Manual 01 for MD++

Introduction to MD++

Keonwook Kang and Wei Cai

October 7, 2005

1 Overview

MD++ code is an acronym of 'Molecular Dynamics in C++', which is originally developed by Prof. Cai during his graduate at MIT and still being updated to equip more powerful functions in it. MD++ is basically run in Unix/Linux environment but, if you have cygwin[1] installed in your Windows machine, you can also enjoy MD++ there. This code, as you can guess from its name, is designed for the molecular dynamics simulation with C++ language especially for studying the defect behavior in solid materials at molecular level. However, it is certain that this code can also be extended to solve other problems such as polymer/fluid interface and bio-material properties at users' will. This manual mainly targets those who do not have background knowledge and explains how to use MD++ with many simple examples.

2 Installation

The latest version of MD++ code can be downloaded from the coursework web site, http://coursework.stanford.edu/. Before you download MD++ code, make the following directory and download the file in this directory.

```
$ mkdir Codes
$ cd Codes
```

Unzip the downloaded MD++ code.

```
$ tar -zxvf cw282601_md++-2005-10-07.tar.gz
```

This command will make "MD++" directory and extract all the files and sub-directories under that directory. Make a "runs" directory if it does not exist.

```
$ cd MD++; mkdir runs
```

[1] http://www.cygwin.com/

To compile/build as release(R) mode, type like

```
$ make all build=R
```

This command gives you all the execution files like *fs_gpp*, *eam_gpp*, and *sw_gpp* in the 'bin' directory.[2] Unless you specfy the build mode, the default is a debugging(D) mode, which is a bit slower when you run the MD simuation. If you want to compile only a single binary file, e.g. *fs_gpp*, you may designate it like

```
$ make fs build=R
```

The name of each execution file stands for what potential you are using in the MD simulation. For example, *fs_gpp* means Finnis-Sinclair[3] potential, *eam_gpp* does embedded-atom method[4] potential, and *sw_gpp* does Stillinger-Weber[5] potential. Each potential has its own category of atoms for MD simulation. Body-centered-cubic (BCC) materials like Mo, Ta, and W atoms belong to the FS potential class so that you have to run *fs_gpp* if you want MD simulation of molybdenum atoms. The binary file, *md_gpp* is used only for displaying the structure and does not give any physical data like potential energy, stress, or temperature.
When you need to recompile after you modifiy the source code, you'd better clean the previous one and then rebuild.

```
$ make clean
$ make all build=R
```

To see more help on `make`, type

```
$ make help
```

To check if the installation process is properly done, run some example script,

```
$ bin/fs_gpp Examples/example01-mo.script
```

You can also compile MD++ code on other platforms, such as cygwin, by typing

```
$ make all SYS=cygwin build=R
```

[2] In the 'cygwin' platform, the file name would be fs_cygwin.exe, eam_cygwin.exe, and sw_cygwin.exe.
[3] M.W. Finnis, J.E. Sinclair, Philos. Mag. A 50(1984) 45
[4] PRB 29, 6443 (1984)
[5] PRB 31, 5262 (1985) and PRB 46, 2250 (1992)

3 MD++ Directories

After the installation of MD++, you will get the following directories under 'MD++' directory.[6]

```
$ ls
bin        Doc        makefile   potentials   scripts   Tools
CSD-book   Examples   matlab     README       src
```

/src	All the source codes are in this 'src' directory.
/bin	Binary (or execution) files will be built in the 'bin' directory.
/Doc	MD++ manual is in the 'Doc' directory.
/scripts	You save your script file in this directory.
/runs	The outputs are stored in the subdirectory of this directory.
/potentials	Potential files are in this directory.
/Examples	'Examples' directory has some example script files.
/Tools	Some tools like 'atomeye' for MD++ simulation.
/matlab	Matlab codes for the postprocessing.

4 Script File

To execute a molecular simuation, the MD++ code reads a script file in which you can store a series of commands and variables for your own simulation. To make or edit a script file, it is recommended to use *vi* or *emacs* editor in Unix/Linux environment. For the usage of these editors, search the mannuals on-line. Basically, to edit a script file, type

```
$ vi scripts/example01-mo.script
```

or

```
$ emacs scripts/example01-mo.script
```

If you open an example script file, you will see what a script file has in it. Script files have the common head part:

```
# -*-shell-script-*-
setnolog                    # No log file saved
setoverwrite                # Overwrite a log file if existed
dirname = runs/mo-example       # Specify output directory
```

In a script file, you can commnet out any line or a part of a line by starting it with the sharp mark, #. You can use # to leave your own comment or explanation anywhere in the script. `setnolog` will make no log file saved and the result will be displayed on your screen. If `setnolog` is commented out, the

[6] The new version of MD++ may have different directories, but the main structure is still almost same.

log file will be saved as "A.log.gz" in the output directory. All the other output files are also to be stored in the output directory designated in the head part. `quit` command stops the MD++ simulation. You can use `quit` alone or use it together with `sleep`.

```
quit                # quit the simulation
```

or

```
sleep quit          # System gets sleep for 600 sec before quit.
```

`sleep` command is very useful especially when you want to look at the graphic window carefully and play with it for some time.

5 Graphic Window Control

When you run MD simulation, usually a new window pops up and you obsserve the motion of atoms in that window. I call this window as 'graphic window'. There are several short keys to handle this window. To see the following help, press F1.

```
Mouse drag to rotate
Hot Keys:
F1        : display this message
Up        : rotate up
Down      : rotate down
Left      : rotate left
Right     : rotate right
PgUp      : rotate counterclockwise
PgDn      : rotate clockwise
Home      : back to initial viewpoint
Space     : stop rotate
p         : toggle pause
t         : translation
s         : scaling
d         : move projection infinity point
r         : rotation
f         : toggle pbc enableness
m         : toggle drawframe
g         : pbc glide
x         : pbc shift in x
y         : pbc shift in y
z         : pbc shift in z
w         : print window specification
F9        : output gif
F10       : output postscript
```