

# Documentation for ljmcmd

Thomas A. Knotts IV

## 1 Overview

ljmcmd is a program that performs Molecular Dynamics (MD) and Monte Carlo (MC) simulations with a Lennard Jones fluid. Specifically, the system is composed of a user-specified number of Lennard Jones atoms where each pair interacts via the potential

$$U^* = 4 \left[ (r^*)^{-12} - (r^*)^{-6} \right]$$

Here,  $U^*$  is the dimensionless potential energy between the pair of atoms that are separated by a dimensionless distance of  $r^*$ .

## 2 Running the Program

The program is run from the command line and requires two arguments—the name of the input file and the name of the output file. For example, for an input file arbitrarily named `simulation.input` and an output file arbitrarily named `simulation.output`, the program would be run with the following command

```
user@computer]$ ./ljmcmd simulation.input simulation.output
```

This command can also be placed in a submission script when submitting the program as a batch job on a computer cluster.

## 3 The Input File

### 3.1 General Description

The input file is composed of a series of *keywords* followed by *values* for the keyword. The values are either strings or numbers, depending on the specific keyword, with some keywords requiring more than one value. One keyword is placed on each line and separated from its value(s) by one or more white-space characters. The keyword must appear on the line first followed immediately by its value(s). Anything after the required value(s) is considered a comment and is disregarded by the program.

The input file controls all the parameters of the simulation. A minimum number of *required* keywords are needed, and several *optional* keywords may be given. Some *optional* keywords have default values as they are required for the simulation. Two other *optional* keywords designate whether a certain feature is turned on in the simulation. These are `rdf` and `movie`. The former triggers the program to calculate the *radial distribution function* while the latter instructs the program to output a movie file. Tables 1 and 2 contain a description of each keyword and its possible values for *required* and *optional* keywords, respectively. Table 3 describes the keywords needed to trigger the optional *features*. Many of the values require specifying an integer or a real (decimal) number, and the following notation is used to indicate permissible values.

- $[1, \infty)$ : An integer that is greater than or equal to 1.
- $[0, \infty)$ : An integer that is greater than or equal to 0.
- $(-\infty, 0)$ : An integer that is less than 0.
- $(0.0, \infty)$ : A real (decimal) number that is greater than 0.

Table 1: Description of keywords **required** in the input file.

Keyword	Possible Values	Description
sim	md	perform an NVE MD simulation
	mc	perform an NVT MC simulation
N	$[1, \infty)$	number of LJ atoms in the system
temp	$(0.0, \infty)$	temperature of the system
rho	$(0.0, \infty)$	density of the system
esteps	$[0, \infty)$	number equilibrium steps to simulate
psteps	$[0, \infty)$	number production steps to simulate
rcut	$(0.0, \infty)$	cut off distance for Lennard Jones potential
dt	$(0.0, \infty)$	time step for md or initial maximum random displacement for mc

Table 2: Description of **optional** keywords with default values.

Keyword	Default Value	Possible Values	Description
coord	generate	generate <i>filename</i>	generate initial coordinates on an FCC lattice read initial coordinates from file <i>filename</i>
vel	generate	generate <i>filename</i>	generate random initial velocities (md only) read initial velocities from file <i>filename</i> (md only)
output	2000	$[0, \infty)$	frequency for output of instantaneous properties
seed	-827165783	$(-\infty, 0)$ generate	seed for random number generator generate the seed from the system clock

Table 3: Description of keywords that trigger *optional* features.

Keyword	Values	Values Description
Generate the radial distribution function		
rdf	$(0.0, \infty)$ $(0.0, \infty)$ $[1, \infty)$ $[1, \infty)$	minimum bin value, maximum bin value, number of bins, frequency of accumulation
Generate a movie file		
movie	<i>filename</i> $[1, \infty)$	movie file name, frequency of frame output

### 3.2 Sample MD Input File

An example input file for an MD simulation appears below. This file specifies a simulation with 500 particles at a dimensionless temperature of 0.85 and a dimensionless density of 0.90. A total of 400000 equilibration steps are performed followed by 600000 production steps. The Lennard Jones potential is cut off at a dimensionless distance of 3.0, and the time step for the integration of the equations of motion is 0.001. The initial coordinates are read from a file named *equilibrated.cor*, and the initial velocities are randomly generated. Instantaneous properties of the system are output every 2500 steps, and the seed for the random number generator is obtained from the system clock. The radial distribution function is calculated where the minimum distance is 0.8, the maximum distance is 4.0, 100 bins are used to create the histogram of distances between 0.8 and 4.0, and the histograms are accumulated every 250 steps. Finally, a movie file is written to a file named *lj.trr* every 4000 steps.

```
sim      md                # simulation type
N        500              # number of particles
temp     0.85             # temperature
rho      0.90             # density
esteps   400000           # equilibrium steps
psteps   600000           # production steps
rcut     3.0              # Lennard-Jones cut off distance
dt       0.001            # time step
coord    equilibrated.cor  # initial coordinates
vel      generate         # initial velocities
output   2500             # frequency for output of instantaneous properties
seed     generate         # seed value for random number generator
rdf      0.8 4.0 100 250  # calculate rdf
movie    lj.trr 4000      # create a movie file
```

### 3.3 Sample MC Input File

An example input file for an MC simulation appears below. This file specifies a simulation with 500 particles at a dimensionless temperature of 0.85 and a dimensionless density of 0.90. A total of 1000 equilibration Monte Carlo *sweeps* are performed followed by 2000 production sweeps. The Lennard Jones potential is cut off at a dimensionless distance of 3.0, and the maximum possible value for the random particle displacement is 0.1. The initial coordinates are read from a file named *equilibrated.cor*, the instantaneous properties of the system are output every 10 sweeps. The seed for the random number generator is obtained from the system clock, and the radial distribution function is calculated where the minimum distance is 0.8, the maximum distance is 4.0, 100 bins are used to create the histogram of distances between 0.8 and 4.0, and the histograms are accumulated every 250 steps.

```
sim      mc                # simulation type
N        500              # number of particles
temp     0.85             # temperature
rho      0.90             # density
esteps   1000             # equilibrium sweeps
psteps   2000             # production sweeps
rcut     3.0              # Lennard-Jones cut off distance
dt       0.1              # maximum displacement for each MC move
coord    equilibrated.cor  # initial coordinates
output   10               # frequency for output of instantaneous properties
seed     generate         # seed value for random number generator
rdf      0.8 4.0 100 250  # calculate rdf
```

## 4 Description of Program Function

### 4.1 General Function

1. All properties of the system are *dimensionless* as defined below.

Property	Dimensionless Form
length	$r^* = \frac{r}{\sigma}$
mass	$m^* = \frac{m}{m} = 1$
time	$t^* = t \left( \frac{\epsilon}{m\sigma^2} \right)^{1/2}$
temperature	$T^* = \frac{k_B T}{\epsilon}$
force	$f^* = f \frac{\sigma}{\epsilon}$
energy	$U^* = \frac{U}{\epsilon}$
pressure	$P^* = P \frac{\sigma^3}{\epsilon}$
density	$\rho^* = \rho \sigma^3 = \frac{N}{V} \sigma^3$
diffusivity	$D^* = D \left( \frac{m}{\epsilon \sigma^2} \right)^{1/2}$

2. The key difference between *equilibration* and *production* steps concerns the averaging of properties. Specifically, instantaneous values of the system properties calculated during equilibration *do not* contribute to the average properties reported at the end of the simulation. Only instantaneous values from production steps contribute.
3. Movie files are written in the `.trr` format and can be read using the program VMD. A `.xyz` file is also generated along with the `.trr` file. The `.xyz` file should be loaded as a new molecule into VMD first, then the `.trr` file is loaded into this molecule. Once loaded, the representation of the system should be set to CPK.
4. In practice, values for the keywords found in Tables 1, 2, and 3 cannot take on values of  $\infty$  or  $-\infty$  but are limited by computational resources.

### 4.2 MD Simulations

1. During equilibration, MD simulations rescale the velocities to obtain the temperature set by keyword `temp`. This is done using the isokinetic ensemble based on the average temperature of the system over every 100 equilibration steps.
2. No velocity rescaling is done during production steps. As such, the final temperature will not correspond *exactly* to what was specified in the input file.
3. If the keyword `vel` is given a value of `generate` then the velocities are randomly assigned to each particle. This is done using a uniform distribution where the average value corresponds to the temperature set in the input file. Because the velocities should follow a Gaussian distribution, equilibration is needed.
4. For MD simulations, the system is propagated through phase space using the velocity verlet algorithm.
5. Each equilibration and production step propagates the system through time by the value set in `dt`. Thus, the total equilibration simulation time is `esteps*dt`, and the total production simulation time is `psteps*dt`.

### 4.3 MC Simulations

1. Each equilibration and production iteration set by `esteps` and `psteps` consists of `N` trial displacements. Thus each equilibration and production iteration is one Monte Carlo *sweep*. So the total number of

equilibrium displacements attempted is `N·esteps`, and the total number of production displacements attempted is `N·psteps`.

2. The program changes the maximum trial displacement during the simulation to achieve an average Monte Carlo move acceptance rate of 30%. If the acceptance rate is too low then the maximum displacement is reduced. If the acceptance it too high then the maximum displacement is increased.
3. The diffusion constant is not calculated for MC simulations.

## 5 Compiling the Program

The program has been designed to be compiled on both Windows and Linux systems.

### 5.1 Windows

The program can be opened in Visual Studio 2015 and later by opening the project file called `ljmdmc.vcxproj`. The program can then be compiled following normal Visual Studio procedures.

### 5.2 Linux

The program can be compiled using the gcc compiler via the Linux command line using the GNU make utility and the included `makefile` with the following command.

```
user@computer]$ make
```

This will create the executable named `ljmdmc` in the same folder. The previous compile can be cleaned with the following command.

```
user@computer]$ make clean
```

The makefile can be edited to use the Intel `icc` compiler if desired.

## 6 Format of Optional, Initial Coordinate and Velocity Files

As mentioned in Table 2, the simulation can be started from a previous state by providing a coordinate file for both MD and MC and, optionally, a velocity file for MD. To create the coordinate file, the coordinates for the first atom are placed on the first line, those for the second atom on the second line, and so forth for all `N` atoms. The x position appears first on the line, the y position second, and the z position third. Each coordinate on the line is separated by one or more whitespace characters. Below is an example of such a file.

```
6.070409583 0.770379849 3.826778865
4.682435100 0.501276711 3.777803414
0.467807771 1.551879071 3.367365787
.
.
.
2.556208915 4.328443569 5.142097909
```

The velocity input file is similar in construction. The velocities for the first atom are placed on the first line, those for the second atom on the second line, and so forth for all  $N$  atoms. The x component of the velocity appears first on the line, the y component second, and the z component third. Each component of the velocity on the line is separated by one or more whitespace characters. Below is an example.

```

0.105537744    -0.890556665    -1.374696491
0.221269507    -1.608868978    -0.033602851
0.670476524     1.430089088    -0.029336158
.
.
.
-0.130046005     0.1067971390     1.105135949

```