

PuReMD Manual

(Purdue Reactive Molecular Dynamics Program)

Hasan Metin Aktulga

June 6, 2010

This manual is for the two simulation programs which have come to existence as a result of our ReaxFF realization efforts. Our initial efforts have led to the SerialReax program, which is a sequential implementation for ReaxFF. SerialReax has helped us in verifying the accuracy of our implementation in C against the original ReaxFF code which was developed in Fortran, such a task would be cumbersome in a parallel context. It has also served as a testbed for quickly implementing new algorithms and numerical techniques and benchmarking their effectiveness before we incorporated them into the parallel version.

PuReMD (Purdue Reactive Molecular Dynamics) program is based on the sequential implementation, SerialReax, and highly efficient and scalable parallelization techniques. It inherits the excellent performance and small memory foot-print features of SerialReax and extends these desirable capabilities to systems of sizes that are of interest to computational scientists.

For reasons described above, setting up a simulation and running it using PuReMD or SerialReax is quite similar to each other. Therefore in this manual, we take PuReMD as our basis and describe it first. In a following section, we describe the extras that come with SerialReax which we hope to incorporate into PuReMD in the near future.

1 Input Files

PuReMD expects 3 input files: a geometry file describing the system to be simulated, a force field file containing ReaxFF parameters and a control file to manage simulation variables.

1.1 Geometry File

Geometry file tells about the types and initial positions of the atoms in the system. PuReMD supports geometry files in two formats: the pdb format and our custom input format. It is also possible to restart from an earlier simulation check-point using a restart file (which can be either in ascii or binary format as explained below).

1.1.1 pdb format

For detailed and up-to-date information on the pdb format, please visit <http://www.wwpdb.org/docs.html>. Input files of various other formats can easily be converted to the pdb format using the freely available OpenBabel software: (http://openbabel.sourceforge.net/wiki/Main_Page).

In the geometry file, each atom is assigned a unique serial id to be able to identify atoms easily during the simulation. pdb format limits the number of digits in the atom serial field to only 5 digits, therefore the maximum number of atoms that can be input using the pdb format is only 100000, way behind what is generally used for parallel MD simulations.

1.1.2 custom format

PuReMD features a very simple custom geometry format to alleviate the maximum number of atoms limitation of the pdb format and to ease the task of preparing a geometry file. The general layout of our custom geo format is as follows: The first line describes the simulation box and the second line gives the total number of atoms in the system. These initial two lines need to be followed by a single line for each atom describing it in detail. Here is what a custom geo file looks like:

```
BOXGEO x_len y_len z_len alpha beta gamma
N
1 ele1 name1 x1 y1 z1
2 ele2 name2 x2 y2 z2
.
.
.
N eleN nameN xN yN zN
```

First three floating point numbers on the first line give the length of the simulation box in x, y, z dimensions, the remaining ones are for the angles between each box dimension. Currently, PuReMD works only with an orthogonal box meaning all angles need to be 90.0 degrees.

There is no limit by the format on the number of atoms that can be input, so N shown on the second line can be as large as allowed by the memory limitations. Each atom line starting from line 3 and until line N+2 consists of 6 fields:

- an integer denoting the atom's serial id
- a string for the chemical symbol of the element (2 characters max, case insensitive)
- a string for the atom name (7 characters max)
- 3 floating point numbers describing the position in cartesian coordinates

1.2 Force Field File

Force field file contains the ReaxFF parameters to be used during the simulation. Adri van Duin is the main developer and distributor for Reax force fields, you can see his contact info at <http://www.mne.psu.edu/vanduin/>.

1.3 Control File

Parameters in the control file allow the user to tune various simulation options. Parameter names are case-sensitive but their order is not important (except that `ensemble_type` needs to precede both `p_mass` and `pressure`). Described below are the fields that you might use in a control file. If a parameter is missing from the control file, its default value (as given in each parameter's description below) will be assumed. Each parameter must be specified in a single line, first token should be the parameter and the second token should be an appropriate value. Comments regarding a parameter can be included after the value field on the same line.

```
simulation_name    test_puremd
```

Output files produced by PuReMD will be in `simulation_name.some_extension` format. Output files will be discussed in more detail in Section 4. Default value is `simulate`.

```
ensemble_type     1
```

`ensemble_type` denotes the type of the ensemble to be produced by PuReMD. Supported ensembles are as follows:

- 0: NVE
- 1: bNVT - NVT with Berendsen thermostat
- 2: nhNVT - NVT with Nose-Hoover thermostat (under testing)
- 3: sNPT - semiisotropic NPT with Berendsen's coupling
- 4: iNPT - isotropic NPT with Berendsen's coupling
- 5: NPT - anisotropic NPT with Parrinello-Rehman coupling (under development)

`ensemble_type` is NVE by default.

```
nsteps           1000
dt                0.25
```

`nsteps` controls the total number of steps for the simulation and `dt` controls the length of each time step (measured in femtoseconds). Number of steps is 0 by default and timestep length is 0.25 fs.

```
proc_by_dim      1 1 3
```

PuReMD uses the domain decomposition technique to distribute the load among processors, it currently does not have dynamic load balancing. `proc_by_dim` denotes the desired decomposition of the simulation box into subdomains (first integer is the number of equal-length partitions in x dimension, second integer is for y dimension and the last one is for z dimension). Each subdomain is subsequently assigned to a single processor. PuReMD constructs a 3D torus based on the `proc_by_dim` parameter. The default is to use a single processor. SerialReax does not accept the `proc_by_dim` parameter.

```
geo_format      0
```

`geo_format` parameter informs PuReMD about the format of the geometry file to be read. Options are:

- 0: custom format
- 1: pdb format
- 2: ASCII restart file
- 3: binary restart file

pdb and custom formats were already discussed in Section 1.1. Another option is to resume from an older run by setting `geo_format` to 2 (for ASCII restarts) or 3 (for binary restarts) and providing the name of the restart file as an argument to PuReMD (instead of the geo file name). Then PuReMD will read the box geometry, positions and velocities for all atoms in the system from the restart file and continue execution from thereon. Default is the custom geometry format.

```
restart_format  1
restart_freq    0
```

PuReMD can output restart files in an ASCII format (when `restart_format` = 0) or in a binary format (when `restart_format` = 1). While ASCII restarts are good for portability, binary restart files are much more compact and does not cause any loss of information due to truncation of floating point numbers. Binary restart is the default.

There will not be any restart files output unless `restart_freq` parameter is set to a positive integer. A restart file is named as follows: `simulation_name.resS` where S denotes the step that the restart file is written.

```
tabulate_long_range 10000
```

When set to m (must be a positive integer), `tabulate_long_range` option turns on the tabulation optimization for computing electrostatics and van der Waals interactions. The range $[0, \text{cutoff}]$ is sampled at m equally spaced points; energy and forces due to long range interactions between each atom type in the system

are computed at each of these sample points and are stored in a table. Then for each interval, coefficients of a fitted cubic spline interpolation function are computed. During the simulation while computing the long range interactions between any two atoms, the appropriate interpolation function is located and energy and forces between the atom pair is approximated by means of cubic spline interpolation. This method gives significant speed-up compared to computing everything from scratch each time and with only 10000 sample points it is able to provide accuracies at the machine precision level. Default is no tabulation.

```
energy_update_freq    10
```

This option controls the frequency of writes into output files described in detail in Section 4 (except for the trajectory and restart files which are controlled by other parameters explained separately). The default value for this parameter is 0, meaning there will not be any energies and performance logs output.

```
remove_CoM_vel       500
```

Removal of translational and rotational velocities around the center of mass needs to be done for NVT and NPT type ensembles to remove the unphysical effects of scaling velocities. In case of NVE, this is unnecessary and is not done regardless of the value of `remove_CoM_vel`. The default is to remove translational and rotational velocities at every 250 steps.

```
nrhood_cutoff       5.0  
thb_cutoff           0.001  
hbond_cutoff        7.50
```

These cutoff parameters are crucial for the correctness and efficiency of PuReMD. Normally, bonded interactions are truncated after 4-5 Å in ReaxFF and this is controlled by the `nrhood_cutoff` parameter whose default value is 4 Å.

`thb_cutoff` sets the bond strength threshold for valence angle interactions. Bonds which are weaker than `thb_cutoff` will not be included in valence angle interactions. Default for `thb_cutoff` is 0.001.

`hbond_cutoff` controls the distance between the donor and acceptor atoms in a hydrogen bond interaction. Its typical value is from 6 Å to 7.5 Å. If `hbond_cutoff` is set to 0, hydrogen bond interactions will be turned off completely (could be useful for improved performance in simulations where it is a priori known that there are no hydrogen bonding interactions). Default is to set `hbond_cutoff` to 0.

```
reneighbor          10  
vlist_buffer         2
```

PuReMD features delayed neighbor generation by using Verlet lists. `reneighbor` controls the reneighboring frequency and `vlist_buffer` controls the buffer space beyond the maximum ReaxFF interaction cutoff. By default, `vlist_buffer` is set to 0 and reneighboring is done at every step.

```

q_err      1e-6
qeq_freq   1

```

PuReMD uses a preconditioned conjugate gradients (PCG) solver with a diagonal preconditioner for the QEq problem. `q_err` denotes the stopping criteria for the PCG solver, the norm of the relative residual. A lower threshold would yield more accurate equilibration of charges at the expense of an increase in computation time. A threshold of 10^{-6} should be good enough for most cases and this is the default value.

`qeq_freq` can be used to perform charge equilibration at every few steps instead of the default behaviour of performing it at every step. Although doing QEq less frequently would save important computational time, it is not recommended. Because this might cause wild fluctuations in energies and forces.

```

temp_init   0.0
temp_final  300.0
t_mass      0.1666

```

Temperature coupling parameters (`temp_final` and `t_mass`) are effective in all types of ensembles except for NVE. Initial temperature is controlled via the `temp_init` parameter including the NVE ensemble. 0 K is the default value for `temp_init` and 300 K is the default value for `temp_final`. PuReMD features both Berendsen [2] and Nose-Hoover [1] type thermostats as was mentioned while explaining the `ensemble_type` parameter. *Important note: Nose-Hoover thermostat in PuReMD is still under testing.*

`t_mass` is the thermal inertia given in femtoseconds. Suggested (and the default) value of `t_mass` is 500.0, and 0.166 for the Berendsen thermostat, and for the Nose-Hoover thermostat, respectively.

```

pressure    0.000101 0.000101 0.000101
p_mass      5000.0   5000.0   5000.0

```

Pressure coupling parameters are needed only when working with NPT-type ensembles. Currently iNPT (isotropic NPT) and sNPT (semi-isotropic NPT) are the available pressure coupling ensembles in PuReMD. Berendsen thermostats and barostats are used in both cases [2]. `pressure` is the desired pressure of the system in GPa and `p_mass` is the virial inertia in femtoseconds. Suggested (and the default) value of `p_mass` is 5000.0 together with a `t_mass` of 500.0 as NPT methods use Berendsen-type thermostats only.

For the iNPT ensemble, `pressure` parameter expects a single floating number (in case there are more, they will simply be ignored) to control pressure. For the sNPT ensemble, `pressure` parameter expects 3 floating point numbers to control pressure on each dimension. Same things apply for `p_mass` as well.

```

write_freq   100
traj_method  1

```

Trajectory of the simulation will be output to the trajectory file (which will automatically be named as `simulation_name.trj`) at every `write_freq` steps. For making analysis easier, the trajectory file is written as an ASCII file. By default, no trajectory file is written.

PuReMD can output trajectories either using simple MPI send/receives (option 0 which is the default) or using MPI I/O calls (option 1) which are part of the MPI-2 standard. The latter option is supposed to be more efficient (not verified by tests though) but may not be available in some MPI implementations. `traj_method` option is not applicable to SerialReax simulations.

```
traj_title      TEST
atom_info       1
atom_forces     1
atom_velocities 1
bond_info       0
angle_info      0
```

Currently PuReMD only outputs trajectories in its custom trajectory format. This custom format starts with a trajectory header detailing the trajectory title and the values of control parameters used for the simulation. A brief description of atoms follow the trajectory header with atom serial ids and what element each atom is.

Then at each `write_freq` steps (including step 0), a trajectory frame is appended to the trajectory file. The frame header which gives information about various potential energies, temperature, pressure and box geometry is standard. However, the latter parts of the frame can be customized using `atom_info`, `atom_forces`, `atom_velocities`, `bond_info` and `angle_info` parameters which are already self-explanatory. The ordering is atoms section, bonds section and angles section assuming that they are all present. By default, all atom, bond and angle information outputting is turned off.

One nice property of the custom trajectory format is that each part of the trajectory is prepended by a number that can be used to skip that part. For example, the trajectory header is prepended by an integer giving the number of characters to skip the control parameters section. The initial atom descriptions is prepended by the number of characters to skip the initial descriptions part and another one that tells the number of atom description lines. Similar numbers are found at the start of each section within a trajectory frame as well, making it easy to skip parts which are not of interest to a particular trajectory analysis procedure. So the general layout of our custom trajectory format is as follows (assuming all trajectory options are turned on):

```
CHARS_TO_SKIP_SECTION
trajectory header
CHARS_TO_SKIP_ATOM_DESCS NUM_LINES
atom descriptions
CHARS_TO_SKIP_FRAME_HEADER
frame1 header
```

```

CHARS_TO_SKIP_ATOM_LINES NUM_ATOM_LINES
frame1 atom info
CHARS_TO_SKIP_BOND_LINES NUM_BOND_LINES
frame1 bond info
CHARS_TO_SKIP_ANGLE_LINES NUM_ANGLE_LINES
frame1 angle info
.
.
.
CHARS_TO_SKIP_FRAME_HEADER
frameN header
CHARS_TO_SKIP_ATOM_LINES NUM_ATOM_LINES
frameN atom info
CHARS_TO_SKIP_BOND_LINES NUM_BOND_LINES
frameN bond info
CHARS_TO_SKIP_ANGLE_LINES NUM_ANGLE_LINES
frameN angle info

```

2 SerialReax Extras

In this section, we explain the parameters found in SerialReax but not in PuReMD. Our work towards adding the same functionalities into PuReMD is underway.

In addition to the PCG solver, SerialReax features a preconditioned GMRES (PGMRES) solver and an incomplete LU factorization (ILU) based preconditioning scheme. An ILU factorization essentially does the same thing as an LU factorization but small terms in the matrix are dropped to expedite the factorization and to prevent a huge number of fill-ins in the factor matrices. Following are the extra control parameters found in SerialReax regarding the QEq solver:

```

ilu_refactor      100
ilu_droptol      0.01

```

`ilu_droptol` sets the threshold for dropping small terms in the resulting ILU factors. Suggested (and the default) value for `ilu_droptol` is 10^{-2} . Despite the drop rules, ILU factorization is still a costly operation. So a user can choose to perform it at every `ilu_refactor` steps. The fact that atoms move very slowly in an MD simulation allows the use of same ILU factors as preconditioners in the subsequent steps with little performance loss. For liquids, this frequency can be on the order of 100-200 steps, for solids it can go up to thousands of steps depending on how fast atoms are moving. The default for `ilu_refactor` is 100.

```

t_mode           0
t_rate           -100.0
t_freq           2.0

```

These options are specifically for being able to change the temperature of the system during a simulation. `t_mode` of 1 gives a step-wise control over temperature, *i.e.* the system maintains its temperature for a simulation time of `t_freq` picoseconds. After that, the target temperature is increased/decreased by `t_rate` K and the thermostat lets the system converge to its new target temperature.

On the other hand, `t_mode` of 2 increases/decreases the target temperature at each time-step by an amount which corresponds to `t_rate / t_freq` K/ps. The overall effect of such a regime is a constant slope (instead of the step pattern with a `t_mode` of 1) in the target temperature–simulation time graph.

```
molec_anal      1
freq_molec_anal 1
bond_graph_cutoff 0.3
ignore          2 0 3
```

Since ReaxFF is a reactive force field, during the simulation molecules present in the system will change. These changes can be traced by turning the `molec_anal` option on by setting it to a non-zero integer. Molecules are determined based on the `bond_graph_cutoff` parameter: bond orders less than this threshold are not counted as a physical bond and do not contribute to molecular structures, all others do. `ignore` allows one to ignore the bondings of specific atom types. The first number after `ignore` denotes how many atom types will be listed and the following numbers (which correspond to the order of elements in the force field file) denote the atom types to be ignored in molecular analysis.

3 Compilation and Execution

PuReMD is distributed in the `tar.gz` compression format which can be extracted under a Unix system with the following command:

```
gtar xvzf PuReMD.tar.gz
```

As results a new directory, named `PuReMD`, will appear in the working directory. It contains the source code directory (`src`) along with a directory for sample systems (`examples`).

PuReMD can be compiled by switching to the `src` directory and running `make`. The executable, `puremd`, will be created inside the source directory. The Makefile that comes in the distribution assumes OpenMPI as the default MPI implementation and `mpicc` as the default MPI compiler. In case you have a different MPI implementation, please set your MPI compiler in the Makefile appropriately.

PuReMD requires 3 input files as mentioned in section 1. For example, the command to run `puremd` with OpenMPI is as follows:

```
mpirun -np #p -machinefile m.txt puremd geo ffield control
```

SerialReax comes in a similar distribution format and Makefile, so instructions for compiling and running PuReMD is applicable for SerialReax as well.

4 Output

PuReMD writes its output files into the directory where it is run. There are a number of output files all of which have the `simulation_name` as the first part of their names followed by its unique extension:

- `.out` contains a summary of the simulation progress. Its format is:

```
Step  Total Energy  Potential  Kinetic
T (in K)  Volume(in A^3)  P(in GP)
```

- `.pot` contains detailed information regarding various types of energies that comprise the total potential energy:

```
Step  Bonds  OverCoor+UnderCoor  LonePair
Angle+Penalty  3-body Coalition  Hydrogen Bonds
Torsion  4-body Conjugation
vander Waals  Coulomb  Polarization
```

- `.log` is intended for performance tracking purposes. It displays the total time per step and what parts of code take up how much time to compute.
- `.prs` is output only when pressure coupling is on. It displays detailed information regarding the pressure and box dimensions as simulation progresses.
- `.trj` is the trajectory file. Atom positions are written into this file at every `write_freq` steps using the desired format as explained before. Each frame is concatenated one below the other.

Apart from these, there might be some text printed to `stderr` for debugging purposes. If you encounter some problems with the code (like a segmentation fault or unexpected termination of the code), please contact `haktulga@cs.purdue.edu` with the error message printed to `stderr` and your input files.

In addition to the output files above, SerialReax can output another file (with extension `.mol`) which contains the fragmentation analysis output.

References

- [1] Glenn J. Martyna, Douglas J. Tobias, and Michael L. Klein. “Constant pressure molecular dynamics algorithms.” *The Journal of Chemical Physics* 101, 4177 (1994).
- [2] H. J. C. Berendsen, J. P. M. Postma, W. F. van Gunsteren, A. DiNola, and J. R. Haak. “Molecular dynamics with coupling to an external bath.” *The Journal of Chemical Physics* 81, 3684-3690 (1984).