ERKALE Users' Guide

Susi Lehtola

July 2, 2019

Contents

1 Introduction

1.1 What is ERKALE?

ERKALE is a code for Hartree-Fock and density-functional theory calculations for atoms and molecules. It uses a Gaussian basis set for representing the molecular orbitals. ERKALE is written in C++ and uses the Armadillo template library for linear algebra operations. ERKALE is designed to be as easily maintained and user friendly as possible, but also to try to be reasonably fast for calculations to take place in practice.

The speciality of ERKALE is the computation of x-ray properties, such as electron momentum densities and Compton profiles, x-ray absorption (XAS) and x-ray Raman scattering (XRS) spectra for core electron excitations through the transition potential approximation, and valence electron excitation spectra through the Casida method.

ERKALE is also the leading program used for completeness-optimization of basis sets.

1.2 Why another code?

I wanted to do some research on the modeling of inelastic x-ray scattering. This would require low-level access to a quantum chemical code. The code would need to have a gentle learning curve, be reasonably fast for "production" use - and be free, so that I and others could use the code anywhere they liked.

As I did not find suitable existing programs on the market, I decided to write my own program. This would mean spending more time in development, although with the benefit of getting to grips with low level stuff. The decision was made a lot simpler due to the availability of fast, free libraries for computing electron repulsion integrals (LIBINT[1]) and exchange-correlation functionals (LIBXC[2, 3]), which meant that most of the requisite, but rather time consuming work was already done by others.

Being free is also important because scientific results need to be reproducible. The path from equations to results is often very long in computational science; the code used to implement the equations is (at least!) as important as the equations themselves. To guarantee that the code stays available, I have chosen the GNU General Public License (GPL), which is commonly used in other scientific software as well.

1.3 Citation

The recommended citation for the use of ERKALE in scientific publications is

> S. Lehtola, ERKALE - HF/DFT from Hel, 2016.
> URL http://github.com/susilehtola/erkale

There is also a scientific publication describing ERKALE [4], which you should cite as well, in addition to further publications relevant to the pertinent functionality in ERKALE [13, 14, 18, 25, 27, 34], as detailed below.

2 Usage

ERKALE is divided into a set of programs, which each are designed to perform a specific task.

erkale is the main program, which is mainly used for performing ground-state SCF calculations.

erkale_bastool is a utility program for working with basis set and can be used to decontract a basis, dump the basis for the wanted element or plot completeness profiles.

erkale_casida performs time-dependent density-functional theory (TDDFT) calculations in the Casida formalism as a postprocessing tool for ERKALE.

erkale_copt creates completeness-optimized primitive basis sets.

erkale_emd is the electronic momentum density (EMD) program that postprocesses the results produced by ERKALE.

erkale_xrs performs XAS/XRS calculations in the transition potential approximation.

2.1 erkale

The first step in computing ground-state electronic momentum density properties and Compton profiles, or nonresonant inelastic x-ray scattering (NRIXS) spectra using time-dependent density-functional theory (DFT) through the Casida method, or the transition potential (TP) approximation in ERKALE is the solution of the ground-state electron density. This is performed by the *erkale* program (*erkale_omp* for the OpenMP parallelized version).

The SCF solver is run with the syntax

```
$ erkale runfile
```

where *runfile* is the file containing the specifics of the calculation to run.

The only keyword that you **have to** define is *Method*, which specifies the type of calculation you wish to run, everything else is defaulted. The full list of keywords is given below.

AtomGuess What method to use for the separated atoms when using the atomic guess (see *Guess* section). The default is *Auto*, in which case *Method* will be used for the atomic guess as well. In some cases (e.g. meta-GGA functionals) the atomic calculations may not converge, and specifying a different *AtomGuess* may prove helpful.

Basis The basis set to use. This actually is the file name that ERKALE will look for, and can be given with or without the *.gbs* filename extension. ERKALE will search for the file in the following order:

1. In the current directory.
2. In the directory specified by the environmental variable *ERKALE_LIBRARY*.
3. In the system-wide basis set directory, defined at compile-time. By default this is */usr/share/erkale/basis*.

By default, the used basis set is aug-cc-pVTZ.

BasisCutoff When P-orthogonalization is enabled, the cutoff to use to drop small primitives from contracted functions (in the intermediate normalization). The default is 10^{-8}.

BasisOrth The method used to orthonormalize the basis set. Can be *Can* for Canonical orthonormalization, *Sym* for symmetric orthonormalization or *Chol* for Cholesky orthonormalization. The default is *Auto*, which uses symmetric orthonormalization if possible (no near-linear dependencies), and otherwise canonical orthonormalization. You probably don't want to modify this setting.

BasisRotate Perform P-orthogonalization [5] of basis set? Default is *true*. To restore functionality to that before revision 1181, set this to *false*.

C1-DIIS When DIIS is used, use the original C1 formulation instead of the newer, more flexible C2 formulation? False by default.

Charge The charge of the system, in units of elementary charge. By default 0.

Cholesky Use the Cholesky decomposition [6] to treat two-electron integrals?

CholeskyMode Mode to run the Cholesky decomposition in, specified with an integer. The default is 0, indicating no special action. Setting the value to 1 will save the generated Cholesky decomposition on disk. Setting the value to -1 will load the decomposition from disk, skipping the potentially time consuming formation. This option is useful for, *e.g.*, running benchmarks on different functionals using the same basis set and geometry.

CholeskyThr The threshold to use for the Cholesky decomposition. The default is 10^{-7}.

CholeskyShThr The same-shell threshold to use in the Cholesky decomposition [7]. The program calculates all two-electron integrals for a given shell, and reuses the same integrals until the maximum Cholesky residual on the shell r_{shell} is smaller than the global maximum r_{max} by the set threshold ϵ: $r_{\text{shell}} < \epsilon r_{\text{max}}$.

ConvThr The orbital gradient convergence threshold. This is the same as the DIIS error, $\max \|[\boldsymbol{F}, \boldsymbol{P}]\|$.

DecFock When forming the Fock matrix in direct SCF, calculate the contributions in the decontracted basis. This may be useful when you have a small system with a very large contracted basis set. *False* by default.

Decontract Indices of the atoms to decontract the basis set for, for example *1,2,6-8*, or * for all atoms. Empty by default, meaning no decontraction.

DensityFitting Use density fitting to evaluate the Coulomb term (and exact exchange if needed)? Default is *true* for DFT calculations and *false* for HF calculations. When calculation of the exact exchange is needed, you need to define the *FittingBasis* explicitly.

DFTDelta Defines when to switch to final integration grid, as a multiple **ConvThr**. The default value is 100.

DFTGrid Integration grid to use for DFT calculations. Can be *Auto* for adaptive grid (thus controlled by **DFTInitialTol** and **DFTFinalTol**), or manually defined by *nrad lmax*, *nrad* being the amount of radial shells and *lmax* the maximum order integrated exactly. For Lobatto grids there is no limitation for *lmax*; for Lebedev grids see Table 1 for supported values. You can also define the angular quadrature with *-npts*, where *npts* is the amount of points in the Lebedev quadrature of the wanted order. The default is 50 -194, for a (50,194) grid.

N.B. The default is aimed **for LDA and GGA calculations on light atoms**. If you want to run heavier atoms and/or meta-GGAs, you should check that the grid is converged! Standard choices for bigger grids are (75,302) and (99,590). If the basis set is large like UGBS, you may need many more radial points.

For preliminiary calculations one may be interested in smaller grids, for which (30,170) might be okay. This should not be used for any production runs.

l_{max}	n_{points}	l_{max}	n_{points}
3	6	41	590
5	14	47	770
7	26	53	974
9	38	59	1202
11	50	65	1454
13	74	71	1730
15	86	77	2030
17	110	83	2354
19	146	89	2702
21	170	95	3074
23	194	101	3470
25	230	107	3890
27	266	113	4334
29	302	119	4802
31	350	125	5294
35	434	131	5810

Table 1: Supported Lebedev grids. l_{max} is the order of the rule, while n_{points} gives the amount of points on the grid.

DFTFinalTol (Available only when *Method* has been set to a DFT keyword) The final tolerance for the DFT exchange-correlation integration grid, as determined by the convergence of the diagonal values of the Kohn-Sham Fock matrix [8]. By default 10^{-5}.

DFTInitialTol (Available only when *Method* has been set to a DFT keyword) The initial tolerance for the DFT exchange-correlation integration grid, as determined by the convergence of the diagonal values of the Kohn-Sham Fock matrix [8]. By default 10^{-4}.

DFTLobatto (Available only when *Method* has been set to a DFT keyword) Use Lobatto grids instead of Lebedev grids for angular quadrature. False by default.

DIISOrder When DIIS is used, how many Fock matrices to use for the interpolation? By default 20.

DIISThr Error threshold for using DIIS interpolated Fock matrices. By default 0.05.

DimerSymmetry If running on a diatomic molecule placed along the z axis, this makes the calculation enforce the known orbital symmetries for diatomic molecules, *i.e.* that the orbital behaves as $e^{im\phi}$ with respect to the angle ϕ around the bond. This option may be useful for setting up fully numerical calculations with X2DHF or HELFEM, as the calculation yields the occupations for the different m values.

Direct Compute two-electron integrals on-the-fly? You need to enable this in order to compute large systems. False by default.

FittingBasis The auxiliary basis set to use for density fitted calculation of the Coulomb and exact exchange (for HF and hybrid DFT) matrices. *Auto* for automatical formation. **N.B.** automatical formation is only

supported for pure DFT calculations, as exchange fitting necessary for HF and hybrid DFT has more stringent requirements for the auxiliary basis set. The default is *Auto*.

Guess (If an initial wavefunction was not given to the solver via *LoadChk*) Which guess to use for the calculation? The default is *Atomic*; good alternatives are *NO, Huckel,* as well as *SAP* that needs precomputed atomic radial potentials, see below. Valid options are

- **Core** for the core guess (very bad for polyatomic systems, especially in the presence of heavy atoms),
- **Huckel** for a Hückel-type guess [9], where the program employs the orbital coefficients and energies from a set of spin-averaged fractionally occupied atomic calculations to build the guess orbitals,
- **SAD** for the superposition of atomic densities (SAD) guess [10], which also employs spin-averaged fractionally occupied atomic calculations,
- **SAP** for the superposition of atomic potentials (SAP) [11], which needs precomputed radial potentials, the location for which must be specified with the ERKALE_SAP_LIBRARY environment variable,
- **GSAP** for a SAP guess computed in the Gaussian basis set (this usually gives poor results and shouldn't be used), and
- **NO** for the purified SAD guess [11].

InputBohr Specifies that the geometry in the xyz file is in Bohr instead of Ångström.

LoadChk Initialize calculation with orbitals from checkpoint file.

Logfile Redirect standard output to file. Set as *stdout* to retain in standard output. By default *erkale.log*.

MaxIter Maximum number of SCF steps to perform before giving up. By default 100.

Method Can be *HF* for (restricted or unrestricted) Hartree-Fock, *ROHF* for restricted open-shell Hartree-Fock or a keyword specifying the used exchange-correlation functional.

The evaluation of the exchange-correlation functionals is performed by the LIBXC library, so for a list of supported exchange, correlation and exchange-correlation functionals please refer to the attached list or the LIBXC documentation (e.g. the file *xc_funcs.h* included in the LIBXC installation).

The exchange-correlation functionals are given in the format *exchange-correlation*, for instance *lda_x-lda_c_vwn* for LDA exchange and Vosko-Wilk-Nusair correlation. If you want, you can also give this in the format 1-7, where the numbers are the relevant LIBXC functional identifiers.

If you want to run a calculation with just exchange, or just correlation, you can do this by specifying, *e.g.*, *lda_x-none* and *none-lda_c_vwn*, respectively, as the method. If you set the method to *none* or *none-none* (or substitute 0 for *none*), you will get a calculation corresponding to Hartree level of theory.

Multiplicity Spin multiplicity $2S + 1$ of the system. 1 by default (singlet $S = 0$).

NLGrid The grid used to evaluate the non-local correlation contribution in VV10. Same syntax as for *DFTGrid*. The default is a (50,194) grid, which typically gives results converged to the grid limit. Note that you will want to use a much bigger *DFTGrid*.

Occupancies Orbital occupancies, given in the format n_1 n_2 n_3 ... n_N for closed shell and $n_{1\alpha}$ $n_{1\beta}$ $n_{2\alpha}$ $n_{2\beta}$ ... $n_{N\alpha}$ $n_{N\beta}$ for open shell systems. If occupancies are not specified, the Aufbau principle is used to form the occupancies.

PZ Use Perdew–Zunger self-interaction correction (PZ-SIC) [12, 13, 14]? Default is *No*.

PZEthr The threshhold for energy convergence in the line search procedure.

PZimag Enable imaginary rotations? The default is *Auto*, indicating the imaginary degrees are turned on by stability analysis (if it is enabled).

PZiter The number of consecutive line search iterations to take in the occupied–occupied and occupied–virtual blocks. The default is *20*.

PZIThr The threshhold for the initial localization procedure specified by *PZlocmet*. The default is 10^{-3}.

PZloc Initial localization of the orbitals? The default is *Auto*, meaning that canonical orbitals will be localized, but if a previous PZ calculation is loaded in, the orbitals will not be touched.

PZlocmet The method used in the initial localization of the orbitals. The possibilities are the same as for the *Method* keyword of *erkale_loc*.

PZmode The interactions to apply the PZ correction to. The default is *XC,* standing for exchange and correlation. Any combination of *X*, *C*, and *D* is allowed, where *X* stands for exchange, *C* for correlation and *D* for dispersion (i.e. VV10 non-local correlation).

PZNRthr The gradient threshold to use to switch to a Newton–Raphson method in the occupied–occupied optimization. The default is *0*, meaning that the Newton–Raphson procedure will not be used. Since the Hessian is evaluated seminumerically, the procedure is rather costly.

PZoo Enable rotations in the occupied–occupied block? *True* by default. You can turn this off if you set *PZw* to 0 (with *PZScale Constant*), in which case the model is invariant to rotations in the occupied–occupied block.

PZOOThr The convergence threshold for the occupied–occupied gradient. The default is 10^{-4}.

PZov Enable rotations in the occupied–virtual (and virtual–occupied) block? *True* by default. Turning this off is primarily useful if you use the PZ code to run Edmiston–Ruedenberg localization.

PZOVThr The convergence threshold for the occupied–virtual gradient. The default is 10^{-5}.

PZprec Preconditioning of the occupied–virtual block in PZ-SIC calculations [14]. Can be 0 for no preconditioning, 1 for preconditioning with the unified Hamiltonian, or 2 for preconditioning with the orbital-dependent Hamiltonian.

PZScale Scaling of the Perdew–Zunger self-interaction correction. Can be *Constant* (default), *Kinetic* or *Density*.

PZScaleExp The exponent in the kinetic or density scaling equation. The default is *1.0*.

PZseed The random seed to use for the initial localization. The default is *0*.

PZstab Run stability analysis on the found solution [14]? Can be 0 for no stability analysis, 1 for stability analysis in the occupied–occupied block, or 2 for stability analysis in the occupied–occupied and occupied–virtual blocks. The orbital Hessian is formed by finite difference of analytic gradients, and is diagonalized using dense matrix algebra. The eigenvalues of the Hessian will be printed out.
The program also accepts negative values which trigger following the instability. E.g., *-1* means that stability analysis is run in the occupied–occupied block, and if negative eigenvalues are found the solution will be displaced on the direction of the largest instability and the optimization is restarted.

PZstabThr The threshold ϵ for a negative eigenvalue ω, interpreted as $\omega < -\epsilon$. If this inequality holds, the wave function is deemed unstable.

PZw The weight used for the Perdew–Zunger self-interaction correction. The default is *1.0*.

SaveChk Save results to checkpoint file. Default is *erkale.chk*.

Shift Level shift in Hartree to use when solving the SCF equations. The default is 0, *i.e.* no level shift.

System The xyz file containing the nuclear coordinates of the system to calculate. By default *atoms.xyz*. Additionally, the fifth column in the file may be used to specify atomic charges for the initial atomic guess, which may be useful for obtaining different classes of solutions (e.g. ionic vs non-ionic). The file is searched in the current directory, and then in the directory specified by the environment variable ERKALE_SYSDIR.
The molecular geometry can also be specified as a Z matrix; in this case the xyz file should start with the line #ZMATRIX.

StrictIntegrals Do tight screening of two-electron integrals in on-the-fly calculations of the Hartree-Fock Coulomb and/or exchange matrices. *False* by default, meaning that adaptive screening based on the change of the density matrix is used to skip small products.

UseADIIS Use the ADIIS convergence acceleration algorithm? True by default.

UseDIIS Use the DIIS convergence acceleration algorithm? (Once the DIIS error has dropped below **DIISThr**.) True by default.

UseBroyden Use the Broyden convergence accelerator algorithm? False by default.

UseLM Use pure spherical harmonics in the basis set, *i.e.*, 5 d functions instead of the 6 cartesians, 7 f functions instead of the 10 cartesians, *etc*. True by default.

UseTRRH Use the Trust Region Roothaan-Hall method for orbital updates instead of diagonalization of the Fock matrix. False by default.

Verbose Be verbose in the calculation? True by default.

VV10 Add in VV10 non-local correlation [15]? *True* or *false*. The default is *Auto*, triggering VV10 contributions when toggled by the functional definition in LIBXC. If you set "VV10 True", you need to specify the parameters to be used with *VV10Pars*. Whether VV10 is enabled by *True* or *Auto*, you need to check that the *NLGrid* is suitable.

VV10Pars The b and C parameters for the VV10 non-local correlation functional. This is only used if VV10 is set to True, if VV10 is Auto the parameters will be read in from LIBXC.

2.2 erkale_bastool

The basis set tool can be used to prepare basis sets for calculations by decontracting the basis set (you can, however, accomplish the same thing with the *Decontract* setting in the runfile) or dumping a specific element from the library. The tool can also be used to plot the completeness profile of an element in the basis set.

To examine the composition of a basis set, use, *e.g.*,

```
$ erkale_bastool aug-cc-pVTZ composition
```

which will print out the contraction scheme of the basis set.

To plot the completeness profile [16] of a basis set use, *e.g.*,[1]

```
$ erkale_bastool aug-cc-pVTZ completeness \
  O O-prof.dat
```

which computes the completeness profile of the aug-cc-pVTZ oxygen basis set and save it to the file *O-prof.dat*. The first column in the file contains the 10-base logarithm of the scanning exponent, $\lg \alpha$, whereas the following columns contain the s, p, d, ... space completenesses of the basis set.

To decontract a basis set, use

```
$ erkale_bastool input.gbs decontract \
  output.gbs
```

which will save the input basis set in decontracted form in the file *output.gbs*.

To dump the basis set for a specific element use

```
$ erkale_bastool input.gbs dump element \
  output.gbs
```

Or, to dump the basis set for all the elements in a *xyz* file, use

```
$ erkale_bastool input.gbs genbas system.xyz \
  output.gbs
```

To convert a basis set file into Dalton format, use

```
$ erkale_bastool input.gbs savedalton \
  output.dal
```

To perform P-orthogonalization[5] of a basis set, use

```
$ erkale_bastool input.gbs Porth cutoff \
  Cortho output.gbs
```

Here, *cutoff* is the cutoff for primitive coefficients in the intermediate normalization (largest coefficient normalized to unity), and *Cortho* is the C_{ortho} parameter used in the rotation which controls linear dependencies, $C_{\mathrm{ortho}} = 0$ being equivalent to Davidson rotation.

[1] \ denotes line continuation; write the commands on a single line.

2.3 erkale_casida

To compute NRIXS spectra with the Casida tool, you need to do first a ground-state calculation.

The runfile format for a Casida calculation is as follows:

CasidaC The correlation functional to be used for the Casida calculation. Only LDA functionals are currently supported. By default *lda_c*.

CasidaCoupling The coupling mode for the Casida calculation. Can be 0 for the independent particle approximation (IPA), 1 for the random phase approximation (RPA) or 2 for the time-dependent local density approximation (TDLDA). By default 2.

CasidaPol Force a polarized Casida calculation, even if a restricted ground-state calculation is used. False by default.

CasidaQval The values of momentum transfer that the spectra should be computed for. For example *0.1:0.1:1.0,2.5,7.4*

CasidaStates The states to include in the Casida calculation, for instance *1:10,14,16:48*. For unrestricted calculations the alpha and beta states need to be specified separately (alpha first, then beta). You can also use "HOMO" and "LUMO" as specifiers, *e.g.*, *1:homo+20*, which will include all of the occupied orbitals and the 20 lowest unoccupied ones as well.

CasidaX The exchange functional to be used for the Casida calculation. Only LDA functionals are currently supported. By default *lda_x*.

DFTFittingBasis The basis set to use as the fitting basis. Auto for automatical formation.

LoadChk The file containing the ground-state density. By default *erkale.chk*.

As a result from the Casida executable you will get the file *casida.dat* that contains the dipole NRIXS spectrum. The first column gives the transition energy and the second gives the transition speed. If you also specified *CasidaQVal*, then you will also get files named *casida-x.yy.dat* that contain the momentum transfer dependent spectra.

If default settings are enough for you, you don't need to specify a runfile.

2.4 erkale_copt

The completeness optimization tool can be used to create completeness-optimized [17, 18] primitive basis sets. The optimization uses conjugate gradients with finite difference gradients [18].

The measure to optimize is

$$\tau_n = \left(\frac{1}{\lg \alpha_{\mathrm{high}} - \lg \alpha_{\mathrm{low}}} \int_{\alpha_{\mathrm{low}}}^{\alpha_{\mathrm{high}}} [1 - Y(\alpha)]^n \, d \lg \alpha \right)^{1/n}, \tag{1}$$

where $n = 1$ corresponds to maximizing the area of completeness

$$\tau_{\text{area}} = 1 - \frac{1}{\lg \alpha_{\text{high}} - \lg \alpha_{\text{low}}} \int_{\alpha_{\text{low}}}^{\alpha_{\text{high}}} Y(\alpha) d \lg \alpha$$

and $n = 2$ to minimizing the rms deviation from completeness

$$\tau_{\text{rms}} = \sqrt{\frac{1}{\lg \alpha_{\text{high}} - \lg \alpha_{\text{low}}} \int_{\alpha_{\text{low}}}^{\alpha_{\text{high}}} [1 - Y(\alpha)]^2 \, d \lg \alpha}.$$

Here $Y(\alpha)$ is the completeness profile, which measures the capability of the basis set to represent the primitive Gaussian basis function the set is scanned with,

$$Y(\alpha) = \sum_{\mu\nu} \langle \alpha | \mu \rangle S^{-1}_{\mu\nu} \langle \nu | \alpha \rangle .$$

When $|\alpha\rangle$ can be expanded exactly in the basis set, $Y(\alpha) = 1$.

The syntax of the program is

```
$ erkale_copt runfile
```

where *runfile* is the file that contains the wanted parameters:

LinDepThresh the linear dependence cutoff, 10^{-5} by default

am the angular momentum of the shell to optimize (0 for S, 1 for P, *etc.*)

coulomb use the Coulomb metric instead of the overlap metric? False by default. Use True for optimizing auxiliary basis sets for density fitting.

max the upper limit of the range of $\lg \alpha$ to optimize for

min the lower limit of the range of $\lg \alpha$ to optimize for

n the measure to use: $n = 0$ for maximal area or $n = 1$ for minimal rms deviation

nfull the number of functions on each side to fully optimize, the exponents in the middle being represented by an even-tempered expansion. 4 by default, which yields practically fully converged results

nfunc the number of primitives to use

output file to save the resulting basis to. Default is *optimized.gbs*.

To generate basis sets for practical calculations you should use the optimization scheme detailed in reference [17] to generate sufficient sets for all shells of all elements. Note that will need to concatenate the shells by hand for each element.

Note that you should not in general generate basis sets by hand, but use an automated procedure such as the one I've used [18, 19, 20]. The ERKALE source code includes the algorithms I've used, under *src/contrib*.

2.5 erkale_copt_plateau

This is another variant of the completeness optimization program, where instead of defining a number of funtions and limits for the wanted completeness plateau, the width of the plateau is determined automatically by targetting exact satisfaction of the specified value for the metric τ_n. This is achieved in teh program by adjustment of the width of the plateau by changing the upper limit of the integral [18]. The syntax of the program is

```
$ erkale_copt_plateau runfile
```

where *runfile* is the file that contains the wanted parameters:

LinDepThresh the linear dependence cutoff, 10^{-5} by default

am the angular momentum of the shell to optimize (0 for S, 1 for P, *etc.*)

coulomb use the Coulomb metric instead of the overlap metric? False by default. Use True for optimizing auxiliary basis sets for density fitting.

min the lower limit of the range of $\lg \alpha$

n the measure to use: $n = 0$ for maximal area or $n = 1$ for minimal rms deviation

nfull the number of functions on each side to fully optimize, the exponents in the middle being represented by an even-tempered expansion. 4 by default, which yields practically fully converged results

nfunc the number of primitives to use

output file to save the resulting basis to. Default is *optimized.gbs*.

Like *erkale_copt*, this program also generates functions one shell at a time and so you will need to concatenate the basis sets by hand.

2.6 erkale_cube

erkale_cube can calculate the electron density and/or the molecular orbitals on a grid. To compute the electron momentum density on a grid, use *erkale_emd* (section 2.9).

erkale_cube generates a file in the GAUSSIAN cube format, which can be visualized, *e.g.*, with Avogadro [21] or JMol [22]. The settings are

AutoBuffer Buffer in Ångström to add in each side of the cube determined by the extremal locations of the atoms. The default is 2.5.

AutoSpacing Grid spacing in Ångström to use in automatic grid generation. The default is 0.1.

Cube The geometry of the cube, same syntax as for **EMDCube** in section 2.9. Alternatively, can be Auto for automatic generation (default).

Density A boolean stating whether the densities are to be calculated. Output saved to *density.cube*; if spin unrestricted also alpha and beta densities are computed and saved in *density-a.cube* and *density-b.cube*, respectively.

ELF Compute the electron localization function [23]? The default is *false*.

LoadChk File to load orbitals and densities from

OrbIdx Indices of orbitals to plot. Same syntax as for **CasidaStates** in section 2.3, but the HOMO and LUMO specifiers are not supported.

Potential Compute the electrostatic potential? Saved in *potential.cube*. Default is *false*.

SplitOrbs A boolean stating whether the orbitals are to be saved in a single file *orbital.cube* (*false*, default), or to be split into single orbital-specific files *orbital.N.cube*, where *N* is the orbital index. If spin unrestricted, the corresponding files are *orbital-a.cube* and *orbital-b.cube*, or *orbital-a.N.cube* and *orbital-b.N.cube*.

2.7 erkale_fchkpt

For compatibility with other software, ERKALE includes a tool for converting formatted checkpoint files into ERKALE format, and vice versa: *erkale_fchkpt*.

The main purpose of this tool is to enable visualization of calculations performed with ERKALE using Avogadro[21] (doesn't support spin polarized calculations) or IQmol[24], and to analyze calculations performed with GAUSSIAN™ or Q-CHEM™.

The accepted keywords for *erkale_fchkpt* are as follows:

LoadChk Load checkpoint in ERKALE format from file.

LoadFchk Load formatted checkpoint from file.

SaveChk Save checkpoint in ERKALE format in file.

SaveFchk Save formatted checkpoint from file.

FchkTol Tolerance for deviation of norm of density matrix. The default is 10^{-8}. (Only applicable to fchk $\rightarrow$ chk)

Renormalize A boolean stating whether the density matrix should be renormalized to the correct amount of electrons. (Only applicable to fchk $\rightarrow$ chk)

Reorthonormalize A boolean stating whether the orbitals are to be reorthonormalized. (Only applicable to fchk $\rightarrow$ chk)

2.8 erkale_geom

erkale_geom performs ground-state geometry optimizations, using analytical derivatives. The program is ran similar to the main executable *erkale*. Upon execution, the electronic structure is first solved in the starting point structure verbosely, after which the executions proceeds in non-verbose mode. The optimization is performed in cartesian coordinates, which is suboptimal for large molecules due to the suboptimal choice of the degrees of freedom. ROHF geometry optimization is not currently supported.

N.B. Analytic VV10 forces are currently incorrect in ERKALE. You should use numerical gradients instead for VV10 calculations, as well as for PZ-SIC calculations.

Additional settings compared to *erkale* are

Criterion The convergence criterion to use: LOOSE, NORMAL, TIGHT or VERYTIGHT. The default is *NORMAL*.

CGReset If using conjugate gradients, reset the search direction every N steps. The default is *5*.

MaxSteps The maximum amount of geometry optimization steps to take. Default is *256*.

NumGrad Use numerical gradients instead of analytic gradients? Default is *False*. You may want to set the *Stencil* and the *Stepsize* as well.

Optimizer The optimization algorithm to use. Currently supported options are

CGFR	Fletcher–Reeves conjugate gradients
CGPR	Polak–Ribière conjugate gradients
BFGS	Broyden–Fletcher–Goldfarb–Shanno optimizer (default)
SD	Steepest descent

OptMovie The file to store the progress of the optimization in. The file contains xyz snapshots of the system, and can be played as a movie in, *e.g.*, JMol. Default is *optimize.xyz*.

Result The file to store the final geometry in. Default is *optimized.xyz*.

Stencil The finite difference stencil to use for numerical gradients. The default is 2, for a two-point central difference stencil.

Stepsize The step size for the finite difference stencil in bohr. The default is 10^{-6}.

It is also possible to fix atoms during the geometry optimization. This is done by adding a *-Fx* suffix to the element symbol in the input geometry file.

2.9 erkale_emd

The EMD executable can be used to calculate the electron momentum density and the Compton profile.

To calculate the radial electron momentum density (isotropic EMD) and the isotropic Compton profile with the algorithm from reference [25] from the checkpoint file called *erkale.chk*, the runfile format is

```
LoadChk erkale.chk
DoEMD true
```

The default relative tolerance for the radial p integral is 10^{-8}, which can be modified with the *EMDTol* keyword.

As a result of the calculation, you will get the following files:

emd.txt The radial EMD. The first column is p, the second is the EMD $n\left(p\right)$.

moments.txt The moments of the radial EMD, $\left\langle p^k \right\rangle$ for $k = -2, \ldots, 4$. $\left\langle p^0 \right\rangle$ is simply the number of electrons.

compton.txt The isotropic Compton profile $J\left(q\right)$. The first column is q, the second is $J\left(q\right)$ and the third gives the error estimate of the numerical integration.

compton-interp.txt The isotropic Compton profile, interpolated on a fixed grid. Contents otherwise same as in *compton.txt*.

Correspondingly, if you want to compute the EMD on a grid, you need to add the *EMDCube* keyword. The syntax is

```
EMDCube -10:.1:10
```

for a cube from -10 bohr to 10 bohr with spacing 0.1 bohr in p_x, p_y and p_z, or, if you want a non-isotropic grid, the syntax is

```
EMDCube -3:.2:5 -2:.5:3 -1:.01:1
```

where the spacings are given separately for p_x, p_y and p_z. The result will then be saved in the file *emdcube.dat*, where the first three columns contain p_x, p_y and p_z and the fourth one contains the value of the EMD.

You can also calculate momentum density overlap integrals

$$S_{AB}(k) = \int p^{2k} n_A(\mathbf{p}) n_B(\mathbf{p}) d^3p$$

as well as the related similarity measures

$$I_{AB}(k) = \int p^{2k} \pi_A(\mathbf{p}) \pi_B(\mathbf{p}) d^3p,$$

where the momentum shape functions are defined as

$$\pi_A(\mathbf{p}) = n_A(\mathbf{p})/N_A,$$

N_A being the amount of electrons

$$N_A = \int n_A(\mathbf{p}) d^3p.$$

Also the similarity integral

$$D_{AB}(k) = \sqrt{I_{AA}(k) + I_{BB}(k) - 2I_{AB}(k)}$$

is calculated. The reference is given with the *LoadChk* keyword, and the similarity calculation is triggered with the *Similarity* keyword, with an argument specifying the checkpoint used for the comparison. The integrals are performed on a grid with 500 point radial points (Chebyshev rule), and 2030 angular points (Lebedev), analogously to [26].

2.9.1 erkale_adf_emd

As a supplementary feature, ERKALE can also calculate momentum densities from calculations performed with the Amsterdam Density Functional program (ADF) in the Slater type orbital (STO) basis. However, we do not recommend the use of ADF for EMD calculations as better results can be obtained faster using GTO basis sets.

2.10 erkale_loc

This is a tool for orbital localization. The used algorithms are described in [27], please cite it when you use the program. Accepted keywords

LoadChk Checkpoint to load calculation from.

SaveChk Checkpoint to which save calculation with localized orbitals.

Maxiter Maximum amount of iterations to take in the unitary optimization. Default is 50000.

Method Method to use for orbital localization. Can be:

ER	Edmiston–Ruedenberg localization. [28]
FB	Foster–Boys, *i.e.*, second moment [29]
FB2	FB with penalty exponent $p = 2$ [30]
FB3	FB with penalty exponent $p = 3$ [30]
FM	fourth moment [31]
FM2	FM with penalty exponent $p = 2$ [31]
FM3	FM with penalty exponent $p = 3$ [31]
MU	Pipek–Mezey (PM) with Mulliken charges [32]
MUH	PM with Mulliken charges, but with exponent $p = 1.5$ instead of $p = 2$ [33]
MU2	PM with Mulliken charges, but with exponent $p = 4$ instead of $p = 2$ [33]
LO	PM with Löwdin charges [33]

LOH PM with Löwdin charges, but with exponent $p = 1.5$ instead of $p = 2$ [33]

LO2 PM with Löwdin charges, but with exponent $p = 4$ instead of $p = 2$ [33]

BA PM with Bader charges [34, 35]

BAH PM with Bader charges, but with exponent $p = 1.5$ instead of $p = 2$ [34, 35]

BA2 PM with Bader charges, but with exponent $p = 4$ instead of $p = 2$ [34, 35]

BE PM with Becke charges [34]

BEH PM with Becke charges, but with exponent $p = 1.5$ instead of $p = 2$ [34]

BE2 PM with Becke charges, but with exponent $p = 4$ instead of $p = 2$ [34]

HI PM with Hirshfeld charges [34, 36]

HIH PM with Hirshfeld charges, but with exponent $p = 1.5$ instead of $p = 2$ [34, 36]

HI2 PM with Hirshfeld charges, but with exponent $p = 4$ instead of $p = 2$ [34, 36]

IHI PM with iterative Hirshfeld charges [34, 37, 38]

IHIH PM with iterative Hirshfeld charges, but with exponent $p = 1.5$ instead of $p = 2$ [34, 37, 38]

IHI2 PM with iterative Hirshfeld charges, but with exponent $p = 4$ instead of $p = 2$ [34, 37, 38]

IAO PM with intrinsic atomic orbital charges [39] (This yields intrinsic bond orbitals, IBOs, in the nomenclature of [39])

IAOH PM with intrinsic atomic orbital charges [34, 39], but with exponent $p = 1.5$ instead of $p = 2$ (This yields intrinsic bond orbitals, IBOs, in the nomenclature of [39])

IAO2 PM with intrinsic atomic orbital charges [39], but with exponent $p = 4$ instead of $p = 2$ (This yields intrinsic bond orbitals, IBOs, in the nomenclature of [39])

ST PM with iterative Stockholder charges [34, 40, 41, 42]

STH PM with iterative Stockholder charges [34, 40, 41, 42], but with exponent $p = 1.5$ instead of $p = 2$

ST2 PM with iterative Stockholder charges [34, 40, 41, 42], but with exponent $p = 4$ instead of $p = 2$

VO PM with Voronoi charges [34]

VOH PM with Voronoi charges [34], but with exponent $p = 1.5$ instead of $p = 2$

VO2 PM with Voronoi charges [34], but with exponent $p = 4$ instead of $p = 2$

The default is FB.

Virtual In addition to occupied orbitals, localize virtual orbitals as well? Default is *false*.

Logfile Where the progress report is saved to. The default is the standard output.

Accelerator The convergence accelerator to use. Can be SDSA for steepest descent / steepest ascent, CGPR for Polak–Ribière conjugate gradients, or CGFR for Fletcher–Reeves conjugate gradients. The default is CGPR. The use of CGFR is discouraged.

LineSearch Method to use for line searches. Can be poly_df for fitting the derivative of the cost function, poly_fdf for fitting both cost function and its derivative, armijo for an Armijo line search or fourier_df for a Fourier transform method.

StartingPoint Where to start the optimization from. Possibilities: *CAN* for canonical orbitals, *ORTH* for a random orthogonal matrix (default), or *UNIT* for a random complex unitary matrix.

Delocalize Instead of localizing, delocalize orbitals? Default is *false*.

Seed The random seed to use when starting from a random matrix.

FThreshold Determine convergence when relative change in function value is less than this. Default is 10^{-7}.

GThreshold Determine convergence when absolute value of the Riemannian derivative is smaller than this. Default is 10^{-7}.

2.11 erkale_pop

To perform additional population analyses in addition to the Mulliken analysis ran by *erkale* by default, you can use *erkale_pop*. The accepted keywords are

LoadChk Checkpoint file to load densities from. Default is *erkale.chk*.

Bader Calculate Bader charges, along the lines of reference [43]. Default is *false*.

Becke Calculate Becke charges. **N.B. These are dependent on the used weighting scheme, which is not physical!** Default is *false*.

DensThr Compute total density thresholds for orbital visualization[34, 39]. Default is *false*.

Hirshfeld Calculate Hirshfeld charges [36]. Default is *false*.

HirshfeldMethod Method to use in computing reference densities for Hirshfeld or iterative Hirshfeld analysis. Default is *HF*. It is also possible to load reference densities from checkpoint files with *Load*, so the use of post-HF densities is also possible. The checkpoint files should be in the current working directory and be named el_Q.chk, el being the element symbol and Q the charge (only 0 for Hirshfeld, $Q \in [-2, 2]$ for iterative Hirshfeld).

IterativeHirshfeld Calculate iterative Hirshfeld charges[37, 38]. Default is *false*.

IAO Calculate intrinsic atomic orbital charges[39]. Default is *false*.

IAOBasis (Minimal) basis set to use for intrinsic atomic orbital analysis. Default is *MINAO.gbs*.

Mulliken Calculate Mulliken charges. Default is *false*.

Lowdin Calculate Löwdin charges. Default is *false*.

OrbThr Compute orbital density thresholds for orbital visualization[34, 39]. Default is *false*.

OrbThrVal Orbital density thresholds for orbital or total density visualization[34, 39]. Default is 0.85, corresponding to 85% of total charge within the threshold.

OrbThrGrid DFT grid tolerance for numerical overlap matrix, for determining orbital or density thresholds for visualization[34, 39]. Default is 10^{-3}.

Stockholder Calculate iterative Stockholder charges [40, 41, 42]. Default is *false*.

Voronoi Calculate Voronoi charges. Default is *false*.

Tol Integration grid tolerance for the diagonal element of the overlap matrix [44]. Default is 10^{-5}.

2.12 erkale_xrs

The runfile for the XRS executable contains the same directives as that of the SCF executable (since the TP calculation involves an SCF calculation), and in addition some directives that are specific to the XRS/XAS calculation. To perform a XAS/XRS calculation, you first need to do a ground-state calculation, from which the initial core state is then identified by localization.

Note: in contrast to the other executables, *erkale_xrs* will check the SaveChk file whether a calculation has been already performed. This can be used to initialize an XCH calculation with a TP calculation (copy TP checkpoint to the SaveChk file of the XCH run), or to compute more TP spectra (*e.g.*, different augmentation scheme, different Q values or different QMethod).

Note: The current implementation (since revision 1262) of the localization of the core hole is based on an algorithm similar to the one by Iannuzzi and Hütter [45], instead of the one described in [4]. This enables

XRSAugment Asks for the double basis set method to be used: after the TP calculation has converged, augment the basis set with diffuse functions on the given atoms to improve the description of virtual orbitals, *e.g.*, 1,2,5-10,13. You should always augment the excited atom; others aren't normally necessary. None by default.

XRSDoubleBasis (If the double basis set method is used) The basis set to add on the augmented centers. The default is *X-AUTO*, which is *X-FIRST* for first-row atoms and *X-SECOND* for second-row atoms.

XRSGridTol The XC integration grid tolerance for the double basis set calculation. 10^{-4} by default.

XRSInitialState The core state to excite. Default is *1s*. The program localizes a set of orbitals on the excited atom, from which the core state is identified. Only the inner core can be excited; the orbital must not be in the valence or the iteration will not converge.

XRSInitialOrbital Index of initial orbital on the wanted shell. Default is 1. For excitations from higher than s shells, can be chosen from the range $1, \ldots, 2l + 1$.

XRSLmax (If the local expansion method is used.) Defines the order of expansion in the angular integrals. 5 by default.

XRSLquad (If the local expansion method is used.) Determines how many points are used in the angular integrals. 30 by default.

XRSMethod Which type of calculation to run. Can be *TP* for transition potential (for x-ray Raman and x-ray absorption spectra), *FCH* for full core-hole (for computing vertical photoionization energy), or *XCH* (for computing absolute energy correction to TP spectrum). The default is *TP*.

XRSNrad (If the local expansion method is used.) Gives the number of points used in the radial integrals. 200 by default.

XRSSpin Which spin to excite? Alpha by default. (This setting is relevant for open-shell ground state systems.)

XRSQval The values of momentum transfer q the spectra should be calculated for. For example *0.1:0.1:1.0,2.5,7.4*

XRSQMethod The method of computing the q dependent spectra. Can be *Local* for the local expansion method [46], *Fourier* for the Fourier transform based method or *Series* for a moment matrix based method (not recommended). The default is *Fourier*, which is also used for q dependent Casida.

The excited atom is specified for *erkale_xrs* in the xyz file with the *-Xc* suffix. You can use the same ground state calculation for studying excitations at different centers, *e.g.*, the nonequivalent carbons on phenol.

2.13 Checkpoint files

ERKALEuses the HDF5 library http://www.hdfgroup.org/HDF5/ to perform checkpointing. The method has both the advantage of the speed and accuracy of binary I/O, and the cross-platform compatibility and easy visual inspection that formatted text I/O is conventionally used for.

You can inspect ERKALE checkpoint files with the *h5dump* utility of HDF5. For instance, you can see what entries the checkpoint file *erkale.chk* contains with

```
$ h5dump -n erkale.chk
```

Sometimes you might want to get a hold on the orbital energies. These are saved in the *E* entry (restricted runs) or the *Ea* and *Eb* entries (unrestricted runs), *e.g.*,

```
$ h5dump -d E erkale.chk
```

will give you the orbital energies in Hartree.

If you want just the orbital energies in eV, you can use, *e.g.*,

```
$ h5dump -d E -m '% .10e' erkale.chk | \
awk '{if(NR>5 && NF==2) {print $2*27.21138505}}'
```

Here, the -m "% .10e" option defines that the output from h5dump is to be in scientific notation with 10 decimals – the default precision of h5dump may not be enough for all purposes.

3 Basis set format

The basis set files use GAUSSIAN '94 format. As an example, the cc-pVDZ basis for hydrogen is

```
H     0
S   3   1.00
 13.0100000              0.0196850
  1.9620000              0.1379770
  0.4446000              0.4781480
S   1   1.00
  0.1220000              1.0000000
P   1   1.00
  0.7270000              1.0000000
****
```

The first row contains the element the basis is for (H) and the atom number for the specific entry (0 for the default basis). The second line indicates that an S shell is to follow with 3 primitives and norm 1.00. The next lines then contain the exponent and the contraction coefficient of the normalized primitive, respectively. **** on the last line marks that there are no more shells in the basis set.

N.B. The convention used in ERKALE for the angular momenta symbols is: S ($l = 0$), P ($l = 1$), D ($l = 2$), F ($l = 3$), G ($l = 4$), H ($l = 5$), I ($l = 6$), J ($l = 7$), K ($l = 8$), L ($l = 9$), M ($l = 10$), N ($l = 11$). **Basis sets from the EMSL basis set exchange don't have J shells, but denote l=7 with K.** To cope with this, you can also define the angular momentum of the shell with, *e.g.*,

```
l=1   1    1.00
  0.7270000              1.0000000
```

If you want to mix basis sets in a calculation, *e.g.*, in order to use a bigger basis set for the excited atom in a XAS/XRS calculation, you can do this quite easily by preparing your own basis set file, which contains a general basis (index 0) for all the atoms in the system, and a specific basis for the atom(s) you want to treat differently. For instance the basis set

```
H     0
S   3   1.00
 13.0100000              0.0196850
  1.9620000              0.1379770
  0.4446000              0.4781480
****
H     1
S   3   1.00
 13.0100000              0.0196850
  1.9620000              0.1379770
  0.4446000              0.4781480
S   1   1.00
  0.1220000              1.0000000
P   1   1.00
  0.7270000              1.0000000
****
```

would define a minimal basis set (1 s function) to be used for hydrogen by default, except for atom number 1 for which a $2s1p$ basis is used. ERKALE will refuse to calculate if you screw up the index blatantly, *e.g.*, in this case if atom number 1 in the xyz file is not hydrogen but, say, oxygen.

If you want to perform counterpoise calculations, then you simply mark the ghost atoms with the *-Bq* suffix in the geometry file.

4 Parallelization

The routines in ERKALE are parallelized using OpenMP statements in the source code. This allows for efficient use of multicore workstations and compute nodes. The number of threads used by ERKALE is controlled as usual with the OMP_NUM_THREADS environment variable. By default, the OpenMP executables will use all the cores available.

It is recommended you compile ERKALE both as a sequential version and as a parallel version, as the OpenMP version includes unnecessary overhead for calculations running on a single core.

5 Examples

The geometry and the runfiles for the examples given here are shipped with the ERKALE source code.

5.1 SCF calculation

As an example we compute the water dimer with the coordinates given by the following xyz file at Hartree-Fock level of theory and the aug-cc-pVTZ basis set.

```
6
Water dimer
O       -1.464   0.099   0.300
H       -1.956   0.624  -0.340
H       -1.797  -0.799   0.206
O        1.369   0.146  -0.395
H        1.894   0.486   0.335
H        0.451   0.165  -0.083
```

This is a rather small system, so using the core guess with

```
Guess Core
```

in the runfile is usually beneficial. The default, using the atomic guess [10], is better in larger systems. However, for demonstration purposes we use a third initialization method - that of a "minimal" basis - with even a completely different method!

```
System h2o-dimer.xyz
Method lda_x-lda_c_vwn
Basis cc-pVDZ
SaveChk lda.chk
Guess Core
```

and run it with *erkale*; on my desktop computer the calculation took a few seconds. Out of the calculation we get the checkpoint file *lda.chk*.

Next, we use the minimal basis DFT calculation to seed the full HF calculation, where we use the runfile

```
System h2o-dimer.xyz
Method HF
Basis aug-cc-pVTZ
LoadChk lda.chk
SaveChk hf.chk
```

and run it once again. For this example, HF converges with 16 iterations starting from the core guess, in 15 iterations starting from atomic densities, and in 13 iterations starting from the cc-pVDZ LDA calculation.

5.1.1 Tough convergence cases

For cases that fail to converge using the default convergence settings, you may want to try using the code developed for PZ-SIC calculations[13, 14] by using the settings

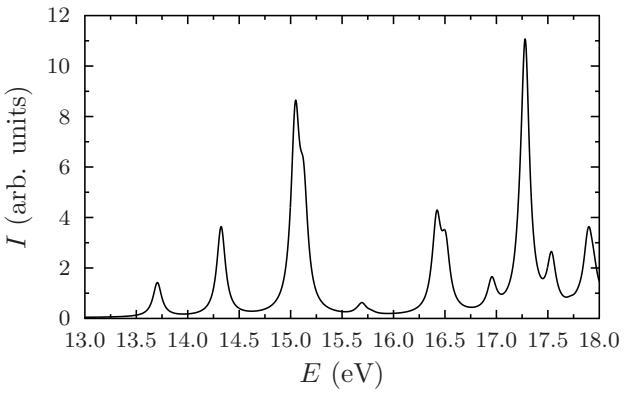

Fig. 1: Photoabsorption spectrum.

```
PZ true
PZscale const
PZw 0.0
PZoo false
PZov true
PZimag false
```

The first line tells ERKALE to use the the PZ-SIC code, whereas the second and third lines make sure that the PZ-SIC correction is turned off. The fourth line turns off occupied–occupied rotations, which are unnecessary for HF and Kohn–Sham DFT, and the fifth line makes sure that occupied–virtual rotations are enabled, the last line making sure imaginary rotations are not enabled.

With these settings, the program will run using the given settings for PZ-SIC. The orbitals saved to disk at every iteration are canonicalized orbitals.

N.B. ROHF and XAS/XRS calculations aren't currently supported with the PZ-SIC code.

5.2 Casida

We compute the valence photoabsorption spectrum of the water dimer with

```
LoadChk hf.chk
CasidaStates 1:70
```

where we have only included the first 70 states in the calculation, giving 600 pairs of occupied and unoccupied orbitals, whereas the full calculation would have 1740 pairs.

The Casida calculation gives the dipole NRIXS intensity, which can be converted into the photoabsorption spectrum by multiplying the dipole intensity with the energy.

Furthermore, the calculation gives just delta peaks, so for plotting purposes we have broadened the peaks with a Lorentzian with a FWHM of 0.05 eV. The result is shown in Fig. 1.

5.3 EMD

We compute the isotropic Compton profile by running *erkale_emd* with the runfile

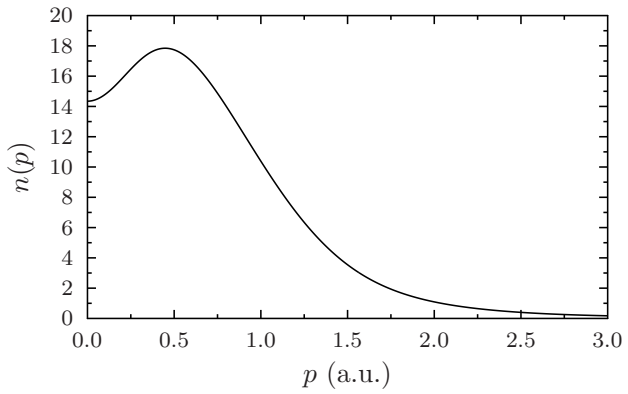

Fig. 2: The electron momentum density.

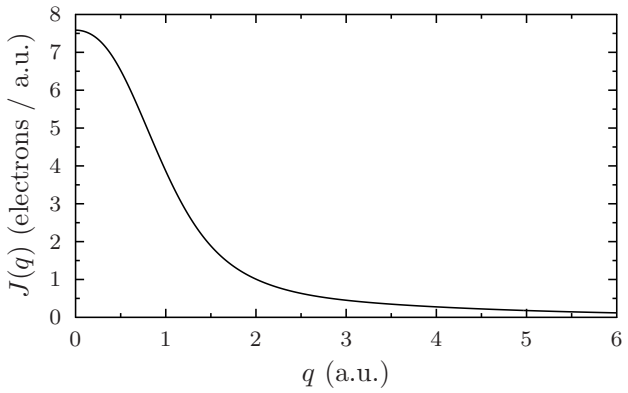

Fig. 3: The Compton profile.

```
LoadChk hf.chk
DoEMD true
```

Out of the calculation we get the electron momentum density and Compton profile shown in Figs. 2 and 3.

5.4 XAS/XRS

Next, we compute the x-ray Raman spectrum. First, we need to define which atom is excited. This we do by creating a new xyz file

```
6
Water dimer
O       -1.464   0.099   0.300
H       -1.956   0.624  -0.340
H       -1.797  -0.799   0.206
O-Xc     1.369   0.146  -0.395
H        1.894   0.486   0.335
H        0.451   0.165  -0.083
```

where we have defined the donor oxygen as the excited atom.

As the input for the XAS/XRS calculation, we use the following input (in reality, you *may* want to use a bigger basis set):

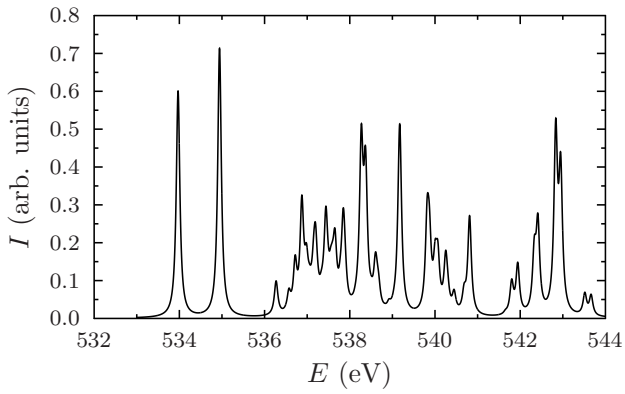

Fig. 4: XAS spectrum of donor oxygen.

```
System h2o-dimer-xrs.xyz
Method lda_x-lda_c_vwn
Basis cc-pVDZ
LoadChk lda.chk
XRSAugment 4
XRSInitialState 1s
```

The *XRSAugment* keyword asks for a double-basis set calculation to be performed. Once the TP calculation has converged, you can play with the double basis set method or the q values used for computing the spectra.

The resulting XAS spectrum[2] is shown in Fig. 4.

To perform an absolute energy scale correction, do another run with the *XRSMethod XCH* setting. You can use the TP result to initialize the calculation by copying the TP checkpoint to the SaveChk file of the XCH run.

Vertical photoionization energies can be obtained with the *XRSMethod FCH* setting.

6 Interfacing with formatted checkpoint files

6.1 Reading in results from other codes

ERKALE can read in results from formatted checkpoint files produced by other codes. Once you have the formatted checkpoint file (per-code instructions below),

1. Convert the formatted checkpoint file into ERKALE format with

   $ erkale_fchkpt runfile

   where *runfile* contains the relevant *LoadFchk* and *SaveChk* directives. Since the formatted checkpoint files may be quite big, you may want to compress them with gzip, bzip2 or xz to save disk space; ERKALE can read compressed formatted

[2] As for Casida, *dipole.dat* contains the XRS spectrum, which you need to multiply with the transition energy to get the XAS spectrum!

checkpoint files as well, the need for decompression is determined by the filename extension.

2. Calculate the properties as you usually would using the newly obtained ERKALE checkpoint file.

6.1.1 GAUSSIAN™

To compute, *e.g.*, EMD properties from calculations performed with GAUSSIAN™ , do the following:

1. At the beginning of the GAUSSIAN™ run input file, specify the keyword

   %Chk=chkpt

   which will give you a binary checkpoint file, chkpt.chk, as a result of the GAUSSIAN™ run.

2. If you use a post-HF level of theory, you need to specify in the route section that the density matrix needs to be computed with *Density=Current*, e.g.,

   #P CCSD/aug-cc-pVTZ Density=Current

3. Run the calculation with GAUSSIAN™.

4. Convert the binary file into formatted checkpoint format (i.e. an ASCII file) with the GAUSSIAN™ *formchk* utility

   $ formchk chkpt.chk chkpt.fchk

6.1.2 Q-CHEM™

Specify "GUI 2" in the Q-CHEM™ runfile. **N.B.** Presently, the usefulness of formatted checkpoint files produced by Q-CHEM™ are limited to SCF (HF/DFT) level results.

6.1.3 PSI4™

An example partial PSI4™ input file to save the results in a formatted checkpoint file is as follows:
energy, wfn = energy('scf', return_wfn=True)
fchk_writer = psi4.core.FCHKWriter(wfn)
fchk_writer.write('output.fchk')

6.2 Saving results from ERKALE for use in other codes

Many visualization programs, such as AVOGADRO or IQMOL, support the use of formatted checkpoint files for visualization of e.g. electron densities and molecular orbitals. To achieve this, use the *erkale_fchkpt* tool as in section 6.1, only now *LoadChk* the Erkale checkpoint and *SaveFchk* a formatted checkpoint. Note that alternatively, electron densities and molecular orbitals can also be visualized in AVOGADRO or JMOL using cube files which were described above in section 2.6.

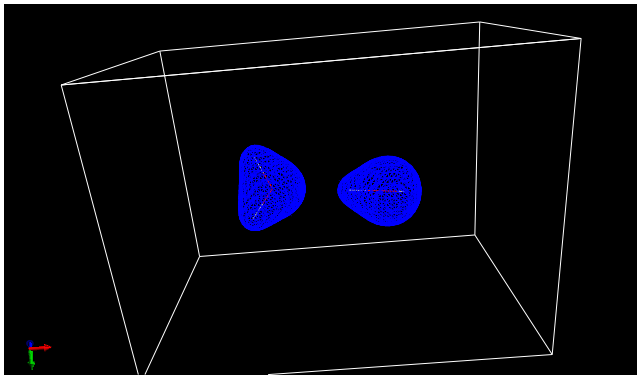

Fig. 5: Electron density plot using Avogadro.

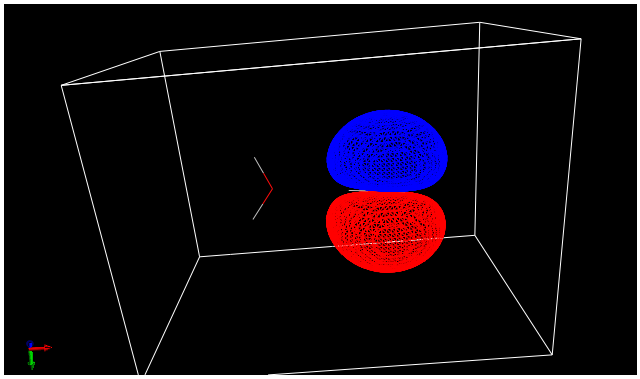

Fig. 6: Orbital plot using Avogadro.

As an example, we visualize the aug-cc-pVTZ -level Hartree–Fock calculation of the water dimer presented above in section 5.1 using AVOGADRO 1.0.3. Saving a formatted checkpoint and loading it in AVOGADRO, setting the resolution to "High" and leaving the isosurface value at 0.1 in AVOGADRO, we obtain the graphics shown in Fig. 5. Correspondingly, plotting the HOMO using "High" resolution and the default isosurface value of 0.02, we obtain the plot shown in Fig. 6.

7 Questions?

If you want to know more about how ERKALE works, or you are interested in contributing, you can contact me via email at *susi.lehtola (at) alumni.helsinki.fi*. As the code is on GitHub, patches can be easily proposed using pull requests.

References

[1] E. F. Valeev, Libint — a high-performance library for computing Gaussian integrals in quantum mechanics. https://github.com/evaleev/libint/ 1.2

[2] M. A. L. Marques and others, Libxc – a library of exchange-correlation functionals for

density-functional theory. http://www.tddft.org/programs/octopus/wiki/index.php/Libxc 1.2

[3] M. A. L. Marques, M. J. T. Oliveira, and T. Burnus, Libxc: a library of exchange and correlation functionals for density functional theory, Comput. Phys. Commun. 183, 2272 (2012) . arXiv:1203.1739 1.2

[4] J. Lehtola, M. Hakala, A. Sakko and K. Hämäläinen, ERKALE — A Flexible Program Package for X-ray Properties of Atoms and Molecules, J. Comput. Chem. 33, 1572 (2012). 1.3, 2.12

[5] F. Jensen, Unifying General and Segmented Contracted Basis Sets. Segmented Polarization Consistent Basis Sets, J. Chem. Theory Comput. 10, 1074 (2014). 2.1, 2.2

[6] N. H. F. Beebe and J. Lindberg, Simplifications in the generation and transformation of two-electron integrals in molecular calculations, Int. J. Quantum Chem. 12, 683 (1977). 2.1

[7] H. Koch, A. Sánchez de Merás, and T. B. Pedersen, Reduced scaling in electronic structure calculations using Cholesky decompositions, J. Chem. Phys. 118, 9481 (2003). 2.1

[8] A. M. Köster, R. Flores-Moreno, and J. U. Reveles, Efficient and reliable numerical integration of exchange-correlation energies and potentials, J. Chem. Phys. 121, 681 (2004). 2.1

[9] P. Norman and H. J. Aa. Jensen, Phosphorescence parameters for platinum (II) organometallic chromophores: A study at the non-collinear four-component Kohn–Sham level of theory, Chem. Phys. Lett. 531 (2012), 229. 2.1

[10] J. H. Van Lenthe *et al.*, Starting SCF calculations by superposition of atomic densities, J. Comput. Chem. 27 (2006), 926. 2.1, 5.1

[11] S. Lehtola, Superposition of Atomic Potentials: a simple yet efficient orbital guess for self-consistent field calculations, arXiv:1810.11659. 2.1

[12] J. P. Perdew and A. Zunger, Self-interaction correction to density-functional approximations for many-electron systems, Phys. Rev. B. 23, 5048 (1981). 2.1

[13] S. Lehtola and H. Jónsson, Variational, Self-Consistent Implementation of the Perdew–Zunger Self-Interaction Correction with Complex Optimal Orbitals, J. Chem. Theory Comput. 10, 5324 (2014). 1.3, 2.1, 5.1.1

[14] S. Lehtola, M. Head-Gordon, and H. Jónsson, Complex orbitals, multiple local minima and symmetry breaking in Perdew--Zunger self-interaction corrected density-functional theory calculations, J. Chem. Theory Comput. 12, 3195 (2016). 1.3, 2.1, 5.1.1

[15] O. A. Vydrov and T. Van Voorhis, Nonlocal van der Waals density functional: The simpler the better, J. Chem. Phys. 133, 244103 (2010). 2.1

[16] D. P. Chong, Completeness profiles of one-electron basis sets, Can. J. Chem. 73, 79 (1995). 2.2

[17] P. Manninen and J. Vaara, Systematic Gaussian basis-set limit using completeness-optimized primitive sets. A case for magnetic properties, J. Comput. Chem. 27, 434 (2006). 2.4, 2.4

[18] S. Lehtola, Automatic algorithms for completeness-optimization of Gaussian basis sets, J. Comput. Chem. 36, 335 (2015). 1.3, 2.4, 2.4, 2.5

[19] J. Lehtola, P. Manninen, M. Hakala and K. Hämäläinen, Completeness-optimized basis sets. Application to ground-state electron momentum densities, J. Chem. Phys. 137, 104105 (2012). 2.4

[20] S. Lehtola, P. Manninen, M. Hakala and K. Hämäläinen, Contraction of completeness-optimized basis sets. Application to ground-state electron momentum densities, J. Chem. Phys. 138, 044109 (2013). 2.4

[21] Avogadro: an open-source molecular builder and visualization tool. http://avogadro.openmolecules.net/ 2.6, 2.7

[22] Jmol: an open-source Java viewer for chemical structures in 3D. http://www.jmol.org/ 2.6

[23] A. D. Becke and K. E. Edgecombe, A simple measure of electron localization in atomic and molecular systems, J. Chem. Phys. 92, 5397 (1990). 2.6

[24] IQmol, a free open-source molecular editor and visualization package. http://iqmol.org 2.7

[25] J. Lehtola, M. Hakala, J. Vaara, and K. Hämäläinen, Calculation of isotropic Compton profiles with Gaussian basis sets, Phys. Chem. Chem. Phys. 13, 5630 (2011) . 1.3, 2.9

[26] J. Vandenbussche, G. Acke, and P. Bultinck, Performance of DFT methods in momentum space: quantum similarity measures versus moments of momentum, J. Chem. Theory Comput. 9, 3908 (2013). 2.9

[27] S. Lehtola and H. Jónsson, Unitary optimization of localized molecular orbitals, J. Chem. Theory Comput. 9, 5365 (2013). 1.3, 2.10

[28] C. Edmiston and K. Ruedenberg, Localized atomic and molecular orbitals, Rev. Mod. Phys. 35, 457 (1963). 2.10

[29] J. M. Foster and S. F. Boys, Canonical Configuration Interaction Procedure, Rev. Mod. Phys. 32, 300 (1960). 2.10

[30] B. Jansík, S. Høst, K. Kristensen, and P. Jørgensen, Local orbitals by minimizing powers of the orbital variance, J. Chem. Phys. 134, 194104 (2011). 2.10

[31] I.-M. Høyvik, B. Jansík, and P. Jørgensen, Orbital localization using fourth central moment minimization, J. Chem. Phys. 137, 224114 (2012). 2.10

[32] J. Pipek and P. G. Mezey, A fast intrinsic localization procedure applicable for ab initio and semiempirical linear combination of atomic orbital wave functions, J. Chem. Phys. 90, 4916 (1989). 2.10

[33] I.M. Høyvik, B. Jansik, and P. Jørgensen, Pipek–Mezey Localization of Occupied and Virtual Orbitals, J. Comput. Chem. 34, 1456 (2013). 2.10

[34] S. Lehtola and H. Jónsson, Pipek–Mezey orbital localization using various partial charge estimates, J. Chem. Theory Comput. 10, 642 (2014). 1.3, 2.10, 2.11

[35] J. Cioslowski, Partitioning of the orbital overlap matrix and the localization criteria, J. Math. Chem. 8, 169 (1991). 2.10

[36] F. L. Hirshfeld, Bonded-atom fragments for describing molecular charge densities, Theor. Chim. Acta 44, 129 (1977). 2.10, 2.11

[37] P. Bultinck, Ch. Van Alsenoy, P. W. Ayers, and R. Carbó-Dorca, Critical analysis and extension of the Hirshfeld atoms in molecules, J. Chem. Phys. 126, 144111 (2007). 2.10, 2.11

[38] P. Bultinck, P. W. Ayers, S. Fias, K. Tiels, and Ch. Van Alsenoy, Uniqueness and basis set dependence of iterative Hirshfeld charges, Chem. Phys. Lett. 444, 205 (2007). 2.10, 2.11

[39] G. Knizia, Intrinsic atomic orbitals: an unbiased bridge between quantum theory and chemical concepts, J. Chem. Theory Comput. 9, 4834 (2013). 2.10, 2.11

[40] T. C. Lillestolen and R. Wheatley, J. Chem. Commun. (Cambridge, U. K.) 7345, 5909 (2008). 2.10, 2.11

[41] T. C. Lillestolen and R. J. Wheatley, J. Chem. Phys. 131, 144101 (2009). 2.10, 2.11

[42] T. C. Lillestolen and R. J. Wheatley, Atomic charge densities generated using an iterative stockholder procedure", J. Chem. Phys. 131, 144101 (2009). 2.10, 2.11

[43] J. I. Rodríguez *et al.*, An efficient grid-based scheme to compute QTAIM atomic properties without explicit calculation of zero-flux surfaces, J. Comput. Chem. 30, 1082 (2009). 2.11

[44] M. Krack and A. M. Köster, An adaptive numerical integrator for molecular integrals, J. Chem. Phys. 108, 3226 (2008). 2.11

[45] M. Iannuzzi and J. Hütter, Inner-shell spectroscopy by the Gaussian and augmented plane wave method, Phys. Chem. Chem. Phys. 9, 1599 (2007). 2.12

[46] A. Sakko, M. Hakala, J. A. Soininen, and K. Hämäläinen, Density functional study of x-ray Raman scattering from aromatic hydrocarbons and polyfluorene, Phys. Rev. B 76, 1 (2007) .

2.12

Changes

2019-07-02 Update documentation on the completeness-optimization tools.

2018-03-18 Better document visualization of orbitals.

2018-01-07 Documented change of convergence thresholds (DeltaPmax, DeltaEmax, DeltaPrms → ConvThr). Documented new SCF guesses.

2016-08-01 Dropped removed keywords Linesearch and UseTRDSM from manual. TightIntegrals should be StrictIntegrals. Document how to use PZ-SIC code for hard to converge cases.

2016-07-14 Specify charge can be specified for atomic guess in the xyz file. Specify geometries can be loaded from ERKALE_SYSDIR. Geometry can also be specified as a Z Matrix. Document VV10 non-local correlation. Documented *AtomGuess*. Documented the Cholesky decomposition. Documented the new Perdew–Zunger self-interaction correction code. XC fitting has been removed.

2015-01-22 Can define angular quadrature by amount of points in rule. Documented Perdew–Zunger stability analysis.

2014-12-18 Documented more changes in the XRS calculation. Documented Perdew–Zunger calculations.

2014-05-16 XRS calculations can now be done with non-1s initial states.

2014-05-03 Added orbital threshold calculations and P-orthogonalization.

2014-02-14 Added iterative Hirshfeld charges and localization.

2013-12-22 Added note about basis set angular momentum.

2013-12-13 Added citation to unitary optimization algorithm, and to Pipek–Mezey methods. Added mention of similarity integrals, and electrostatic potential cubes.

2013-11-06 Added intrinsic atomic orbital charges.

2013-11-05 Added Voronoi charges.

2013-10-30 Added genbas option of *erkale_bastool*.

2013-09-30 Added entry on *erkale_pop*. Updated entries on *erkale* and *erkale_loc*.

2013-07-24 Updated entries on *erkale_cube* and *erkale_fchkpt*.

2013-06-05 Added the DFTGrid keyword for the use of a manually defined integration grid (mainly for more exact comparison to literacy values).

2013-05-18 Added the geometry optimizer *erkale_geom*.

2013-01-19 Can now initialize *erkale_xrs* runs by copying a previously computed checkpoint to the SaveChk file before running. Added section explaining checkpoint files.

2013-01-07 Added exchange-correlation fitting.

2012-09-04 TRDSM and line search minimization are now implemented.

2012-08-28 Changes to *erkale_xrs*. The *FullHole* keyword is now obsolete, having been replaced by the *XRSMethod* keyword.

2012-08-23 Added mention about *erkale_cube*, *erkale_adf_emd* and customizable tolerance for the radial integrals in *erkale_emd*. RI-HF is now supported.

2012-07-30 Added basis set composition and tight integrals, updated orthogonalization method.

2012-06-21 Fixed syntax for CasidaStates.

2012-03-02 The DFTDirect setting is now deprecated, as starting from revision 428 all DFT calculations are done on-the-fly.

2012-02-23 Added UseTRRH setting for trust-region Roothaan-Hall updates.

2012-01-13 Choosing the density-fitting basis is now possible.

2012-01-06 Updated completeness-optimization section.