LATTE Manual

Marc Cawkwell*

Theoretical Division, Los Alamos National Laboratory,

Los Alamos, New Mexico 87545, USA

(Dated: October 21, 2016)

I. AUTHORS

The LATTE code and the parameterizations distributed with the source were developed at Los Alamos National Laboratory by:

Nick Bock (T-1)
Marc Cawkwell (T-1)
Josh Coe (T-1)
Aditi Krishnapriyan (T-1 and Stanford)
Matt Kroonblawd (T-1 and Mizzou)
Adam Lang (T-1)
Enrique Martinez-Saez (MST-8)
Sue Mniszewski (CCS-3)
Christian Negre (T-1)
Anders Niklasson (T-1)
Ed Sanville (T-1)
Mitch Wood (T-1 and Purdue)
Ping Yang (T-1)

II. OVERVIEW

LATTE enables atomistic simulations using a semi-empirical, tight binding description of interatomic interactions. A number of options are available for the type of calculation and the flavor of tight binding model used. The user may specify an orthogonal or non-orthogonal tight basis, self-consistent charge transfer or local charge neutrality, triclinic simulation cells gas-phase, periodic boundary conditions, or k-space integration, static calculations, geometry optimization, or molecular dynamics in various ensembles. Furthermore, molecular dynamics simulations can be performed using the 'regular' Born-Oppenheimer scheme, the extended Lagrangian Born-Oppenheimer formalism, or the fast, SCF-free QMMD formalism.

LATTE is controlled by setting parameters in the two files `TBparam/control.in` and `MDcontroller`. Both of these files currently have a fixed format that cannot be deviated from. The format and options available in these files are described in the next sections.

Atomic coordinates and the dimensions of the periodically repeated simulation cell are read from the file `bl/inputblock.dat`. Again, the format of this file is specified, but it is very similar to that of an `.xyz` file, making it straightforward to transfer between programs.

III. BL/INPUTBLOCK.DAT

The coordinates of atoms and the dimensions of the (optionlly) periodic simulation cell are read from the file `bl/inputblock.dat`. If we are restarting a relaxation or MD simulation then LATTE will read in the coordinates, box dimensions, and other data from the file `bl/restart.dat`. This is controlled through the `RESTART=` flag in `TBparam/control.in`.

For a box of organic molecules, in this case 4 nitromethane molecules in a box measuring $7.085 \times 7.085 \times 7.085$ Å^3 we have,

```
28
7.085 0.0 0.0
0.0 7.085 0.0
0.0 0.0 7.085
O 0.002801 -1.096869 2.047388
O 0.002801 1.096869 2.047388
N -0.002783 0.000000 1.501569
C -0.005764 0.000000 -0.014521
H 1.022793 -0.000000 -0.329858
H -0.509925 -0.890752 -0.336103
H -0.509925 0.890752 -0.336103
O 3.392555 2.405927 3.426924
O 3.385873 4.595527 3.426979
N 3.404757 3.480579 2.876310
C 3.374865 3.506232 1.363798
H 4.385123 3.511524 1.064563
H 2.895865 2.586348 1.040565
H 2.877072 4.399569 1.038416
O 0.948641 2.896140 7.635026
```

```
O 0.991151 5.093637 7.611981
N 1.011220 3.978711 7.034277
C 0.986130 3.993088 5.550872
H 2.023798 3.985066 5.276074
H 0.476657 3.103615 5.227966
H 0.480595 4.840965 5.219954
O 2.452549 -0.947530 5.389024
O 2.466222 1.272741 5.392669
N 2.412477 0.121939 4.846992
C 2.424496 0.131335 3.335140
H 3.491371 0.135739 3.028514
H 1.947525 -0.723242 3.027513
H 1.955704 1.041502 3.024550
```

Line 1: The total number of atoms to be read in.

Line 2 - line 4: The three periodic lattice vectors of the periodic simulation box in Å. The origin of the box is always 0 0 0. The periodic box is not used when performing gas phase simulations (`PBCON= 0`), but in that make sure you also compute the electrostatics in the gas phase (without Ewald summation) in order to avoid issues.

Lines 5 to `NATS`+3: List of the species and atomic coordinates in `.xyz` format. The species label is the usual chemical element symbol and is case sensitive. It must correspond to the entries in the files containing the parameterization.

IV. TBPARAM/CONTROL.IN

`TBparam/control.in` is used to tell LATTE what type of calculation (MD, relaxation etc) to do and how to do it.

A. Structure and format of `TBparam/control.in`

```
CONTROL= 1
BASISTYPE= ORTHO
DEBUGON= 0
FERMIM= 6
CGORLIB= 1 CGTOL= 1.0e-6
KBT= 0.0
NORECS= 1
ENTROPYKIND= 1
PPOTON= 1 VDWON= 1
SPINON= 0 SPINTOL= 1.0e-4
ELECTRO= 1 ELECMETH= 0 ELEC_ETOL= 0.001 ELEC_QTOL= 1.0e-6
COULACC= 1.0e-6 COULCUT= -500.0 COULR1= 500.0
MAXSCF= 100
BREAKTOL= 1.0E-12 MINSP2ITER= 35 SP2CONV= REL
FULLQCONV= 0 QITER= 0
QMIX= 0.25 SPINMIX= 0.25 MDMIX= 0.25
ORDERNMOL= 0
SPARSEON= 0 THRESHOLDON= 1 NUMTHRESH= 1.0e-5 FILLINSTOP= 100 BLKSZ= 4
LCNON= 0 LCNITER= 4 CHTOL= 0.01
SKIN= 1.0
RELAX= 0 RELAXTYPE= SD MAXITER= 100 RLXFTOL= 0.001
MDON= 1
PBCON= 1
RESTART= 0
CHARGE= 0
XBO= 1
XBODISON= 1
XBODISORDER= 5
NGPU= 2
KON= 0
```

```
COMPFORCE= 0
DOSFIT= 0 INTS2FIT= 1 BETA= 1000.0 NFITSTEP= 5000 QFIT= 0 MCSIGMA= 0.2
PPFITON= 0
ALLFITON= 0
PPSTEP= 500 BISTEP= 500 PP2FIT= 2 BINT2FIT= 6
PPBETA= 1000.0 PPSIGMA= 0.01 PPNMOL= 10 PPNGEOM= 200
PARREP= 0
ER= 1.0
```

B. Setting values in `TBparam/control.in`

- `CONTROL=` *integer*

Compute the density matrix using:
1: Diagonalization
2: SP2 purification algorithm of Niklasson. Setting `SPARSEON= 1` switches to an $\mathcal{O}(N)$ implementation of the algorithm in sparse matrix algebra
3: Recursive expansion of the Fermi operator
4, 5: Truncated SP2-type expansions of the Fermi operator to yield an approximate finite electron temperature (*experimental*)

- `BASISTYPE=` *character*

`ORTHO`: Orthogonal basis of atomic orbitals. No overlap matrix, or more precisely, $S = I$.
`NONORTHO`: Non-orthogonal basis of atomic orbitals. The overlap matrix, Pulay force, etc. is computed. Parameterizations for the non-orthogonal model are currently not in a refined state.

- `DEBUGON=` *integer*

1: Produce a range of files useful in debugging. These include printing out the Hamiltonian, density matrix, and the radial dependence of the bond integrals and pair potentials.
0: Do not produce files that can be useful in diagnosing problems in calculations

- `FERMIM=` *integer*

Selects the order of the recursion in the expansion of the density matrix (`CONTROL= 3`)

- `CGORLIB=` *integer*

Controls whether we use Niklasson's algorithm or a LAPACK subroutine to solve $AX = B$ in the recursive expansion of the Fermi operator (`CONTROL= 3`)
0: LAPACK subroutine
1: Niklasson's conjugate gradient algorithm.

- `CGTOL=` *real*

Tolerance for the conjugate gradient solution of $AX = B$ (`CGORLIB= 1`)

- `KBT=` *real*

Electron temperature in eV in the Fermi-Dirac distribution when occupying energy levels in the calculation of the density matrix. Does not apply when using the regular SP2 algorithm (`CONTROL= 2`) to calculate the density matrix since that is strictly zero electronic temperature.

- `NORECS=` *integer*

Number of recursion steps for the truncated SP2 expansion that mimicks the finite temperature Fermi-Dirac distribution (`CONTROL= 4`).

- `ENTROPYKIND=` *integer*

Controls how the electronic entropy is calculated if the density matrix is calculated at finite electronic temperature. If diagonalization is used then obtain get the exact entropy essentially for free during the calculation.
0: Do not calculate the electronic entropy
1: Exact Mermin $X \ln X$ expression for the Fermi-Dirac distribution.
2: Different form of the exact expression that can be useful when using the (`CONTROL = 5`) (*experimental*)
3: Approximate 4th order expansion of the exact form (*experimental*)
4: Approximate 8th order expansion of the exact form (*experimental*)

- `PPOTON=` *integer*

0: Do not include the pair potential when computing energy and forces

1: Include the analytic pair potential when computing energy and forces. An example is distributed with the source.

2: Read a tabular pair potential from `TBparam/ppots.dftb` when computing the energy and forces. LATTE will interpolate between points using the recipe from Numerical Recipes.

- `VDWON=` *integer*

No longer used.

- `SPINON=` *integer*

0: Assume double occupancy when occupying the energy levels

1: Perform a self-consistent, spin-polarized calculation. Self-consistent charge transfer must be enabled too (`ELECTRO= 1`).

- `SPINTOL=` *real*

Tolerance for determining self-consistency in the atom-centered spin densities.

- `ELECTRO=` *integer*

0: No self-consistent charge transfer. Apply local-charge neutrality.

1: Enable self-consistent charge transfer

- `ELECMETH=` *integer*

Controls how the electrostatics are computed when charge transfer is enabled

0: Ewald summation

1: Compute electrostatics in real space

- `ELEC_ETOL=` *real*

No longer used.

- `ELEC_QTOL=` *real*

Tolerance on the maximum change in the Mulliken partial charge on any atom between successive SCF cycles when applying self-consistent charge transfer. When applying local

charge neutrality, the SCF procedure will run until all partial charges are less than this value.

- `COULACC=` *real*

Tolerance on the accuracy of the electrostatic energy when using Ewald summation.

- `COULCUT=` *real*

When using Ewald summation `COULCUT` is the radius in Å within which the electrostatic are evaluated in real space. Setting `COULCUT` < 0 enables the optimal value of the cut-off radius to be calculated during the initialization of the electrostatic solver. When computing the electrostatics in real space, `ELECMETH = 1`, as in a gas-phase calculation, we add a cut-off tail to the $1/R$ dependence that terminates at $R =$ `COULCUT`.

- `COULR1=` *real*

When computing the electrostatics in real space, a cut-off tail is added to the $1/R$ dependence at $R =$ `COULR1` and terminated at $R =$ `COULCUT`.

- `MAXSCF=` *integer*

Maximum number of SCF cycles when optimizing the self-consistent charge transfer or local charge neutrality.

- `BREAKTOL=` *real*

Tolerance on the error in the total number of electrons when *i*) computing the chemical potential via diagonalization or, *ii*) determining the convergence of the SP2 algorithm.

- `MINSP2ITER=` *integer*

Minimum number of recursive cycles in the SP2 algorithm for computing the density matrix before we check for its convergence. This parameter helps us to avoid terminating the expansion too early.

- `SP2CONV=` *character*

Controls how the convergence of the SP2 algorithm is identified after we have passed `MINSP2ITER` cycles:

REL: Apply a relative measure whereby the expansion is terminated once the idempotency error increases between two successive iterations.
ABS: Continue the recursive expansion until the error in the electronic occupation is less than BREAKTOL.

- FULLQCONV= *integer*

0: Do not run to full SCF convergence (as specified by ELEC_QTOL) at every MD time step. Instead run only QITER SCF cycles at each MD time step.
1: Fully converge the SCF optimization to ELEC_QTOL at each MD time step

- QITER= *integer*

Run QITER SCF cycles per MD time step under self-consistent charge transfer (ELECTRO = 1) or local charge neutrality (ELECTRO= 0). This applies only if FULLQCONV= 0.

- QMIX= *real*

Parameter in the linear mixing scheme when computing the set of self-consistent Mulliken partial charges, that is, $q_{i+1} = \texttt{QMIX} q_{\text{new}} + (1 - \texttt{QMIX}) q_i$, where i lables the SCF cycle, q a Mulliken partial charge, and q_{new} is the Mulliken partial charge computed directly from the density matrix at SCF cycle $i + 1$.

- SPINMIX= *real*

Parameter in the linear mixing scheme for the spin densities.

- MDMIX= *real*

Parameter in the linear mixing scheme for the Mulliken partial charges during an MD simulation. During MD, we converge the SCF fully for the first 10 time steps. Following that we run QITER SCF cycles with a linear mixing in the SCF proceedure of MDMIX. When we apply the SCF-free, QITER= 0, propagation, MDMIX has a different meaning and should be tuned to ensure the stability of the algorithm.

- ORDERNMOL= *integer*

No longer used.

- `SPARSEON=` *integer*

0: Run the SP2 algorithm in dense matrix algebra with $\mathcal{O}(N^3)$ scaling with the number of atoms.

1: Run the SP2 algorithm in sparse matrix algebra (now OpenMP parallelized) with $\mathcal{O}(N)$ scaling with the number of orbitals. The best performance will be found for systems that are insulators with a large number of small molecules

2: Employ a graph-based partitioning of the system over shared memory cores. Currently experimental.

- `THRESHOLDON=` *integer*

0: Do not perform numerical thresholding of matrix elements in the sparse matrix implementation of the SP2 algorithm.

1: Remove matrix elements of abolute value $<$ `NUMTHRESH` at each iteration in the sparse matrix implementation of the SP2 algorithm.

- `NUMTHRESH=` *real*

Value of the numerical threshold on matrix elements in the sparse matrix implementation of the SP2 algorithm. Matrix elements with absolute value $<$ `NUMTHRESH` are removed at each recursive iteration if `THRESHOLDON= 1`.

- `FILLINSTOP=` *integer*

Stop allowing the fill-in of the sparse matrices in the $\mathcal{O}(N)$ SP2 algorithm after `FILLINSTOP` iterations.

- `BLKSZ=` *integer*

LATTE can optionally use the DBCSR parallel sparse matrix algebra package that is distributed with the `cp2k` package. `BLKSZ` defines the size of the sub-matrix blocks used in the blocked compressed sparse row storage format in DBCSR.

- `MSPARSE=` *integer*

Initializes the maximum number of non-zero elements per row when putting the dense

arrays into a CSR-like sparse storage formal. If `MSPARSE= 0` then on initialization the number of non-zeros will be set equal to the size of the dense arrays. This dimension is optimized in subsequent interations of the $\mathcal{O}(N)$ implementations of the SP2 algorithm.

- `LCNON=` *integer*

No longer used.

- `LCNITER=` *integer*

No longer used.

- `CHTOL=` *integer*

No longer used.

- `SKIN=` *real*

Width in Å of the skin, or buffer region, in the neighbor list constructions.

- `RELAX=` *integer*

1: Perform a molecular statics relaxation / geometry optimization.

0: Do not perform a molecular statics relaxation (that is, optimize the geometry).

- `RELAXTYPE=` *character*

`SD`: optimize geometry using a simple steepest descent algorithm.

`CG`: optimize geometry using a conjugate gradient algorithm (broken).

- `MAXITER=` *integer*

Maximum number of steps in the molecular statics relaxation/geometry optimization provided `RLXFTOL` is not reached first.

- `RLXFTOL=` *real*

The relaxation will stop once the absolute value of the maximum force acting on any atom falls below `RLXFTOL`, in eV Å^{-1}.

- `MDON=` *integer*

1: Perform an MD simulation. How the forces are calculated is controlled in `TBparam/control.in` whereas the type of MD simulation and other MD-specific parameters are defined in the file `MDcontroller`. .

0: Do not perform an MD simulation. If both `RELAX= 0` and `MDON= 0` then a static calculation of the total energy and forces will be performed.

- `PBCON=` *integer*

1: Apply periodic boundary conditions using the three lattice vectors defined in the file `bl/inputblock.dat`.

0: Do not apply periodic boundary conditions - perform a gas-phase calculation. Care should be taken to ensure the electrostatic interactions are computed accordingly.

- `RESTART=` *integer*

0: Read the input atomic coordinates (and nothing else) from `bl/inputblock.dat`

1: Restart a calculation by reading coordinates, electronic information, and velocities (if restarting an MD run) from the file `bl/restart.dat`.

- `CHARGE=` *integer*

Defines how many electrons should be added to or removed from the system. `CHARGE= 0` corresponds to a charge neutral system whereas `CHARGE= 1`, for example, corresponds the removal of one electron from the system. The total number of electrons $N_e = \sum_{i=1}^{N_{\text{atoms}}} N_e^i - \texttt{CHARGE}$, where N_e^i is the number of valence electrons on neutral, isolated atom i.

- `XBO=` *integer*

1: Use the extended Lagrangian Born-Oppenheimer MD formalism to time-reverisbly propagate the chemical potential, partial charges, spin densities, or on-site Hamiltonian matrix element shifts during MD, depending on the type of calculation you are performing. `XBO = 1` is also required in zero-SCF, fast QMMD, that is MD with `QITER = 0`.

0: Do not use the extended Lagrangian Born-Oppenheimer formalism in your MD simulation.

- `XBODISON=` *integer*

1: Apply dissipation of kind `XBODISORDER` to counteract the accumulation of numerical noise in extended Lagrangian Born-Oppenheimer MD and fast-QMMD.
0: Do not apply any dissipation in the propagation of the electronic degrees of freedom in extended Lagrangian Born-Oppenheimer MD.

- `XBODISORDER=` *integer*

Selects the dissipation scheme used during extended Lagrangian Born-Oppenheimer MD and fast-QMMD when `XBODIS= 1`. $3 \leq$ `XBODISORDER` ≤ 9, with 3 being the most severe, and 9 the weakest dissipation.

- `NGPU=` *integer*

When the Nvidia CUDA library for the SP2 algorithm is compiled into LATTE, `NGPU` controls the number of GPUs the SP2 algorithm (in $\mathcal{O}(N^3)$ form), should be parallelized over. Tested up to and including `NGPU= 3`.

- `KON=` *integer*

0: Perform all calculations in real space with real numbers.
1: Use k-space integration to compute the density matrix. This has currently only been implemented with diagonalization (`CONTROL= 1`). The k-space mesh is defined (simply) in the file `TBparam/kmesh.in`.

- `COMPFORCE=` *integer*

0: When performing a static calculation do not compute the band structure forces. These calculations are expensive when k-space integration is used if one is interested only in obtaining the electronic densities of states or total energy, for example.
1: Compute the band structure forces during a static calculation.

The subsequent lines in `TBparam/control.in` are for experimental features of the code that are not being actively maintained. It's safe to ignore them, but they must be there are LATTE by default will read them in.

V. MDCONTROLLER

The `MDcontroller` file controls which kind of MD simulation to perform (NVE, NVT etc.), the time step, how often energies and restart files are written etc.

A. Structure and format of `MDcontroller`

```
MAXITER= 20000
UDNEIGH= 25
DT= 0.5
TEMPERATURE= 300.0 RNDIST= UNIFORM SEEDINIT= RANDOM
DUMPFREQ= 250
RSFREQ= 1000
WRTFREQ= 25
TOINITTEMP= 1
THERMPER= 500
THERMRUN= 10000
NVTON= 0 NPTON= 0 AVEPER= 1000 FRICTION= 100.0 SEED= 54
PTARGET= 0.0 NPTTYPE= ISO
SHOCKON= 0
SHOCKSTART= 15000
SHOCKDIR= 1
UPARTICLE= 2600.0 USHOCK= -4590.0 C0= 990.0
MDADAPT= 0
GETHUG= 0 E0= -795.725 V0= 896.984864 P0= 0.083149
```

B. Setting values in `MDcontroller`

- `MAXITER=` *integer*

Run the MD simulation until `MAXITER` MD time steps have been completed. This is the total number of time steps starting from time step 1, not the number of time steps in addition to those already completed when restarting a simulation from a restart file.

- `UDNEIGH=` *integer*

Re-build the neighbor lists every `UDNEIGH` time steps.

- `DT=` *real*

Length of the MD time step in femtoseconds.

- `TEMPERATURE=` *real*

The temperature used in the initialization of the velocities and the target temperature for the various thermostats.

- `RNDIST=` *character*

`UNIFORM`: Draw random numbers from a uniform distribution when initializing the velocities.
`GAUSSIAN`: Draw random numbers from a Gaussian distribution when initializing the velocities.

- `SEEDINIT=` *character*

`UNIFORM`: Do not change the seed when initializing the random number generator.
`RANDOM`: Generate a new seed when initializing the random number generator that depends on the computer clock, that is, a different sequence of random numbers will be generated for every simulation. This will effect the initialization of velocities as well the Langevin and Andersen thermostats.

- `DUMPFREQ=` *integer*

Write to file a snapshot of the simulation in every `DUMPFREQ` time steps. These files are put in the directory `animate/` and are of `dump2atomeye.*.cfg` format for visualization with `Atomeye` if periodic boundary conditions are used. Trajectories from gas-phase MD simulations are appended to the file `animate/myXYZfile.xyz` for visualization in `VMD` or a compatible viewer.

- `RSFREQ=` *integer*

Write a restart file every `RSFREQ` time steps. All restart files, labeled by the time step, are

placed in the directory `Restarts`. The last restart file also overwrites the file `Restarts/-restartMD.last.dat`, which helps with restarting simulations using scripts. To restart an MD simulation, set `RESTART= 1` in `TBparam/control.in` and copy your restart file to `bl/restart.dat`.

- `WRTFREQ=` *integer*

Output the instantanous total energy, temperature, pressure, HOMO-LUMO gap, chemical potential, stress tensor, or other diagnostics as specified in `src/tbmd.f90` every `WRTFREQ` time steps. By default the output is written to screen. It can easily be stored using `./LATTE_DOUBLE > myfile.dat`, for example, when running the code.

- `TOINITTEMP=` *integer*

0: Do not initialize velocities at the start of a new MD simulation (does not affect restarts)
1: Initialize velocites at the start of a new MD simulation (does not affect restarts). It is hard to foresee a situation where one would specify `TOINITTEMP` $\neq 1$.

- `THERMPER=` *integer*

The frequency (in time steps) where the velocities or velocities and box size are updated when using the velocity rescaling thermostat (`NVTON= 1`) or the simple barostat (`NPTON= 1`), respectively. The velocities or box size are updated in response to averages of the temperature or pressure over `AVEPER` time steps.

- `THERMRUN=` *integer*

Run an MD simulation with a thermostat or barostat until the total number of time steps reaches `THERMRUN` time steps. This is not the number of additional time steps when starting from a restart file. When the total number of time steps exceeds `THERMRUN` the simulation reverts to NVE dynamics.

- `NVTON=` *integer*

Controls which, if any, thermostat is used in the simulation.
0: No thermostat. A microcanonical, NVE trajectory will be computed.
1: Simple velocity rescaling. This does not yield dynamics consistent with the canonical

ensemble but it is a very simple method for preparing a system for NVE dynamics at a specified temperature. Velocities are rescaled every `THERMPER` time steps based on the temperature averaged over `AVEPER` time steps.
2: Langevin dynamics. The `FRICTION` input is used to derive the Langevin friction parameter, $\alpha_i = m_i/$`FRICTION`, where i labels atoms and m_i is the mass of atom i. Hence, `FRICTION` has units of femtoseconds.
3: Andersen thermostat. Currently broken.
4: Nosé-Hoover thermostat. Here the `FRICTION` input again has units of femtoseconds and sets the strength of the coupling between the atomic trajectories and the Nosé-Hoover heat bath.

- `NPTON=` *integer*

0: Do not change the size or shape of the box during the simulation in response to the computed stress tensor.
1: Perform velocity rescaling and adjust the simulation box according to achieve the target temperature and hydrostatic pressure. The velocities and box are rescaled every `THERMPER` time steps based on temperatures and components of the stress tensor averaged over the previous `AVEPER` time steps and the targets for the temperature (`TEMPERATURE`) and hydrostatic pressure (`PTARGET`). The box can be rescaled isotropically (dependent only on the hydrostatic pressure) or anisotropically (dependent on σ_{11}, σ_{22}, and σ_{33}) depending on the setting of `NPTTYPE`.

- `AVEPER=` *integer*

Number of time steps over which the temperature, or temperature and pressure, are averaged in the velocity rescaling thermostat (`NVTON= 1`) and the simple barostat (`NPTON= 1`).

- `FRICTION=` *real*

A parameter with units of femtoseconds used in the Langevin (`NVTON= 2`), Andersen (`NVTON= 3`), and Nosé-Hoover (`NVTON= 4`) thermostats.

- `SEED=` *integer*

No longer used.

- `PTARGET=` *real*

Target hydrostatic pressure in GPa for the simple barostat (`NPTON= 1`). The size and optionally shape of the simulation box will be adjusted until so that the computed hydrostatic pressure is equal to, on average, `PTARGET`.

- `NPTTYPE=` *character*

`ISO`: Expand or contract the three axes of the box isotropically due to the computed hydrostatic pressure and the value of the target pressure, `PTARGET`.
`ANISO`: Allow each orthogonal axis of the simulation box to expand or contract independently of the others in response to the diagonal components of the stress tensor and the target pressure, `PTARGET`.

- `SHOCKON=` *integer*

1: Starting at MD time step `SHOCKSTART` shrink the simulation box along axis `SHOCKDIR` to mimick shock compression. The 'shrinking-cell' boudary condition yields a uniaxial compression that is consistent with the Rankine-Hugoniot jump conditions.
0: Do not apply the shrinking-cell boundary condition.

- `SHOCKSTART=` *integer*

The MD time step when the shrinking cell boundary condition is applied. The time step at which simulation will return to an NVE trajectory depends on the size of the box, U_s and U_p, and is calculated during initialization.

- `SHOCKDIR=` *integer*

Selects which axis of the simulation cell is uniaxially compressed during the application of the shrinking cell boundary condition.
1: x
2: y
3: z

- `UPARTICLE=` *real*

The user specified particle velocity, U_p, in m s^{-1} for the shrinking cell boundary condition.

- `USHOCK=` *real*

The shock wave velocity, U_s, in m s^{-1} for the shrinking cell boundary condition. If `USHOCK` < 0 in the input then it is computed internally using the universal liquid Hugoniot. This requires the input of the sound velocity under ambient conditions.

- `C0=` *real*

The sound velocity of the liquid under ambient conditions in m s^{-1}. This is used in the calculation of U_s using the universal liquid Hugoniot.

- `MDADAPT=` *integer*

An experimental feature whereby the number of SCF cycles and mixing and adjusted on-the-fly when the code detects the HOMO-LUMO gap closing. This feature helps energy conservation when chemistry is going on. The number of SCF cycles etc. are currently hard-coded in `src/bodirect.f90`.
0: Do not adapt the SCF procedure
1: Adjust SCF procedure on-the-fly

- `GETHUG=` *integer*

Compute states on the principal (unreacted) Hugoniot by means of a simple Hugoniostat. LATTE self-consistently updates the temperature and density until $(E-E_0)-1/2(P+P_0)(V-V_0)=0$, where P is the target pressure on the Hugoniot and P_0, V_0, and E_0 are the pressure, specific volume, and internal energy, respectively, of the initial state. The specific volume is updated by a volume rescaling barostat and the temperature by the Langevin thermostat. The target pressure, in GPa, is specified using `PTARGET`.

- `E0=` *real*

Internal energy, potential plus kinetic, of the initial, pre-shocked state in eV.

- `V0=` *real*

Volume of the simulation cell at the initial, pre-shocked state in Å^3.

- `P0=` *real*

Pressure of the simulation cell at the initial, pre-shocked state in GPa.

---

* Electronic address: `cawkwell@lanl.gov`