**­­­­ ChIMES LSQ and ChIMES MD User Guide**

**(4-body code)**

**Contributing Authors:**

*Rebecca Lindsey (RKL)*

*Laurence Fried (LEF)*

*Nir Goldman (NG)*

*Lukas Koziol (LK)*

*Manual last updated: Aug. 29 2018 (RKL)*

**Limitations/To do:**

* Update the test suite/manual text surrounding test suite
* Add new tests to suite
* Test LAMMPS linking
* Clean up ancillary options/utilities:
  + Excluding many-body interactions from (LSQ and MD Codes)
  + Penalty function modification (LSQ and MD Codes)
  + Self-consistent machinery in the LSQ code
  + Ensembles
  + PES Scan generator
  + Small utilities
* Overbonding won’t work properly for non-H2O and when multiple “to” overbonding atom types are requested
* MD code functionality when parameter files has USEPOVER true but FITPOVER false is not working. This needs to be fixed.

**Quick Navigation:**

**Getting Started**

* [Compiling/running the code](#Compiling/running the code:)
* [Running the test suites](file:///usr/WS1/fried/git/chimes_lsq.develop/doc/correct)
* [Compiler options](#Some Compiler options)
* [Makefile 101](#Makefile 101)

**Digressions**

* [Determination of “allowed” 3-body Power Sets](file:///usr/WS1/fried/git/chimes_lsq.develop/doc/allowed" \l "_Determination_of_)
* A Note on Units

**LSQ Code**

* [Main Control Variables](#_LSQ_CODE:_Main)
* [Topology Variables](#_LSQ_CODE:_Topology)
* [Extras](#_Extras)
* [The LSQ python script](#LSQ Python Codes)

**MD Code**

* [Main Input File](#_MD_CODE:_Main)
* [Parameter file](#_MD_CODE:_Parameter) (This section of the manual is under construction)

**Utilities**

* [PES Scan generator](#_PES_Scan_generator)
* [Small utilities](#Small Utilities – ANCILLARY SUPPORT)

# Compiling/running the code:

The md code supports both serial and MPI functionality. To switch between the two, change the “CXX” command in the Makefile. Refer to the examples therein.

* Before compiling, navigate to the src directory and remove “chimes\_md” and “chimes \_lsq”
* If compiling the MD code on LLNL HPC, load module intel and impi (or mvapichX/X.X)
* Makefile options include the following tabulated values. Note that “xx” stands for either “lsq” or “md.” The make command is executed as make <options>. The resulting executables are of the form chimes\_xx.

|  |  |
| --- | --- |
| chimes \_xx | Compile the xx code. |
| travis | Compile Travis trajectory analyzer. See the .pdf file in contrib/Travis for more information. |
| packmol | Compile packmol initial configuration generator. See the packmol website for more information |

* In serial mode, the lsq and md codes can be run with:   
    
  /path/to/executable < input\_file.name > output\_file.name
* In MPI mode, the md code can be run with:   
    
  srun –n x /path/to/executable < input\_file.name > output\_file.name  
    
  on LLNL HPC (using slurm, and submitted to the queue). Note that the “mpiicc” option should be used for this compilation.  
    
  or:   
    
  mpirun –np x –stdin all /path/to/executable < input\_file.name > output\_file.name  
    
  on machines that do not have slurm. The “-stdin all” flag gives all processors access to stdin. For this application, the “mpicxx” option is recommended.

Note that a few special compiler options can be specified for the lsq and md codes. For example, by adding “-D VERBOSITY=1,” to the “MYFLAGS” argument list given in the Makefile, greater detail will be printed to output files. Including “-D FORCECHECK=1” will print out the “3b\_results.dat” file at each time step.

# Running the test suites/generating “correct” test suite reference files

**[This section needs updating as of 08/29/18 – RKL ]**

Note that the suite is intended to use lsq.py version. Immediately after pulling code (i.e. before making any modifications):

* Run from test\_suite-xx with ./generate\_test\_suite.sh

Once edits have been made:

* Run from test\_suite-xx with ./run\_test\_suite.sh

**Notes for Reading the Makefile**

**Flags and options:**

* LINK2LMPS:  
  no Do not link MD code to LAMMPS  
  yes Link MD code to LAMMPS
* FORCECHECK:  
  1 Print out 3b forces on each atom at each time step (generally for debugging)  
  0 Don't print out the aforementioned (default)
* VERBOSITY:  
  1 Verbose (default)  
  0 Not verbose (not recommended)
* FPENALTY\_POWER  
  (Any double) The power on the fpenalty function – 3.0 is typically default
* CHECK\_CHEBY\_RANGE: (i.e. check for proper behavior of transformed distance at end of cheby interval)  
  0 Do not check  
  1 Check (default)
* WARN:  
  TRUE when potential problems are encountered, output a warning, but do not kill the code (default)  
  FALSE kill the code when potential problems are encountered

# Some Compiler options

|  |  |
| --- | --- |
| CXX=mpiicc -g -D USE\_MPI -std=c++11 | Use for LLNL HPC; SLURM compatible; MPI |
| CXX=mpicc -g -D USE\_MPI -std=c++11 | Use for GNU MPI |
| CXX=mpicxx -g -D USE\_MPI -std=c++11 | Use for Debian (OSX) MPI |
| CXX=g++ -g -std=c++11 | Serial with debugging information |
| CXX=g++ -O3 -std=c++11 | Serial, extreme optimizations |
| CXX=g++ -g -std=c++11 -Wall -Wextra -Wuninitialized -O1 | Serial Specialty debugging |
| CXX=g++ -g -std=c++11 -Wall –Wextra | Serial Specialty debugging |
| CXX=icpc -std=c++11 -fast -Wall | Serial Specialty debugging |

# Makefile 101

|  |  |
| --- | --- |
| “Target” | The name of a compiled object (i.e. the "\*" of "\*.o", or name of an exectuable) |
| “Rule” | How to compile something, for example, g++ -o my\_exec file1.cpp file2.cpp is a rule |
| “Prerequisite” | The files that are required to make a target (the "file1.cpp and file2.cpp from above) |
| $@ | Name of target of a rule |
| $< | Name of the first prerequisite of a rule |
| $^ | Name of all prerequisites of a rule, with spaces between them |
| -c | Compile file directly into an object... i.e. g++ -c myfile.cpp produces myfile.o |

# Determination of “Allowed” Many-body Power Sets

ChIMES force fields are *defined* on an *atom-pair basis*, but are invariant with respect to *atom ordering*. Thus, in order to determine atom pairs that can be generated for a set of atoms **a** = {*a*1, *a*2, *a*3, … , *a*n}, we first generate all permutations of a. Possible atom pairs are then generated from each permutation of **a**.

For example:

**a** = {C,C,O} = **a**p0; which yields pairs {CC, CO, CO}; sorting pairs by alpha. Order: {CC, CO, CO}

**a**p1 = {C,O,C}; which yields pairs {CO, CC, OC}; sorting pairs by alpha. Order: {CO, CC, CO}

**a**p2 = {O,C,C}; which yields pairs {OC, OC, CC}; sorting pairs by alpha. Order: {CO, CO, CC}

So for the set of 3 atoms, {C,C,O}, we determine the possible (equivalent) sets of pairs: {CC, CO, CO}, {CO, CC, CO}, {CO, CO, CC}.

# Important Note on Units:

* For all pair types *except DFTBPOLY*, the LSQ C++ code expects the .xyzf file to have **coordinates in Å**, but **forces in Hartree/Bohr**
* For all pair types *except DFTBPOLY*, the LSQ C++/Python codes output forces/potential parameters in terms of kcal/mol/ **Å** and Kcal/mol
* For all pair types *except DFTBPOLY*, the LSQ C++ code expects the .xyzf file to have tensors given in GPa. For DFTBPOLY, tensors are expected in atomic units (Hartree/Bohr3)

**Note on comments:**

The 4-body version of the code (v 399 and after) treats the character strings “!” or “##” as starting the beginning of a comment.

Prior 4-body versions and the 3-body version do not have a clear concept of comment. In 4 body v 399 and greater, a warning will be printed if input is not understood. In prior versions of the code input that is not understood is silently passed over.



# LSQ CODE: Main Control Variables

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Allowed Values** | **Usage** |
| # TRJFILE # | <any string>  or  MULTI <any string>  or  <any string> ORTHO  or  <any string> NON-ORTHO | Provides the name of the simulation trajectory file. Files use a .xyzf format, which is like the standard .xyz format, with two exceptions: (1) the line after that specifying number of atoms will contain information on box dimensions and optionally stress tensor and system energy, and (2) each coordinate line has x, y, and z forces on the corresponding atom appended. Option <any string> is the trajectory file name.  For “ MULTI <any string>”, “MULTI” tells the program to expect multiple trajectory files. Here, <any string> is the name of a file structured like:  <nfiles>  <frames to read> path/to/file-1.xyzf  <frames to read> path/to/file-2.xyzf  ...  <frames to read> path/to/file-n.xyzf  **NOTE:** When using this option, “# NFRAMES #” be equal to the sum of <frames to read> for each of the <nfiles>  Option “<any string>” and option “<any string> ORTHO” means the trajectory parser will expect the line following the atom number line to be formatted like:  <box x-len> <box y-len> <box z-len>  Option “NON-ORTHO” means the trajectory parser will expect  the line following the atom number line to be formatted like: <latvec-1\_x> <latvec-1\_y> <latvec-1\_z> <latvec-2\_x> <latvec-2\_y> <latvec-2\_z> <latvec-3\_x> <latvec-3\_y> <latvec-3\_z>, where “latvec-a\_b” is the bth component of the ath lattice vector.  **Currently only option “<any string>” is implemented.** |
| # WRAPTRJ # | “true” or “false” | Defines whether coordinates should be wrapped (i.e. application of post-simulation periodic boundary conditions). Automatically disabled when ghost atoms (layers) are used |
| # NFRAMES # | Any integer > 0 | Number of frames in the .xyzf file |
| # NLAYERS # | Any integer => 0 | Number of supercell ghost layers to create from simulation box (i.e. replicate images). A value of 0 yields the original box. A value of 1 yields a single shell of replicated boxes around the original box (i.e. 27 boxes). |
| # FITCOUL # | “true” or “false” | Defines whether charges should be fit, or held fixed at user-defined values. Note that currently, functionality is only supported when # FITCOUL # is true, or when  # FITCOUL # is false and all charges are zero.  If # FITCOUL # is false, but charges are non-zero, program will attempt to subtract charge contributions from forces. |
| # CNSCOUL # | “true” or “false” | Apply constraints to charges? See “Adding Charge Constraints” in “[Extras](#_Extras_1)” section for more information on use. |
| # FITSTRS # | “true” or “false” or “ALL” or “FIRST <integer>” or ALL  To be included soon: “ALLFIRST <integer>” | Defines whether stress tensors should be included in fit. If true, they should be listed at the end of the box dimension line, in the .xyzf file. For DFTPOLY pair types, tensors are expected in atomic units, otherwise, tensors are expected in GPa.  For option “true” the xx, yy, and zz tensors should be listed at the end of the box dimension line in the xyz file, for each frame.  For option “ALL” the xx, yy, zz, xy, xz, yz tensors should be listed at the end of the box dimension line in the xyz file, for each frame.  Option “FIRST <integer>” is like option “true,” but tensors are only read/processed for the first <integer> frames. Do not include tensors for the other frames.  Code currently does not support use of this feature along with Ewald functionality.  See notes in “[Extras](#_Extras_1)” section for more information on use. |
| # FITENER # | “true” or “false”  or “FIRST <integer>” | Defines whether to consider energies in fit. If true, energies should be the last entry on the box length line of the .xyzf files (regardless of what other options are on the line). Should be in units of kcal/mol. When energies are included in the fit, an additional column is added to the A-matrix where values are zero for forces and stresses, and unity for energies.  Functionality has not been verified for compatibility with Ewald functionality. |
| # FITPOVR # | “true” or “false” | Defines whether a ReaxFF type over-coordination force should be applied. Note that currently, functionality is only supported when # FITPOVR # is true, or when  # FITPOVR # is false and all # USEOVRP # are set to false.  If # FITPOVR # is set to false and at least one # USEOVRP # is set to true, program will attempt to subtract over-coordination contributions from forces. |
| # PAIRTYP # | “SPLINE”, “CHEBYSHEV”, “DFTBPOLY”, or “INVRSE\_R” | The form of the inter-molecular potential to be used for fitting.  If the # PAIRTYP # is **DFTBPOLY**, an integer must also be given to indicate the polynomial order. A value of n corresponds to monomials that range from order 1 to n.  If the # PAIRTYP # is **CHEBYSHEV**, 5 integers must be provided, where the first 3 the order for 2-, 3-, and 4-body interactions. Note that 2-body order follows the same convention as for DFTBPOLY types, while the many body integers provides the order for 3-body interactions, and a value of n indicates monomials ranging in order from 0 to n-1.  The final two integers specify the range for the Chebyshev distance transformation. Values must fall between -1 and 1.  If the # PAIRTYP # is **INVRSE\_R**, an integer must also be given to indicate the number of terms. |
| # CHBTYPE # | “DEFAULT”, “INVRSE\_R”, “MORSE” | Determines how distance is transformed for a Chebyshev-type fit. Default means no transformation will be applied. Only used if # PAIRTYP # is CHEBYSHEV. |

**­­­­**

# LSQ CODE: Topology Variables

# 

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Allowed Values** | **Usage** |
| **# NATMTYP #** | Any integer > 0 | Number of unique atom types in trajectory |
| # TYPEIDX # | Integers > 0 | Ascending integers, ranging from 1 to # NATMTYP#. Index for each unique atom type. Values form a column below  # TYPEIDX #. |
| # ATM\_TYP # | Any chemical symbol  (i.e. C or Si) | Chemical symbol for each unique atom type. Values form a column below # ATM\_TYP #. |
| # ATMCHRG # | Any float OR “+” or “-“ | If # FITCOUL # is false: user-defined partial atomic charge for each unique atom type. If # FITCOUL # is true: either a positive or negative sign, to indicate how pair charge signs should be assigned. |
| # ATMMASS # | Any float > 0 | User-defined atomic mass for each unique atom type. Values form a column below # ATMMASS #. Only used to set masses for MD in the resulting parameter file. |
| # PAIRIDX # | Integers > 0 | Ascending integers, ranging from 1 to the total number of unique atom pair types. Values form a column below  # PAIRIDX #. |
| # ATM\_TYX # | Any chemical symbol  (i.e. C or Si) | # ATM\_TY1 # or # ATM\_TY2 #. Used to define interaction pair types. Order does not matter. |
| # S\_MINIM # | Any float > 0 | Inner cutoff for pair potential in Angstroms. |
| # S\_MAXIM # | Any float > 0 | Outer cutoff for pair potential in Angstroms. |
| # S\_DELTA # | Any float > 0 | Grid spacing for splines. Number of splines for a given pair type is (# S\_MAXIM # – # S\_MINIM #)/# S\_DELTA #. Only used if # PAIRTYP # is SPLINE, but regardless, something (such as “NA”) still needs to be entered in this field. |
| # MORSE\_LAMBDA # | Any float > 0 | Morse-type lambda to use when a Morse transformation is used on pair distance for a CHEBYSHEV type pair. Only used if # PAIRTYP # is CHEBYSHEV. |
| # USEOVRP # | “true” or “false” | Determines whether overbonding should be included for pair type. |
| # TO\_ATOM # | Any chemical symbol  (i.e. C or Si) | The atom that overbonding is considered for. For example, “O” for water-type systems. Only used if # USEOVRP # is true. |
| # P\_OVERB # | Any float > 0 | *p*over from the ReaxFF overbonding equation of order 1. Only used if # USEOVRP # is true. |
| # R\_0\_VAL # | Any float > 0 | *r*0 from the ReaxFF overbonding equation of order 1. Only used if # USEOVRP # is true. |
| # P\_1\_VAL # | Any float > 0 | *p*bX,1 from the ReaxFF overbonding equation of order 1. “X” refers to the “TO\_ATOM.” Only used if # USEOVRP # is true. |
| # P\_2\_VAL # | Any float > 0 | *p*bX,2 from the ReaxFF overbonding equation of order 1. “X” refers to the “TO\_ATOM.” Only used if # USEOVRP # is true. |
| # LAMBDA6 # | Any float > 0 | *λ*6 from the ReaxFF overbonding equation of order 1. Only used if # USEOVRP # is true. |
| # NIJBINS #  # NIKBINS #  # NJKBINS # | Any integer > 0 | Optional. Used for generating {rij, rij, rik} histograms. Integers give the number of bins to consider for each pair distance. Default values are {0, 0, 0} |
| # FCUTTYP # | “CUBIC”  or  “TERSOFF <float>” | Cutoff style for 3b Chebyshev potentials. Should be specified on its own line, under # PAIRIDX# entries. Applies to all-body interactions. For “TERSOFF <float>”, the float gives the fcut function kick-in distance as: rcut - <float>\*rcut, and should take on values between 0 and 1. |

# Extras

**Including Stress Tensors in Fit**

This functionality allows the XX, YY, and ZZ (or XX, YY, ZZ, XY, XZ, YZ) components of the stress to be included in the least squares fit. Weighting is generally required due to the imbalance between atomic forces and stress tensors in. Weights can be quickly generated using a bash command something like the following, where 50 is the number of frames, and 600 is 3 times the number of atoms in the system.

factor=500.0; rm -f ${factor}-weights.dat; for i in {1..50}; do for j in {1..600}; do echo "1.0" >> ${factor}-weights.dat; done; for j in {1..3}; do echo $factor >> ${factor}-weights.dat; done; done

**Excluding 3-body Interactions – ANCILLARY SUPPORT**

Interactions corresponding to specific atom triplets can be excluded from the fitting process by including the following lines above the # NATMTYP # entry in the input file:

EXCLUDE 3B INTERACTION: 2

OOOOOO

COCOOO

**Adding Charge Constraints**

If the user desires to fit charges during the force matching process, they can either do so with no constraints (the default option), or by specifying n\_atom\_pairs -1 constraints. These constraints are added to the end of the input file, as discussed in the “Modifying the Penalty Function” section above. Take water, as an example; here we have 3 pair types, OO, OH, and HH. We want to enforce that the sum of all charges is zero, and that H has half the charge of O, and we can do so by adding the following lines:

CHARGE CONSTRAINTS:

OO HH OH 1000.0 -4000.0 0.0 0.0

OO HH OH 1000.0 4000.0 4000.0 0.0

The first line can be re-written as the equation: 1000\*qOqO - 4000\*qHqH = 0, and enforces the relationship that |qO| = 2|qH|.

The second line can be re-written as the equation: 1000\*qOqO + 4000\*qHqH + 4000\*qOqH = 0, and enforces that the sign of qOH needs to be opposite of qHH and qOH (i.e. negative), and the relationship |qOH| = 2|qHH|. Note that each line needs entries for all atom pairs.

**Separating 2-body and 3-body inner and outer cutoffs**

Currently, 3-body outer cutoffs are taken to be equivalent to the constituent 2-body outer cutoffs, by default. One has the option of setting all outer cutoffs to an equivalent value by adding the following line to the end of the fm\_setup,in file:

SPECIAL 3B S\_MAXIM: ALL 4.0

Otherwise, if one wants to specify each cutoff separately, syntax similar to the following should be used:

SPECIAL 3B S\_MAXIM: SPECIFIC 4

CCCCCC CC CC CC 4.4 4.4 4.4

COCOCC CC CO CO 4.4 4.0 4.0

OOCOCO CO CO OO 4.0 4.0 6.5

OOOOOO OO OO OO 6.5 6.5 6.5

Where the “4” is the number of cutoffs to be listed. Any 3-body type for which a line is not provided will use the same S\_MAXIM as the 2-body interactions, where constituent pairs determine the cutoff. Similar syntax is used for 4-body interactions.

Similarly, special inner cutoffs can be requested, which is particularly useful when extrapolating the 2-body interactions. Keep in mind that separate values need to be specified for each pair within the triplet, when the “SPECIFIC” keyword is used:

SPECIAL 3B S\_MINIM: ALL 0.0

or

SPECIAL 3B S\_MINIM: SPECIFIC 4

OOOOOO OO OO OO 2.00000 2.00000 2.00000

OOOHOH OO OH OH 2.00000 0.80000 0.80000

HHOHOH OH OH HH 0.80000 0.80000 1.00000

HHHHHH HH HH HH 1.00000 1.00000 1.00000

Similar syntax is used for 4-body interactions.

# LSQ Python Codes

|  |  |  |
| --- | --- | --- |
| **Name** | **Usage** | **Notes** |
| lsq2.py | See script “help”. Uses several defaults. For our standard SVD fit, using our standard file names:  Python lsq2.py > params.txt | Produces output compatible with the 4-body version of the MD code. |
| lsq.py | python lsq.py A.txt b.txt params.header  > params.txt | Produce output compatible with original version of MD code. **Ancillary support.** |

# MD CODE: Main Input File

|  |  |  |
| --- | --- | --- |
| **Keyword** | **Allowed Values** | **Usage** |
| # PLOTPES # | See PES section | If true, this is the only keyword needed in the file. Defaults to false. **Ancillary support.** |
| **# RNDSEED #** | Any integer > 0 | Seed for random number generator. |
| # TEMPERA # | Any float > 0 | Target simulation temperature, in Kelvin. |
| # PRESSUR # | Any float > 0 | Pressure in GPa for an NPT ensemble simulation. See section “[ensembles](#_MD_CODE:_Ensembles).” |
| # CONVCUT # | float between 0 and 1 | Maximum allowed fraction deviation from input temperature. Defaults to 0.10 (10%). |
| # CMPRFRC # | “true <file name>”  or “false” | If true, forces are computed for a single time step and compared to the forces in the given file. |
| **# CHECKFRC #** | “true” or “false” | If true, forces are computed numerically from derivatives of the energy. This is very expensive, but it is a valuable consistency check on the code. |
| # SLFCNST # | “true” or “false” | If true, prints a VASP POSCAR file – Assumes a cell with all angles orthogonal. Defaults to false. **Ancillary support.** |
| # TIMESTP # | Any float > 0 | MD time step in femtoseconds (fs). |
| # N\_MDSTP # | Any integer > 0 | Number of MD steps. |
| # PENTHRS # | float between 0 and 1 | If a penalty kick-in for any atom pair exceeds <value>\*per-atom-econs, the job is killed. Default setting is to turn this feature off. |
| # NLAYERS # | Any integer => 0  Or  <int> REPLICATE <int> | First int: number of ghost supercell layers to create from simulation box. A value of 0 yields the original box, and pbc are used to determine MIC distances. A value of 1 yields a single shell of replicated boxes around the original box, in the (+)/(-) direction, and requires use of neighbor lists to find distances. **When 3- or greater body interactions are included, use NLAYERS >= 1. LJ pair type does not use layers.**  Second int: Number of real replicates of the system to create, prior to ghost atom construction. Provides a convenient way to easily explicitly increase system size. “Real” replicates are build in the positive direction, so 1 layer results in 8\*NATOMS atoms. |
| # USENEIG # | “true” or “false” | If true, neighbor lists are used. Will automatically determine padding based on velocities and defines separate 2-, 3-, and 4-body neighbors as atoms within <longest n-body FF cutoff> + <padding>. Defaults to false, unless ghost atoms are used – in that case, defaults to true. |
| # PRMFILE # | A file name | Name of the input parameter file. |
| # CRDFILE # | Many options. | See section “[input coordinate files](#MD CODE: Input Coordinate Files).” |
| # VELINIT # | “READ”, “GEN”, or  “RESTART” | Initialize velocities using Box Muller (GEN), or read from coordinate file (READ). The RESTART option will restart a simulation using a previously generated restart file. The restart file should be listed as the CRDFILE. |
| # CONSRNT # | Many options. | See section “[ensembles](#_MD_CODE:_Ensembles).” |
| # PRSCALC # | “ANALYTICAL” or “NUMERICAL” | Compute pressures analytically or numerically? |
| # WRPCRDS # | true or false | Apply PBC to coordinates? If not specified, simulation defaults to true, unless ghost atoms are used – in that case, defaults to false. Use false if post processing for properties such as mean squared displacements. |
| # FRQTRAJ #  Depreciated: “FRQDFTB” | Any integer > 0 | How frequently should the DFTB .gen file be printed? **Note: All other “FRQ” and “PRNT” type variables must be specified below this one.** |
| # TRAJEXT # | “GEN”, “XYZF”, or “LAMMPSTRJ” | Format for main trajectory file. “GEN” produces a dftb+ style .gen file, “XYZF” a standard .xyzf file (coordinates only), and “LAMMPSTRJ” a .lammpstrj file with coordinates and forces. The latter format is recommended when interfacing with visualization software, as many features file format features will be interpreted natively. Default value is “GEN”. |
| # FRQENER # | Any integer > 0 | How frequently should the energies, etc be generated? |
| # PRNTVEL # | “false” or  “true” and  “FRQDFTB” or an integer > 0 | Should the velocities be printed durng the simulation? If so, specify the frequency for output. “FRQDFTB” means to print with the same frequency as the .gen file. Defaults to false. |
| # PRNTFRC # | “false” or  “true” and  “FRQDFTB” or an integer > 0 | Should the forces be printed during the simulation? Same as for # PRNTVEL # |
| # PRNTBAD # | “false” or “true” | If true, prints two additional xyz files, one for r\_ij < r\_cut,in, and one for r\_cut,in < r\_ij < r\_cut,in+d\_penalty |
| # PENTHRS # | float between 0 and 1 | If included, the float value is between 0 and 1. If a penalty kick-in for any atom pair exceeds <value>\*per-atom-econs, the job is killed. Default setting is to turn this feature off. |
|  |  |  |

# 

# MD CODE: Ensembles

The following are possible entries for the “# CONSRNT #” line. Note: “TTC” is the thermostat time constant, and “BTC” is the barostat time constant.

|  |  |  |
| --- | --- | --- |
| **Name** | **Support** | **Notes** |
| NVT-SCALE | Supported |  |
| NVT-MTK HOOVER <TTC> | Supported |  |
| NVE | Supported |  |
| *NPT-MTK HOOVER <TTC> <BTC>* | Supported |  |
| *NPT-BEREND <TTC> <BTC>* | Ancillary |  |
| NVT-BEREND <TTC> | Supported |  |
| *NPT-BEREND-ANISO <TTC> <BTC>* | Ancillary |  |
| LMP-NVT <TTC> | Under testing/verification | Requires compilation with LINK2LMPS flag true |
| LMP-NPT <TTC> <BTC> | Under testing/verification | Requires compilation with LINK2LMPS flag true |

# MD CODE: Input Coordinate Files

The MD code’s initial system can be specified in numerous ways:

1. <string>

Name of file containing initial coordinates. Should be a .xyz or .xyzf file where the comment line specifies box dimensions, and each coordinate line optionally has the has the x, y, and z velocity or force components listed. The CRDFILE can also be a restart file. See documentation for VELINIT.

1. CAT <int> <string #1> … <string #int>

Concatenate <int> files (<string #1> … <string #int>) along the z-axis. Assumes an orthorhombic box and that all files have the same x- and y-dimensions. Expects files in a .xyz or .xyzf file format.

1. SCALE <double> <string>

Scale coordinates of input file <string> by a factor of <double>. Expects files in a .xyz or .xyzf file format.

1. INITIALIZE <string> <string> <double> <int>

Construct coordinates for a system based on an input structure. First string specifies if input structure is for a molecule or an atom. Accepted values are “MOLECULAR” or “ATOMIC.” Second string is either the name of the initial coordinate file (for option MOLECULAR), or the atom type for the system being constructed (for option ATOMIC). The double is the desired box length, in Angstroms; a cubic box is assumed. The final int value is the number of molecules/atoms to pack into the box. Systems are initialized on a cubic lattice. Expects files in a .xyz format (i.e. no velocities or forces)

# MD CODE: Parameter File

Note: This file is automatically generated by the LSQ C++ and python scripts. Parameter files can be generated by hand as well, but formatting is very important. If building by hand, it is easiest to start from a reference file, like those found in the test suites.

**Features:**

* In general, vertical spacing should NOT be modified
* In general, horizontal spacing should NOT be modified when in line with heading text
* In general, section ordering should NOT be modified. Exceptions are:
  + Lines with atom types info can be in any order
  + Lines with pair type info can be in any order
  + Note that subsequent ordering of related sections should be consistent, as these sections are used to determine type indices. For example, pair and triplet maps relate parameter type index to the pair/triplet type
* The heading section, prepended by exclamation points can contain whatever text the user wants. It is ignored by the MD code.
* Ordering of the Boolean variables in the next section is very important, as is the formatting/casing
* The PAIRTYPE line uses the same conventions as in the LSQ CODE: Main Control Variables section
* Charges are given in e, atom masses in amu

**Special Controls:**

A penalty function is added to the molecular mechanics force field to manage cases where a distance may fall below the rmin cutoff, and has the functional form: Penalty = penalty\_scaling(smin + penalty\_dist – rlen)^3, where parameters penalty scaling and penalty\_dist can be user-specified by adding the following lines above the “ATOM PAIR TRIPLETS” line, for example:

!# PAIRIDX # # ATM\_TY1 # # ATM\_TY1 # # S\_MINIM # # S\_MAXIM # . . .

0 C C 1 4.8 . . .

PAIR CHEBYSHEV PENALTY DIST: 0.05

PAIR CHEBYSHEV PENALTY SCALING: 2E8

PAIR CHEBYSHEV CUBIC SCALING: 1.0

ATOM PAIR TRIPLETS: 0

The “PAIR CHEBYSHEV CUBIC SCALING:” also controls a penalty function (“fpenalty” in the code), which helps to discourage small pair distances. See the LSQ C++ section for more details.

**Excluding 3-body Interactions – ANCILLARY SUPPORT**

Interactions corresponding to specific atom triplets can be excluded from the simulation. If the parameter file was generated by the house\_lsq code, and triplet exclusion was requested during the fitting process, nothing more needs to be done to turn off specific 3-body interactions. Otherwise, the number of triplet types needs to be updated to reflect the smaller number of triplets, triplet parameters for the excluded types need to be removed, and map keys for all variants of the excluded type need to be set to -2.

Note: If triplet types are excluded, it is important that they are not included in any special 3-body inner or outer cutoff lists.

# PES Scan generator – ANCILLARY SUPPORT

This utility, which is built into the house\_md generates potential energy surface scans of Chebyshev-type force fields, based on an input parameter file. Below are a few examples of usage. As a word of caution, S\_MINIM, S\_MAXIM, and S\_DELTA are used to define the range and spacing between scan data points.

Example 1: Scan of only 2-body pair interactions (3-body energies not included in reported energy)

1. # PLOTPES #
2. true 3 params.txt-scan-2b
3. PAIRTYPE PARAMS: 0
4. PAIRTYPE PARAMS: 1
5. PAIRTYPE PARAMS: 2

Line 2 tells the program that 3 scans will be performed, and that parameters should be read from the provided parameter file. Lines 3,4, and 5 are taken **directly** from the parameter file, and specify the specific interaction types scan should be run for. The scan range and fineness is specified by S\_MINIM, S\_MAXIM, and S\_DELTA in the parameter file.

Notes:

* The cubic smoothing function (fcut) can be excluded from 2-body scans by adding “EXCLUDE FCUT” at the end of line 2, above.
* Charges can be excluded by adding “EXCLUDE CHARGES.”
* The penalty function can be excluded by adding “EXCLUDE PENALTY”
* Any combination of these three options can be used, and order does not matter. This is particularly useful when re-fitting potentials

Example 2: Scan of 3-body pair interactions

1. # PLOTPES #
2. true 2 params-2+3b-refit.txt
3. TRIPLETTYPE PARAMS: 2
4. TRIPLETTYPE PARAMS: 3

Notes:

* Yields heat map that scan ik/jk for a given ij slice, and is designed to be visualized with gnuplot “splot.”
* Currently, ij slices are hardcoded to start at ij’s rmin, and step by 0.25 Angstroms at a time.
* “EXCLUDE FCUT,” “EXCLUDE CHARGES,” and “EXCLUDE PENALTY” should not be used with 3-body scans
* 3-body scans automatically include constituent 2-body contributions, including penalty functions.

# Small Utilities – ANCILLARY SUPPORT

These utilities can be found in the contrib (and sub-) folder(s). Usage notes can generally be found at the top of each script. For the most part, these are personal analysis scripts and come with no guarantees/support.

|  |  |
| --- | --- |
| **Utility** | **Summary** |
| combine\_xyz\_force.py | Take a b.txt or forceout.txt type file, a .xyz file, and a file with the box lengths for each frame and create a .xyzf file |
| dftbgen\_to\_xyz.py | Convert a dftbgen .gen file to a .xyz file |
| lmp2xyz.py | Convert a lammps .lammpstrj file to .xyz. Expects a specific .lammpstrj format. |
| xyz2gen.pl | Convert a .xyz file to a .gen file. |
| vasp2xyzf.py | Convert a VASP OUTCAR file to a .xyzf file. Extraction of boxlengths is often buggy. |
| gen\_custom\_xyzf.sh | Re-parse a .xyzf file. Useful when .xyzf file contains energy/stress data, that isn’t desired for the fit. |
| subtract\_forces.sh | Subtract forces from force field specified by parameter file from a .xyzf file’s forces |
| break\_apart\_xyz.py | Breaks a .xyzf trajectory file into its constituent frames |
| post\_proc\_lsq2.py | Scrubs parameter files of interactions with coefficients = 0 |