Manual 04 for MD++

Molecular Dynamics Simulation

Keonwook Kang and Wei Cai

January 19, 2007

1 NVE Ensemble

The easiest Molecular Dynamics (MD) simulation is the one with the micro-canonical, or NVE, ensemble, in which the total energy E and total volume E is conserved.[1] The command to run MD simulation in MD++ is simply `run`. Before calling this command, we need to set quite a few variables, such as `integrator_type`, `ensemble_type`, `timestep`, `totalsteps`, `atommass`, as well as other parameters that controls the simulation output. You can run an example by typing[2]

```
$ bin/fs_gpp scripts/mo_NVE.script
```

The content of `mo_NVE.script` is the following.

```
# -*-shell-script-*-
# MD simulation of Mo crystal
#------------------------------------------------------------
setnolog
setoverwrite
dirname = runs/mo-NVE
#-------------------------------------------
#Read in potential file
#
potfile = ~/Codes/MD++/potentials/mo_pot readpot
#------------------------------------------------------------
# Create Perfect Crystal
crystalstructure = body-centered-cubic
latticeconst = 3.1472           # in Angstrom for Mo
latticesize = [ 1 0 0 5
                0 1 0 5
                0 0 1 5 ]
```

[1] N is the total number of atoms, which is also conserved.

[2] `mo_NVE.script` can be downloaded from coursework.

```
makecrystal finalcnfile = "perf.cn" writecn
#
#-------------------------------------------------------------
#Plot Configuration
atomradius = 1.0 bondradius = 0.3 bondlength = 0
atomcolor = cyan highlightcolor = purple backgroundcolor = gray
plot_atom_info = 1  plotfreq = 10
win_width = 600 win_height = 600
openwin alloccolors rotate saverot refreshnnlist eval plot
#-------------------------------------------------------------
#Molecular Dynamics setting (equilibration run, NVE)
T_OBJ = 300 #(initial temperature, in K)
timestep = 0.0005 # (in picosecond)
atommass = 95.94  # Mo (g/mol)
randseed = 12345 srand48  #randomize random number generator
initvelocity writeall = 1 finalcnfile = init.cn writecn

integrator_type = "VVerlet"
ensemble_type   = "NVE"

totalsteps = 2001

#save property every 20 steps
saveprop = 1 savepropfreq = 10 openpropfile

#format of prop.out file
output_fmt = "curstep EPOT KATOM"

#save atoms to file every 100 steps
savecn = 1 savecnfreq = 100 openintercnfile

#MD simulation
run

#reverse velocity (to test reversibility)
input = -1  multiplyvelocity

#MD simulation (reverse)
run

finalcnfile = final.cn writecn
sleep  quit
```

In this example, MD++ first creates a perfect crystal of Molybdenum with supercell size $5[1\,0\,0] \times 5[0\,1\,0] \times 5[0\,0\,1]$ and then runs MD simulations using the Finnis-Sinclair (FS) potential. The initial velocity is first assigned as ran-

dom numbers but then scaled to correspond to an instantaneous temperature of $T = 300$K. This is done by command `initvelocity`. The initial condition is saved into file `init.cn`. `writeall = 1` tells MD++ to save both atom positions and velocities when calling `writecn`. In the future, we may load this file using `incnfile = init.cn readcn` to re-run the simulation with identical initial condition.

The random number generator is initialized with `randseed` (integer). This means they are pseudo-random numbers. Running the simulation again with the same `randseed` will give you the same result (which is good for debugging). Running the simulation again with a different `randseed` will give you a different initial velocity condition (which is good for collecting statistics). You may also use command `srand48bytime`, which will use the current time as the random seed (in this way, your simulation will always have a different initial velocity condition whenever you run it again).

The time step is specified in unit of picosecond (ps). Here $\Delta t = 0.0005$ ps $= 0.5$ femtosecond (fs). The atommass is specified in unit of g/mol.[3] The choices for `integrator_type` are `VVerlet` (for Velocity Verlet) and `Gear6` (for Gear 6-th order predictor-corrector). The quotation marks are optional. The choices for `ensemble_type` are `NVE`, `NVT`, `NPH`, and `NPT`. The total number of simulation steps is specified by `totalsteps`. Notice that for technical reasons, in MD++ the first step of `VVerlet` only evaluates the potential energy and atomic forces but do not move the atoms, while `Gear6` moves atoms even in the first step. This means that to reach the same number of simulation steps, the value for `totalsteps` when `VVerlet` is used needs to be larger by one than that when `Gear6` is used.

`openpropfile` creates a file `prop.out` in the directory `runs/mo-NVE` (specified by `dirname`). Every 10 steps (as specified by `savepropfreq`) a line of data is written into `prop.out`. The content is specified by `output_fmt`. `curstep` is the current simulation step. `EPOT` is the current potential energy. `KATOM` is the current kinetic energy. The total energy is the sum of `EPOT` and `KATOM`.[4] No output is written into `prop.out` if `saveprop = 0`.

`openintercnfile` creates a file `inter0001.cn` in the directory `runs/mo-NVE`. Every 100 steps (as specified by `savecnfreq`) the current configuration is saved and the next filename automatically is incremented by one (`inter0002.cn`, `inter0003.cn`, etc.). These are called intermediate files which can be used for future analysis (e.g. to make a movie). No intermediate files are saved when `savecn = 0`.

The format of the configuration (`*.cn`) files is the following. The first line specifies the total number of atoms (NP). After that there are NP lines of data, each correspond to one atom. The first three columns of these data are the scaled coordinates (s_x, s_y, s_z) of the atoms, and the next three columns are the scaled velocities of the atoms. After that, there are three lines of data specifies the matrix H. The real coordinates of an atom (x, y, z) and the scaled

[3] Atom mass for all elements can be found at `http://www.webelements.com`

[4] A simple way to plot the total energy using `gnuplot` (a free software on Unix/Linux) is to use the command `plot "prop.out" u ($1):($2+$3) with line`.

coordinates are related by the following,

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{pmatrix} \cdot \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix} \tag{1}$$

`multiplyvelocity` multiplies the number specified by the `input` to all velocity components of all atoms. After the simulation (`run`) has finished, MD++ saves the final configuration file into `final.cn`. The `sleep` command allows the graphics window to stay alive for a while so that we can look at the final atomic structure. You can press `ctrl-c` in the terminal to exit. You can also comment out the `sleep` command so that MD++ quits after finishing `run`.