**­­­­Limitations/To do:**

* Overbonding won’t work right for non-H2O and when multiple “to” overbonding atom types are requested
* Charge enforcement/conservation is hard coded and will only work for systems containing O and H.
* Units have not been annotated in all necessary locations within code/input files.
* MD code functionality when parameter files has USEPOVER true but FITPOVER false is not working. This needs to be fixed.
* MD code functionality when parameter files has USECOUL true but FITCOUL false is not working. This needs to be fixed.

**Compiling/running the code:**

* The lsq and md codes both share source files, and have several compilation options.
* Before compiling, navigate to the src directory
* 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 house\_xx.

|  |  |
| --- | --- |
| house\_xx | Compile the xx code. |
| clean\_xx | Remove .o files related to xx code. Remember, some .o files are shared. |
| realclean\_xx | Remove xx code executable. |
| build\_tests\_xx | Generate fresh “correct” files for test suite . |
| travis | Compile Travis trajectory analyzer. See the .pdf file in contrib/Travis for more information. |
| molanal | Compile the molanal trajectory analysis suite. See the readme.txt file in contrib/molanal for more information. |

* The lsq and md codes can be run with: /path/to/executable < input\_file.name

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

* Navigate to the test\_suite-xx folder, and open up the run\_test\_suite.sh and generate\_test\_suite.sh files
* Ensure the “PATH\_TO\_LSQ\_PY\_CODE” variable is set properly. The suite is intended to use lsq-new-md-fmt.py version.
* Run from test\_suite-xx with ./run\_test\_suite.sh or and generate\_test\_suite.sh

Note on test suite usage: If the older version of the LSQ python script is used, the test suite should indicate a technical pass, because the format of the params files will be different, though the resultant parameters should be consistent.

**Quick Navigation:**

**LSQ C++ Code**

* [Main Control Variables](#_LSQ_CODE:_Main)
* [Topology Variables](#_LSQ_CODE:_Topology)

**LSQ Python Codes**

* [Original script](#_LSQ_Python_Codes)
* [Original MKL script](#_LSQ_Python_Codes_1)
* [New single-threaded script](#_LSQ_Python_Codes_2)
* [New MKL script](#_LSQ_Python_Codes_3)

**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)

# LSQ CODE: Main Control Variables

|  |  |  |
| --- | --- | --- |
| Keyword | Allowed Values | Usage |
| # TRJFILE # | any string | Provides the name of the simulation trajectory file. Files use a .xyzf format, which is like the standard .xyz format, with two exceptions: (1) after the line containing the number of atoms, the x, y, and z box lengths are given, and (2) each coordinate line has x, y, and z forces on the corresponding atom appended. |
| # WRAPTRJ # | “true” or “false” | Defines whether coordinates should be wrapped (i.e. application of post-simulation periodic boundary conditions). |
| # NFRAMES # | Any integer > 0 | Number of frames in the .xyzf file |
| # NLAYERS # | Any integer > 0 | Number of supercells to create from simulation box (i.e. replicate images). |
| # 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” | Will constraints (conservation) be enforced on charges? Only used if # FITCOUL # is true. Only supported for H2O systems. |
| # 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+1.  If the # PAIRTYP # is **CHEBYSHEV**, two integers must be provided, where the first provides the order for 2-body interactions, and follows the same convention as for DFTBPOLY types, while the second integer provides the order for 3-body interactions, and a value of n indicates monomials ranging in order from 0 to n.  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 | User-defined partial atomic charge for each unique atom type. Values form a column below # ATMCHRG #. Only used if # FITCOUL # is false. |
| # 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. |

# LSQ Python Codes

|  |  |  |
| --- | --- | --- |
| Name | Usage | Notes |
| lsq.py | python lsq.py A.txt b.txt params.header  > params.txt | Produce output compatible with old version of MD code. |
| lsq-new-md-fmt.py | python lsq-new-md-fmt.py lsq.py A.txt b.txt params.header ff\_groups.map > params.txt | Produce output compatible with new version of MD code. |
| lsq-new-md-fmt-weighted.py | python lsq-new-md-fmt.py lsq.py A.txt b.txt params.header ff\_groups.map WEIGHTFILE my\_weightfile.dat > params.txt | Located in the contrib/hist\_forces directory. Allows weights to be specified for forces. The input weight file should have one line for each in b.txt that specifies the weight to be applied. |

# MD CODE: Main Input File

|  |  |  |
| --- | --- | --- |
| Keyword | Allowed Values | Usage |
| # RNDSEED # | Any integer > 0 | Seed for random number generator. |
| # TEMPERA # | Any float > 0 | Target simulation temperature, in Kelvin. |
| # 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. |
| # TIMESTP # | Any float > 0 | MD time step in femtoseconds (fs). |
| # N\_MDSTP # | Any integer > 0 | Number of MD steps. # TIMESTP # \* # N\_MDSTEP # = simulation length in fs. |
| # NLAYERS # | Any integer > 0 | Number of supercells to create from simulation box (i.e. replicate images). |
| # PRMFILE # | A file name | Name of the input parameter file. |
| # CRDFILE # | A file name | Name of file containing initial coordinates. Should be a .xyz file where the comment line specifies box dimensions, and each coordinate line has the x, y, and z velocity components listed. |
| # VELINIT # | “true” or “false” | Initialize velocities using Box Muller, or read from coordinate file? |
| # THERMOST # | “HOOVER <Hoover time>” or “VELSCALE <scaling freq>” | Thermostat using Nose-Hoover approach, or velocity scaling. |
| # PRESSUR # | “ANALYTICAL” or “NUMERICAL” | Compute pressures using analytical equations, or numerically? |
| # FRQDFTB # | Any integer > 0 | How frequently should the DFTB gen file be printed? |
| # FRQENER # | Any integer > 0 | How frequently should the energies, etc be generated? |
| # PRNTFRC # | “true” or “false” | Should the forces be printed at the end of the simulation? |

# 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 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 indicies. 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

# PES Scan generator

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 (2-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.

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

1. # PLOTPES #
2. true 4 params.txt-scan-3b
3. TRIPLETTYPE PARAMS: 0 SCAN 1
4. FIX IJ 3.0000 IK 3.0000 SCAN JK
5. TRIPLETTYPE PARAMS: 1 SCAN 2
6. FIX IJ 3.0000 IK 0.9572 SCAN JK
7. FIX IK 0.9572 JK 3.0000 SCAN IJ
8. TRIPLETTYPE PARAMS: 2 SCAN 2
9. FIX IJ 0.9572 IK 0.9572 SCAN JK
10. FIX IJ 0.9572 JK 1.5130 SCAN IK
11. TRIPLETTYPE PARAMS: 3 SCAN 1
12. FIX IJ 1.6238 IK 1.6238 SCAN JK

This example has similar syntax to the previous example, where now the first 3 “words” of lines 3, 5, 8, and 11 are taken directly from the parameter file. The following words are used to specify how many pairs within the 3 that define a 3-body interaction should be scanned. Lines 4, 6, 7, 9, 10, and 12 specify how those scans should be run. Scans for 3-body potentials are similar to an angle scan where bonds are held rigid. In other words, two distances are held fixed while a third is scanned. Words “IJ, IK, and JK” refer to a specific pair from the 3 atoms and correspond to the types given on lines containing “TRIPLETTYPE PARAMS:” in the parameter file.

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

1. # PLOTPES #
2. true 4 params.txt-scan-3b
3. PAIRTYPE PARAMS: 0
4. PAIRTYPE PARAMS: 1
5. PAIRTYPE PARAMS: 2
6. TRIPLETTYPE PARAMS: 0 SCAN 1
7. FIX IJ 3.0000 IK 3.0000 SCAN JK
8. TRIPLETTYPE PARAMS: 1 SCAN 2
9. FIX IJ 3.0000 IK 0.9572 SCAN JK
10. FIX IK 0.9572 JK 3.0000 SCAN IJ
11. TRIPLETTYPE PARAMS: 2 SCAN 2
12. FIX IJ 0.9572 IK 0.9572 SCAN JK
13. FIX IJ 0.9572 JK 1.5130 SCAN IK
14. TRIPLETTYPE PARAMS: 3 SCAN 1
15. FIX IJ 1.6238 IK 1.6238 SCAN JK

This example simply shows that Examples 1 and 2 can be achieved with a single input file

Example 4: Scan of only 3-body pair interactions, with 2-body energies included

1. # PLOTPES #
2. true 4 params.txt-scan-3b
3. TRIPLETTYPE PARAMS: 0 SCAN 1 INCLUDE 2B PAIRTYPE PARAMS IJ 0 IK 0 JK 0
4. FIX IJ 3.0000 IK 3.0000 SCAN JK
5. TRIPLETTYPE PARAMS: 1 SCAN 2 INCLUDE 2B PAIRTYPE PARAMS IJ 0 IK 2 JK 2
6. FIX IJ 3.0000 IK 0.9572 SCAN JK
7. FIX IK 0.9572 JK 3.0000 SCAN IJ
8. TRIPLETTYPE PARAMS: 2 SCAN 2 INCLUDE 2B PAIRTYPE PARAMS IJ 2 IK 2 JK 1
9. FIX IJ 0.9572 IK 0.9572 SCAN JK
10. FIX IJ 0.9572 JK 1.5130 SCAN IK
11. TRIPLETTYPE PARAMS: 3 SCAN 1 INCLUDE 2B PAIRTYPE PARAMS IJ 1 IK 1 JK 1
12. FIX IJ 1.6238 IK 1.6238 SCAN JK

This example has syntax similar to Example 2. Do not combine this type of calculation with the type given in Example 1. The extra words on lines 3, 5, 8, and 11 specify that 2-body contributions should be included in the reported energies, and specifies what 2-body type each of the three pairs should be, where type index is taken from the parameter file.