Definition at line 23 of file [velverlet.f90](#).

Here is the call graph for this function:



Here is the caller graph for this function:



8.442 velverlet.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010.  Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos  !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National      !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has  !
00006 ! rights to use, reproduce, and distribute this software.  NEITHER THE      !

```



```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE velverlet (CURRITER)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE mdarray
00027   USE myprecision
00028   USE constraints_mod
00029
00030   IMPLICIT NONE
00031
00032   INTEGER :: I, J, K, CURRITER
00033   REAL(LATTEPREC) :: PREFACTOR, PREPREFACT
00034   IF (existerror) RETURN
00035
00036   preprefact = half*f2v*dt
00037
00038   !
00039   ! Half timestep advance in V
00040   !
00041
00042   ! IF (FREEZE .EQ. 1) CALL FREEZE_ATOMS(V,FTOT)
00043
00044   DO i = 1, nats
00045
00046     prefactor = preprefact/mass(elempointer(i))
00047
00048     v(1,i) = v(1,i) + prefactor*ftot(1,i)
00049     v(2,i) = v(2,i) + prefactor*ftot(2,i)
00050     v(3,i) = v(3,i) + prefactor*ftot(3,i)
00051
00052   ENDDO
00053
00054   cr = cr + dt*v
00055
00056   !
00057   ! Get new force to complete advance in V
00058   !
00059
00060   CALL getmdf(1, curriter)
00061
00062   !
00063   ! Now finish advancing V with F(t + dt)
00064   !
00065
00066   DO i = 1, nats
00067
00068     prefactor = preprefact/mass(elempointer(i))
00069
00070     v(1,i) = v(1,i) + prefactor*ftot(1,i)
00071     v(2,i) = v(2,i) + prefactor*ftot(2,i)
00072     v(3,i) = v(3,i) + prefactor*ftot(3,i)
00073
00074   ENDDO
00075
00076   RETURN
00077
00078 END SUBROUTINE velverlet
00079

```

8.443 virialarray.f90 File Reference

Modules

- module [virialarray](#)

Variables

- real(latteprec), dimension(6) [virialarray::keten](#)
- real(latteprec) [virialarray::pressure](#)
- real(latteprec), dimension(6) [virialarray::strten](#)
- real(latteprec) [virialarray::sysvol](#)
- real(latteprec), dimension(6) [virialarray::virbond](#)
- real(latteprec), dimension(6) [virialarray::virscoul](#)
- real(latteprec), dimension(6) [virialarray::virial](#)
- real(latteprec), dimension(6) [virialarray::virpair](#)
- real(latteprec), dimension(6) [virialarray::virpul](#)
- real(latteprec), dimension(6) [virialarray::virscoul](#)
- real(latteprec), dimension(6) [virialarray::virsspin](#)

8.444 virialarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE virialarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027
00028   SAVE
00029
00030   !
00031   ! We're going to put everthing into 1 array. The 6 elements are:
00032   !
00033   ! 1 = xx
00034   ! 2 = yy
00035   ! 3 = zz
00036   ! 4 = xy
00037   ! 5 = yz
00038   ! 6 = zx
00039   !
00040
00041   REAL(LATTEPREC) :: virial(6), virbond(6), virscoul(6),
00042   virpair(6)
00042   REAL(LATTEPREC) :: virpul(6), virscoul(6), virsspin(6)
00043   REAL(LATTEPREC) :: keten(6), strten(6)
00044   REAL(LATTEPREC) :: pressure, sysvol
00045
00046 END MODULE virialarray

```

8.445 wigner.f90 File Reference

Functions/Subroutines

- real(latteprec) function [wigner](#) (L, M, MP, COSBETA)

8.445.1 Function/Subroutine Documentation

8.445.1.1 `real(latteprec)` function `wignerd` (`integer L`, `integer M`, `integer MP`, `real(latteprec) COSBETA`)

Definition at line 23 of file `wigner.d.f90`.

8.446 `wigner.d.f90`

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was      !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National    !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE     !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS         !
00009 ! SOFTWARE. If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as    !
00015 ! published by the Free Software Foundation; version 2.0 of the License.   !
00016 ! Accordingly, this program is distributed in the hope that it will be     !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of   !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 FUNCTION wignerd(L, M, MP, COSBETA)
00023
00024 ! Builds Wigner d function
00025 ! notation conforms to that in PRB 72 165107 (2005), eq. (9)
00026
00027 USE myprecision
00028
00029 IMPLICIT NONE
00030
00031 REAL(LATTEPREC) :: BETA, PREF, SUM, NUMER, DENOM, WIGNERD
00032 REAL(LATTEPREC) :: COSBETA
00033 INTEGER, EXTERNAL :: FACTORIAL
00034 INTEGER :: K, L, M, MP
00035
00036 ! PREF = TWO ** (-L) * REAL( (-1)**(L - MP)) * &
00037 !      Sqrt( REAL(FACTORIAL(L + M) * FACTORIAL(L - M)) * &
00038 !           FACTORIAL(L + MP) * FACTORIAL(L - MP)) )
00039
00040 pref = REAL( (-1)**(L - MP)) * &
00041       Sqrt( REAL(FACTORIAL(L + M) * FACTORIAL(L - M)) * &
00042            FACTORIAL(L + MP) * FACTORIAL(L - MP)) ) / REAL(2**1)
00043
00044 sum = zero
00045
00046 DO k = max(0, -m - mp), min(1 - m, 1 - mp)
00047     numer = REAL( (-1)**K ) * &
00048            (one - cosbeta) ** (1 - k - half * (m + mp)) * &
00049            (one + cosbeta) ** (k + half * (m + mp))
00050
00051     denom = REAL(FACTORIAL(K) * FACTORIAL(L - M - K) * &
00052                FACTORIAL(L - MP - K) * FACTORIAL(m + mp + k))
00053
00054     sum = sum + numer / denom
00055
00056 ENDDO
00057 wignerd = pref * sum
00058
00059 RETURN
00060
00061 END FUNCTION wignerd
00062

```


8.448 wrtcfgs.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE wrtcfgs(ITER)
00023
00024 !
00025 ! This subroutine will write out all quantities during calculated
00026 ! during a MD run to .cfg files so they can be read into
00027 ! Ju Li's most awesome (and most free) Atomeye visualization code directly.
00028 !
00029 ! http://mt.seas.upenn.edu/Archive/Graphics/A/
00030 !
00031 ! I suppose that coordinates, velocity, forces, atomic temperature,
00032 ! partials charges, and atomic spin densities would come handy
00033 !
00034
00035 USE constants_mod
00036 USE setuparray
00037 USE mdarray
00038 USE spinarray
00039 USE myprecision
00040 #ifdef MPI_ON
00041 USE mpi
00042 #endif
00043
00044 IMPLICIT NONE
00045
00046 INTEGER :: I, J, L
00047 INTEGER :: ITER, NOMELEMENTS, MYINDEX
00048 INTEGER :: MYID, IERR, IPIV(3), INFO
00049 REAL(LATTEPREC), ALLOCATABLE :: T(:), MAGF(:), ATSPIN(:)
00050 REAL(LATTEPREC) :: V2, MYMASS, MASSMYELEMENTS(4)
00051 REAL(LATTEPREC) :: WORK(3), BOXINV(3,3), S(3)
00052 ! REAL(LATTEPREC), PARAMETER :: CONV = 1.66053D6/(3.0D0*1.38062D0)
00053 CHARACTER(LEN=50) :: FLNM
00054 CHARACTER(LEN=2) :: MYELEMENTS(92)
00055 IF (existererror) RETURN
00056
00057 ALLOCATE(t(nats), magf(nats))
00058
00059 !
00060 ! Let's first get the temperature and |F| for each atom
00061 !
00062
00063
00064 IF (mdon .EQ. 1) THEN
00065
00066 IF (.NOT. ALLOCATED(v)) THEN
00067 ALLOCATE(v(3,nats)); v = 0.0d0
00068 ENDIF
00069
00070 DO i = 1, nats
00071
00072 v2 = v(1,i)*v(1,i) + v(2,i)*v(2,i) + v(3,i)*v(3,i)
00073
00074 t(i) = mass(elempointer(i))*v2
00075
00076 magf(i) = sqrt(ftot(1,i)*ftot(1,i) + ftot(2,i)*ftot(2,i) + &
00077 ftot(3,i)*ftot(3,i))
00078
00079 ENDDO
00080
00081 ELSE
00082
00083 t = zero
00084

```

```

00085      DO i = 1, nats
00086
00087          magf(i) = sqrt(ftot(1,i)*ftot(1,i) + ftot(2,i)*ftot(2,i) + &
00088                      ftot(3,i)*ftot(3,i))
00089
00090      ENDDO
00091
00092  ENDIF
00093
00094  t = t*mvv2t/three
00095
00096  !
00097  ! Now let's figure out how many elements we've got and the cfg
00098  ! formats is most efficient when they're all written out in the same
00099  ! block
00100  !
00101
00102  myelements = "XX"
00103  nomyelements = 1
00104  myelements(nomyelements) = atele(1)
00105  massmyelements = mass(elempointer(1))
00106
00107  DO j = 2, nats
00108
00109      IF (atele(j) .NE. myelements(1) &
00110          .AND. atele(j) .NE. myelements(2) &
00111          .AND. atele(j) .NE. myelements(3) &
00112          .AND. atele(j) .NE. myelements(4) ) THEN
00113
00114          nomyelements = nomyelements + 1
00115          myelements(nomyelements) = atele(j)
00116          massmyelements(nomyelements) = mass(elempointer(j))
00117
00118      ENDIF
00119
00120  ENDDO
00121
00122  IF (pbcon .EQ. 1) THEN
00123
00124      ! Write CFG files if periodic boundary conditions are on
00125
00126      IF (iter .LT. 0) THEN
00127          WRITE(flnm, ' ("animate/dump2atomeye.PANIC.cfg")')
00128          IF (iter .EQ. -999) WRITE(flnm, ' ("lastsystem.cfg")')
00129      ENDIF
00130
00131      IF (parrep .EQ. 0) THEN
00132
00133          IF (iter .GE. 0 .AND. iter .LT. 10) THEN
00134              WRITE(flnm, ' ("animate/dump2atomeye.",I1,".cfg")') iter
00135          ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00136              WRITE(flnm, ' ("animate/dump2atomeye.",I2,".cfg")') iter
00137          ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00138              WRITE(flnm, ' ("animate/dump2atomeye.",I3,".cfg")') iter
00139          ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00140              WRITE(flnm, ' ("animate/dump2atomeye.",I4,".cfg")') iter
00141          ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00142              WRITE(flnm, ' ("animate/dump2atomeye.",I5,".cfg")') iter
00143          ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00144              WRITE(flnm, ' ("animate/dump2atomeye.",I6,".cfg")') iter
00145          ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00146              WRITE(flnm, ' ("animate/dump2atomeye.",I7,".cfg")') iter
00147          ELSEIF (iter .GE. 10000000 .AND. iter .LT. 100000000) THEN
00148              WRITE(flnm, ' ("animate/dump2atomeye.",I8,".cfg")') iter
00149          ENDIF
00150
00151          OPEN(unit = 23, status="UNKNOWN", file=flnm)
00152
00153      ELSE
00154
00155      #ifdef MPI_ON
00156
00157          CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00158
00159      #endif
00160
00161      IF (myid .LT. 10) THEN
00162
00163          IF (iter .GE. 0 .AND. iter .LT. 10) THEN
00164              WRITE(flnm, ' (I1,"/animate/dump2atomeye.",I1,".cfg")') myid, iter
00165          ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00166              WRITE(flnm, ' (I1,"/animate/dump2atomeye.",I2,".cfg")') myid,iter
00167          ELSEIF (iter .GE.100 .AND. iter .LT. 1000) THEN
00168              WRITE(flnm, ' (I1,"/animate/dump2atomeye.",I3,".cfg")') myid,iter
00169          ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00170              WRITE(flnm, ' (I1,"/animate/dump2atomeye.",I4,".cfg")') myid,iter
00171          ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN

```

```

00172         WRITE(flnm,'(I1,"/animate/dump2atomeye.",I5,".cfg)")' myid,iter
00173     ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00174         WRITE(flnm,'(I1,"/animate/dump2atomeye.",I6,".cfg)")' myid,iter
00175     ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00176         WRITE(flnm,'(I1,"/animate/dump2atomeye.",I7,".cfg)")' myid,iter
00177     ELSEIF (iter .GE. 10000000 .AND. iter .LT. 100000000) THEN
00178         WRITE(flnm,'(I1,"/animate/dump2atomeye.",I8,".cfg)")' myid,iter
00179     ENDIF
00180
00181     ELSEIF (myid .GE. 10 .AND. myid .LT. 100) THEN
00182
00183         IF (iter .GE. 0 .AND. iter .LT. 10) THEN
00184             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I1,".cfg)")' myid, iter
00185         ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00186             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I2,".cfg)")' myid,iter
00187         ELSEIF (iter .GE.100 .AND. iter .LT. 1000) THEN
00188             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I3,".cfg)")' myid,iter
00189         ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00190             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I4,".cfg)")' myid,iter
00191         ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00192             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I5,".cfg)")' myid,iter
00193         ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00194             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I6,".cfg)")' myid,iter
00195         ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00196             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I7,".cfg)")' myid,iter
00197         ELSEIF (iter .GE. 10000000 .AND. iter .LT. 100000000) THEN
00198             WRITE(flnm,'(I2,"/animate/dump2atomeye.",I8,".cfg)")' myid,iter
00199         ENDIF
00200
00201     ELSEIF (myid .GE. 100 .AND. myid .LT. 1000) THEN
00202
00203         IF (iter .GE. 0 .AND. iter .LT. 10) THEN
00204             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I1,".cfg)")' myid, iter
00205         ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00206             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I2,".cfg)")' myid,iter
00207         ELSEIF (iter .GE.100 .AND. iter .LT. 1000) THEN
00208             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I3,".cfg)")' myid,iter
00209         ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00210             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I4,".cfg)")' myid,iter
00211         ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00212             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I5,".cfg)")' myid,iter
00213         ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00214             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I6,".cfg)")' myid,iter
00215         ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00216             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I7,".cfg)")' myid,iter
00217         ELSEIF (iter .GE. 10000000 .AND. iter .LT. 100000000) THEN
00218             WRITE(flnm,'(I3,"/animate/dump2atomeye.",I8,".cfg)")' myid,iter
00219         ENDIF
00220
00221     ENDIF
00222
00223     OPEN(unit = 23, status="UNKNOWN", file=flnm)
00224
00225 ENDIF
00226
00227
00228 WRITE(23,10) "Number of particles = ", nats
00229 WRITE(23,'("A = 1.0 Angstrom")')
00230 WRITE(23,11) "H0(1,1) = ", box(1,1), " A"
00231 WRITE(23,11) "H0(1,2) = ", box(1,2), " A"
00232 WRITE(23,11) "H0(1,3) = ", box(1,3), " A"
00233 WRITE(23,11) "H0(2,1) = ", box(2,1), " A"
00234 WRITE(23,11) "H0(2,2) = ", box(2,2), " A"
00235 WRITE(23,11) "H0(2,3) = ", box(2,3), " A"
00236 WRITE(23,11) "H0(3,1) = ", box(3,1), " A"
00237 WRITE(23,11) "H0(3,2) = ", box(3,2), " A"
00238 WRITE(23,11) "H0(3,3) = ", box(3,3), " A"
00239
00240 ! Box inverse
00241
00242 boxinv = box
00243
00244 CALL dgetrf(3, 3, boxinv, 3, ipiv, info)
00245
00246 CALL dgetri(3, boxinv, 3, ipiv, work, 3, info)
00247
00248
00249 !
00250 ! No partial charges, no spins
00251 !
00252 ! Write: x, y, z, vx, vy, vz, fx, fy, fz, |F|, temperature
00253 !
00254
00255 IF (mdon .EQ. 1) THEN
00256
00257     IF (spinon .EQ. 0) THEN
00258

```

```

00259      !
00260      ! Same as above, but now we add partial charges too
00261      !
00262
00263      WRITE(23,'("entry_count = 12")')
00264      WRITE(23,13) "auxiliary[0] = fx "
00265      WRITE(23,13) "auxiliary[1] = fy "
00266      WRITE(23,13) "auxiliary[2] = fz "
00267      WRITE(23,13) "auxiliary[3] = |F|"
00268      WRITE(23,13) "auxiliary[4] = T/K"
00269      WRITE(23,13) "auxiliary[5] = q "
00270
00271      DO i = 1, nomyelements
00272
00273          WRITE(23,14) massmyelements(i)
00274          WRITE(23,15) myelements(i)
00275
00276          DO j = 1, nats
00277
00278              IF (atele(j) .EQ. myelements(i)) THEN
00279
00280                  CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,j), 1, zero, s, 1)
00281                  WRITE(23,17) s(1), s(2), s(3), v(1,j), v(2,j), v(3,j), &
00282                      ftot(1,j), ftot(2,j), ftot(3,j), &
00283                      magf(j), t(j), deltaq(j)
00284
00285              ENDIF
00286
00287          ENDDO
00288
00289      ENDDO
00290
00291      ELSEIF (spinon .EQ. 1) THEN
00292
00293          !
00294          ! Now we add the net spin per atom too
00295          !
00296
00297          ALLOCATE(atspin(nats))
00298
00299          myindex = 0
00300
00301          DO i = 1, nats
00302
00303              IF (basis(elempointer(i)) .EQ. "s") THEN
00304
00305                  myindex = myindex + 1
00306                  atspin(i) = deltaspin(myindex)
00307
00308              ELSEIF (basis(elempointer(i)) .EQ. "sp") THEN
00309
00310                  atspin(i) = deltaspin(myindex + 1) + deltaspin(myindex + 2)
00311
00312                  myindex = myindex + 2
00313
00314              ENDIF
00315
00316          ENDDO
00317
00318      WRITE(23,'("entry_count = 10")')
00319      WRITE(23,13) "auxiliary[0] = |F|"
00320      WRITE(23,13) "auxiliary[1] = T/K"
00321      WRITE(23,13) "auxiliary[2] = q "
00322      WRITE(23,13) "auxiliary[3] = spn"
00323
00324      DO i = 1, nomyelements
00325
00326          WRITE(23,14) massmyelements(i)
00327          WRITE(23,15) myelements(i)
00328
00329          DO j = 1, nats
00330
00331              IF (atele(j) .EQ. myelements(i)) THEN
00332
00333                  CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,j), 1, zero, s, 1)
00334
00335                  WRITE(23,18) s(1), s(2), s(3), v(1,j), v(2,j), v(3,j), &
00336                      magf(j), t(j), deltaq(j), atspin(j)
00337
00338              ENDIF
00339
00340          ENDDO
00341
00342      ENDDO
00343
00344      DEALLOCATE(atspin)
00345

```



```

00346         ENDIF
00347
00348     ELSE
00349
00350         ! No MD so no velocities
00351
00352         IF (spinon .EQ. 0) THEN
00353
00354             WRITE(23,'(".NO_VELOCITY."')
00355             WRITE(23,'("entry_count = 8"')
00356             WRITE(23,13) "auxiliary[0] = fx "
00357             WRITE(23,13) "auxiliary[1] = fy "
00358             WRITE(23,13) "auxiliary[2] = fz "
00359             WRITE(23,13) "auxiliary[3] = |F| "
00360             WRITE(23,13) "auxiliary[4] = q "
00361
00362             DO i = 1, nomyelements
00363
00364                 WRITE(23,14) massmyelements(i)
00365                 WRITE(23,15) myelements(i)
00366
00367                 DO j = 1, nats
00368
00369                     IF (atele(j) .EQ. myelements(i)) THEN
00370
00371                         CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,j), 1, zero, s, 1)
00372
00373                         WRITE(23,22) s(1), s(2), s(3), &
00374                             ftot(1,j), ftot(2,j), ftot(3,j), &
00375                             magf(j), deltaq(j)
00376
00377                     ENDIF
00378
00379                 ENDDO
00380
00381             ENDDO
00382
00383         ELSEIF (spinon .EQ. 1) THEN
00384
00385             WRITE(23,'(".NO_VELOCITY."')
00386             WRITE(23,'("entry_count = 9"')
00387             WRITE(23,13) "auxiliary[0] = fx "
00388             WRITE(23,13) "auxiliary[1] = fy "
00389             WRITE(23,13) "auxiliary[2] = fz "
00390             WRITE(23,13) "auxiliary[3] = |F| "
00391             WRITE(23,13) "auxiliary[4] = q "
00392             WRITE(23,13) "auxiliary[5] = spn"
00393
00394             ALLOCATE(atspin(nats))
00395
00396             myindex = 0
00397
00398             DO i = 1, nats
00399
00400                 IF (basis(elempointer(i)) .EQ. "s") THEN
00401
00402                     myindex = myindex + 1
00403                     atspin(i) = deltaspin(myindex)
00404
00405                 ELSEIF (basis(elempointer(i)) .EQ. "sp") THEN
00406
00407                     atspin(i) = deltaspin(myindex + 1) + deltaspin(myindex + 2)
00408
00409                     myindex = myindex + 2
00410
00411                 ENDIF
00412
00413             ENDDO
00414
00415             DO i = 1, nomyelements
00416
00417                 WRITE(23,14) massmyelements(i)
00418                 WRITE(23,15) myelements(i)
00419
00420                 DO j = 1, nats
00421
00422                     IF (atele(j) .EQ. myelements(i)) THEN
00423
00424                         CALL dgemv('T', 3, 3, one, boxinv, 3, cr(1,j), 1, zero, s, 1)
00425
00426                         WRITE(23,23) s(1), s(2), s(3), &
00427                             ftot(1,j), ftot(2,j), ftot(3,j), &
00428                             magf(j), deltaq(j), atspin(j)
00429
00430                     ENDIF
00431
00432                 ENDDO

```

```

00433
00434         ENDDO
00435
00436         DEALLOCATE(atspin)
00437
00438     ENDIF
00439
00440 ENDIF
00441
00442 !     DEALLOCATE(T, MAGF)
00443
00444 CLOSE(23)
00445
00446 ELSE
00447
00448     ! No PBCs? write an .xyz file instead
00449
00450     IF (iter .EQ. 0) THEN
00451
00452         WRITE(flnm,'("animate/myXYZfile.xyz")')
00453
00454         OPEN(unit=23, status="UNKNOWN", file=flnm)
00455
00456     ELSEIF (iter .LT. 0) THEN
00457
00458         WRITE(flnm,'("animate/myXYZfile.PANIC.xyz")')
00459         IF (iter .EQ. -999) WRITE(flnm,'("lastsystem.xyz")')
00460
00461         OPEN(unit=23, status="UNKNOWN", file=flnm)
00462
00463     ENDIF
00464
00465
00466
00467     WRITE(23,20) nats
00468     IF (iter .EQ. -999) THEN
00469         WRITE(23,'("Last atomic configuration")')
00470     ELSE
00471         WRITE(23,'("LATTE MD: dt = ", F8.3, " Time step = ", I9)') dt, iter
00472     ENDIF
00473
00474     DO i = 1, nats
00475         WRITE(23,21) atele(i), cr(1,i), cr(2,i), cr(3,i)
00476     ENDDO
00477
00478     IF (iter .EQ. -999) CLOSE(23)
00479
00480     !     IF (ITER .LT. 0) THEN
00481
00482     !         WRITE(FLNM,'("animate/myXYZfile.PANIC.xyz")')
00483     !         IF (ITER .EQ. -999) WRITE(FLNM,'("lastsystem.xyz")')
00484
00485     !         OPEN(UNIT=24, STATUS="UNKNOWN", FILE=FLNM)
00486
00487     !         WRITE(24,20) NATS
00488     !         WRITE(24,'("LATTE MD: dt = ", F8.3, " Time step = ", I9)') DT, ITER
00489     !         DO I = 1, NATS
00490     !             WRITE(24,21) ATELE(I), CR(1,I), CR(2,I), CR(3,I)
00491     !         ENDDO
00492
00493     !         CLOSE (24)
00494
00495     !     ENDIF
00496
00497 END IF
00498
00499 IF (mdon .EQ. 1) THEN
00500     DEALLOCATE(t, magf)
00501 ELSE
00502     DEALLOCATE(magf)
00503 ENDIF
00504
00505 10 FORMAT(a22, 1x, i6)
00506 11 FORMAT(a10,1x,f18.9, a2)
00507 12 FORMAT(a15)
00508 13 FORMAT(a18)
00509 14 FORMAT(f5.1)
00510 15 FORMAT(a2)
00511 16 FORMAT(3(f8.6,1x), 3(f12.8,1x), f12.6, 1x, f12.2)
00512 17 FORMAT(3(f8.6,1x), 6(f15.8,1x), f12.6, 1x, f12.2, 1x, f10.4)
00513 18 FORMAT(3(f8.6,1x), 3(f12.8,1x), f12.6, 1x, f12.2, 1x, 2(f10.4, 1x))
00514 20 FORMAT(i6)
00515 21 FORMAT(a2, 1x, 3f12.2)
00516
00517 22 FORMAT(3(g14.6,1x), 3(g14.6,1x), g14.6, 1x, 1(g14.6, 1x))
00518 23 FORMAT(3(g14.6,1x), 3(g14.6,1x), g14.6, 1x, 2(g14.6, 1x))
00519

```

```
00520
00521     RETURN
00522
00523 END SUBROUTINE wrtcfgs
```

8.449 wrtrestart.f90 File Reference

Functions/Subroutines

- subroutine [wrtrestart](#) (ITER)

8.449.1 Function/Subroutine Documentation

8.449.1.1 subroutine wrtrestart (integer *ITER*)

Definition at line 23 of file [wrtrestart.f90](#).


```

00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY,      !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS          !
00009 ! SOFTWARE.  If software is modified to produce derivative works, such      !
00010 ! modified software should be clearly marked, so as not to confuse it        !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as      !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                          !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE wrtrestart(ITER)
00023
00024   USE constants_mod
00025   USE setuparray
00026   USE kspacearray
00027   USE mdarray
00028   USE spinarray
00029   #ifdef MPI_ON
00030     USE mpi
00031   #endif
00032
00033   IMPLICIT NONE
00034
00035   INTEGER :: I, K, ITER
00036   INTEGER :: MYID, IERR
00037   COMPLEX (LATTEPREC) :: KSUM
00038   CHARACTER (LEN=100) :: FLNM
00039   IF (existerror) RETURN
00040
00041   IF ( verbose < 0 ) RETURN
00042
00043   IF (mdon .EQ. 1) THEN
00044
00045     IF (parrep .EQ. 0) THEN
00046
00047       IF (iter .LT. 10) THEN
00048         WRITE(flnm, ' ("Restarts/restartMD.", I1, ".dat")') iter
00049       ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00050         WRITE(flnm, ' ("Restarts/restartMD.", I2, ".dat")') iter
00051       ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00052         WRITE(flnm, ' ("Restarts/restartMD.", I3, ".dat")') iter
00053       ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00054         WRITE(flnm, ' ("Restarts/restartMD.", I4, ".dat")') iter
00055       ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00056         WRITE(flnm, ' ("Restarts/restartMD.", I5, ".dat")') iter
00057       ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00058         WRITE(flnm, ' ("Restarts/restartMD.", I6, ".dat")') iter
00059       ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00060         WRITE(flnm, ' ("Restarts/restartMD.", I7, ".dat")') iter
00061       ENDIF
00062
00063       OPEN (unit = 21, status="UNKNOWN", file=flnm)
00064       OPEN (unit = 19, status="UNKNOWN", &
00065         file="Restarts/restartMD.last.dat")
00066
00067     ELSE
00068
00069     #ifdef MPI_ON
00070       CALL mpi_comm_rank( mpi_comm_world, myid, ierr )
00071     #endif
00072
00073     IF (myid .LT. 10) THEN
00074
00075       IF (iter .LT. 10) THEN
00076         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I1, ".dat")') myid, iter
00077       ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00078         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I2, ".dat")') myid, iter
00079       ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00080         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I3, ".dat")') myid, iter
00081       ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00082         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I4, ".dat")') myid, iter
00083       ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00084         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I5, ".dat")') myid, iter
00085       ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00086         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I6, ".dat")') myid, iter
00087       ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00088         WRITE(flnm, ' (I1, "/Restarts/restartMD.", I7, ".dat")') myid, iter
00089       ENDIF
00090
00091     ELSEIF ( myid .GE. 10 .AND. myid .LT. 100) THEN
00092
00093       IF (iter .LT. 10) THEN

```

```

00094         WRITE(flnm,'(I2,"/Restarts/restartMD.", I1,".dat")') myid,iter
00095     ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00096         WRITE(flnm,'(I2,"/Restarts/restartMD.", I2,".dat")') myid,iter
00097     ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00098         WRITE(flnm,'(I2,"/Restarts/restartMD.", I3,".dat")') myid,iter
00099     ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00100         WRITE(flnm,'(I2,"/Restarts/restartMD.", I4,".dat")') myid,iter
00101     ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00102         WRITE(flnm,'(I2,"/Restarts/restartMD.", I5,".dat")') myid,iter
00103     ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00104         WRITE(flnm,'(I2,"/Restarts/restartMD.", I6,".dat")') myid,iter
00105     ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00106         WRITE(flnm,'(I2,"/Restarts/restartMD.", I7,".dat")') myid,iter
00107     ENDIF
00108
00109     ELSEIF ( myid .GE. 100 .AND. myid .LT. 1000) THEN
00110
00111         IF (iter .LT. 10) THEN
00112             WRITE(flnm,'(I3,"/Restarts/restartMD.", I1,".dat")') myid,iter
00113         ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00114             WRITE(flnm,'(I3,"/Restarts/restartMD.", I2,".dat")') myid,iter
00115         ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00116             WRITE(flnm,'(I3,"/Restarts/restartMD.", I3,".dat")') myid,iter
00117         ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00118             WRITE(flnm,'(I3,"/Restarts/restartMD.", I4,".dat")') myid,iter
00119         ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00120             WRITE(flnm,'(I3,"/Restarts/restartMD.", I5,".dat")') myid,iter
00121         ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00122             WRITE(flnm,'(I3,"/Restarts/restartMD.", I6,".dat")') myid,iter
00123         ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00124             WRITE(flnm,'(I3,"/Restarts/restartMD.", I7,".dat")') myid,iter
00125         ENDIF
00126
00127     ENDIF
00128
00129     OPEN (unit = 21, status="UNKNOWN", file=flnm)
00130
00131     IF (myid .LT. 10) THEN
00132         WRITE(flnm,'(I1,"/Restarts/restartMD.last.dat")') myid
00133     ELSEIF (myid .GE. 10 .AND. myid .LT. 100) THEN
00134         WRITE(flnm,'(I2,"/Restarts/restartMD.last.dat")') myid
00135     ELSEIF (myid .GE. 100 .AND. myid .LT. 1000) THEN
00136         WRITE(flnm,'(I3,"/Restarts/restartMD.last.dat")') myid
00137     ENDIF
00138
00139     OPEN (unit = 19, status="UNKNOWN", file=flnm)
00140
00141     ENDIF
00142
00143     ELSEIF (relaxme .EQ. 1) THEN
00144
00145         OPEN (unit = 21, status="UNKNOWN", file="restartREL.dat")
00146
00147     ELSEIF (relaxme .EQ. 0 .AND. mdon .EQ. 0) THEN
00148
00149         OPEN (unit = 21, status="UNKNOWN", file="restart_singlepoint.dat")
00150
00151     ENDIF
00152
00153     IF (mdon .EQ. 1) WRITE(21,16) "Iter= ", iter
00154     WRITE(21,*) nats
00155     WRITE(21,*) box(1,1), box(1,2), box(1,3)
00156     WRITE(21,*) box(2,1), box(2,2), box(2,3)
00157     WRITE(21,*) box(3,1), box(3,2), box(3,3)
00158
00159     ! WRITE(21,10) "Nats= ", NATS
00160     ! WRITE(21,11) "1.0"
00161     ! WRITE(21,12) BOX(1,1), BOX(2,1), BOX(1,2), BOX(2,2), &
00162     ! BOX(1,3), BOX(2,3)
00163
00164     DO i = 1, nats
00165         WRITE(21,14) atele(i), cr(1,i), cr(2,i), cr(3,i)
00166     ENDDO
00167
00168     WRITE(21,17) chempot
00169
00170     IF (spinon .EQ. 0) THEN
00171
00172         WRITE(21,18) hdim
00173
00174         IF (kon .EQ. 0) THEN
00175
00176             DO i = 1, hdim
00177                 WRITE(21,19) bo(i,i)/two, bo(i,i)/two
00178             ENDDO
00179
00180

```

```

00181      ELSE
00182
00183          DO i = 1, hdim
00184              ksum = cmplx(zero)
00185
00186              DO k = 1, nktot
00187                  ksum = ksum + kbo(i,i,k)
00188              ENDDO
00189
00190              WRITE(21,19) REAL(ksum)/(two*REAL(nktot)), &
00191                  REAL(ksum)/(two*REAL(nktot))
00192          ENDDO
00193      ENDIF
00194
00195      ELSEIF (spinon .EQ. 1) THEN
00196
00197          WRITE(21,18) hdim
00198          DO i = 1, hdim
00199              WRITE(21,19) rhoup(i,i), rhodown(i,i)
00200          ENDDO
00201      ENDIF
00202
00203      IF (mdon .EQ. 1) THEN
00204
00205          DO i = 1, nats
00206              WRITE(21,15) v(1,i), v(2,i), v(3,i)
00207          ENDDO
00208      ENDIF
00209
00210      CLOSE(21)
00211
00212      IF (mdon .EQ. 1) THEN
00213
00214          WRITE(19,16) "Iter= ", iter
00215          WRITE(19,*) nats
00216          WRITE(19,*) box(1,1), box(1,2), box(1,3)
00217          WRITE(19,*) box(2,1), box(2,2), box(2,3)
00218          WRITE(19,*) box(3,1), box(3,2), box(3,3)
00219
00220          DO i = 1, nats
00221              WRITE(19,14) atele(i), cr(1,i), cr(2,i), cr(3,i)
00222          ENDDO
00223
00224          WRITE(19,17) chempot
00225
00226          IF (spinon .EQ. 0) THEN
00227
00228              WRITE(19,18) hdim
00229
00230              IF (kon .EQ. 0) THEN
00231
00232                  DO i = 1, hdim
00233                      WRITE(19,19) bo(i,i)/two, bo(i,i)/two
00234                  ENDDO
00235
00236              ELSE
00237
00238                  DO i = 1, hdim
00239                      ksum = cmplx(zero)
00240
00241                      DO k = 1, nktot
00242                          ksum = ksum + kbo(i,i,k)
00243                      ENDDO
00244
00245                      WRITE(19,19) REAL(ksum)/(two*REAL(nktot)), &
00246                          REAL(ksum)/(two*REAL(nktot))
00247                  ENDDO
00248              ENDIF
00249          ENDIF
00250
00251      ELSEIF (spinon .EQ. 1) THEN
00252
00253          WRITE(19,18) hdim
00254          DO i = 1, hdim
00255              WRITE(19,19) rhoup(i,i), rhodown(i,i)
00256          ENDDO
00257      ENDIF
00258
00259      DO i = 1, nats
00260          WRITE(19,15) v(1,i), v(2,i), v(3,i)
00261      ENDDO
00262
00263      CLOSE(19)
00264
00265      ENDIF
00266
00267  ENDIF

```

```

00268
00269 10 FORMAT(a6,1x,i7)
00270 11 FORMAT(a3)
00271 12 FORMAT(6(e24.16,1x))
00272 !13 FORMAT(3(E24.16,1X),A2,1X,3(I4,1X))
00273 14 FORMAT(a2, 1x, 3g24.14)
00274 15 FORMAT(3(e24.16,1x))
00275 16 FORMAT(a6, 1x, i10)
00276 17 FORMAT(e24.16)
00277 18 FORMAT(i12)
00278 19 FORMAT(2e24.16)
00279
00280 RETURN
00281
00282 END SUBROUTINE wrtrestart

```

8.451 wrtrestartlib.f90 File Reference

Functions/Subroutines

- subroutine [wrtrestartlib](#) (ITER)

8.451.1 Function/Subroutine Documentation

8.451.1.1 subroutine wrtrestartlib (integer *ITER*)

Definition at line 23 of file [wrtrestartlib.f90](#).

Here is the caller graph for this function:



8.452 wrtrestartlib.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !

```



```

00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE wrtrestartlib(ITER)
00023
00024     USE kspacearray
00025     USE neblistarray
00026     USE ppotarray
00027     USE restartarray
00028     USE sparsearray
00029     USE univarray
00030     USE xboarray
00031     USE coulombarray
00032     USE diagarray
00033     USE mdarray
00034     USE nonoarray
00035     USE purearray
00036     USE setuparray
00037     USE spinarray
00038     USE virialarray
00039
00040     USE constants_mod
00041 #ifdef MPI_ON
00042     USE mpi
00043 #endif
00044
00045     IMPLICIT NONE
00046
00047     INTEGER :: I, K, ITER
00048     INTEGER :: MYID, IERR
00049     INTEGER :: CHARACTERCHECK(9), COMPLEXCHECK(15), INTEGERCHECK(20), REALCHECK(98)
00050     COMPLEX(LATTEPREC) :: KSUM
00051     REAL(LATTEPREC), ALLOCATABLE :: TMPBODIAGUP(:), TMPBODIAGDOWN(:)
00052     CHARACTER(LEN=100) :: FLNM, NAME
00053
00054     IF (existererror) RETURN
00055
00056     IF (iter .LT. 10) THEN
00057         WRITE(flnm, '("restart.latte.", I1, ".dat")') iter
00058     ELSEIF (iter .GE. 10 .AND. iter .LT. 100) THEN
00059         WRITE(flnm, '("restart.latte.", I2, ".dat")') iter
00060     ELSEIF (iter .GE. 100 .AND. iter .LT. 1000) THEN
00061         WRITE(flnm, '("restart.latte.", I3, ".dat")') iter
00062     ELSEIF (iter .GE. 1000 .AND. iter .LT. 10000) THEN
00063         WRITE(flnm, '("restart.latte.", I4, ".dat")') iter
00064     ELSEIF (iter .GE. 10000 .AND. iter .LT. 100000) THEN
00065         WRITE(flnm, '("restart.latte.", I5, ".dat")') iter
00066     ELSEIF (iter .GE. 100000 .AND. iter .LT. 1000000) THEN
00067         WRITE(flnm, '("restart.latte.", I6, ".dat")') iter
00068     ELSEIF (iter .GE. 1000000 .AND. iter .LT. 10000000) THEN
00069         WRITE(flnm, '("restart.latte.", I7, ".dat")') iter
00070     ENDIF
00071
00072     OPEN(14, file=flnm, form="UNFORMATTED", access="SEQUENTIAL")
00073
00074     charactercheck = 0
00075     integercheck = 0
00076     complexcheck = 0
00077     realcheck = 0
00078
00079     IF(ALLOCATED( atele ) .AND. rslevel == 0) charactercheck( 2 ) = 1
00080
00081     IF(ALLOCATED( basis ) .AND. rslevel == 0) charactercheck( 3 ) = 1
00082
00083     IF(ALLOCATED( btype ) .AND. rslevel == 0) charactercheck( 4 ) = 1
00084
00085     IF(ALLOCATED( ele ) .AND. rslevel == 0) charactercheck( 5 ) = 1
00086
00087     IF(ALLOCATED( ele1 ) .AND. rslevel == 0) charactercheck( 6 ) = 1
00088
00089     IF(ALLOCATED( ele2 ) .AND. rslevel == 0) charactercheck( 7 ) = 1
00090
00091     IF(ALLOCATED( ppele1 ) .AND. rslevel == 0) charactercheck( 8 ) = 1
00092
00093     IF(ALLOCATED( ppele2 ) .AND. rslevel == 0) charactercheck( 9 ) = 1
00094
00095     IF(ALLOCATED( diag_zwork ) .AND. rslevel == 0) complexcheck( 2 ) = 1
00096
00097     IF(ALLOCATED( hk ) .AND. rslevel >= 2) complexcheck( 3 ) = 1
00098
00099     IF(ALLOCATED( hk0 ) .AND. rslevel >= 2) complexcheck( 4 ) = 1
00100
00101     IF(ALLOCATED( hkdiag ) .AND. rslevel >= 1) complexcheck( 5 ) = 1
00102
00103     IF(ALLOCATED( kbo ) .AND. rslevel >= 2) complexcheck( 6 ) = 1
00104
00105     IF(ALLOCATED( kevecs ) .AND. rslevel >= 2) complexcheck( 7 ) = 1
00106

```

```

00107 IF(ALLOCATED( kf ) .AND. rslevel >= 1) complexcheck( 8 ) = 1
00108
00109 IF(ALLOCATED( khtmp ) .AND. rslevel >= 1) complexcheck( 9 ) = 1
00110
00111 IF(ALLOCATED( korthoh ) .AND. rslevel >= 2) complexcheck( 10 ) = 1
00112
00113 IF(ALLOCATED( kxmat ) .AND. rslevel >= 2) complexcheck( 11 ) = 1
00114
00115 IF(ALLOCATED( sk ) .AND. rslevel >= 2) complexcheck( 12 ) = 1
00116
00117 IF(ALLOCATED( zbo ) .AND. rslevel >= 1) complexcheck( 13 ) = 1
00118
00119 IF(ALLOCATED( zheevd_work ) .AND. rslevel == 0) complexcheck( 14 ) = 1
00120
00121 IF(ALLOCATED( zhjj ) .AND. rslevel == 0) complexcheck( 15 ) = 1
00122
00123 IF(ALLOCATED( diag_iwork ) .AND. rslevel == 0) integercheck( 2 ) = 1
00124
00125 IF(ALLOCATED( elempointer ) .AND. rslevel == 0) integercheck( 3 ) = 1
00126
00127 IF(ALLOCATED( ifail ) .AND. rslevel == 0) integercheck( 4 ) = 1
00128
00129 IF(ALLOCATED( matindlist ) .AND. rslevel == 0) integercheck( 5 ) = 1
00130
00131 IF(ALLOCATED( molid ) .AND. rslevel == 0) integercheck( 6 ) = 1
00132
00133 IF(ALLOCATED( nebcoul ) .AND. rslevel >= 2) integercheck( 7 ) = 1
00134
00135 IF(ALLOCATED( nebpp ) .AND. rslevel >= 2) integercheck( 8 ) = 1
00136
00137 IF(ALLOCATED( nebtb ) .AND. rslevel >= 2) integercheck( 9 ) = 1
00138
00139 IF(ALLOCATED( nono_iwork ) .AND. rslevel == 0) integercheck( 10 ) = 1
00140
00141 IF(ALLOCATED( pptablenth ) .AND. rslevel == 0) integercheck( 11 ) = 1
00142
00143 IF(ALLOCATED( rx ) .AND. rslevel == 0) integercheck( 12 ) = 1
00144
00145 IF(ALLOCATED( rxtmp ) .AND. rslevel == 0) integercheck( 13 ) = 1
00146
00147 IF(ALLOCATED( signlist ) .AND. rslevel == 0) integercheck( 14 ) = 1
00148
00149 IF(ALLOCATED( spinindlist ) .AND. rslevel == 0) integercheck( 15 ) = 1
00150
00151 IF(ALLOCATED( totnebcoul ) .AND. rslevel == 0) integercheck( 16 ) = 1
00152
00153 IF(ALLOCATED( totnebpp ) .AND. rslevel == 0) integercheck( 17 ) = 1
00154
00155 IF(ALLOCATED( totnebtb ) .AND. rslevel == 0) integercheck( 18 ) = 1
00156
00157 IF(ALLOCATED( xb ) .AND. rslevel == 0) integercheck( 19 ) = 1
00158
00159 IF(ALLOCATED( zheevd_iwork ) .AND. rslevel == 0) integercheck( 20 ) = 1
00160
00161 IF(ALLOCATED( atocc ) .AND. rslevel == 0) realcheck( 2 ) = 1
00162
00163 IF(ALLOCATED( bo ) .AND. rslevel >= 1) realcheck( 3 ) = 1
00164
00165 IF(ALLOCATED( bond ) .AND. rslevel >= 1) realcheck( 4 ) = 1
00166
00167 IF(ALLOCATED( bozero ) .AND. rslevel == 0) realcheck( 5 ) = 1
00168
00169 IF(ALLOCATED( bo_padded ) .AND. rslevel >= 1) realcheck( 6 ) = 1
00170
00171 IF(ALLOCATED( chempot_pnk ) .AND. rslevel == 0) realcheck( 7 ) = 1
00172
00173 IF(ALLOCATED( coslist ) .AND. rslevel == 0) realcheck( 8 ) = 1
00174
00175 IF(ALLOCATED( coulombv ) .AND. rslevel == 0) realcheck( 9 ) = 1
00176
00177 IF(ALLOCATED( cplist ) .AND. rslevel == 0) realcheck( 10 ) = 1
00178
00179 IF(ALLOCATED( cr ) .AND. rslevel >= 1) realcheck( 11 ) = 1
00180
00181 IF(ALLOCATED( deltaq ) .AND. rslevel == 0) realcheck( 12 ) = 1
00182
00183 IF(ALLOCATED( deltaspin ) .AND. rslevel == 0) realcheck( 13 ) = 1
00184
00185 IF(ALLOCATED( diag_rwork ) .AND. rslevel == 0) realcheck( 14 ) = 1
00186
00187 IF(ALLOCATED( diag_work ) .AND. rslevel == 0) realcheck( 15 ) = 1
00188
00189 IF(ALLOCATED( downevals ) .AND. rslevel == 0) realcheck( 16 ) = 1
00190
00191 IF(ALLOCATED( downevects ) .AND. rslevel >= 1) realcheck( 17 ) = 1
00192
00193 IF(ALLOCATED( ehist ) .AND. rslevel == 0) realcheck( 18 ) = 1

```

```

00194
00195     IF(ALLOCATED( evals ) .AND. rslevel == 0) realcheck( 19 ) = 1
00196
00197     IF(ALLOCATED( evecs ) .AND. rslevel >= 1) realcheck( 20 ) = 1
00198
00199     IF(ALLOCATED( f ) .AND. rslevel >= 1) realcheck( 21 ) = 1
00200
00201     IF(ALLOCATED( fcoul ) .AND. rslevel >= 1) realcheck( 22 ) = 1
00202
00203     IF(ALLOCATED( fpp ) .AND. rslevel >= 1) realcheck( 23 ) = 1
00204
00205     IF(ALLOCATED( fpul ) .AND. rslevel >= 1) realcheck( 24 ) = 1
00206
00207     IF(ALLOCATED( franprev ) .AND. rslevel >= 1) realcheck( 25 ) = 1
00208
00209     IF(ALLOCATED( fscoul ) .AND. rslevel >= 1) realcheck( 26 ) = 1
00210
00211     IF(ALLOCATED( fsspin ) .AND. rslevel >= 1) realcheck( 27 ) = 1
00212
00213     IF(ALLOCATED( ftot ) .AND. rslevel >= 1) realcheck( 28 ) = 1
00214
00215     IF(ALLOCATED( h ) .AND. rslevel >= 1) realcheck( 29 ) = 1
00216
00217     IF(ALLOCATED( h0 ) .AND. rslevel >= 1) realcheck( 30 ) = 1
00218
00219     IF(ALLOCATED( h2vect ) .AND. rslevel == 0) realcheck( 31 ) = 1
00220
00221     IF(ALLOCATED( hdiag ) .AND. rslevel == 0) realcheck( 32 ) = 1
00222
00223     IF(ALLOCATED( hdown ) .AND. rslevel >= 1) realcheck( 33 ) = 1
00224
00225     IF(ALLOCATED( hed ) .AND. rslevel == 0) realcheck( 34 ) = 1
00226
00227     IF(ALLOCATED( hef ) .AND. rslevel == 0) realcheck( 35 ) = 1
00228
00229     IF(ALLOCATED( hep ) .AND. rslevel == 0) realcheck( 36 ) = 1
00230
00231     IF(ALLOCATED( hes ) .AND. rslevel == 0) realcheck( 37 ) = 1
00232
00233     IF(ALLOCATED( hjj ) .AND. rslevel == 0) realcheck( 38 ) = 1
00234
00235     IF(ALLOCATED( hr0 ) .AND. rslevel == 0) realcheck( 39 ) = 1
00236
00237     IF(ALLOCATED( hubbardu ) .AND. rslevel == 0) realcheck( 40 ) = 1
00238
00239     IF(ALLOCATED( hup ) .AND. rslevel >= 1) realcheck( 41 ) = 1
00240
00241     IF(ALLOCATED( kevals ) .AND. rslevel >= 1) realcheck( 42 ) = 1
00242
00243     IF(ALLOCATED( lcnshift ) .AND. rslevel == 0) realcheck( 43 ) = 1
00244
00245     IF(ALLOCATED( mass ) .AND. rslevel == 0) realcheck( 44 ) = 1
00246
00247     IF(ALLOCATED( mycharge ) .AND. rslevel == 0) realcheck( 45 ) = 1
00248
00249     IF(ALLOCATED( nonotmp ) .AND. rslevel >= 1) realcheck( 46 ) = 1
00250
00251     IF(ALLOCATED( nono_evals ) .AND. rslevel == 0) realcheck( 47 ) = 1
00252
00253     IF(ALLOCATED( nono_work ) .AND. rslevel == 0) realcheck( 48 ) = 1
00254
00255     IF(ALLOCATED( olddeltaqs ) .AND. rslevel == 0) realcheck( 49 ) = 1
00256
00257     IF(ALLOCATED( olddeltaspin ) .AND. rslevel == 0) realcheck( 50 ) = 1
00258
00259     IF(ALLOCATED( orthoh ) .AND. rslevel >= 1) realcheck( 51 ) = 1
00260
00261     IF(ALLOCATED( orthohdown ) .AND. rslevel >= 1) realcheck( 52 ) = 1
00262
00263     IF(ALLOCATED( orthohup ) .AND. rslevel >= 1) realcheck( 53 ) = 1
00264
00265     IF(ALLOCATED( orthorho ) .AND. rslevel >= 1) realcheck( 54 ) = 1
00266
00267     IF(ALLOCATED( overl ) .AND. rslevel >= 1) realcheck( 55 ) = 1
00268
00269     IF(ALLOCATED( pair ) .AND. rslevel >= 1) realcheck( 56 ) = 1
00270
00271     IF(ALLOCATED( phist ) .AND. rslevel == 0) realcheck( 57 ) = 1
00272
00273     IF(ALLOCATED( phistx ) .AND. rslevel == 0) realcheck( 58 ) = 1
00274
00275     IF(ALLOCATED( phisty ) .AND. rslevel == 0) realcheck( 59 ) = 1
00276
00277     IF(ALLOCATED( phistz ) .AND. rslevel == 0) realcheck( 60 ) = 1
00278
00279     IF(ALLOCATED( pnk ) .AND. rslevel >= 1) realcheck( 61 ) = 1
00280

```

```

00281 IF(ALLOCATED( potcoef ) .AND. rslevel >= 1) realcheck( 62 ) = 1
00282
00283 IF(ALLOCATED( ppr ) .AND. rslevel >= 1) realcheck( 63 ) = 1
00284
00285 IF(ALLOCATED( ppspl ) .AND. rslevel >= 1) realcheck( 64 ) = 1
00286
00287 IF(ALLOCATED( ppval ) .AND. rslevel >= 1) realcheck( 65 ) = 1
00288
00289 IF(ALLOCATED( qlist ) .AND. rslevel == 0) realcheck( 66 ) = 1
00290
00291 IF(ALLOCATED( respchi ) .AND. rslevel == 0) realcheck( 67 ) = 1
00292
00293 IF(ALLOCATED( rhodown ) .AND. rslevel >= 1) realcheck( 68 ) = 1
00294
00295 IF(ALLOCATED( rhodownzero ) .AND. rslevel == 0) realcheck( 69 ) = 1
00296
00297 IF(ALLOCATED( rhoup ) .AND. rslevel >= 1) realcheck( 70 ) = 1
00298
00299 IF(ALLOCATED( rhoupzero ) .AND. rslevel == 0) realcheck( 71 ) = 1
00300
00301 IF(ALLOCATED( sh2 ) .AND. rslevel >= 1) realcheck( 72 ) = 1
00302
00303 IF(ALLOCATED( sinlist ) .AND. rslevel == 0) realcheck( 73 ) = 1
00304
00305 IF(ALLOCATED( smat ) .AND. rslevel >= 1) realcheck( 74 ) = 1
00306
00307 IF(ALLOCATED( spinlist ) .AND. rslevel == 0) realcheck( 75 ) = 1
00308
00309 IF(ALLOCATED( spintmp ) .AND. rslevel >= 1) realcheck( 76 ) = 1
00310
00311 IF(ALLOCATED( spin_pnk ) .AND. rslevel >= 1) realcheck( 77 ) = 1
00312
00313 IF(ALLOCATED( thist ) .AND. rslevel == 0) realcheck( 78 ) = 1
00314
00315 IF(ALLOCATED( tmpbodiag ) .AND. rslevel == 0) realcheck( 79 ) = 1
00316
00317 IF(ALLOCATED( tmprhodown ) .AND. rslevel == 0) realcheck( 80 ) = 1
00318
00319 IF(ALLOCATED( tmprhoup ) .AND. rslevel == 0) realcheck( 81 ) = 1
00320
00321 IF(ALLOCATED( twoxx2 ) .AND. rslevel >= 1) realcheck( 82 ) = 1
00322
00323 IF(ALLOCATED( umat ) .AND. rslevel >= 1) realcheck( 83 ) = 1
00324
00325 IF(ALLOCATED( upevals ) .AND. rslevel == 0) realcheck( 84 ) = 1
00326
00327 IF(ALLOCATED( upevecs ) .AND. rslevel >= 1) realcheck( 85 ) = 1
00328
00329 IF(ALLOCATED( v ) .AND. rslevel >= 1) realcheck( 86 ) = 1
00330
00331 IF(ALLOCATED( atele ) .AND. rslevel >= 0) charactercheck( 2 ) = 1
00332
00333 IF(ALLOCATED( basis ) .AND. rslevel >= 0) charactercheck( 3 ) = 1
00334
00335 IF(ALLOCATED( btype ) .AND. rslevel >= 0) charactercheck( 4 ) = 1
00336
00337 IF(ALLOCATED( ele ) .AND. rslevel >= 0) charactercheck( 5 ) = 1
00338
00339 IF(ALLOCATED( ele1 ) .AND. rslevel >= 0) charactercheck( 6 ) = 1
00340
00341 IF(ALLOCATED( ele2 ) .AND. rslevel >= 0) charactercheck( 7 ) = 1
00342
00343 IF(ALLOCATED( ppele1 ) .AND. rslevel >= 0) charactercheck( 8 ) = 1
00344
00345 IF(ALLOCATED( ppele2 ) .AND. rslevel >= 0) charactercheck( 9 ) = 1
00346
00347 IF(ALLOCATED( diag_zwork ) .AND. rslevel >= 0) complexcheck( 2 ) = 1
00348
00349 IF(ALLOCATED( hk ) .AND. rslevel >= 2) complexcheck( 3 ) = 1
00350
00351 IF(ALLOCATED( hk0 ) .AND. rslevel >= 2) complexcheck( 4 ) = 1
00352
00353 IF(ALLOCATED( hkdiag ) .AND. rslevel >= 1) complexcheck( 5 ) = 1
00354
00355 IF(ALLOCATED( kbo ) .AND. rslevel >= 2) complexcheck( 6 ) = 1
00356
00357 IF(ALLOCATED( kevecs ) .AND. rslevel >= 2) complexcheck( 7 ) = 1
00358
00359 IF(ALLOCATED( kf ) .AND. rslevel >= 1) complexcheck( 8 ) = 1
00360
00361 IF(ALLOCATED( khtmp ) .AND. rslevel >= 1) complexcheck( 9 ) = 1
00362
00363 IF(ALLOCATED( korthoh ) .AND. rslevel >= 2) complexcheck( 10 ) = 1
00364
00365 IF(ALLOCATED( kxmat ) .AND. rslevel >= 2) complexcheck( 11 ) = 1
00366
00367 IF(ALLOCATED( sk ) .AND. rslevel >= 2) complexcheck( 12 ) = 1

```

```

00368
00369 IF(ALLOCATED( zbo ) .AND. rslevel >= 1) complexcheck( 13 ) = 1
00370
00371 IF(ALLOCATED( zheevd_work ) .AND. rslevel >= 0) complexcheck( 14 ) = 1
00372
00373 IF(ALLOCATED( zhjj ) .AND. rslevel >= 0) complexcheck( 15 ) = 1
00374
00375 IF(ALLOCATED( diag_iwork ) .AND. rslevel >= 0) integercheck( 2 ) = 1
00376
00377 IF(ALLOCATED( elempointer ) .AND. rslevel >= 0) integercheck( 3 ) = 1
00378
00379 IF(ALLOCATED( ifail ) .AND. rslevel >= 0) integercheck( 4 ) = 1
00380
00381 IF(ALLOCATED( matindlist ) .AND. rslevel >= 0) integercheck( 5 ) = 1
00382
00383 IF(ALLOCATED( molid ) .AND. rslevel >= 0) integercheck( 6 ) = 1
00384
00385 IF(ALLOCATED( nebcoul ) .AND. rslevel >= 2) integercheck( 7 ) = 1
00386
00387 IF(ALLOCATED( nebpp ) .AND. rslevel >= 2) integercheck( 8 ) = 1
00388
00389 IF(ALLOCATED( nebtb ) .AND. rslevel >= 2) integercheck( 9 ) = 1
00390
00391 IF(ALLOCATED( nono_iwork ) .AND. rslevel >= 0) integercheck( 10 ) = 1
00392
00393 IF(ALLOCATED( pptablenth ) .AND. rslevel >= 0) integercheck( 11 ) = 1
00394
00395 IF(ALLOCATED( rx ) .AND. rslevel >= 0) integercheck( 12 ) = 1
00396
00397 IF(ALLOCATED( rxtmp ) .AND. rslevel >= 0) integercheck( 13 ) = 1
00398
00399 IF(ALLOCATED( signlist ) .AND. rslevel >= 0) integercheck( 14 ) = 1
00400
00401 IF(ALLOCATED( spinindlist ) .AND. rslevel >= 0) integercheck( 15 ) = 1
00402
00403 IF(ALLOCATED( totnebcoul ) .AND. rslevel >= 0) integercheck( 16 ) = 1
00404
00405 IF(ALLOCATED( totnebpp ) .AND. rslevel >= 0) integercheck( 17 ) = 1
00406
00407 IF(ALLOCATED( totnebtb ) .AND. rslevel >= 0) integercheck( 18 ) = 1
00408
00409 IF(ALLOCATED( xb ) .AND. rslevel >= 0) integercheck( 19 ) = 1
00410
00411 IF(ALLOCATED( zheevd_iwork ) .AND. rslevel >= 0) integercheck( 20 ) = 1
00412
00413 IF(ALLOCATED( atocc ) .AND. rslevel >= 0) realcheck( 2 ) = 1
00414
00415 IF(ALLOCATED( bo ) .AND. rslevel >= 1) realcheck( 3 ) = 1
00416
00417 IF(ALLOCATED( bond ) .AND. rslevel >= 1) realcheck( 4 ) = 1
00418
00419 IF(ALLOCATED( bozero ) .AND. rslevel >= 0) realcheck( 5 ) = 1
00420
00421 IF(ALLOCATED( bo_padded ) .AND. rslevel >= 1) realcheck( 6 ) = 1
00422
00423 IF(ALLOCATED( chempot_pnk ) .AND. rslevel >= 0) realcheck( 7 ) = 1
00424
00425 IF(ALLOCATED( coslist ) .AND. rslevel >= 0) realcheck( 8 ) = 1
00426
00427 IF(ALLOCATED( coulombv ) .AND. rslevel >= 0) realcheck( 9 ) = 1
00428
00429 IF(ALLOCATED( cplist ) .AND. rslevel >= 0) realcheck( 10 ) = 1
00430
00431 IF(ALLOCATED( cr ) .AND. rslevel >= 1) realcheck( 11 ) = 1
00432
00433 IF(ALLOCATED( deltaq ) .AND. rslevel >= 0) realcheck( 12 ) = 1
00434
00435 IF(ALLOCATED( deltaspin ) .AND. rslevel >= 0) realcheck( 13 ) = 1
00436
00437 IF(ALLOCATED( diag_rwork ) .AND. rslevel >= 0) realcheck( 14 ) = 1
00438
00439 IF(ALLOCATED( diag_work ) .AND. rslevel >= 0) realcheck( 15 ) = 1
00440
00441 IF(ALLOCATED( downevals ) .AND. rslevel >= 0) realcheck( 16 ) = 1
00442
00443 IF(ALLOCATED( downevects ) .AND. rslevel >= 1) realcheck( 17 ) = 1
00444
00445 IF(ALLOCATED( ehist ) .AND. rslevel >= 0) realcheck( 18 ) = 1
00446
00447 IF(ALLOCATED( evals ) .AND. rslevel >= 0) realcheck( 19 ) = 1
00448
00449 IF(ALLOCATED( evecs ) .AND. rslevel >= 1) realcheck( 20 ) = 1
00450
00451 IF(ALLOCATED( f ) .AND. rslevel >= 1) realcheck( 21 ) = 1
00452
00453 IF(ALLOCATED( fcoul ) .AND. rslevel >= 1) realcheck( 22 ) = 1
00454

```

```
00455 IF(ALLOCATED( fpp ) .AND. rslevel >= 1) realcheck( 23 ) = 1
00456
00457 IF(ALLOCATED( fpul ) .AND. rslevel >= 1) realcheck( 24 ) = 1
00458
00459 IF(ALLOCATED( franprev ) .AND. rslevel >= 1) realcheck( 25 ) = 1
00460
00461 IF(ALLOCATED( fscoul ) .AND. rslevel >= 1) realcheck( 26 ) = 1
00462
00463 IF(ALLOCATED( fsspin ) .AND. rslevel >= 1) realcheck( 27 ) = 1
00464
00465 IF(ALLOCATED( ftot ) .AND. rslevel >= 1) realcheck( 28 ) = 1
00466
00467 IF(ALLOCATED( h ) .AND. rslevel >= 1) realcheck( 29 ) = 1
00468
00469 IF(ALLOCATED( h0 ) .AND. rslevel >= 1) realcheck( 30 ) = 1
00470
00471 IF(ALLOCATED( h2vect ) .AND. rslevel >= 0) realcheck( 31 ) = 1
00472
00473 IF(ALLOCATED( hdiag ) .AND. rslevel >= 0) realcheck( 32 ) = 1
00474
00475 IF(ALLOCATED( hdown ) .AND. rslevel >= 1) realcheck( 33 ) = 1
00476
00477 IF(ALLOCATED( hed ) .AND. rslevel >= 0) realcheck( 34 ) = 1
00478
00479 IF(ALLOCATED( hef ) .AND. rslevel >= 0) realcheck( 35 ) = 1
00480
00481 IF(ALLOCATED( hep ) .AND. rslevel >= 0) realcheck( 36 ) = 1
00482
00483 IF(ALLOCATED( hes ) .AND. rslevel >= 0) realcheck( 37 ) = 1
00484
00485 IF(ALLOCATED( hjj ) .AND. rslevel >= 0) realcheck( 38 ) = 1
00486
00487 IF(ALLOCATED( hr0 ) .AND. rslevel >= 0) realcheck( 39 ) = 1
00488
00489 IF(ALLOCATED( hubbardu ) .AND. rslevel >= 0) realcheck( 40 ) = 1
00490
00491 IF(ALLOCATED( hup ) .AND. rslevel >= 1) realcheck( 41 ) = 1
00492
00493 IF(ALLOCATED( kevals ) .AND. rslevel >= 1) realcheck( 42 ) = 1
00494
00495 IF(ALLOCATED( lcnshift ) .AND. rslevel >= 0) realcheck( 43 ) = 1
00496
00497 IF(ALLOCATED( mass ) .AND. rslevel >= 0) realcheck( 44 ) = 1
00498
00499 IF(ALLOCATED( mycharge ) .AND. rslevel >= 0) realcheck( 45 ) = 1
00500
00501 IF(ALLOCATED( nonotmp ) .AND. rslevel >= 1) realcheck( 46 ) = 1
00502
00503 IF(ALLOCATED( nono_evals ) .AND. rslevel >= 0) realcheck( 47 ) = 1
00504
00505 IF(ALLOCATED( nono_work ) .AND. rslevel >= 0) realcheck( 48 ) = 1
00506
00507 IF(ALLOCATED( olddeltags ) .AND. rslevel >= 0) realcheck( 49 ) = 1
00508
00509 IF(ALLOCATED( olddeltaspin ) .AND. rslevel >= 0) realcheck( 50 ) = 1
00510
00511 IF(ALLOCATED( orthoh ) .AND. rslevel >= 1) realcheck( 51 ) = 1
00512
00513 IF(ALLOCATED( orthohdown ) .AND. rslevel >= 1) realcheck( 52 ) = 1
00514
00515 IF(ALLOCATED( orthohup ) .AND. rslevel >= 1) realcheck( 53 ) = 1
00516
00517 IF(ALLOCATED( orthorho ) .AND. rslevel >= 1) realcheck( 54 ) = 1
00518
00519 IF(ALLOCATED( overl ) .AND. rslevel >= 1) realcheck( 55 ) = 1
00520
00521 IF(ALLOCATED( pair ) .AND. rslevel >= 1) realcheck( 56 ) = 1
00522
00523 IF(ALLOCATED( phist ) .AND. rslevel >= 0) realcheck( 57 ) = 1
00524
00525 IF(ALLOCATED( phistx ) .AND. rslevel >= 0) realcheck( 58 ) = 1
00526
00527 IF(ALLOCATED( phisty ) .AND. rslevel >= 0) realcheck( 59 ) = 1
00528
00529 IF(ALLOCATED( phistz ) .AND. rslevel >= 0) realcheck( 60 ) = 1
00530
00531 IF(ALLOCATED( pnk ) .AND. rslevel >= 1) realcheck( 61 ) = 1
00532
00533 IF(ALLOCATED( potcoef ) .AND. rslevel >= 1) realcheck( 62 ) = 1
00534
00535 IF(ALLOCATED( ppr ) .AND. rslevel >= 1) realcheck( 63 ) = 1
00536
00537 IF(ALLOCATED( ppspl ) .AND. rslevel >= 1) realcheck( 64 ) = 1
00538
00539 IF(ALLOCATED( ppval ) .AND. rslevel >= 1) realcheck( 65 ) = 1
00540
00541 IF(ALLOCATED( qlist ) .AND. rslevel >= 0) realcheck( 66 ) = 1
```

```

00542
00543   IF(ALLOCATED( respchi ) .AND. rslevel >= 0) realcheck( 67 ) = 1
00544
00545   IF(ALLOCATED( rhodown ) .AND. rslevel >= 1) realcheck( 68 ) = 1
00546
00547   IF(ALLOCATED( rhodownzero ) .AND. rslevel >= 0) realcheck( 69 ) = 1
00548
00549   IF(ALLOCATED( rhoup ) .AND. rslevel >= 1) realcheck( 70 ) = 1
00550
00551   IF(ALLOCATED( rhoupzero ) .AND. rslevel >= 0) realcheck( 71 ) = 1
00552
00553   IF(ALLOCATED( sh2 ) .AND. rslevel >= 1) realcheck( 72 ) = 1
00554
00555   IF(ALLOCATED( sinlist ) .AND. rslevel >= 0) realcheck( 73 ) = 1
00556
00557   IF(ALLOCATED( smat ) .AND. rslevel >= 1) realcheck( 74 ) = 1
00558
00559   IF(ALLOCATED( spinlist ) .AND. rslevel >= 0) realcheck( 75 ) = 1
00560
00561   IF(ALLOCATED( spintmp ) .AND. rslevel >= 1) realcheck( 76 ) = 1
00562
00563   IF(ALLOCATED( spin_pnk ) .AND. rslevel >= 1) realcheck( 77 ) = 1
00564
00565   IF(ALLOCATED( thist ) .AND. rslevel >= 0) realcheck( 78 ) = 1
00566
00567   IF(ALLOCATED( tmpbodiag ) .AND. rslevel >= 0) realcheck( 79 ) = 1
00568
00569   IF(ALLOCATED( tmprhodown ) .AND. rslevel >= 0) realcheck( 80 ) = 1
00570
00571   IF(ALLOCATED( tmprhoup ) .AND. rslevel >= 0) realcheck( 81 ) = 1
00572
00573   IF(ALLOCATED( twoxx2 ) .AND. rslevel >= 1) realcheck( 82 ) = 1
00574
00575   IF(ALLOCATED( umat ) .AND. rslevel >= 1) realcheck( 83 ) = 1
00576
00577   IF(ALLOCATED( upevals ) .AND. rslevel >= 0) realcheck( 84 ) = 1
00578
00579   IF(ALLOCATED( upevecs ) .AND. rslevel >= 1) realcheck( 85 ) = 1
00580
00581   IF(ALLOCATED( v ) .AND. rslevel >= 1) realcheck( 86 ) = 1
00582
00583   IF(ALLOCATED( vhist ) .AND. rslevel >= 0) realcheck( 87 ) = 1
00584
00585   IF(ALLOCATED( wdd ) .AND. rslevel >= 0) realcheck( 88 ) = 1
00586
00587   IF(ALLOCATED( wff ) .AND. rslevel >= 0) realcheck( 89 ) = 1
00588
00589   IF(ALLOCATED( work ) .AND. rslevel >= 0) realcheck( 90 ) = 1
00590
00591   IF(ALLOCATED( wpp ) .AND. rslevel >= 0) realcheck( 91 ) = 1
00592
00593   IF(ALLOCATED( wss ) .AND. rslevel >= 0) realcheck( 92 ) = 1
00594
00595   IF(ALLOCATED( x2 ) .AND. rslevel >= 1) realcheck( 93 ) = 1
00596
00597   IF(ALLOCATED( x2down ) .AND. rslevel >= 1) realcheck( 94 ) = 1
00598
00599   IF(ALLOCATED( x2hrho ) .AND. rslevel >= 1) realcheck( 95 ) = 1
00600
00601   IF(ALLOCATED( x2up ) .AND. rslevel >= 1) realcheck( 96 ) = 1
00602
00603   IF(ALLOCATED( xmat ) .AND. rslevel >= 1) realcheck( 97 ) = 1
00604
00605   IF(ALLOCATED( zheevd_rwork ) .AND. rslevel >= 0) realcheck( 98 ) = 1
00606
00607   WRITE(14)charactercheck
00608   WRITE(14)complexcheck
00609   WRITE(14)integercheck
00610   WRITE(14)realcheck
00611
00612   WRITE(14)iter
00613
00614   IF(charactercheck( 2 ) == 1)THEN
00615       WRITE(14)SIZE( atele ,dim=1)
00616       WRITE(14) atele
00617   ENDIF
00618
00619   IF(charactercheck( 3 ) == 1)THEN
00620       WRITE(14)SIZE( basis ,dim=1)
00621       WRITE(14) basis
00622   ENDIF
00623
00624   IF(charactercheck( 4 ) == 1)THEN
00625       WRITE(14)SIZE( btype ,dim=1)
00626       WRITE(14) btype
00627   ENDIF
00628

```

```

00629 IF(charactercheck( 5 ) == 1)THEN
00630     WRITE(14)SIZE( ele ,dim=1)
00631     WRITE(14) ele
00632 ENDIF
00633
00634 IF(charactercheck( 6 ) == 1)THEN
00635     WRITE(14)SIZE( ele1 ,dim=1)
00636     WRITE(14) ele1
00637 ENDIF
00638
00639 IF(charactercheck( 7 ) == 1)THEN
00640     WRITE(14)SIZE( ele2 ,dim=1)
00641     WRITE(14) ele2
00642 ENDIF
00643
00644 IF(charactercheck( 8 ) == 1)THEN
00645     WRITE(14)SIZE( ppele1 ,dim=1)
00646     WRITE(14) ppele1
00647 ENDIF
00648
00649 IF(charactercheck( 9 ) == 1)THEN
00650     WRITE(14)SIZE( ppele2 ,dim=1)
00651     WRITE(14) ppele2
00652 ENDIF
00653
00654 IF(complexcheck( 2 ) == 1)THEN
00655     WRITE(14)SIZE( diag_zwork ,dim=1)
00656     WRITE(14) diag_zwork
00657 ENDIF
00658
00659 IF(complexcheck( 3 ) == 1)THEN
00660     WRITE(14)SIZE( hk ,dim=1) , SIZE( hk ,dim=2) , SIZE( hk ,dim=3)
00661     WRITE(14) hk
00662 ENDIF
00663
00664 IF(complexcheck( 4 ) == 1)THEN
00665     WRITE(14)SIZE( hk0 ,dim=1) , SIZE( hk0 ,dim=2) , SIZE( hk0 ,dim=3)
00666     WRITE(14) hk0
00667 ENDIF
00668
00669 IF(complexcheck( 5 ) == 1)THEN
00670     WRITE(14)SIZE( hkdiag ,dim=1) , SIZE( hkdiag ,dim=2)
00671     WRITE(14) hkdiag
00672 ENDIF
00673
00674 IF(complexcheck( 6 ) == 1)THEN
00675     WRITE(14)SIZE( kbo ,dim=1) , SIZE( kbo ,dim=2) , SIZE( kbo ,dim=3)
00676     WRITE(14) kbo
00677 ENDIF
00678
00679 IF(complexcheck( 7 ) == 1)THEN
00680     WRITE(14)SIZE( kevecs ,dim=1) , SIZE( kevecs ,dim=2) , SIZE(
kevecs ,dim=3)
00681     WRITE(14) kevecs
00682 ENDIF
00683
00684 IF(complexcheck( 8 ) == 1)THEN
00685     WRITE(14)SIZE( kf ,dim=1) , SIZE( kf ,dim=2)
00686     WRITE(14) kf
00687 ENDIF
00688
00689 IF(complexcheck( 9 ) == 1)THEN
00690     WRITE(14)SIZE( khtmp ,dim=1) , SIZE( khtmp ,dim=2)
00691     WRITE(14) khtmp
00692 ENDIF
00693
00694 IF(complexcheck( 10 ) == 1)THEN
00695     WRITE(14)SIZE( korthoh ,dim=1) , SIZE( korthoh ,dim=2) , SIZE(
korthoh ,dim=3)
00696     WRITE(14) korthoh
00697 ENDIF
00698
00699 IF(complexcheck( 11 ) == 1)THEN
00700     WRITE(14)SIZE( kxmat ,dim=1) , SIZE( kxmat ,dim=2) , SIZE( kxmat ,dim=3)
00701     WRITE(14) kxmat
00702 ENDIF
00703
00704 IF(complexcheck( 12 ) == 1)THEN
00705     WRITE(14)SIZE( sk ,dim=1) , SIZE( sk ,dim=2) , SIZE( sk ,dim=3)
00706     WRITE(14) sk
00707 ENDIF
00708
00709 IF(complexcheck( 13 ) == 1)THEN
00710     WRITE(14)SIZE( zbo ,dim=1) , SIZE( zbo ,dim=2)
00711     WRITE(14) zbo
00712 ENDIF
00713

```



```

00714 IF(complexcheck( 14 ) == 1)THEN
00715     WRITE(14)SIZE( zheevd_work ,dim=1)
00716     WRITE(14) zheevd_work
00717 ENDIF
00718
00719 IF(complexcheck( 15 ) == 1)THEN
00720     WRITE(14)SIZE( zhjj ,dim=1)
00721     WRITE(14) zhjj
00722 ENDIF
00723
00724 IF(integercheck( 2 ) == 1)THEN
00725     WRITE(14)SIZE( diag_iwork ,dim=1)
00726     WRITE(14) diag_iwork
00727 ENDIF
00728
00729 IF(integercheck( 3 ) == 1)THEN
00730     WRITE(14)SIZE( elempointer ,dim=1)
00731     WRITE(14) elempointer
00732 ENDIF
00733
00734 IF(integercheck( 4 ) == 1)THEN
00735     WRITE(14)SIZE( ifail ,dim=1)
00736     WRITE(14) ifail
00737 ENDIF
00738
00739 IF(integercheck( 5 ) == 1)THEN
00740     WRITE(14)SIZE( matindlist ,dim=1)
00741     WRITE(14) matindlist
00742 ENDIF
00743
00744 IF(integercheck( 6 ) == 1)THEN
00745     WRITE(14)SIZE( molid ,dim=1)
00746     WRITE(14) molid
00747 ENDIF
00748
00749 IF(integercheck( 7 ) == 1)THEN
00750     WRITE(14)SIZE( nebcoul ,dim=1) , SIZE( nebcoul ,dim=2) , SIZE(
nebcoul ,dim=3)
00751     WRITE(14) nebcoul
00752 ENDIF
00753
00754 IF(integercheck( 8 ) == 1)THEN
00755     WRITE(14)SIZE( nebpp ,dim=1) , SIZE( nebpp ,dim=2) , SIZE( nebpp ,dim=3)
00756     WRITE(14) nebpp
00757 ENDIF
00758
00759 IF(integercheck( 9 ) == 1)THEN
00760     WRITE(14)SIZE( nebtb ,dim=1) , SIZE( nebtb ,dim=2) , SIZE( nebtb ,dim=3)
00761     WRITE(14) nebtb
00762 ENDIF
00763
00764 IF(integercheck( 10 ) == 1)THEN
00765     WRITE(14)SIZE( nono_iwork ,dim=1)
00766     WRITE(14) nono_iwork
00767 ENDIF
00768
00769 IF(integercheck( 11 ) == 1)THEN
00770     WRITE(14)SIZE( pptablength ,dim=1)
00771     WRITE(14) pptablength
00772 ENDIF
00773
00774 IF(integercheck( 12 ) == 1)THEN
00775     WRITE(14)SIZE( rx ,dim=1)
00776     WRITE(14) rx
00777 ENDIF
00778
00779 IF(integercheck( 13 ) == 1)THEN
00780     WRITE(14)SIZE( rxtmp ,dim=1)
00781     WRITE(14) rxtmp
00782 ENDIF
00783
00784 IF(integercheck( 14 ) == 1)THEN
00785     WRITE(14)SIZE( signlist ,dim=1)
00786     WRITE(14) signlist
00787 ENDIF
00788
00789 IF(integercheck( 15 ) == 1)THEN
00790     WRITE(14)SIZE( spinindlist ,dim=1)
00791     WRITE(14) spinindlist
00792 ENDIF
00793
00794 IF(integercheck( 16 ) == 1)THEN
00795     WRITE(14)SIZE( totnebcoul ,dim=1)
00796     WRITE(14) totnebcoul
00797 ENDIF
00798
00799 IF(integercheck( 17 ) == 1)THEN

```

```

00800      WRITE(14)SIZE( totnebpp ,dim=1)
00801      WRITE(14) totnebpp
00802  ENDIF
00803
00804  IF(integercheck( 18 ) == 1)THEN
00805      WRITE(14)SIZE( totnebtb ,dim=1)
00806      WRITE(14) totnebtb
00807  ENDIF
00808
00809  IF(integercheck( 19 ) == 1)THEN
00810      WRITE(14)SIZE( xb ,dim=1)
00811      WRITE(14) xb
00812  ENDIF
00813
00814  IF(integercheck( 20 ) == 1)THEN
00815      WRITE(14)SIZE( zheevd_iwork ,dim=1)
00816      WRITE(14) zheevd_iwork
00817  ENDIF
00818
00819  IF(realcheck( 2 ) == 1)THEN
00820      WRITE(14)SIZE( atocc ,dim=1)
00821      WRITE(14) atocc
00822  ENDIF
00823
00824  IF(realcheck( 3 ) == 1)THEN
00825      WRITE(14)SIZE( bo ,dim=1) , SIZE( bo ,dim=2)
00826      WRITE(14) bo
00827  ENDIF
00828
00829  IF(realcheck( 4 ) == 1)THEN
00830      WRITE(14)SIZE( bond ,dim=1) , SIZE( bond ,dim=2)
00831      WRITE(14) bond
00832  ENDIF
00833
00834  IF(realcheck( 5 ) == 1)THEN
00835      WRITE(14)SIZE( bozero ,dim=1)
00836      WRITE(14) bozero
00837  ENDIF
00838
00839  IF(realcheck( 6 ) == 1)THEN
00840      WRITE(14)SIZE( bo_padded ,dim=1) , SIZE( bo_padded ,dim=2)
00841      WRITE(14) bo_padded
00842  ENDIF
00843
00844  IF(realcheck( 7 ) == 1)THEN
00845      WRITE(14)SIZE( chempot_pnk ,dim=1)
00846      WRITE(14) chempot_pnk
00847  ENDIF
00848
00849  IF(realcheck( 8 ) == 1)THEN
00850      WRITE(14)SIZE( coslist ,dim=1)
00851      WRITE(14) coslist
00852  ENDIF
00853
00854  IF(realcheck( 9 ) == 1)THEN
00855      WRITE(14)SIZE( coulombv ,dim=1)
00856      WRITE(14) coulombv
00857  ENDIF
00858
00859  IF(realcheck( 10 ) == 1)THEN
00860      WRITE(14)SIZE( cplist ,dim=1)
00861      WRITE(14) cplist
00862  ENDIF
00863
00864  IF(realcheck( 11 ) == 1)THEN
00865      WRITE(14)SIZE( cr ,dim=1) , SIZE( cr ,dim=2)
00866      WRITE(14) cr
00867  ENDIF
00868
00869  IF(realcheck( 12 ) == 1)THEN
00870      WRITE(14)SIZE( deltaq ,dim=1)
00871      WRITE(14) deltaq
00872  ENDIF
00873
00874  IF(realcheck( 13 ) == 1)THEN
00875      WRITE(14)SIZE( deltaspin ,dim=1)
00876      WRITE(14) deltaspin
00877  ENDIF
00878
00879  IF(realcheck( 14 ) == 1)THEN
00880      WRITE(14)SIZE( diag_rwork ,dim=1)
00881      WRITE(14) diag_rwork
00882  ENDIF
00883
00884  IF(realcheck( 15 ) == 1)THEN
00885      WRITE(14)SIZE( diag_work ,dim=1)
00886      WRITE(14) diag_work

```

```

00887     ENDIF
00888
00889     IF(realcheck( 16 ) == 1)THEN
00890         WRITE(14)SIZE( downevals ,dim=1)
00891         WRITE(14) downevals
00892     ENDIF
00893
00894     IF(realcheck( 17 ) == 1)THEN
00895         WRITE(14)SIZE( downvecs ,dim=1) , SIZE( downvecs ,dim=2)
00896         WRITE(14) downvecs
00897     ENDIF
00898
00899     IF(realcheck( 18 ) == 1)THEN
00900         WRITE(14)SIZE( ehist ,dim=1)
00901         WRITE(14) ehist
00902     ENDIF
00903
00904     IF(realcheck( 19 ) == 1)THEN
00905         WRITE(14)SIZE( evals ,dim=1)
00906         WRITE(14) evals
00907     ENDIF
00908
00909     IF(realcheck( 20 ) == 1)THEN
00910         WRITE(14)SIZE( evecs ,dim=1) , SIZE( evecs ,dim=2)
00911         WRITE(14) evecs
00912     ENDIF
00913
00914     IF(realcheck( 21 ) == 1)THEN
00915         WRITE(14)SIZE( f ,dim=1) , SIZE( f ,dim=2)
00916         WRITE(14) f
00917     ENDIF
00918
00919     IF(realcheck( 22 ) == 1)THEN
00920         WRITE(14)SIZE( fcoul ,dim=1) , SIZE( fcoul ,dim=2)
00921         WRITE(14) fcoul
00922     ENDIF
00923
00924     IF(realcheck( 23 ) == 1)THEN
00925         WRITE(14)SIZE( fpp ,dim=1) , SIZE( fpp ,dim=2)
00926         WRITE(14) fpp
00927     ENDIF
00928
00929     IF(realcheck( 24 ) == 1)THEN
00930         WRITE(14)SIZE( fpul ,dim=1) , SIZE( fpul ,dim=2)
00931         WRITE(14) fpul
00932     ENDIF
00933
00934     IF(realcheck( 25 ) == 1)THEN
00935         WRITE(14)SIZE( franprev ,dim=1) , SIZE( franprev ,dim=2)
00936         WRITE(14) franprev
00937     ENDIF
00938
00939     IF(realcheck( 26 ) == 1)THEN
00940         WRITE(14)SIZE( fscoul ,dim=1) , SIZE( fscoul ,dim=2)
00941         WRITE(14) fscoul
00942     ENDIF
00943
00944     IF(realcheck( 27 ) == 1)THEN
00945         WRITE(14)SIZE( fsspin ,dim=1) , SIZE( fsspin ,dim=2)
00946         WRITE(14) fsspin
00947     ENDIF
00948
00949     IF(realcheck( 28 ) == 1)THEN
00950         WRITE(14)SIZE( ftot ,dim=1) , SIZE( ftot ,dim=2)
00951         WRITE(14) ftot
00952     ENDIF
00953
00954     IF(realcheck( 29 ) == 1)THEN
00955         WRITE(14)SIZE( h ,dim=1) , SIZE( h ,dim=2)
00956         WRITE(14) h
00957     ENDIF
00958
00959     IF(realcheck( 30 ) == 1)THEN
00960         WRITE(14)SIZE( h0 ,dim=1) , SIZE( h0 ,dim=2)
00961         WRITE(14) h0
00962     ENDIF
00963
00964     IF(realcheck( 31 ) == 1)THEN
00965         WRITE(14)SIZE( h2vect ,dim=1)
00966         WRITE(14) h2vect
00967     ENDIF
00968
00969     IF(realcheck( 32 ) == 1)THEN
00970         WRITE(14)SIZE( hdiag ,dim=1)
00971         WRITE(14) hdiag
00972     ENDIF
00973

```

```

00974 IF(realcheck( 33 ) == 1)THEN
00975     WRITE(14)SIZE( hdown ,dim=1) , SIZE( hdown ,dim=2)
00976     WRITE(14) hdown
00977 ENDIF
00978
00979 IF(realcheck( 34 ) == 1)THEN
00980     WRITE(14)SIZE( hed ,dim=1)
00981     WRITE(14) hed
00982 ENDIF
00983
00984 IF(realcheck( 35 ) == 1)THEN
00985     WRITE(14)SIZE( hef ,dim=1)
00986     WRITE(14) hef
00987 ENDIF
00988
00989 IF(realcheck( 36 ) == 1)THEN
00990     WRITE(14)SIZE( hep ,dim=1)
00991     WRITE(14) hep
00992 ENDIF
00993
00994 IF(realcheck( 37 ) == 1)THEN
00995     WRITE(14)SIZE( hes ,dim=1)
00996     WRITE(14) hes
00997 ENDIF
00998
00999 IF(realcheck( 38 ) == 1)THEN
01000     WRITE(14)SIZE( hjj ,dim=1)
01001     WRITE(14) hjj
01002 ENDIF
01003
01004 IF(realcheck( 39 ) == 1)THEN
01005     WRITE(14)SIZE( hr0 ,dim=1)
01006     WRITE(14) hr0
01007 ENDIF
01008
01009 IF(realcheck( 40 ) == 1)THEN
01010     WRITE(14)SIZE( hubbardu ,dim=1)
01011     WRITE(14) hubbardu
01012 ENDIF
01013
01014 IF(realcheck( 41 ) == 1)THEN
01015     WRITE(14)SIZE( hup ,dim=1) , SIZE( hup ,dim=2)
01016     WRITE(14) hup
01017 ENDIF
01018
01019 IF(realcheck( 42 ) == 1)THEN
01020     WRITE(14)SIZE( kevals ,dim=1) , SIZE( kevals ,dim=2)
01021     WRITE(14) kevals
01022 ENDIF
01023
01024 IF(realcheck( 43 ) == 1)THEN
01025     WRITE(14)SIZE( lcnshift ,dim=1)
01026     WRITE(14) lcnshift
01027 ENDIF
01028
01029 IF(realcheck( 44 ) == 1)THEN
01030     WRITE(14)SIZE( mass ,dim=1)
01031     WRITE(14) mass
01032 ENDIF
01033
01034 IF(realcheck( 45 ) == 1)THEN
01035     WRITE(14)SIZE( mycharge ,dim=1)
01036     WRITE(14) mycharge
01037 ENDIF
01038
01039 IF(realcheck( 46 ) == 1)THEN
01040     WRITE(14)SIZE( nonotmp ,dim=1) , SIZE( nonotmp ,dim=2)
01041     WRITE(14) nonotmp
01042 ENDIF
01043
01044 IF(realcheck( 47 ) == 1)THEN
01045     WRITE(14)SIZE( nono_evals ,dim=1)
01046     WRITE(14) nono_evals
01047 ENDIF
01048
01049 IF(realcheck( 48 ) == 1)THEN
01050     WRITE(14)SIZE( nono_work ,dim=1)
01051     WRITE(14) nono_work
01052 ENDIF
01053
01054 IF(realcheck( 49 ) == 1)THEN
01055     WRITE(14)SIZE( olddeltags ,dim=1)
01056     WRITE(14) olddeltags
01057 ENDIF
01058
01059 IF(realcheck( 50 ) == 1)THEN
01060     WRITE(14)SIZE( olddeltaspin ,dim=1)

```

```

01061     WRITE(14) olddeltaspin
01062   ENDIF
01063
01064   IF(realcheck( 51 ) == 1) THEN
01065     WRITE(14) SIZE( orthoh ,dim=1) , SIZE( orthoh ,dim=2)
01066     WRITE(14) orthoh
01067   ENDIF
01068
01069   IF(realcheck( 52 ) == 1) THEN
01070     WRITE(14) SIZE( orthohdown ,dim=1) , SIZE( orthohdown ,dim=2)
01071     WRITE(14) orthohdown
01072   ENDIF
01073
01074   IF(realcheck( 53 ) == 1) THEN
01075     WRITE(14) SIZE( orthohup ,dim=1) , SIZE( orthohup ,dim=2)
01076     WRITE(14) orthohup
01077   ENDIF
01078
01079   IF(realcheck( 54 ) == 1) THEN
01080     WRITE(14) SIZE( orthorho ,dim=1) , SIZE( orthorho ,dim=2)
01081     WRITE(14) orthorho
01082   ENDIF
01083
01084   IF(realcheck( 55 ) == 1) THEN
01085     WRITE(14) SIZE( overl ,dim=1) , SIZE( overl ,dim=2)
01086     WRITE(14) overl
01087   ENDIF
01088
01089   IF(realcheck( 56 ) == 1) THEN
01090     WRITE(14) SIZE( pair ,dim=1) , SIZE( pair ,dim=2)
01091     WRITE(14) pair
01092   ENDIF
01093
01094   IF(realcheck( 57 ) == 1) THEN
01095     WRITE(14) SIZE( phist ,dim=1)
01096     WRITE(14) phist
01097   ENDIF
01098
01099   IF(realcheck( 58 ) == 1) THEN
01100     WRITE(14) SIZE( phistx ,dim=1)
01101     WRITE(14) phistx
01102   ENDIF
01103
01104   IF(realcheck( 59 ) == 1) THEN
01105     WRITE(14) SIZE( phisty ,dim=1)
01106     WRITE(14) phisty
01107   ENDIF
01108
01109   IF(realcheck( 60 ) == 1) THEN
01110     WRITE(14) SIZE( phistz ,dim=1)
01111     WRITE(14) phistz
01112   ENDIF
01113
01114   IF(realcheck( 61 ) == 1) THEN
01115     WRITE(14) SIZE( pnk ,dim=1) , SIZE( pnk ,dim=2)
01116     WRITE(14) pnk
01117   ENDIF
01118
01119   IF(realcheck( 62 ) == 1) THEN
01120     WRITE(14) SIZE( potcoef ,dim=1) , SIZE( potcoef ,dim=2)
01121     WRITE(14) potcoef
01122   ENDIF
01123
01124   IF(realcheck( 63 ) == 1) THEN
01125     WRITE(14) SIZE( ppr ,dim=1) , SIZE( ppr ,dim=2)
01126     WRITE(14) ppr
01127   ENDIF
01128
01129   IF(realcheck( 64 ) == 1) THEN
01130     WRITE(14) SIZE( ppspl ,dim=1) , SIZE( ppspl ,dim=2)
01131     WRITE(14) ppspl
01132   ENDIF
01133
01134   IF(realcheck( 65 ) == 1) THEN
01135     WRITE(14) SIZE( ppval ,dim=1) , SIZE( ppval ,dim=2)
01136     WRITE(14) ppval
01137   ENDIF
01138
01139   IF(realcheck( 66 ) == 1) THEN
01140     WRITE(14) SIZE( qlist ,dim=1)
01141     WRITE(14) qlist
01142   ENDIF
01143
01144   IF(realcheck( 67 ) == 1) THEN
01145     WRITE(14) SIZE( respchi ,dim=1)
01146     WRITE(14) respchi
01147   ENDIF

```

```

01148
01149 IF(realcheck( 68 ) == 1)THEN
01150     WRITE(14)SIZE( rhodown ,dim=1) , SIZE( rhodown ,dim=2)
01151     WRITE(14) rhodown
01152 ENDIF
01153
01154 IF(realcheck( 69 ) == 1)THEN
01155     WRITE(14)SIZE( rhodownzero ,dim=1)
01156     WRITE(14) rhodownzero
01157 ENDIF
01158
01159 IF(realcheck( 70 ) == 1)THEN
01160     WRITE(14)SIZE( rhoup ,dim=1) , SIZE( rhoup ,dim=2)
01161     WRITE(14) rhoup
01162 ENDIF
01163
01164 IF(realcheck( 71 ) == 1)THEN
01165     WRITE(14)SIZE( rhoupzero ,dim=1)
01166     WRITE(14) rhoupzero
01167 ENDIF
01168
01169 IF(realcheck( 72 ) == 1)THEN
01170     WRITE(14)SIZE( sh2 ,dim=1) , SIZE( sh2 ,dim=2)
01171     WRITE(14) sh2
01172 ENDIF
01173
01174 IF(realcheck( 73 ) == 1)THEN
01175     WRITE(14)SIZE( sinlist ,dim=1)
01176     WRITE(14) sinlist
01177 ENDIF
01178
01179 IF(realcheck( 74 ) == 1)THEN
01180     WRITE(14)SIZE( smat ,dim=1) , SIZE( smat ,dim=2)
01181     WRITE(14) smat
01182 ENDIF
01183
01184 IF(realcheck( 75 ) == 1)THEN
01185     WRITE(14)SIZE( spinlist ,dim=1)
01186     WRITE(14) spinlist
01187 ENDIF
01188
01189 IF(realcheck( 76 ) == 1)THEN
01190     WRITE(14)SIZE( spintmp ,dim=1) , SIZE( spintmp ,dim=2)
01191     WRITE(14) spintmp
01192 ENDIF
01193
01194 IF(realcheck( 77 ) == 1)THEN
01195     WRITE(14)SIZE( spin_pnk ,dim=1) , SIZE( spin_pnk ,dim=2)
01196     WRITE(14) spin_pnk
01197 ENDIF
01198
01199 IF(realcheck( 78 ) == 1)THEN
01200     WRITE(14)SIZE( thist ,dim=1)
01201     WRITE(14) thist
01202 ENDIF
01203
01204 IF(realcheck( 79 ) == 1)THEN
01205     WRITE(14)SIZE( tmpbodiag ,dim=1)
01206     WRITE(14) tmpbodiag
01207 ENDIF
01208
01209 IF(realcheck( 80 ) == 1)THEN
01210     WRITE(14)SIZE( tmprhodown ,dim=1)
01211     WRITE(14) tmprhodown
01212 ENDIF
01213
01214 IF(realcheck( 81 ) == 1)THEN
01215     WRITE(14)SIZE( tmprhoup ,dim=1)
01216     WRITE(14) tmprhoup
01217 ENDIF
01218
01219 IF(realcheck( 82 ) == 1)THEN
01220     WRITE(14)SIZE( twoxx2 ,dim=1) , SIZE( twoxx2 ,dim=2)
01221     WRITE(14) twoxx2
01222 ENDIF
01223
01224 IF(realcheck( 83 ) == 1)THEN
01225     WRITE(14)SIZE( umat ,dim=1) , SIZE( umat ,dim=2)
01226     WRITE(14) umat
01227 ENDIF
01228
01229 IF(realcheck( 84 ) == 1)THEN
01230     WRITE(14)SIZE( upevals ,dim=1)
01231     WRITE(14) upevals
01232 ENDIF
01233
01234 IF(realcheck( 85 ) == 1)THEN

```

```

01235      WRITE(14)SIZE( upevecs ,dim=1) , SIZE( upevecs ,dim=2)
01236      WRITE(14) upevecs
01237  ENDIF
01238
01239  IF(realcheck( 86 ) == 1)THEN
01240      WRITE(14)SIZE( v ,dim=1) , SIZE( v ,dim=2)
01241      WRITE(14) v
01242  ENDIF
01243
01244  IF(realcheck( 87 ) == 1)THEN
01245      WRITE(14)SIZE( vhist ,dim=1)
01246      WRITE(14) vhist
01247  ENDIF
01248
01249  IF(realcheck( 88 ) == 1)THEN
01250      WRITE(14)SIZE( wdd ,dim=1)
01251      WRITE(14) wdd
01252  ENDIF
01253
01254  IF(realcheck( 89 ) == 1)THEN
01255      WRITE(14)SIZE( wff ,dim=1)
01256      WRITE(14) wff
01257  ENDIF
01258
01259  IF(realcheck( 90 ) == 1)THEN
01260      WRITE(14)SIZE( work ,dim=1)
01261      WRITE(14) work
01262  ENDIF
01263
01264  IF(realcheck( 91 ) == 1)THEN
01265      WRITE(14)SIZE( wpp ,dim=1)
01266      WRITE(14) wpp
01267  ENDIF
01268
01269  IF(realcheck( 92 ) == 1)THEN
01270      WRITE(14)SIZE( wss ,dim=1)
01271      WRITE(14) wss
01272  ENDIF
01273
01274  IF(realcheck( 93 ) == 1)THEN
01275      WRITE(14)SIZE( x2 ,dim=1) , SIZE( x2 ,dim=2)
01276      WRITE(14) x2
01277  ENDIF
01278
01279  IF(realcheck( 94 ) == 1)THEN
01280      WRITE(14)SIZE( x2down ,dim=1) , SIZE( x2down ,dim=2)
01281      WRITE(14) x2down
01282  ENDIF
01283
01284  IF(realcheck( 95 ) == 1)THEN
01285      WRITE(14)SIZE( x2hrho ,dim=1) , SIZE( x2hrho ,dim=2)
01286      WRITE(14) x2hrho
01287  ENDIF
01288
01289  IF(realcheck( 96 ) == 1)THEN
01290      WRITE(14)SIZE( x2up ,dim=1) , SIZE( x2up ,dim=2)
01291      WRITE(14) x2up
01292  ENDIF
01293
01294  IF(realcheck( 97 ) == 1)THEN
01295      WRITE(14)SIZE( xmat ,dim=1) , SIZE( xmat ,dim=2)
01296      WRITE(14) xmat
01297  ENDIF
01298
01299  IF(realcheck( 98 ) == 1)THEN
01300      WRITE(14)SIZE( zheevd_rwork ,dim=1)
01301      WRITE(14) zheevd_rwork
01302  ENDIF
01303
01304  CLOSE(14)
01305
01306  END SUBROUTINE wrtrestartlib

```

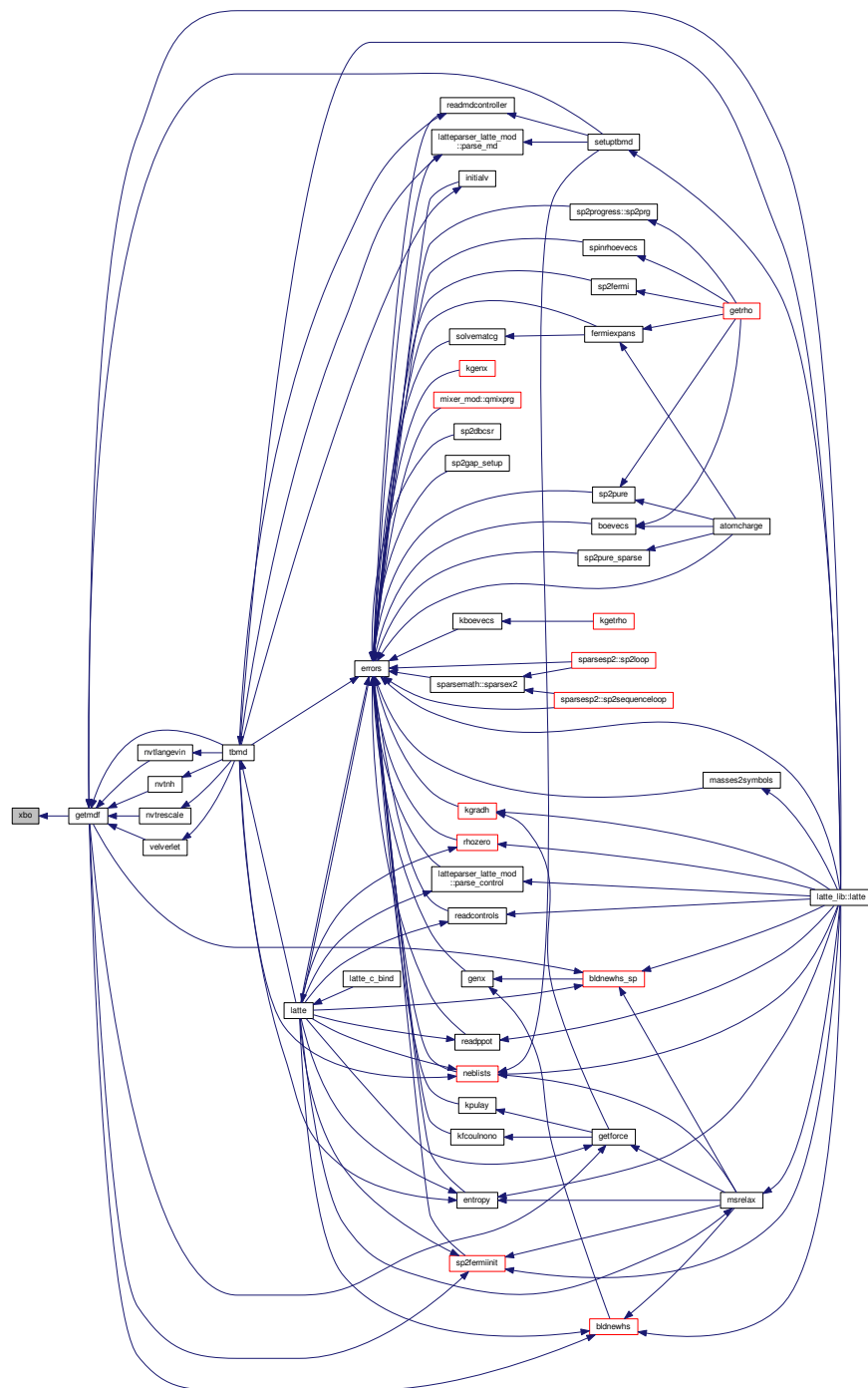
8.453 xbo.f90 File Reference

Functions/Subroutines

- subroutine [xbo](#) (ITER)

8.453.1.1 subroutine xbo (integer *ITER*)

Here is the caller graph for this function:



8.454 xbo.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !
00010 ! modified software should be clearly marked, so as not to confuse it !
00011 ! with the version available from LANL. !
00012 ! !
00013 ! Additionally, this program is free software; you can redistribute it !
00014 ! and/or modify it under the terms of the GNU General Public License as !
00015 ! published by the Free Software Foundation; version 2.0 of the License. !
00016 ! Accordingly, this program is distributed in the hope that it will be !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details. !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 SUBROUTINE xbo(ITER)
00023
00024 !
00025 ! This here subroutine implements Niklasson's eXtended
00026 ! Born-Oppenheimer, time-reversible, and most excellent MD thing.
00027 !
00028 ! Niklasson, PRL, vol. 100, p 123004, (2008)
00029 !
00030
00031 USE constants_mod
00032 USE xboarray
00033 USE setuparray
00034 USE myprecision
00035
00036 IMPLICIT NONE
00037
00038 INTEGER :: I, J, ITER
00039 IF (existerror) RETURN
00040
00041 !
00042 ! If we have no damping:
00043 !
00044 ! Our time reversible guess for the next set of diagonal H-matrix
00045 ! elements depend on those from the last guess, the guess before that, and
00046 ! the H-matrix elements we calculated subject to constraints on Delta_q
00047 ! from the last iteration.
00048 !
00049 ! P(t) - last guess
00050 ! P(t - dt) - guess before that
00051 ! D(t) - the elements calculated from the last run
00052 !
00053 !  $P(t + dt) = 2P(t) - P(t - dt) + 2[D(t) - P(t)]$ 
00054 !
00055 ! With damping, it's the same general idea but we use guesses from even
00056 ! earlier iterations too. Preliminary testing shows better energy
00057 ! conservation with high 'K'.
00058 !
00059
00060 ! Required for the Fast-QMMD scheme
00061
00062 IF (qiter.EQ. 0) THEN
00063     kappa_scale = mdmix
00064 ELSE
00065     kappa_scale = one
00066 ENDIF
00067
00068
00069 IF (iter.EQ. 1) THEN
00070
00071     IF (electro.EQ. 1) THEN
00072
00073         IF (xbodison.EQ. 0) THEN
00074
00075             DO i = 1, nats
00076                 pnk(1,i) = deltaq(i)
00077                 pnk(2,i) = pnk(1,i)
00078             ENDDO
00079
00080             ELSEIF (xbodison.EQ. 1) THEN
00081
00082                 DO i = 1, nats
00083                     pnk(1,i) = deltaq(i)
00084

```

```

00085         DO j = 2, xbodisorder + 1
00086             pnk(j,i) = pnk(j-1,i)
00087         ENDDO
00088     ENDDO
00089
00090
00091     ENDIF
00092
00093     ELSEIF (electro .EQ. 0) THEN
00094
00095         IF (xbodison .EQ. 0) THEN
00096
00097             DO i = 1, nats
00098                 pnk(1,i) = lcnshtft(i)
00099                 pnk(2,i) = pnk(1,i)
00100             ENDDO
00101
00102         ELSEIF (xbodison .EQ. 1) THEN
00103
00104             DO i = 1, nats
00105                 pnk(1,i) = lcnshtft(i)
00106
00107                 DO j = 2, xbodisorder + 1
00108                     pnk(j,i) = pnk(j-1,i)
00109                 ENDDO
00110
00111             ENDDO
00112
00113         ENDIF
00114
00115     ENDIF
00116
00117     ELSEIF (iter .GT. 1) THEN
00118
00119         IF (xbodison .EQ. 0) THEN
00120
00121             IF (electro .EQ. 0) THEN
00122
00123                 DO i = 1, nats
00124
00125                     lcnshtft(i) = two*pnk(1,i) - pnk(2,i) + &
00126                         two*(lcnshtft(i) - pnk(1,i))
00127
00128                     pnk(2,i) = pnk(1,i)
00129                     pnk(1,i) = lcnshtft(i)
00130
00131                 ENDDO
00132
00133             ELSEIF (electro .EQ. 1) THEN
00134
00135                 DO i = 1, nats
00136
00137                     deltaq(i) = two*pnk(1,i) - pnk(2,i) + &
00138                         two*(deltaq(i) - pnk(1,i))
00139
00140                     pnk(2,i) = pnk(1,i)
00141                     pnk(1,i) = deltaq(i)
00142
00143                 ENDDO
00144
00145             ENDIF
00146
00147         ELSEIF (xbodison .EQ. 1) THEN
00148
00149             IF (electro .EQ. 0) THEN
00150
00151                 DO i = 1, nats
00152
00153                     lcnshtft(i) = two*pnk(1,i) - pnk(2,i) + &
00154                         kappa_scale*kappa_xbo*(lcnshtft(i) -
00155 pnk(1,i))
00156
00157                     DO j = 1, xbodisorder+1
00158                         lcnshtft(i) = lcnshtft(i) + alpha_xbo*
00159 cnk(j)*pnk(j,i)
00160                     ENDDO
00161
00162                     DO j = 1, xbodisorder
00163                         pnk(xbodisorder+2 - j,i) = pnk(xbodisorder+1 - j,i)
00164                     ENDDO
00165
00166                     pnk(1,i) = lcnshtft(i)
00167
00168
00169

```

```

00170          ENDDO
00171
00172          ELSEIF (electro .EQ. 1) THEN
00173
00174              DO i = 1, nats
00175
00176                  deltaq(i) = two*pnk(1,i) - pnk(2,i) + &
00177                      kappa_scale*kappa_xbo*(deltaq(i) -
00178                          pnk(1,i))
00179
00180                  DO j = 1, xbodisorder+1
00181
00182                      deltaq(i) = deltaq(i) + alpha_xbo*cnk(j)*
00183                          pnk(j,i)
00184
00185                  ENDDO
00186
00187                  DO j = 1, xbodisorder
00188
00189                      pnk(xbodisorder+2 - j,i) = pnk(xbodisorder+1 - j,i)
00190
00191                  ENDDO
00192
00193                      pnk(1,i) = deltaq(i)
00194
00195                  ENDDO
00196
00197          ENDF
00198
00199      ENDF
00200
00201      RETURN
00202
00203  END SUBROUTINE xbo

```

8.455 xboarray.f90 File Reference

Modules

- module [xboarray](#)

Variables

- real(latteprec) [xboarray::alpha_xbo](#)
- real(latteprec), dimension(:), allocatable [xboarray::chempot_pnk](#)
- real(latteprec), dimension(10) [xboarray::cnk](#)
- real(latteprec) [xboarray::kappa_scale](#)
- real(latteprec) [xboarray::kappa_xbo](#)
- real(latteprec) [xboarray::lastguess_chempot](#)
- real(latteprec), dimension(:,), allocatable [xboarray::pnk](#)
- real(latteprec) [xboarray::prevlastguess_chempot](#)
- real(latteprec), dimension(:,), allocatable [xboarray::spin_pnk](#)

8.456 xboarray.f90

```

00001 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00002 ! Copyright 2010. Los Alamos National Security, LLC. This material was !
00003 ! produced under U.S. Government contract DE-AC52-06NA25396 for Los Alamos !
00004 ! National Laboratory (LANL), which is operated by Los Alamos National !
00005 ! Security, LLC for the U.S. Department of Energy. The U.S. Government has !
00006 ! rights to use, reproduce, and distribute this software. NEITHER THE !
00007 ! GOVERNMENT NOR LOS ALAMOS NATIONAL SECURITY, LLC MAKES ANY WARRANTY, !
00008 ! EXPRESS OR IMPLIED, OR ASSUMES ANY LIABILITY FOR THE USE OF THIS !
00009 ! SOFTWARE. If software is modified to produce derivative works, such !

```

```
00010 ! modified software should be clearly marked, so as not to confuse it      !
00011 ! with the version available from LANL.                                     !
00012 !                                                                           !
00013 ! Additionally, this program is free software; you can redistribute it      !
00014 ! and/or modify it under the terms of the GNU General Public License as     !
00015 ! published by the Free Software Foundation; version 2.0 of the License.    !
00016 ! Accordingly, this program is distributed in the hope that it will be      !
00017 ! useful, but WITHOUT ANY WARRANTY; without even the implied warranty of    !
00018 ! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General !
00019 ! Public License for more details.                                           !
00020 !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
00021
00022 MODULE xboarray
00023
00024   USE myprecision
00025
00026   IMPLICIT NONE
00027   SAVE
00028
00029   REAL(LATTEPREC), ALLOCATABLE :: pnk(:, :)
00030   REAL(LATTEPREC), ALLOCATABLE :: chempot_pnk(:)
00031   REAL(LATTEPREC), ALLOCATABLE :: spin_pnk(:, :)
00032   REAL(LATTEPREC) :: lastguess_chempot, prevlastguess_chempot
00033
00034   REAL(LATTEPREC) :: alpha_xbo, kappa_xbo, kappa_scale
00035   REAL(LATTEPREC) :: cnk(10)
00036
00037 END MODULE xboarray
```

Bibliography

- [1] S. M. Mniszewski, M. J. Cawkwell, M. E. Wall, J. Mohd-Yusof, N. Bock, T. C. Germann, and A. M. N. Niklasson. Efficient parallel linear scaling construction of the density matrix for born–oppenheimer molecular dynamics. *Journal of Chemical Theory and Computation*, 11(10):4644–4654, 2015. PMID: 26574255. [138](#), [140](#)
- [2] Christian F. A. Negre, Susan M. Mniszewski, Marc J. Cawkwell, Nicolas Bock, Michael E. Wall, and Anders M. N. Niklasson. Recursive factorization of the inverse overlap matrix in linear-scaling quantum molecular dynamics simulations. *Journal of Chemical Theory and Computation*, 12(7):3063–3073, 2016. PMID: 27267207. [45](#)

Index

/home/christian/LATTE/README.md, 143

a

fermicommon, 44

addqdep

addqdep.f90, 144

addqdep.f90, 144

addqdep, 144

algo.f90, 151

allfit

allfit.f90, 151

allfit.f90, 151

allfit, 151

allfiton

constants_mod, 17

allocatécoulomb

allocatécoulomb.f90, 161

allocatécoulomb.f90, 161

allocatécoulomb, 161

allocatediag

allocatediag.f90, 163

allocatediag.f90, 163

allocatediag, 163

allocatenebarrays

allocatenebarrays.f90, 166

allocatenebarrays.f90, 166

allocatenebarrays, 166

allocatenono

allocatenono.f90, 168

allocatenono.f90, 168

allocatenono, 168

allocatepure

allocatepure.f90, 171

allocatepure.f90, 170

allocatepure, 171

allocatesubgraph

allocatesubgraph.f90, 173

allocatesubgraph.f90, 172

allocatesubgraph, 173

allocatexbo

allocatexbo.f90, 173

allocatexbo.f90, 173

allocatexbo, 173

allocest

neblastarray, 77

alpha_xbo

xboarray, 135

am

am.f90, 177

am.f90, 177

am, 177

assessocc

assessocc.f90, 178

assessocc.f90, 177

assessocc, 178

atele

setuparray, 89

atocc

setuparray, 89

atomcharge

atomcharge.f90, 179

atomcharge.f90, 179

atomcharge, 179

aveper

mdarray, 65

avepress

avepress.f90, 184

avepress.f90, 184

avepress, 184

avesforhug

avesforhug.f90, 187

avesforhug.f90, 187

avesforhug, 187

avet

mdarray, 65

avetemp

avetemp.f90, 189

avetemp.f90, 189

avetemp, 189

basis

setuparray, 90

basis type

constants_mod, 17

beta0

purearray, 86

binfitstep

constants_mod, 17

bint2fit

constants_mod, 17

bldnewH.f90, 191

bldnewhs, 191

bldnewHS_sp.f90, 201

bldnewhs_sp, 201

bldnewhs

bldnewH.f90, 191

bldnewhs_sp

bldnewHS_sp.f90, 201

bldspinH.f90, 211

bldspinh, 211

- bldspinh
 - bldspinH.f90, 211
- blksz
 - constants_mod, 17
- bm
 - bm.f90, 217
- bm.f90, 217
 - bm, 217
- bml_dmode
 - latteparser_latte_mod::latte_type, 139
- bml_type
 - latteparser_latte_mod::latte_type, 139
- bndfil
 - constants_mod, 18
- bo
 - setuparray, 90
- bo_padded
 - sparsearray, 96
- bodirect.f90, 217
 - boevects, 218
- bodirectprogress.f90, 222
 - boevectsprg, 222
- boevects
 - bodirect.f90, 218
- boevectsprg
 - bodirectprogress.f90, 222
- bond
 - univarray, 133
- box
 - constants_mod, 18
- box_old
 - constants_mod, 18
- boxdims
 - constants_mod, 18
- bozero
 - setuparray, 90
- breaktol
 - constants_mod, 18
- btype
 - setuparray, 90
- c0
 - mdarray, 65
- calpha
 - coulombarray, 31
- calpha2
 - coulombarray, 31
- caselist.f90, 225
- cbs
 - dbcsr_var_mod, 36
- cgolib
 - constants_mod, 18
- cgtol
 - fermicommon, 44
- cgtol2
 - fermicommon, 44
- charge
 - constants_mod, 18
- chempot
 - constants_mod, 18
- chempot_pnk
 - xboarray, 135
- chksum
 - dbcsr_var_mod, 36
- chksum2
 - dbcsr_var_mod, 36
- chtol
 - constants_mod, 18
- cnk
 - xboarray, 135
- col_blk_sizes
 - dbcsr_var_mod, 37
- col_dist_a
 - dbcsr_var_mod, 37
- compforce
 - constants_mod, 18
- conjgradient
 - conjgradient.f90, 226
- conjgradient.f90, 225
 - conjgradient, 226
- consmot
 - mdarray, 65
- constants_mod, 15
 - allfiton, 17
 - basistype, 17
 - binfitstep, 17
 - bint2fit, 17
 - blksz, 17
 - bndfil, 18
 - box, 18
 - box_old, 18
 - boxdims, 18
 - breaktol, 18
 - cgolib, 18
 - charge, 18
 - chempot, 18
 - chtol, 18
 - compforce, 18
 - control, 19
 - coordsfile, 19
 - cove, 19
 - debugon, 19
 - dosfiton, 19
 - ecoul, 19
 - egap, 19
 - ehomo, 19
 - elec_etol, 19
 - elec_qtol, 19
 - elecmeth, 20
 - electro, 20
 - elumo, 20
 - emeth, 20
 - ente, 20
 - entropykind, 20
 - eps, 20
 - erep, 20
 - espin, 20

espin_zero, 20
existerror, 21
f2v, 21
fiton, 21
freeze, 21
fullqconv, 21
hdim, 21
int2fit, 21
job, 21
kbt, 21
ke2t, 21
kee, 22
kon, 22
latteinexists, 22
lcniter, 22
lcnon, 22
libcalls, 22
libinit, 22
librun, 22
massden, 22
maxeval, 22
maxminusmin, 23
maxscf, 23
mcbeta, 23
mcsigma, 23
mdadapt, 23
mdmix, 23
mdon, 23
mineval, 23
minsp2iter, 23
mixer, 23
mvv2ke, 24
mvv2t, 24
nats, 24
nfitstep, 24
ngpu, 24
nmat, 24
noelem, 24
noint, 24
norecs, 24
numscf, 24
occerrlimit, 25
occsteps, 25
ordernmol, 25
parampath, 25
parrep, 25
pbcon, 25
pi, 25
pp2fit, 25
ppbeta, 25
ppfiton, 25
ppnfitstep, 26
ppngeom, 26
ppnmol, 26
ppoton, 26
ppsigma, 26
qfit, 26
qiter, 26
qmix, 26
relaxme, 26
restart, 26
restartlib, 27
rslevel, 27
scfs, 27
scfs_ii, 27
scfstep, 27
sp2conv, 27
sparseon, 27
spinmix, 27
spinon, 27
spintol, 27
sponly, 28
summass, 28
togpa, 28
tote, 28
totne, 28
tracelimit, 28
trrhoh, 28
tscale, 28
vardt, 28
vdwon, 28
verbose, 29
xbodison, 29
xbodisorder, 29
xboon, 29
constants_mod.f90, 228
constraints_mod, 29
 freeze_atoms, 29
constraints_mod.f90, 231
constraints_mod::constrains_type, 137
 method, 137
 verbose, 137
contiter
 mdarray, 65
control
 constants_mod, 19
coordsfile
 constants_mod, 19
 latteparser_latte_mod::latte_type, 139
coslist
 coulombarray, 31
coul_acc
 latteparser_latte_mod::latte_type, 139
coulacc
 coulombarray, 31
coulb
 coulombarray, 32
coulcut
 coulombarray, 32
coulcut2
 coulombarray, 32
coulomb_ewald.f90, 233
 coulombewald, 233
coulomb_oldskool.f90, 236
 gaspccoulomb, 236
coulomb_rspace.f90, 240

- coulombrspace, 240
- coulombarray, 31
 - calpha, 31
 - calpha2, 31
 - coslist, 31
 - coulacc, 31
 - coulb, 32
 - coulcut, 32
 - coulcut2, 32
 - coulr1, 32
 - coulvol, 32
 - eightpi, 32
 - fourcalpha2, 32
 - k1_list, 32
 - k2_list, 32
 - k3_list, 32
 - kcutoff, 33
 - kcutoff2, 33
 - keconst, 33
 - ksq_list, 33
 - latticevecs, 33
 - lmax, 33
 - mmax, 33
 - nk, 33
 - nmax, 33
 - olddeltaqs, 33
 - pi2, 34
 - recipvecs, 34
 - relperm, 34
 - sinlist, 34
 - sqrtpi, 34
 - tfact, 34
 - twopi, 34
- coulombarray.f90, 244
- coulombewald
 - coulomb_ewald.f90, 233
- coulombrspace
 - coulomb_rspace.f90, 240
- coulombv
 - setuparray, 90
- coulr1
 - coulombarray, 32
- coultailcoef
 - coultailcoef.f90, 246
- coultailcoef.f90, 246
 - coultailcoef, 246
- coulvol
 - coulombarray, 32
- cove
 - constants_mod, 19
- cpilist
 - diagarray, 41
- cr
 - setuparray, 90
- cumdt
 - mdarray, 65
- d1
 - relaxcommon, 87
- dbcsr_var_mod, 34
 - cbs, 36
 - chksum, 36
 - chksum2, 36
 - col_blk_sizes, 37
 - col_dist_a, 37
 - diag, 37
 - dist_a, 37
 - dist_b, 37
 - dist_c, 37
 - error, 37
 - found, 37
 - grid_dist, 37
 - group, 37
 - ierr, 38
 - iter, 38
 - matrix_a, 38
 - matrix_b, 38
 - mp_comm, 38
 - mp_env, 38
 - mp_group, 38
 - my_block, 38
 - mynode, 38
 - myploc, 38
 - myset_dist, 35
 - n, 39
 - nblkcols_total, 39
 - nblkrows_total, 39
 - npdims, 39
 - numnodes, 39
 - pcol, 39
 - pgrid, 39
 - proc_holds_blk, 39
 - prow, 39
 - rbs, 39
 - row_blk_sizes, 40
 - row_dist_a, 40
 - temp, 40
 - tr, 40
- dbcsr_var_mod.f90, 248
- deallocateall
 - deallocateall.f90, 251
- deallocateall.f90, 251
 - deallocateall, 251
- deallocatecoulomb
 - deallocatecoulomb.f90, 255
- deallocatecoulomb.f90, 255
 - deallocatecoulomb, 255
- deallocatediag
 - deallocatediag.f90, 257
- deallocatediag.f90, 257
 - deallocatediag, 257
- deallocatenebarrays
 - deallocatenebarrays.f90, 259
- deallocatenebarrays.f90, 259
 - deallocatenebarrays, 259
- deallocatenono
 - deallocatenono.f90, 261

- deallocatenono.f90, 261
 - deallocatenono, 261
- deallocatepure
 - deallocatepure.f90, 263
- deallocatepure.f90, 263
 - deallocatepure, 263
- deallocatesubgraph
 - deallocatesubgraph.f90, 265
- deallocatesubgraph.f90, 265
 - deallocatesubgraph, 265
- deallocatexbo
 - deallocatexbo.f90, 266
- deallocatexbo.f90, 266
 - deallocatexbo, 266
- debugon
 - constants_mod, 19
- deltadim
 - spinarray, 109
- deltaq
 - setuparray, 90
- deltaspin
 - spinarray, 109
- dense2sparse
 - sparsesp2, 102
- dense2sparse_timer
 - timer_mod, 130
- dense2sparsegraph
 - subgraph, 111
- deorthomyrho
 - deorthomyrho.f90, 268
- deorthomyrho.f90, 268
 - deorthomyrho, 268
- deorthomyrhoprg
 - deorthomyrhoprogress.f90, 271
- deorthomyrhoprogress.f90, 271
 - deorthomyrhoprg, 271
- dfda
 - dfda.f90, 274
- dfda.f90, 274
 - dfda, 274
- dfdb
 - dfdb.f90, 275
- dfdb.f90, 275
 - dfdb, 275
- dfdr
 - dfdr.f90, 276
- dfdr.f90, 276
 - dfdr, 276
- dgamma
 - mdarray, 65
- dgspdr
 - dgspdr.f90, 277
- dgspdr.f90, 277
 - dgspdr, 277
- diag
 - dbcsr_var_mod, 37
- diag_iwork
 - diagarray, 41
- diag_liwork
 - diagarray, 41
- diag_lrwork
 - diagarray, 41
- diag_lwork
 - diagarray, 41
- diag_lzwork
 - diagarray, 41
- diag_rwork
 - diagarray, 41
- diag_work
 - diagarray, 41
- diag_zwork
 - diagarray, 41
- diagarray, 40
 - cplist, 41
 - diag_iwork, 41
 - diag_liwork, 41
 - diag_lrwork, 41
 - diag_lwork, 41
 - diag_lzwork, 41
 - diag_rwork, 41
 - diag_work, 41
 - diag_zwork, 41
 - downevals, 41
 - downvecs, 41
 - evals, 42
 - evecs, 42
 - exptol, 42
 - ifail, 42
 - kevals, 42
 - kevecs, 42
 - khtmp, 42
 - numlimit, 42
 - upevals, 42
 - upevecs, 42
 - zbo, 43
 - zheevd_iwork, 43
 - zheevd_liwork, 43
 - zheevd_lrwork, 43
 - zheevd_lwork, 43
 - zheevd_rwork, 43
 - zheevd_work, 43
- diagarray.f90, 279
- diagmyh
 - diagmyh.f90, 280
- diagmyh.f90, 280
 - diagmyh, 280
- dimlist
 - neblastarray, 77
- dist_a
 - dbcsr_var_mod, 37
- dist_b
 - dbcsr_var_mod, 37
- dist_c
 - dbcsr_var_mod, 37
- dmbuild_timer
 - timer_mod, 130

- dosfit
 - dosfit.f90, 283
- dosfit.f90, 283
 - dosfit, 283
- dosfiton
 - constants_mod, 19
- downevals
 - diagarray, 41
- downvecs
 - diagarray, 41
- dqin
 - mixer_mod, 72
- dqout
 - mixer_mod, 72
- dslmmpda
 - dslmmpda.f90, 291
- dslmmpda.f90, 291
 - dslmmpda, 291
- dslmmpdb
 - dslmmpdb.f90, 292
- dslmmpdb.f90, 292
 - dslmmpdb, 292
- dt
 - mdarray, 65
- dtlmpda
 - dtlmpda.f90, 293
- dtlmpda.f90, 293
 - dtlmpda, 293
- dtlmpdb
 - dtlmpdb.f90, 294
- dtlmpdb.f90, 294
 - dtlmpdb, 294
- dtzero
 - mdarray, 65
- dumpfreq
 - mdarray, 65
- dunivscale
 - dunivscaling_function.f90, 297
- dunivscale_sub
 - dunivscaling.f90, 295
- dunivscaling.f90, 295
 - dunivscale_sub, 295
- dunivscaling_function.f90, 297
 - dunivscale, 297
- dwignerddb
 - dwignerddb.f90, 300
- dwignerddb.f90, 300
 - dwignerddb, 300
- e0
 - mdarray, 65
- ecoul
 - constants_mod, 19
- efermi
 - latteparser_latte_mod::latte_type, 139
- egap
 - constants_mod, 19
- ehist
 - mdarray, 66
- ehomo
 - constants_mod, 19
- eight
 - myprecision, 74
- eightpi
 - coulombarray, 32
- ele
 - setuparray, 90
- ele1
 - setuparray, 90
- ele2
 - setuparray, 90
- elec_etol
 - constants_mod, 19
- elec_qtol
 - constants_mod, 19
- elecmeth
 - constants_mod, 20
- electro
 - constants_mod, 20
- elempointer
 - setuparray, 91
- eleven
 - myprecision, 74
- elumo
 - constants_mod, 20
- emeth
 - constants_mod, 20
- ente
 - constants_mod, 20
- entropy
 - entropy.f90, 301
- entropy.f90, 301
 - entropy, 301
- entropyiter
 - mdarray, 66
- entropykind
 - constants_mod, 20
- eps
 - constants_mod, 20
- erep
 - constants_mod, 20
- error
 - dbcsr_var_mod, 37
- errors
 - errors.f90, 311
- errors.f90, 311
 - errors, 311
- espin
 - constants_mod, 20
- espin_zero
 - constants_mod, 20
- evals
 - diagarray, 42
- evects
 - diagarray, 42
- existerror
 - constants_mod, 21

- exptol
 - diagarray, [42](#)
- extractsubgraph
 - subgraphsp2, [118](#)
- f
 - setuparray, [91](#)
- f2v
 - constants_mod, [21](#)
- factorial
 - factorial.f90, [314](#)
- factorial.f90, [314](#)
 - factorial, [314](#)
- fcoul
 - setuparray, [91](#)
- fcoulnono
 - fcoulnono.f90, [315](#)
- fcoulnono.f90, [315](#)
 - fcoulnono, [315](#)
- fcoulnono_sp
 - fcoulnono_sp.f90, [322](#)
- fcoulnono_sp.f90, [321](#)
 - fcoulnono_sp, [322](#)
- fermiallocate
 - fermiallocate.f90, [335](#)
- fermiallocate.f90, [335](#)
 - fermiallocate, [335](#)
- fermicommon, [43](#)
 - a, [44](#)
 - cgtol, [44](#)
 - cgtol2, [44](#)
 - fermim, [44](#)
 - p0, [44](#)
 - r0, [44](#)
 - tmpmat, [44](#)
 - vala, [44](#)
 - valp0, [44](#)
 - valr0, [44](#)
 - valrho, [44](#)
 - valtmp, [45](#)
 - x2, [45](#)
- fermicommon.f90, [337](#)
- fermideallocate
 - fermideallocate.f90, [338](#)
- fermideallocate.f90, [338](#)
 - fermideallocate, [338](#)
- fermiexpans
 - fermiexpans.f90, [340](#)
- fermiexpans.f90, [340](#)
 - fermiexpans, [340](#)
- fermim
 - fermicommon, [44](#)
- fifteen
 - myprecision, [74](#)
- fillinstop
 - sparsearray, [96](#)
- first_step
 - subgraph, [116](#)
- fiton
 - constants_mod, [21](#)
- fittingoutput
 - fittingoutput.f90, [345](#)
- fittingoutput.f90, [345](#)
 - fittingoutput, [345](#)
- five
 - myprecision, [74](#)
- fortyeight
 - myprecision, [74](#)
- found
 - dbcsr_var_mod, [37](#)
- four
 - myprecision, [74](#)
- fourcalpha2
 - coulombarray, [32](#)
- fourteen
 - myprecision, [74](#)
- fpp
 - setuparray, [91](#)
- fpul
 - setuparray, [91](#)
- franprev
 - mdarray, [66](#)
- freeze
 - constants_mod, [21](#)
- freeze_atoms
 - constraints_mod, [29](#)
- friction
 - mdarray, [66](#)
- fscoul
 - setuparray, [91](#)
- fspinnono
 - fspinnono.f90, [348](#)
- fspinnono.f90, [347](#)
 - fspinnono, [348](#)
- fspinnono_sp
 - fspinnono_sp.f90, [354](#)
- fspinnono_sp.f90, [354](#)
 - fspinnono_sp, [354](#)
- fsspin
 - setuparray, [91](#)
- ftot
 - setuparray, [91](#)
- fullqconv
 - constants_mod, [21](#)
- g
 - subgraph, [116](#)
- gamma
 - mdarray, [66](#)
- gaspcoulomb
 - coulomb_oldskool.f90, [236](#)
- gaussrn
 - gaussrn.f90, [363](#)
- gaussrn.f90, [363](#)
 - gaussrn, [363](#)
- genX.f90, [366](#)
 - genx, [366](#)
- genXprogress.f90, [370](#)

- gendiag
 - gendiag.f90, [364](#)
- gendiag.f90, [364](#)
- gendiag, [364](#)
- genx
 - genX.f90, [366](#)
- genxbml
 - genxprogress, [46](#)
- genxprogress, [45](#)
- genxbml, [46](#)
- igenx, [46](#)
- over_bml, [46](#)
- zk1_bml, [47](#)
- zk2_bml, [47](#)
- zk3_bml, [47](#)
- zk4_bml, [47](#)
- zk5_bml, [47](#)
- zk6_bml, [47](#)
- zmat_bml, [47](#)
- zsp, [47](#)
- geps
 - subgraph, [116](#)
- gershgorin
 - gershgorin.f90, [372](#)
- gershgorin.f90, [372](#)
- gershgorin, [372](#)
- get_end
 - get_end_scope, [48](#)
- get_end_scope, [47](#)
- get_end, [48](#)
- main, [48](#)
- get_end_scope.py, [375](#)
- get_file_unit
 - openfiles_mod, [81](#)
- get_k_lists
 - initcoulombklist.f90, [471](#)
- getbndfil
 - getbndfil.f90, [376](#)
- getbndfil.f90, [376](#)
- getbndfil, [376](#)
- getcoule
 - getcoule.f90, [379](#)
- getcoule.f90, [379](#)
- getcoule, [379](#)
- getdeltaq
 - getdeltaq.f90, [381](#)
- getdeltaq.f90, [381](#)
- getdeltaq, [381](#)
- getdeltaspin
 - getdeltaspin.f90, [385](#)
- getdeltaspin.f90, [385](#)
- getdeltaspin, [385](#)
- getdensity
 - getdensity.f90, [395](#)
- getdensity.f90, [395](#)
- getdensity, [395](#)
- getdipole
 - getdipole.f90, [397](#)
- getdipole.f90, [397](#)
- getdipole, [397](#)
- getforce
 - getforce.f90, [399](#)
- getforce.f90, [399](#)
- getforce, [399](#)
- gethdim
 - gethdim.f90, [403](#)
- gethdim.f90, [403](#)
- gethdim, [403](#)
- gethug
 - mdarray, [66](#)
- getke
 - getke.f90, [407](#)
- getke.f90, [407](#)
- getke, [407](#)
- getmatindlist
 - getmatindlist.f90, [409](#)
- getmatindlist.f90, [409](#)
- getmatindlist, [409](#)
- getmaxf
 - getmaxf.f90, [412](#)
- getmaxf.f90, [412](#)
- getmaxf, [412](#)
- getmdf
 - getmdf.f90, [414](#)
- getmdf.f90, [414](#)
- getmdf, [414](#)
- getpressure
 - getpressure.f90, [419](#)
- getpressure.f90, [419](#)
- getpressure, [419](#)
- getrespf
 - getrespf.f90, [422](#)
- getrespf.f90, [422](#)
- getrespf, [422](#)
- getrho
 - getrho.f90, [424](#)
- getrho.f90, [424](#)
- getrho, [424](#)
- getspinE.f90, [428](#)
- getspine, [428](#)
- getspinE_zero.f90, [432](#)
- getspinezero, [432](#)
- getspine
 - getspinE.f90, [428](#)
- getspinezero
 - getspinE_zero.f90, [432](#)
- getsplinenr.f90, [433](#)
- spline, [433](#)
- splint, [433](#)
- gradH.f90, [435](#)
- gradh, [435](#)
- gradH_sp.f90, [441](#)
- gradhsp, [441](#)
- gradh
 - gradH.f90, [435](#)
- gradhsp

- gradH_sp.f90, 441
- grid_dist
 - dbcsr_var_mod, 37
- group
 - dbcsr_var_mod, 37
- h
 - setuparray, 91
- h0
 - setuparray, 91
- h2vect
 - spinarray, 109
- half
 - myprecision, 74
- hdiag
 - setuparray, 92
- hdim
 - constants_mod, 21
- hdown
 - spinarray, 109
- hed
 - setuparray, 92
- hef
 - setuparray, 92
- hep
 - setuparray, 92
- hes
 - setuparray, 92
- hg
 - mdarray, 66
- hjj
 - nonoarray, 79
- hk
 - kspacearray, 53
- hk0
 - kspacearray, 53
- hkdiag
 - kspacearray, 53
- homolumo, 48
 - homolumogap, 49
 - sp2sequence, 49
- homolumo.f90, 454
- homolumogap
 - homolumo, 49
- hr0
 - setuparray, 92
- hthresh
 - sparsearray, 96
- hubbardu
 - setuparray, 92
- hugrescale
 - hugrescale.f90, 457
- hugrescale.f90, 457
 - hugrescale, 457
- hup
 - spinarray, 109
- ierr
 - dbcsr_var_mod, 38
- ifail
 - diagarray, 42
- ifrestart
 - ifrestart.f90, 460
- ifrestart.f90, 460
 - ifrestart, 460
- igenx
 - genxprogress, 46
- im
 - myprecision, 74
- init_dbcsr
 - init_dbcsr.f90, 463
- init_dbcsr.f90, 463
 - init_dbcsr, 463
- init_timer
 - timer_mod, 125
- initcoulomb
 - initcoulomb.f90, 467
- initcoulomb.f90, 467
 - initcoulomb, 467
- initcoulombklist.f90, 470
 - get_k_lists, 471
- initialv
 - initialv.f90, 474
- initialv.f90, 474
 - initialv, 474
- initrng
 - initrng.f90, 478
- initrng.f90, 478
 - initrng, 478
- initshockcomp
 - initshockcomp.f90, 480
- initshockcomp.f90, 480
 - initshockcomp, 480
- int2fit
 - constants_mod, 21
- iset
 - mdarray, 66
- iter
 - dbcsr_var_mod, 38
- ix
 - sparsemath, 101
 - subgraphsp2, 123
- jjb
 - sparsemath, 101
- jjn
 - subgraphsp2, 123
- jjp
 - subgraphsp2, 123
- job
 - constants_mod, 21
- jobname
 - latteparser_latte_mod::latte_type, 140
- k1_list
 - coulombarray, 32
- k2_list
 - coulombarray, 32

- k3_list
 - coulombarray, 32
- kappa_scale
 - xboarray, 135
- kappa_xbo
 - xboarray, 135
- kbldnewh
 - kbldnewh.f90, 482
 - newkbldnewh.f90, 634
- kbldnewh.f90, 482
 - kbldnewh, 482
- kbo
 - kspacearray, 53
- kbodirect.f90, 493
 - kboevects, 493
- kboevects
 - kbodirect.f90, 493
- kbt
 - constants_mod, 21
- kcutoff
 - coulombarray, 33
- kcutoff2
 - coulombarray, 33
- kdeorthomyrho
 - kdeorthomyrho.f90, 500
- kdeorthomyrho.f90, 500
 - kdeorthomyrho, 500
- kdiagmyh
 - kdiagmyh.f90, 502
- kdiagmyh.f90, 502
 - kdiagmyh, 502
- ke2t
 - constants_mod, 21
- keconst
 - coulombarray, 33
- kee
 - constants_mod, 22
- kernelparser_mod, 50
 - parsing_kernel, 51
- kernelparser_mod.f90, 506
- keten
 - virialarray, 133
- kevals
 - diagarray, 42
- kevecs
 - diagarray, 42
- kf
 - kspacearray, 53
- kfcoulnono
 - kfcoulnono.f90, 511
- kfcoulnono.f90, 511
 - kfcoulnono, 511
- kgenX.f90, 519
 - kgenx, 520
- kgenx
 - kgenX.f90, 520
- kgetdos
 - kgetdos.f90, 523
- kgetdos.f90, 522
 - kgetdos, 523
- kgetrho
 - kgetrho.f90, 525
- kgetrho.f90, 525
 - kgetrho, 525
- kgradH.f90, 528
 - kgradh, 528
- kgradh
 - kgradH.f90, 528
- khtmlp
 - diagarray, 42
- kon
 - constants_mod, 22
- korthoh
 - kspacearray, 53
- korthomyH.f90, 536
 - korthomyh, 537
- korthomyh
 - korthomyH.f90, 537
- kpulay
 - kpulay.f90, 538
- kpulay.f90, 538
 - kpulay, 538
- kshift
 - kspacearray, 53
- kspacearray, 52
 - hk, 53
 - hk0, 53
 - hkdiag, 53
 - kbo, 53
 - kf, 53
 - korthoh, 53
 - kshift, 53
 - kxmat, 53
 - nktot, 53
 - nkx, 54
 - nky, 54
 - nkz, 54
 - sk, 54
 - virbondk, 54
 - zhjj, 54
- kspacearray.f90, 547
- ksq_list
 - coulombarray, 33
- kxmat
 - kspacearray, 53
- lastguess_chempot
 - xboarray, 135
- latte
 - latte.f90, 548
 - latte_lib, 55
- latte.f90, 548
 - latte, 548
- latte_abiversion
 - latte_lib, 55
- latte_c_abiversion
 - latte_c_bind.f90, 555

- latte_c_bind
 - latte_c_bind.f90, 555
- latte_c_bind.f90, 555
 - latte_c_abiversion, 555
 - latte_c_bind, 555
- latte_lib, 54
 - latte, 55
 - latte_abiversion, 55
- latte_lib.f90, 558
- latte_timer
 - timer_mod, 131
- latteinexists
 - constants_mod, 22
- latteparser_latte_mod, 55
 - lt, 61
 - parse_control, 56
 - parse_kmesh, 58
 - parse_md, 59
- latteparser_latte_mod.f90, 569
- latteparser_latte_mod::latte_type, 138
 - bml_dmode, 139
 - bml_type, 139
 - coordsfile, 139
 - coul_acc, 139
 - efermi, 139
 - jobname, 140
 - maxscf, 140
 - mdim, 140
 - mdsteps, 140
 - method, 140
 - mixcoeff, 140
 - mpulay, 140
 - nlisteach, 140
 - parampath, 141
 - pulaycoeff, 141
 - restart, 141
 - scftol, 141
 - threshold, 141
 - timeratio, 141
 - timestep, 141
 - verbose, 141
 - zmat, 142
- latteprec
 - myprecision, 74
- latticevecs
 - coulombarray, 33
- lcniter
 - constants_mod, 22
- lcnon
 - constants_mod, 22
- lcnshift
 - setuparray, 92
- lg
 - subgraphsp2, 123
- libcalls
 - constants_mod, 22
- libinit
 - constants_mod, 22
- librun
 - constants_mod, 22
- lmax
 - coulombarray, 33
- lt
 - latteparser_latte_mod, 61
- main
 - get_end_scope, 48
- mass
 - mdarray, 66
- massden
 - constants_mod, 22
- masses2symbols
 - masses2symbols.f90, 578
- masses2symbols.f90, 578
 - masses2symbols, 578
- matindlist
 - setuparray, 92
- matrix_a
 - dbcsr_var_mod, 38
- matrix_b
 - dbcsr_var_mod, 38
- matrixio, 62
 - writedmatrix, 62
 - writehmatrix, 62
 - writemtx, 63
- matrixio.f90, 581
- maxcut
 - neblastarray, 77
- maxdim
 - purearray, 86
- maxdimcoul
 - neblastarray, 77
- maxdimpp
 - neblastarray, 77
- maxdimtb
 - neblastarray, 77
- maxeval
 - constants_mod, 22
- maxiter
 - mdarray, 66
- maxminusmin
 - constants_mod, 23
- maxscf
 - constants_mod, 23
 - latteparser_latte_mod::latte_type, 140
- mcbeta
 - constants_mod, 23
- mcsigma
 - constants_mod, 23
- mdadapt
 - constants_mod, 23
- mdarray, 64
 - aveper, 65
 - avet, 65
 - c0, 65
 - consmot, 65
 - contiter, 65

- cumdt, 65
- dgamma, 65
- dt, 65
- dtzero, 65
- dumpfreq, 65
- e0, 65
- ehist, 66
- entropyiter, 66
- franprev, 66
- friction, 66
- gamma, 66
- gethug, 66
- hg, 66
- iset, 66
- mass, 66
- maxiter, 66
- npton, 67
- npttype, 67
- nvton, 67
- p0, 67
- phist, 67
- phistx, 67
- phisty, 67
- phistz, 67
- ptarget, 67
- rndist, 67
- rsfreq, 68
- seedinit, 68
- seedth, 68
- setth, 68
- shockdir, 68
- shockon, 68
- shockstart, 68
- shockstop, 68
- temperature, 68
- thermper, 68
- thermrun, 69
- thist, 69
- toinittemp, 69
- ttarget, 69
- tzero, 69
- uparticle, 69
- ushock, 69
- v, 69
- v0, 69
- vhist, 69
- wrtfreq, 70
- mdarray.f90, 583
- mdim
 - latteparser_latte_mod::latte_type, 140
- mdmix
 - constants_mod, 23
- mdon
 - constants_mod, 23
- mdsteps
 - latteparser_latte_mod::latte_type, 140
- method
 - constraints_mod::constrains_type, 137
- latteparser_latte_mod::latte_type, 140
- mineval
 - constants_mod, 23
- minsp2iter
 - constants_mod, 23
- minusone
 - myprecision, 74
- mixcoeff
 - latteparser_latte_mod::latte_type, 140
- mixer
 - constants_mod, 23
- mixer_mod, 70
 - dqin, 72
 - dqout, 72
 - mixinit, 73
 - mx, 73
 - qmixprg, 70
 - scferror, 73
- mixer_mod.f90, 585
- mixinit
 - mixer_mod, 73
- mmax
 - coulombarray, 33
- mofit
 - mofit.f90, 586
- mofit.f90, 586
 - mofit, 586
- mofit_plato.f90, 592
 - mofitplato, 592
- mofitplato
 - mofit_plato.f90, 592
- molid
 - neblastarray, 78
- mp_comm
 - dbcsr_var_mod, 38
- mp_env
 - dbcsr_var_mod, 38
- mp_group
 - dbcsr_var_mod, 38
- mpulay
 - latteparser_latte_mod::latte_type, 140
- msparse
 - sparsearray, 96
- msrelax
 - msrelax.f90, 600
- msrelax.f90, 600
 - msrelax, 600
- mvv2ke
 - constants_mod, 24
- mvv2t
 - constants_mod, 24
- mx
 - mixer_mod, 73
- mxrlx
 - relaxcommon, 87
- my_block
 - dbcsr_var_mod, 38
- mycharge

- setuparray, 92
- mynode
 - dbcsr_var_mod, 38
- myoloc
 - dbcsr_var_mod, 38
- myprecision, 73
 - eight, 74
 - eleven, 74
 - fifteen, 74
 - five, 74
 - fortyeight, 74
 - four, 74
 - fourteen, 74
 - half, 74
 - im, 74
 - latteprec, 74
 - minusone, 74
 - nine, 75
 - one, 75
 - quarter, 75
 - re, 75
 - seven, 75
 - six, 75
 - sixteen, 75
 - sqrt2, 75
 - ten, 75
 - third, 75
 - thousand, 76
 - three, 76
 - threequart, 76
 - twelve, 76
 - twenty, 76
 - twentyfour, 76
 - twentysix, 76
 - two, 76
 - zero, 76
- myprecision.f90, 607
- myset_dist
 - dbcsr_var_mod, 35
- n
 - dbcsr_var_mod, 39
- nats
 - constants_mod, 24
- nblkcols_total
 - dbcsr_var_mod, 39
- nblkrows_total
 - dbcsr_var_mod, 39
- nebcoul
 - neblastarray, 78
- neblast_cell.f90, 608
- neblasts, 608
- neblastarray, 77
 - allocst, 77
 - dimlist, 77
 - maxcut, 77
 - maxdimcoul, 77
 - maxdimpp, 77
 - maxdimtb, 77
- molid, 78
- nebcoul, 78
- nebpp, 78
- nebtb, 78
- nomol, 78
- ppmax2, 78
- rcutcoul2, 78
- rcuttb2, 78
- skin, 78
- totnebcoul, 78
- totnebpp, 79
- totnebtb, 79
- udneigh, 79
- neblastarray.f90, 625
- neblasts
 - neblast_cell.f90, 608
 - neblasts.f90, 627
- neblasts.f90, 627
 - neblasts, 627
- nebpp
 - neblastarray, 78
- nebtb
 - neblastarray, 78
- newkbldnewh.f90, 634
 - kbldnewh, 634
- nfitstep
 - constants_mod, 24
- ngpu
 - constants_mod, 24
- nine
 - myprecision, 75
- nk
 - coulombarray, 33
- nktot
 - kspacarray, 53
- nkx
 - kspacarray, 54
- nky
 - kspacarray, 54
- nkz
 - kspacarray, 54
- nlisteach
 - latteparser_latte_mod::latte_type, 140
- nmat
 - constants_mod, 24
- nmax
 - coulombarray, 33
- nnz
 - sparsearray, 96
- nnz.f90, 643
 - nnzend, 643
 - nnzstart, 643
- nnz_max
 - sparsearray, 96
- nnz_pad
 - sparsearray, 97
- nnzend
 - nnz.f90, 643

- nnzstart
 - nnz.f90, 643
- node_in_part
 - subgraph, 117
- noelec
 - noelec.f90, 644
- noelec.f90, 644
 - noelec, 644
- noelem
 - constants_mod, 24
- noint
 - constants_mod, 24
- nomol
 - neblistarray, 78
- nono_evals
 - nonoarray, 79
- nono_iwork
 - nonoarray, 80
- nono_liwork
 - nonoarray, 80
- nono_lwork
 - nonoarray, 80
- nono_work
 - nonoarray, 80
- nonoarray, 79
 - hjj, 79
 - nono_evals, 79
 - nono_iwork, 80
 - nono_liwork, 80
 - nono_lwork, 80
 - nono_work, 80
 - nonotmp, 80
 - nonzero, 80
 - orthoh, 80
 - orthohdown, 80
 - orthohup, 80
 - sh2, 80
 - smat, 81
 - spintmp, 81
 - umat, 81
 - x2hrho, 81
 - xmat, 81
- nonoarray.f90, 645
- nonotmp
 - nonoarray, 80
- nonzero
 - nonoarray, 80
- nopps
 - ppotarray, 85
- norecs
 - constants_mod, 24
- normalizesubgraph
 - subgraphsp2, 118
- norms
 - norms.f90, 646
- norms.f90, 646
 - norms, 646
- npdims
 - dbcsr_var_mod, 39
- npton
 - mdarray, 67
- nptrescale
 - nptrescale.f90, 649
- nptrescale.f90, 649
 - nptrescale, 649
- npttype
 - mdarray, 67
- nr_nodes
 - subgraph, 117
- nr_of_nodes_in_part
 - subgraph, 117
- nr_part
 - subgraph, 117
- nr_sp2_iter
 - purearray, 86
- num_timers
 - timer_mod, 131
- numlimit
 - diagarray, 42
- numnodes
 - dbcsr_var_mod, 39
- numscf
 - constants_mod, 24
- numthresh
 - sparsearray, 97
- nvtNH.f90, 658
 - nvtnh, 658
- nvtandersen
 - nvtandersen.f90, 652
- nvtandersen.f90, 652
 - nvtandersen, 652
- nvtlangevin
 - nvtlangevin.f90, 654
- nvtlangevin.f90, 654
 - nvtlangevin, 654
- nvtnh
 - nvtNH.f90, 658
- nvton
 - mdarray, 67
- nvtrescale
 - nvtrescale.f90, 662
- nvtrescale.f90, 662
 - nvtrescale, 662
- occerrlimit
 - constants_mod, 25
- occsteps
 - constants_mod, 25
- oldd
 - relaxcommon, 87
- olddeltaqs
 - coulombarray, 33
- olddeltaspin
 - spinarray, 109
- oldf
 - relaxcommon, 87
- one

- myprecision, 75
- open_file
 - openfiles_mod, 82
- open_file_to_read
 - openfiles_mod, 83
- openfiles_mod, 81
 - get_file_unit, 81
 - open_file, 82
 - open_file_to_read, 83
- openfiles_mod.f90, 666
- ordernmol
 - constants_mod, 25
- orthoh
 - nonoarray, 80
- orthohdown
 - nonoarray, 80
- orthohup
 - nonoarray, 80
- orthomyH.f90, 667
 - orthomyh, 667
- orthomyHprogress.f90, 670
 - orthomyhprg, 670
- orthomyh
 - orthomyH.f90, 667
- orthomyhprg
 - orthomyHprogress.f90, 670
- orthomyrho
 - orthomyrho.f90, 673
- orthomyrho.f90, 673
 - orthomyrho, 673
- orthorho
 - setuparray, 93
- over_bml
 - genxprogress, 46
- overl
 - univarray, 133
- p0
 - fermicommon, 44
 - mdarray, 67
- pair
 - univarray, 133
- pairpot
 - pairpot.f90, 675
- pairpot.f90, 674
 - pairpot, 675
- pairpot_noneb.f90, 678
 - pairpotnoneb, 678
- pairpotnoneb
 - pairpot_noneb.f90, 678
- pairpotspline
 - pairpotspline.f90, 682
- pairpotspline.f90, 682
 - pairpotspline, 682
 - ppheavi, 683
- pairpottab
 - pairpottab.f90, 685
- pairpottab.f90, 685
 - pairpottab, 685
- panic
 - panic.f90, 688
- panic.f90, 688
 - panic, 688
- parafileopen
 - parafileopen.f90, 694
- parafileopen.f90, 693
 - parafileopen, 694
- parampath
 - constants_mod, 25
 - latteparser_latte_mod::latte_type, 141
- parawrite
 - parawrite.f90, 695
- parawrite.f90, 695
 - parawrite, 695
- parrep
 - constants_mod, 25
- parse_control
 - latteparser_latte_mod, 56
- parse_kmesh
 - latteparser_latte_mod, 58
- parse_md
 - latteparser_latte_mod, 59
- parsing_kernel
 - kernelparser_mod, 51
- partitiongraph
 - subgraph, 112
- pbcb
 - pbcb.f90, 697
- pbcb.f90, 697
 - pbcb, 697
- pbcon
 - constants_mod, 25
- pcol
 - dbcsr_var_mod, 39
- pgrid
 - dbcsr_var_mod, 39
- phist
 - mdarray, 67
- phistx
 - mdarray, 67
- phisty
 - mdarray, 67
- phistz
 - mdarray, 67
- pi
 - constants_mod, 25
- pi2
 - coulombarray, 34
- plot_ppot.f90, 699
 - plotppot, 700
- plot_univ.f90, 702
 - plotuniv, 702
- plotppot
 - plot_ppot.f90, 700
- plotuniv
 - plot_univ.f90, 702
- pnk

- xboarray, 135
- potcoef
- ppotarray, 85
- pp
 - purearray, 86
- pp2fit
 - constants_mod, 25
- ppak
 - ppotarray, 85
- ppbeta
 - constants_mod, 25
- ppele1
 - ppotarray, 85
- ppele2
 - ppotarray, 85
- ppfit
 - ppfit.f90, 705
- ppfit.f90, 705
 - ppfit, 705
- ppfiton
 - constants_mod, 25
- ppheavi
 - pairpotspline.f90, 683
- ppmax2
 - neblastarray, 78
- ppnfitstep
 - constants_mod, 26
- ppngeom
 - constants_mod, 26
- ppnk
 - ppotarray, 85
- ppnmol
 - constants_mod, 26
- ppotarray, 84
 - nopps, 85
 - potcoef, 85
 - ppak, 85
 - ppele1, 85
 - ppele2, 85
 - ppnk, 85
 - ppr, 85
 - pprk, 85
 - ppspl, 85
 - pptablenth, 85
 - ppval, 85
- ppotarray.f90, 715
- ppoton
 - constants_mod, 26
- ppr
 - ppotarray, 85
- pprk
 - ppotarray, 85
- ppsiga
 - constants_mod, 26
- ppspl
 - ppotarray, 85
- pptablenth
 - ppotarray, 85
- ppval
 - ppotarray, 85
- pressure
 - virialarray, 133
- prevf
 - relaxcommon, 88
- prevlastguess_chempot
 - xboarray, 135
- printspase
 - printspase.f90, 716
- printspase.f90, 716
 - printspase, 716
- proc_holds_blk
 - dbcsr_var_mod, 39
- progressloop
 - subgraphsp2, 119
- propchempot
 - propchempot_xbo.f90, 717
- propchempot_xbo.f90, 717
 - propchempot, 717
- propspins
 - propspins_xbo.f90, 720
- propspins_xbo.f90, 720
 - propspins, 720
- prow
 - dbcsr_var_mod, 39
- ptarget
 - mdarray, 67
- pulay
 - pulay.f90, 723
- pulay.f90, 723
 - pulay, 723
- pulay_sp
 - pulay_sp.f90, 731
- pulay_sp.f90, 731
 - pulay_sp, 731
- pulay_spprogress
 - pulay_spprogress.f90, 742
- pulay_spprogress.f90, 741
 - pulay_spprogress, 742
- pulaycoeff
 - latteparser_latte_mod::latte_type, 141
- purearray, 86
 - beta0, 86
 - maxdim, 86
 - nr_sp2_iter, 86
 - pp, 86
 - signlist, 86
 - twoxx2, 86
 - vv, 86
 - x2, 87
 - x2down, 87
 - x2up, 87
- purearray.f90, 751
- qconsistency
 - qconsistency.f90, 753
- qconsistency.f90, 752
 - qconsistency, 753

- qfit
 - constants_mod, 26
- qiter
 - constants_mod, 26
- qlist
 - setuparray, 93
- qmix
 - constants_mod, 26
- qmixprg
 - mixer_mod, 70
- qneutral
 - qneutral.f90, 761
- qneutral.f90, 760
 - qneutral, 761
- quarter
 - myprecision, 75
- r0
 - fermicommon, 44
- rbs
 - dbcsr_var_mod, 39
- rcutcoul2
 - neblistarray, 78
- rcuttb2
 - neblistarray, 78
- re
 - myprecision, 75
- readcontrols
 - readcontrols.f90, 766
- readcontrols.f90, 766
 - readcontrols, 766
- readcr
 - readcr.f90, 772
- readcr.f90, 772
 - readcr, 772
- readmdcontroller
 - readmdcontroller.f90, 775
- readmdcontroller.f90, 775
 - readmdcontroller, 775
- readppot
 - readppot.f90, 780
- readppot.f90, 779
 - readppot, 780
- readppotspline
 - readppotspline.f90, 783
- readppotspline.f90, 782
 - readppotspline, 783
- readppottab
 - readppottab.f90, 785
- readppottab.f90, 785
 - readppottab, 785
- readrestart
 - readrestart.f90, 788
- readrestart.f90, 788
 - readrestart, 788
- readrestartlib
 - readrestartlib.f90, 792
- readrestartlib.f90, 792
 - readrestartlib, 792
- readtb
 - readtb.f90, 803
- readtb.f90, 803
 - readtb, 803
- recipvecs
 - coulombarray, 34
- relaxcommon, 87
 - d1, 87
 - mxrlx, 87
 - oldd, 87
 - oldf, 87
 - prevf, 88
 - relconst, 88
 - reltype, 88
 - rlxftol, 88
- relaxcommon.f90, 806
- relaxme
 - constants_mod, 26
- relconst
 - relaxcommon, 88
- relperm
 - coulombarray, 34
- reltype
 - relaxcommon, 88
- resetprodhd
 - resetprodhd.f90, 807
- resetprodhd.f90, 807
 - resetprodhd, 807
- respchi
 - setuparray, 93
- restart
 - constants_mod, 26
 - latteparser_latte_mod::latte_type, 141
- restartarray, 88
 - tmpbodiag, 88
 - tmphdim, 88
 - tmprhdown, 88
 - tmprhoup, 88
- restartarray.f90, 810
- restartlib
 - constants_mod, 27
- rhodown
 - spinarray, 109
- rhodownzero
 - spinarray, 109
- rhoup
 - spinarray, 109
- rhoupzero
 - spinarray, 110
- rhozero
 - rhozero.f90, 811
- rhozero.f90, 811
 - rhozero, 811
- rlxftol
 - relaxcommon, 88
- rndist
 - mdarray, 67
- row_blk_sizes

- dbcsr_var_mod, 40
- row_dist_a
 - dbcsr_var_mod, 40
- rsfreq
 - mdarray, 68
- rslevel
 - constants_mod, 27
- rx
 - sparsearray, 97
- rxtmp
 - sparsearray, 97
- scferror
 - mixer_mod, 73
- scfs
 - constants_mod, 27
- scfs_ii
 - constants_mod, 27
- scfstep
 - constants_mod, 27
- scftol
 - latteparser_latte_mod::latte_type, 141
- seedinit
 - mdarray, 68
- seedth
 - mdarray, 68
- setth
 - mdarray, 68
- setuparray, 89
 - atele, 89
 - atocc, 89
 - basis, 90
 - bo, 90
 - bozero, 90
 - btype, 90
 - coulombv, 90
 - cr, 90
 - deltaq, 90
 - ele, 90
 - ele1, 90
 - ele2, 90
 - elempointer, 91
 - f, 91
 - fcoul, 91
 - fpp, 91
 - fpul, 91
 - fscoul, 91
 - fsspin, 91
 - ftot, 91
 - h, 91
 - h0, 91
 - hdiag, 92
 - hed, 92
 - hef, 92
 - hep, 92
 - hes, 92
 - hr0, 92
 - hubbardu, 92
 - lcnsht, 92
 - matindlist, 92
 - mycharge, 92
 - orthorho, 93
 - qlist, 93
 - respchi, 93
 - spinindlist, 93
 - setuparray.f90, 834
 - setuptbmd
 - setuptbmd.f90, 836
 - setuptbmd.f90, 836
 - setuptbmd, 836
 - seven
 - myprecision, 75
 - sh2
 - nonoarray, 80
 - shiftH.f90, 838
 - shifh, 838
 - shifh
 - shiftH.f90, 838
 - shockcomp
 - shockcomp.f90, 843
 - shockcomp.f90, 843
 - shockcomp, 843
 - shockdir
 - mdarray, 68
 - shockon
 - mdarray, 68
 - shockstart
 - mdarray, 68
 - shockstop
 - mdarray, 68
 - shutdown_dbcsr
 - shutdown_dbcsr.f90, 845
 - shutdown_dbcsr.f90, 845
 - shutdown_dbcsr, 845
 - shutdown_timer
 - timer_mod, 125
 - signlist
 - purearray, 86
 - sinlist
 - coulombarray, 34
 - six
 - myprecision, 75
 - sixteen
 - myprecision, 75
 - sk
 - kspacearray, 54
 - skin
 - neblastarray, 78
 - slmmp
 - slmmp.f90, 847
 - slmmp.f90, 847
 - slmmp, 847
 - smat
 - nonoarray, 81
 - solvematcg
 - solvematcg.f90, 848
 - solvematcg_sparse.f90, 855

- solvematcg.f90, [848](#)
 - solvematcg, [848](#)
- solvematcg_sparse.f90, [855](#)
 - solvematcg, [855](#)
- solvematlapack
 - solvematlapack.f90, [861](#)
- solvematlapack.f90, [861](#)
 - solvematlapack, [861](#)
- sp2T.f90, [917](#)
 - sp2t, [917](#)
- sp2all_timer
 - timer_mod, [131](#)
- sp2conv
 - constants_mod, [27](#)
- sp2d
 - sp2progress, [95](#)
- sp2dbcsr
 - sp2dbcsr.f90, [864](#)
- sp2dbcsr.f90, [864](#)
 - sp2dbcsr, [864](#)
- sp2fermi
 - sp2fermi.f90, [867](#)
- sp2fermi.f90, [867](#)
 - sp2fermi, [867](#)
- sp2fermi_init.f90, [874](#)
 - sp2fermiinit, [874](#)
- sp2fermiinit
 - sp2fermi_init.f90, [874](#)
- sp2gap
 - sp2gap.f90, [883](#)
- sp2gap.f90, [883](#)
 - sp2gap, [883](#)
- sp2gap_setup
 - sp2gap_setup.f90, [885](#)
- sp2gap_setup.f90, [885](#)
 - sp2gap_setup, [885](#)
- sp2init
 - sp2progress, [95](#)
- sp2loop
 - sparsesp2, [103](#)
- sp2prg
 - sp2progress, [94](#)
- sp2progress, [93](#)
 - sp2d, [95](#)
 - sp2init, [95](#)
 - sp2prg, [94](#)
- sp2progress.f90, [889](#)
- sp2pure
 - sp2pure.f90, [890](#)
- sp2pure.f90, [890](#)
 - sp2pure, [890](#)
- sp2pure_sparse
 - sp2pure_sparse.f90, [896](#)
- sp2pure_sparse.f90, [896](#)
 - sp2pure_sparse, [896](#)
- sp2pure_sparse_parallel
 - sp2pure_sparse_parallel.f90, [906](#)
- sp2pure_sparse_parallel.f90, [906](#)
 - sp2pure_sparse_parallel, [906](#)
- sp2pure_sparse_parallel, [906](#)
 - sp2pure_sparse_parallel, [906](#)
- sp2pure_sparse_parallel_simple
 - sp2pure_sparse_parallel_simple.f90, [910](#)
- sp2pure_sparse_parallel_simple.f90, [910](#)
 - sp2pure_sparse_parallel_simple, [910](#)
- sp2pure_subgraph_parallel
 - sp2pure_subgraph_parallel.f90, [914](#)
- sp2pure_subgraph_parallel.f90, [914](#)
 - sp2pure_subgraph_parallel, [914](#)
- sp2sequence
 - homolumo, [49](#)
- sp2sequenceloop
 - sparsesp2, [105](#)
- sp2sparse_timer
 - timer_mod, [131](#)
- sp2t
 - sp2T.f90, [917](#)
- sparse2dense
 - sparsesp2, [107](#)
- sparse2dense_timer
 - timer_mod, [131](#)
- sparseadd
 - sparsemath, [98](#)
- sparsearray, [96](#)
 - bo_padded, [96](#)
 - fillinstop, [96](#)
 - hthresh, [96](#)
 - mparse, [96](#)
 - nnz, [96](#)
 - nnz_max, [96](#)
 - nnz_pad, [97](#)
 - numthresh, [97](#)
 - rx, [97](#)
 - rxtmp, [97](#)
 - thresholdon, [97](#)
 - work, [97](#)
 - xb, [97](#)
- sparsearray.f90, [922](#)
- sparsemath, [97](#)
 - ix, [101](#)
 - jib, [101](#)
 - sparseadd, [98](#)
 - sparsesetx2, [98](#)
 - sparsex2, [99](#)
 - x, [101](#)
 - y, [101](#)
- sparsemath.f90, [923](#)
- sparseon
 - constants_mod, [27](#)
- sparsesetx2
 - sparsemath, [98](#)
- sparsesp2, [102](#)
 - dense2sparse, [102](#)
 - sp2loop, [103](#)
 - sp2sequenceloop, [105](#)
 - sparse2dense, [107](#)
- sparsesp2.f90, [926](#)
- sparsex2

- sparsemath, 99
- spin_pnk
 - xboarray, 135
- spinarray, 108
 - deltadim, 109
 - deltaspin, 109
 - h2vect, 109
 - hdown, 109
 - hup, 109
 - olddeltaspin, 109
 - rhodown, 109
 - rhodownzero, 109
 - rhoup, 109
 - rhoupzero, 110
 - spinlist, 110
 - wdd, 110
 - wff, 110
 - wpp, 110
 - wss, 110
- spinarray.f90, 931
- spinindlist
 - setuparray, 93
- spinlist
 - spinarray, 110
- spinmix
 - constants_mod, 27
- spinon
 - constants_mod, 27
- spinrhodirect.f90, 932
 - spinrhoevcs, 932
- spinrhoevcs
 - spinrhodirect.f90, 932
- spintmp
 - nonoarray, 81
- spintol
 - constants_mod, 27
- spline
 - getsplinennr.f90, 433
- splint
 - getsplinennr.f90, 433
- sponly
 - constants_mod, 28
- sqrt2
 - myprecision, 75
- sqrtpi
 - coulombarray, 34
- start_timer
 - timer_mod, 126
- stdescent
 - stdescent.f90, 938
- stdescent.f90, 938
 - stdescent, 938
- stop_timer
 - timer_mod, 127
- strten
 - virialarray, 133
- subgraph, 110
 - dense2sparsegraph, 111
 - first_step, 116
 - g, 116
 - geps, 116
 - node_in_part, 117
 - nr_nodes, 117
 - nr_of_nodes_in_part, 117
 - nr_part, 117
 - partitiongraph, 112
 - thresholdgraph, 113
 - tracegraph, 114
 - trnormgraph, 115
 - vvx, 117
- subgraph.f90, 940
- subgraphsp2, 117
 - extractsubgraph, 118
 - ix, 123
 - jjn, 123
 - jjp, 123
 - lg, 123
 - normalizesubgraph, 118
 - progressloop, 119
 - subgraphsp2loop, 121
 - subgraphsp2dense, 122
 - xs, 124
- subgraphsp2.f90, 943
- subgraphsp2loop
 - subgraphsp2, 121
- subgraphsp2dense
 - subgraphsp2, 122
- summary
 - summary.f90, 947
- summary.f90, 947
 - summary, 947
- summass
 - constants_mod, 28
- sysvol
 - virialarray, 133
- tabtest
 - tabtest.f90, 956
- tabtest.f90, 956
 - tabtest, 956
- tavg
 - timer_mod, 131
- tbmd
 - tbmd.f90, 958
- tbmd.f90, 957
 - tbmd, 958
- tclock_max
 - timer_mod, 131
- tclock_rate
 - timer_mod, 131
- tcount
 - timer_mod, 131
- temp
 - dbcsr_var_mod, 40
- temperature
 - mdarray, 68
- ten

- myprecision, 75
- tfact
 - coulombarray, 34
- thermper
 - mdarray, 68
- thermrn
 - mdarray, 69
- third
 - myprecision, 75
- thist
 - mdarray, 69
- thousand
 - myprecision, 76
- three
 - myprecision, 76
- threequart
 - myprecision, 76
- threshold
 - latteparser_latte_mod::latte_type, 141
- thresholdgraph
 - subgraph, 113
- thresholdon
 - sparsearray, 97
- time_mls
 - timer_mod, 128
- timedate_tag
 - timer_mod, 128
- timer_mod, 124
 - dense2sparse_timer, 130
 - dmbuild_timer, 130
 - init_timer, 125
 - latte_timer, 131
 - num_timers, 131
 - shutdown_timer, 125
 - sp2all_timer, 131
 - sp2sparse_timer, 131
 - sparse2dense_timer, 131
 - start_timer, 126
 - stop_timer, 127
 - tavg, 131
 - tclock_max, 131
 - tclock_rate, 131
 - tcount, 131
 - time_mls, 128
 - timedate_tag, 128
 - timer_results, 129
 - tname, 131
 - tpercent, 132
 - tstart, 132
 - tstart_clock, 132
 - tstop_clock, 132
 - tsum, 132
 - ttotal, 132
 - tx, 132
- timer_mod.f90, 965
- timer_results
 - timer_mod, 129
- timeratio
 - latteparser_latte_mod::latte_type, 141
- timestep
 - latteparser_latte_mod::latte_type, 141
- tlmmp
 - tlmmp.f90, 968
- tlmmp.f90, 968
 - tlmmp, 968
- tmpbodiag
 - restartarray, 88
- tmphdim
 - restartarray, 88
- tmpmat
 - fermicommon, 44
- tmprhodown
 - restartarray, 88
- tmprhoup
 - restartarray, 88
- tname
 - timer_mod, 131
- togpa
 - constants_mod, 28
- toinittemp
 - mdarray, 69
- tote
 - constants_mod, 28
- toteng
 - toteng.f90, 969
- toteng.f90, 969
 - toteng, 969
- totne
 - constants_mod, 28
- totnebcoul
 - neblastarray, 78
- totnebpp
 - neblastarray, 79
- totnebtb
 - neblastarray, 79
- tpercent
 - timer_mod, 132
- tr
 - dbcsr_var_mod, 40
- tracegraph
 - subgraph, 114
- tracelimit
 - constants_mod, 28
- trnormgraph
 - subgraph, 115
- trrhoh
 - constants_mod, 28
- tscale
 - constants_mod, 28
- tstart
 - timer_mod, 132
- tstart_clock
 - timer_mod, 132
- tstop_clock
 - timer_mod, 132
- tsum

- timer_mod, 132
- ttarget
 - mdarray, 69
- ttotal
 - timer_mod, 132
- twelve
 - myprecision, 76
- twenty
 - myprecision, 76
- twentyfour
 - myprecision, 76
- twentysix
 - myprecision, 76
- two
 - myprecision, 76
- twopi
 - coulombarray, 34
- twoxx2
 - purearray, 86
- tx
 - timer_mod, 132
- tzero
 - mdarray, 69
- udneigh
 - neblastarray, 79
- umat
 - nonoarray, 81
- univarray, 132
 - bond, 133
 - overl, 133
 - pair, 133
- univarray.f90, 972
- univscale
 - univscaling_function.f90, 975
- univscale_sub
 - univscaling.f90, 973
- univscaling.f90, 973
 - univscale_sub, 973
- univscaling_function.f90, 975
 - univscale, 975
- univtailcoef
 - univtailcoef.f90, 978
- univtailcoef.f90, 978
 - univtailcoef, 978
- uparticle
 - mdarray, 69
- upevals
 - diagarray, 42
- upevecs
 - diagarray, 42
- ushock
 - mdarray, 69
- v
 - mdarray, 69
- v0
 - mdarray, 69
- vala
 - fermicommon, 44
- valp0
 - fermicommon, 44
- valr0
 - fermicommon, 44
- valrho
 - fermicommon, 44
- valtmp
 - fermicommon, 45
- vardt
 - constants_mod, 28
- vdwon
 - constants_mod, 28
- vdwtailcoef
 - vdwtailcoef.f90, 981
- vdwtailcoef.f90, 981
 - vdwtailcoef, 981
- velverlet
 - velverlet.f90, 984
- velverlet.f90, 984
 - velverlet, 984
- verbose
 - constants_mod, 29
 - constraints_mod::constrains_type, 137
 - latteparser_latte_mod::latte_type, 141
- vhist
 - mdarray, 69
- virbond
 - virialarray, 134
- virbondk
 - kspacearray, 54
- vircoul
 - virialarray, 134
- virial
 - virialarray, 134
- virialarray, 133
 - keten, 133
 - pressure, 133
 - strten, 133
 - sysvol, 133
 - virbond, 134
 - vircoul, 134
 - virial, 134
 - virpair, 134
 - virpul, 134
 - virscoul, 134
 - virsspin, 134
- virialarray.f90, 987
- virpair
 - virialarray, 134
- virpul
 - virialarray, 134
- virscoul
 - virialarray, 134
- virsspin
 - virialarray, 134
- vv
 - purearray, 86

- vvx
 - subgraph, [117](#)
- wdd
 - spinarray, [110](#)
- wff
 - spinarray, [110](#)
- wignerd
 - wignerd.f90, [989](#)
- wignerd.f90, [988](#)
- wignerd, [989](#)
- work
 - sparsearray, [97](#)
- wpp
 - spinarray, [110](#)
- writedmatrix
 - matrixio, [62](#)
- writehmatrix
 - matrixio, [62](#)
- writemtx
 - matrixio, [63](#)
- wrtcfs
 - wrtcfs.f90, [990](#)
- wrtcfs.f90, [990](#)
- wrtcfs, [990](#)
- wrtfreq
 - mdarray, [70](#)
- wrtrestart
 - wrtrestart.f90, [997](#)
- wrtrestart.f90, [997](#)
- wrtrestart, [997](#)
- wrtstartlib
 - wrtstartlib.f90, [1002](#)
- wrtstartlib.f90, [1002](#)
- wrtstartlib, [1002](#)
- wss
 - spinarray, [110](#)
- x
 - sparsemath, [101](#)
- x2
 - fermicommon, [45](#)
 - purearray, [87](#)
- x2down
 - purearray, [87](#)
- x2hrho
 - nonoarray, [81](#)
- x2up
 - purearray, [87](#)
- xb
 - sparsearray, [97](#)
- xbo
 - xbo.f90, [1018](#)
- xbo.f90, [1017](#)
- xbo, [1018](#)
- xboarray, [134](#)
 - alpha_xbo, [135](#)
 - chempot_pnk, [135](#)
 - cnk, [135](#)
 - kappa_scale, [135](#)
 - kappa_xbo, [135](#)
 - lastguess_chempot, [135](#)
 - pnk, [135](#)
 - prevlastguess_chempot, [135](#)
 - spin_pnk, [135](#)
- xboarray.f90, [1021](#)
- xbodison
 - constants_mod, [29](#)
- xbodisorder
 - constants_mod, [29](#)
- xboon
 - constants_mod, [29](#)
- xmat
 - nonoarray, [81](#)
- xs
 - subgraphsp2, [124](#)
- y
 - sparsemath, [101](#)
- zbo
 - diagarray, [43](#)
- zero
 - myprecision, [76](#)
- zheevd_iwork
 - diagarray, [43](#)
- zheevd_liwork
 - diagarray, [43](#)
- zheevd_lrwork
 - diagarray, [43](#)
- zheevd_lwork
 - diagarray, [43](#)
- zheevd_rwork
 - diagarray, [43](#)
- zheevd_work
 - diagarray, [43](#)
- zhjj
 - kspacearray, [54](#)
- zk1_bml
 - genxprogress, [47](#)
- zk2_bml
 - genxprogress, [47](#)
- zk3_bml
 - genxprogress, [47](#)
- zk4_bml
 - genxprogress, [47](#)
- zk5_bml
 - genxprogress, [47](#)
- zk6_bml
 - genxprogress, [47](#)
- zmat
 - latteparser_latte_mod::latte_type, [142](#)
- zmat_bml
 - genxprogress, [47](#)
- zsp
 - genxprogress, [47](#)