RXMD User Guide v 1.0.1

September 16, 2017

rxmd : Linear Scalable Parallel ReaxFF Molecular Dynamics Simulator

rxmd has been developed to simulate large-scale Reactive Force Field molecular dynamics (MD) simulations on from commodity laptops to high-end supercomputing platforms. **rxmd** has been used in a various class of material studies, such as shock-induced chemical reactions, stress corrosion cracking, underwater bubble collapse, fracture of self-healing ceramics and oxidation of nanoparticles.

0. Prerequisites

rxmd is designed to be simple, portable and minimally dependent on 3rd party library. You will need 1) a Fortran compiler that supports OpenMP, and 2) MPI (Message Passing Interface) library for parallel and distributed simulation. Modern Fortran compilers natively support OpenMP, and you can find many freely available MPI libraries online. Please refer to MPI library developer website about how to install their library.

rxmd has been tested on following environments.

- Fortan Compiler:

```
GNU Fortran (GCC) 6.1.0
Intel Fortran (IFORT) 17.0.4
IBM XL Fortran V14.1
```

- MPI library:

```
OpenMPI 1.8.8
MPICH2
MVAPICH2
Cray Mpich 7.6.0
```

1. Getting Started

To get started, clone this repository to your computer.

```
~$ git clone https://github.com/USCCACS/rxmd.git
```

2. How to build RXMD

2.1 Working Directory

Frist, change working directory to **rxmd/**

```
~$ cd rxmd
```

you will see following files and directories.

```
rxmd $ ls
DAT/          conf/         ffield        regtests/     src/          util/
Makefile.inc  doc/          init/         rxmd.in       unittests/
```

Here, two directories, **src/** and **init/**, are especially important for you. **src/** contains all rxmd source codes and **init/** has a program and input files to generate an initial configurations for simulation.

2.2 Configure Makefiles

There are two Makefile files **Makefile.inc** and **init/Makefile** that you might need to modify according to your computing environment.

- **Makefile.inc** defines which compiler you like to use to build the **rxmd** executable. We have several predefined compiler settings in **Makefile.inc**. Please enable the macro **FC** for the Fortran compiler and compiler flags you want to use. Also do not forget disable macros you don't want to use.
- **init/Makefile** is used to build software to generate intial configuration, called **geninit**. Any Fortran or MPI compiler that supports the stream I/O can be used here.

Example 1) Linux Computer with Intel Compiler Many HPC centers have Intel Fortran compiler and its MPI binding installed. If this is the case, enable following lines in **Makefile.inc** and **init/Makefile**. - **Makefile.inc**

```
# Intel Compiler
FC = mpif90 -O3
```

- **init/Makefile**

```
FC = ifort
```

Example 2) BlueGene/Q

IBM provides Fortran XL Compiler and MPI library for BlueGene Series. Please enable following lines in **Makefile.inc** and **init/Makefile**.

- **Makefile.inc**

```
# xl fortran
FC = mpif90 -O3 -qhot
```

- **init/Makefile**

```
FC = xlf
```

2.3 Prepare Initial Geometry

Next step is to generate initial MD geometry. Type the make command shown below.

```
rxmd $ make -C init/
```

This compiles the standalone application **geninit**, read a geometry file (init.xyz by default) in **init/** directory, replicate the geometry and save the entire initial MD geometry into **rxff.bin** file, and then place **rxff.bin** file in **DAT/** directory.

2.4 Build RXMD

Type the command below to build the **rxmd** executable.

```
rxmd $ make -C src/
```

Check to see if you the **rxmd** executable and the initial geomerty input **DAT/rxff.bin** in place, then you are ready to start a simulation.

```
rxmd $ ls
DAT/          conf/         ffield        regtests/     rxmd.in       unittests/
Makefile.inc  doc/          init/         rxmd*         src/          util/
```

```
rxmd $ ls DAT/
rxff.bin
```

3. How to run

Default input parameters are set to run a single process job. Type

```
rxmd $ ./rxmd
```

How to run a multi process job depends on which MPI library you use, but most likely **mpirun** just works for you.

```
rxmd $ mpirun -np nprocessors ./rxmd
```

If you see following outputs, congratulations! You have everything working.

```
rxmd $ ./rxmd
              rxmd has started
----------------------------------------------------------------
         req/alloc # of procs:        1  /        1
         req proc arrengement:        1        1        1
                parameter set:Reactive MD-force field: nitramines
(RDX/HMX/TATB/PETN)
                time step[fs]:    2.50E-01
 MDMODE CURRENTSTEP NTIMESTPE:  1         0       100
  isQEq,QEq_tol,NMAXQEq,qstep:     1   1.0E-07   500     1
                Lex_fqs,Lex_k:   1.000   2.000
            treq,vsfact,sstep:     300.000   1.000      100
                  fstep,pstep:   100    10
               NATOMS GNATOMS:                   168                     168
                         LBOX:       1.000       1.000       1.000
                  Hmatrix [A]:         13.180          0.000          0.000
                  Hmatrix [A]:          0.000         11.570          0.000
                  Hmatrix [A]:          0.000          0.000         10.710
               lata,latb,latc:      13.180      11.570      10.710
          lalpha,lbeta,lgamma:      90.000      90.000      90.000
               density [g/cc]:    1.8061
         # of linkedlist cell:     4     3     3
            maxrc, lcsize [A]:     3.160        3.29      3.86      3.57
    # of linkedlist cell (NB):     4     3     3
              lcsize [A] (NB):      3.29      3.86      3.57
     MAXNEIGHBS, MAXNEIGHBS10:    30   700
            NMINCELL, NBUFFER:     3    30000
    FFPath, DataDir, ParmPath:      ffield          DAT      rxmd.in
          # of atoms per type:          24 - 1          48 - 2          48 - 3
48 - 4
----------------------------------------------------------------
nstep  TE  PE  KE: 1-Ebond 2-(Elnpr,Eover,Eunder) 3-(Eval,Epen,Ecoa) 4-(Etors,Econj)
5-Ehbond 6-(Evdw,EClmb,Echarge)
        0 -9.82464E+01 -9.82464E+01  0.00000E+00 -1.369E+02  1.287E+00 -1.362E+00
5.208E-01 -1.398E-03  3.821E+01     0.00    0.00    0.00  41    0.36    0.23
       10 -9.82465E+01 -9.82467E+01  2.32025E-04 -1.369E+02  1.290E+00 -1.364E+00
5.214E-01 -1.397E-03  3.821E+01     0.08    0.00   -0.00  32    0.36    0.27
       20 -9.82466E+01 -9.82471E+01  4.80178E-04 -1.369E+02  1.287E+00 -1.366E+00
5.202E-01 -1.408E-03  3.821E+01     0.16    0.00   -0.00   4    0.36    0.25

...


       total (sec):       2.9980         2.9980
----------------------------------------------
    rxmd successfully finished
```

To learn more about **rxmd**, please refer to RXMD Manual.

4. License

This project is licensed under the GPU 3.0 license - see the LICENSE.md file for details

5. Publications

- Mechanochemistry of shock-induced nanobubble collapse near silica in water K. Nomura, R. K. Kalia, A. Nakano, and P. Vashishta, Applied Physics Letters 101, 073108: 1-4 (2012)
- Structure and dynamics of shock-induced nanobubble collapse in water M. Vedadi, A. Choubey, K. Nomura, R. K. Kalia, A. Nakano, P. Vashishta, and A. C. T. van Duin, Physical Review Letters 105, 014503: 1-4 (2010)
- Embrittlement of metal by solute segregation-induced amorphization H. Chen,R. K. Kalia, E. Kaxiras, G. Lu, A. Nakano, K. Nomura, A. C. T. van Duin, P. Vashishta, and Z. Yuan, Physical Review Letters 104, 155502: 1-4 (2010)
- Metascalable molecular dynamics simulation of nano-mechano-chemistry F. Shimojo, R. K. Kalia, A. Nakano, K. Nomura, and P. Vashishta, Journal of Physics: Condensed Matter 20, 294204: 1-9 (2008)
- A scalable parallel algorithm for large-scale reactive force-field molecular dynamics simulations K. Nomura, R. K. Kalia, A. Nakano, and P. Vashishta, Computer Physics Communications 178, 73-87 (2008)

1. Initial System Preparation

1. Obtain the unit cell or the prepared system coordinates in xyz file format.
   For example: If properties of a system such as RDX, FOX-7, TATB, CaDPA etc. is to be determined, obtain the unit cell for the desired system from literature. Or, if the simulation is to be performed for some other systems containing voids, vacuums etc. prepare the system accordingly and verify the coordinates in xyz format using some visualization software like VMD, Ovito, VESTA etc. to check if the system looks right or not.
2. After verification of the system by visualizing the coordinates in xyz file format with some molecular visualization software as mentioned above, the system is read by geninit.f90 code. This code will prepare initial binary files which will be read by rxff code to perform the simulation.

Parameters and format required by geninit.f90 file:

2.1.Specific format read by the code

```
As mentioned above the coordinates of the system under consideration should be in
xyz file format and they should be unscaled units. The format of xyz file read by
geninit.f90 file requires some extra specifications as compared to a regular xyz
file.
The format is as follows:
Line 1: Natoms (number of atoms)     tag (some string to describe your system)
Line 2: System size (lata   latb    latc    alpha   beta    gamma)
Line 3 - Line Natoms + 2: Atomic coordinates (atype x   y   z)
where atype is a string for describing the atom type such as C, H, O, N, S, Mo, Ni,
Al and x, y, z specifies the Cartesian coordinates.
```

2.2.Parameters to be specified for geninit.f90 file

```
1.Number of Processors: The number of processors required in x, y and z format
should be specified as arguments to vprocs array as:

vprocs(3) = (/2,2,2/) if the simulation requires a total of 8 processors such that
2 processors are required in each of x, y and z directions.

Important: To estimate the number of processor for correct computation please
ensure that you divide the system length in x, y and z (in angstrom (A)) direction
by 12 A and use the quotient as the arguments to vprocs(3).

Note: If the number of processor  in direction  comes out to be odd then number of
processors in direction  should be equal to . This is important for MPI
communications performed by the code.

For Example:
1.If your system is  then the total number of processors in x, y and z direction
should be .

2.If your system is  then the total number of processors in x, y and z direction
should be . Since 3 and 5 processors cannot be used in x and y direction as they
will violate the MPI communications performed in the code and the computation will
experience a deadlock due to communication error.
```

```
```

2.3.Multiplicity in x, y and z directions

```
If the simulation needs to be performed on a scaled system then the integral
multiples in x, y and z directions should be specified as arguments to array mc(3).

For example:
1.If the system is  13x13x17 Aand you specify mc(3) = (/2,3,2/) then the final
system size will be  and hence the number of processors      should be specified
accordingly as mentioned above in 2.2.
```

These steps will ensure that the binaries created are correct and hence will ensure appropriate simulation results, if the simulation is performed properly

2. Input file description (rxmd.in)

The input file rxmd.in has following format:

Here the keywords mean:

```
  dt  :Increment in time during a single molecular dynamics step (usually in fs).

  time_step: Total number of time step the simulation needs to be run. So the total
time of simulation will be (time_step*dt) fs.

  treq: The temperature at which simulation is required to be done.

  vsfact: The factor by which velocity is to be scaled. This factor will ensure
quenching or heating of the system, as a value above 1 will ensure heating while a
value below 1 will ensure cooling of the system.

  sstep: This parameter dictates the rate of heating or cooling of the system. A
higher
         value will mean gradual heating or cooling while a lower value will mean
rapid heating or cooling.

  fstep: This parameter controls when the trajectory in various output file formats
will be saved to a file.

  pstep:  This parameter controls the print out information to the standard output
or to a file related to various properties such as energy, temperature pressure etc.

  vprocs: This parameter should be same as specified for geninit.f90 file so that
```

```
the number of processors remain consistent.

  Mode : mode for doing the simulation. The Rxff code provides following mode the
perform simulations.
       1. Mode 0: This mode provide velocity to the atoms distributed according to
a Gaussian distribution samples using box-muller algorithm and scaled to be at
temperature = treq. Also, it initializes the charges on each atom to estimate
electro static forces experienced on each atom.
       2. Mode 4: Scale velocities of the atoms every sstep by a factor of vsfact.
       3. Mode 5: Scale velocities of the atoms according to the total kinetic
energy of the system.
       4. Mode 6: This mode provide velocity to the atoms distributed according to
a Gaussian distribution samples using box muller algorithm and scaled to be at
temperature = treq every sstep.
       5. Mode 7: This mode scales the velocities of the atoms based on their
individual kinetic energies every sstep to obtain an NVE ensemble.

  isQEq: This parameter is true or false specifying whether the charge
equilibration scheme is required or not.

  QEq_tol: The required tolerance for electro-static term relative to its previous
value

  NMAXQEq: The maximum number of iterations to perform for charge equilibration
step if electro-static terms do not approach to a value less than QEq_tol between
any   two consecutive iterations.

  qstep: This parameter defines the frequency at which charge equilibration is to
be done during the simulation.

  isBinary: The parameter can be given a true value or a false value and defines
              whether a binary file will be saved or not.

  isBondfile: This parameter can be given a true or a false value and defines
whether   information related to the bond list of every atom will be saved to a
file or not.

  isPDB: This parameter can be given a true or a false value and defines whether
information related to atomic trajectories is saved to the file in PDB format or
not.
```

3.Running a parallel job:

1.The first step is to create binaries corresponding to your system which can be done by following the steps in 1 in init folder.
2.After creating binaries ensure that you are at the root directory such that init directory is one level below you. Then create a folder called DAT and copy all the binaries of the form of rxff* from init to DAT by doing `cp init/rxff* DAT/`.
3.Then run mkdir.sh in the unit folder as sh mkdir.sh from the root directory as mentioned in step 2. This will create as many folders as binary files for every processor to read and write data.
4. Then prepare your input file by specifying reasonable values for various parameters as mentioned in 2.
5. Then in the src folder compile the code using make command and prepare an executable. This executable rxmd will be used for doing the computation.
6. From your root directory type this command for generating an output log file.

```
mpirun -np ${nprocs} ./rxmd > log
```

mpirun is the compiler which was used for compiling the code and preparing the executable in step 5. You can use other compilers as well which are compatible to the requirements of our code.
${nprocs} is the total number of processors required for performing the simuation.
log is the name of output file where the output results will be saved.