

Visualization

Keonwook Kang and Wei Cai

December 10, 2005

1 MD++ Visualization

1.1 Visualization Parameters

In MD++, there are several parameters related with displaying the simulation on the graphic window. I'll start with the basic parameters for the plotting configuration. In the very first script of the manual M02, you see the following plot configuration part.

```

:
:
#-----
# Plot Configuration
atomradius = 1.0  bondradius = 0.3  bondlength = 0
atomcolor = blue  highlightcolor = purple backgroundcolor = gray
bondcolor = red   fixatomcolor = yellow
plotfreq = 10 win_width = 600  win_height = 600
rotateangles = [ 0 0 0 1.0 ]
openwin alloccolors rotate saverot eval plot
:
:
```

The command, `openwin` opens the graphic window, whose size is determined by `win_width` and `win_height` in pixel. `atomradius` and `bondradius` are the radii of an atom and of a bond between atoms, respectively, and they are nondimensionalized by the half of the maximum length among three periodicity vectors. If the system is composed of more than two different atoms, each atom's radius can be defined like

```

atomradius = [ radius for the 1st species
               radius for the 2nd species ...].
```

`bondlength` is the length of a bond in Angstrom. If the interatomic distance is larger than the `bondlength`, the bond will not be displayed. In the script, since the bond length is set to zero, no bond will be shown. Using `atomcolor` and `bondcolor`, the following colors can be declared for atoms and

bonds: red, blue, green, yellow, orange, gold, black, white, gray, cyan, purple, magenta and so on. More colors are listed in the “namecolor.c” file. If you use more than two different atoms, you can specify their colors like¹

```
atomcolor = [ color for the 1st species
              color for the 2nd species ...].
```

`fixatomcolor` and `backgroundcolor` set the colors of fixed atoms and the background. `highlightcolor` determines the color of simulation box frame and at the same time the color of atoms to be highlighted. The atoms can be highlighted by specifying the index numbers like

```
plot_highlight_atoms = [flag indexnumbers]
```

Only if `flag=1`, the atoms of enumerated index numbers are to be highlighted. The command, `alloccolors` allocates RGB values to the color names used in the script file. See `alloccolors()` in the source code `md.cpp` if you want to know how the colors are allocated.²

`plotfreq` decides how frequently the atoms are displayed in the window. If `plotfreq=10`, the window is refreshed at every 10 steps. `rotateangles` decides the angles of view point in degree and the scaling factor.

```
rotateangles = [ horizontal rotation angle
                 vertical rotation angle
                 spin rotation angle
                 scale factor ]
```

```
rotate saverot
```

Look into “display.cpp” and “display.h” if you want to know how `rotate` and `saverot` are functioning.

Let’s see another script, “example01-mo.script” in the Example directory. In this script, the atoms are visualized according to their individual energy. See how it works from the following script.

```
#-----
# Plot Configuration
atomradius = 1.0 bondradius = 0.3 bondlength = 0
atomcolor = cyan highlightcolor = purple backgroundcolor = gray
bondcolor = red fixatomcolor = yellow
plot_color_bar = [ 1 -6.8 -6.55 ]
plot_atom_info = 3
plotfreq = 10 win_width = 600 win_height = 600
rotateangles = [ 0 0 0 1.5 ]
openwin alloccolors rotate saverot eval plot
```

`plot_color_bar` maps the value of color index, which can be energy or topology, of each atom to appropriate colors by the predefined colormaps. Its format is

¹The elements can also be declared like `nspecies = 2 element0 = “Si” element1 = “Ge”`.

²`colors = [1st user-declared color, 2nd user-declared color, ..., the maximum user-declared color, bond color, highlight color, fixatomcolor, background color, 1st species color, 2nd species color, ..., the maximum species color]`. The maximum numbers of user-defined color and species color are defined as 10 in “md.h”. To tell how users can declare colors, see the 1st script in the manual 04.

`plot_color_bar = [n low high]`.
`n=0` makes switch off. If `n=1`, mapping of $\alpha \equiv (\text{color_ind} - \text{low}) / (\text{high} - \text{low})$ is done by the rule `colormap1`. If `n=2`, mapping is done by the rule `colormap2`. For more information on colormaps, look into “`colormap.h`” file. By default, the color index of each atom expresses the potential energy of an individual atom. If you want it to be topological value, declare `plot_color_axis = 2` beforehand. And if you want it back, declare again `plot_color_axis = 0` or `1`. You may notice that `atomcolor = cyan` in the script effects no longer.

`plot_atom_info` has already been explained briefly in the manual M04 together with `plot_color_windows`³, and I will give full description of it here. `plot_atom_info` shows the following information on the event of clicking an atom with the right mouse button: `plot_atom_info = 1` gives the selected atom’s index number and reduced coordiante, `plot_atom_info = 2` gives its index number and real coordiante, `plot_atom_info = 3` gives the atom’s index number, color index (energy or topology value), whether it is fixed or not, and its species, `plot_atom_info = 4` gives index number and the force acting on that atom. `plot_atom_info = 5` gives RGB color of the atom only if `plot_color_bar` is activated. If neither `plot_color_bar` nor `plot_color_windows` is used, the species of the atom will be displayed by `plot_atom_info = 5`.

`plot_limits` will show certain atoms following the rules below.

```
plot_limits = [  flag
                x_lowerbound x_upperbound
                y_lowerbound y_upperbound
                z_lowerbound z_upperbound]
```

If `flag` equals to 0: then this option is ignored, 1: then only plot atoms within the specified limits, and 2: then atoms within the limits will be plotted with the same color as the fixed atoms. Note that the boundaries are given as reduced coordinates.

1.2 calcentralsymmetry

When you try to identify a defect in the crystal structure, sometimes it is more graphically indicating to use atoms’ local positions than their individual energies because the individual energy is easy to thermally fluctuate at nonzero finite temperature. The *centro-symmetry deviation* (CSD)[1] parameter is one of quantitative measures of deformation of an ideal centro-symmetry in the atom’s nearest neighborhood.

In the following example script⁴, `colorXX` (where `XX` is any number between 0 and 9.) is used to define specific colors. Once you allocate color name to certain number, you can call any color you want by its number as `plot_color_windows` does. The format of `plot_color_window` is

³For the usage of `plot_color_windows`, see the 1st script in the manual 04.

⁴Uploaded as “`mo_CSD.script`” in the coursework website. You also need to download the configuration file, “`relaxed_sgdisl.cn`”.

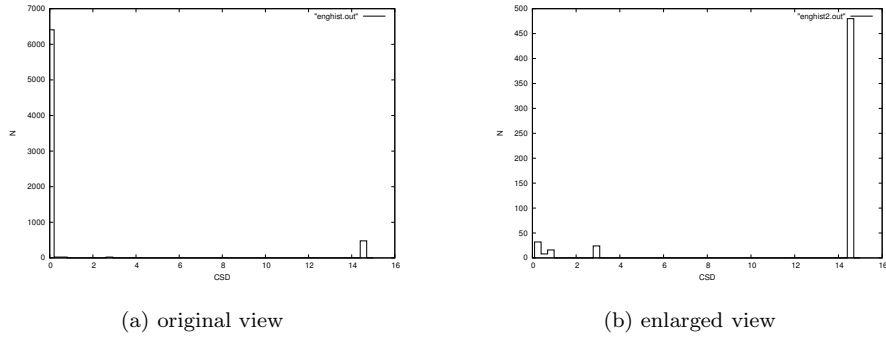


Figure 1: Histogram of atoms according to their CSD parameters when `input = [-0.1 15 50]`. N is number of atoms.

```
plot_color_window = [ Total number of colors
                      color_index_range color_number
                      color_index_range color_number
                      ...                ]
```

The atoms of which color index is within the range will be displayed in the corresponding color.

If you run the script, you will see a new window pop up as shown in Fig.1(b).⁵ `GnuPlotHistogram`⁶ command plots histogram of atoms according to their CSD parameters from 0.1 to 15 with bin number 50 defined in `input`. Since atoms in the perfect structure have 0 CSD values, those atoms are not counted. If you see the Fig.1(a), number of atoms at 0 CSD value is larger than 6000 when `input=[-0.1 15 50]`, and it means most atoms form perfect structure according to CSD.

```
# --shell-script--
setnolog setoverwrite
dirname = runs/mo_CSD
potfile = ~/Codes/MD++/potentials/mo_pot readpot
incnfile = relaxed_sgdisl.cn readcn
#-----
# colors for Central symmetry view
color00 = "red"      color01 = "blue"  color02 = "green"
color03 = "magenta" color04 = "cyan"   color05 = "purple"
```

⁵In case that the GnuPlot window does not show up even if you see the Fig.2, your `tcsh` may be the reason of the problem. What you can do is simply to comment out the 3 lines from 7027 to 7029 in the `md.cpp` and compile it again.

```
#!/bin/tcsh\n
setenv LD_LIBRARY_PATH ~/Tools/Lib:\n
module add gnuplot\n
```

⁶For `GnuPlotHistogram`, GnuPlot is required.

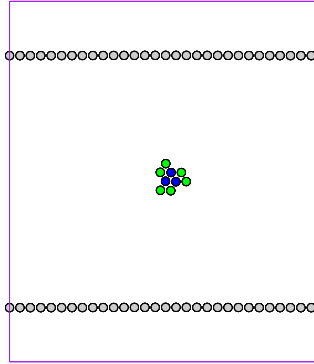


Figure 2: A single dislocation in BCC Molybdenum. Top and bottom parts are cut away and gray atoms are free surface atoms. Atoms in perfect structure are not displayed.

```

color06 = "gray80"  color07 = "white"
#-----
# Plot Configuration
atomradius = 1.0 bondlength = 0
atomcolor = cyan highlightcolor = purple backgroundcolor = gray
fixatomcolor = yellow plotfreq = 10
plot_color_bar = [ 1 -6.8 -6.55 ] plot_atom_info = 3
rotateangles = [ 0 0 0 1.5 ] win_width = 600 win_height = 600
#-----
# (central symmetry)
plot_color_axis = 2 # In case of coloraxis = 2, color_ind = _TOPOL
NCS = 8             # FCC = 6*2, BCC = 4*2
refreshnlist calcentralsymmetry
input = [ 0.1 15 50 ] GnuPlotHistogram
plot_color_windows = [ 3
                      0.3 2 2 #color02 = green
                      2 4 1 #color01 = blue
                      10 30 6 #color06 = gray80
                      ]
openwin alloccolors rotate saverot plot
sleep quit

```

According to the histogram (Fig.1(b)), `plot_color_window` has three different color index regions: 0.3 to 2, 2 to 4, and 10 to 30. Atoms in each region will be plotted in green, blue, and gray80, respectively. The graphic result is shown in the Fig.2.

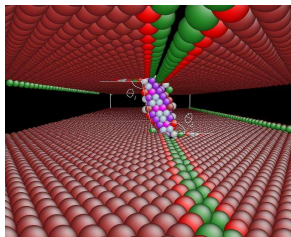


Figure 3: A dislocation in Mo thin film is visualized by Atomeye.

1.3 Using Atomeye

Atomeye[2] is one of frequently used atomistic configuration viewer program and can be freely downloaded from the website, <http://164.107.79.177/Archive/Graphics/A/>. To use Atomeye, the configuration *cn* file needs to be exported to *cfg* file, which is standard datafile format for Atomeye. If the following line is added in the script file

```
finalcnfile = relaxed.cfg writeatomeyecfg
```

the “relaxed.cfg” file is generated in the output directory. Then typing the command

```
$ cd ~/Codes/MD++/runs/(output directory)
$ ../../Tools/Atomeye/A relaxed.cfg
```

will run Atomeye program. For more explanation on Atomeye, read the manual in the above website. A sample graphic result with Atomeye is shown in Fig.3. If you want to directly display the result with Atomeye, include the following lines in the script.⁷

```
#-----
#View by AtomEye (Li Ju)
atomeyepath = ~/Codes/MD++/Tools/AtomEye
atomeyeexe = A element0 = Mo
#atomeyerepeat = [ 1 2 1 2 ]
eval atomeye
```

References

- [1] Vasily V. Bulatov and Wei Cai, “Computer Simulation of Dislocations” unprinted
- [2] J. Li, Modelling Simul. Mater. Sci. Eng. **11** 173 (2003)

⁷Currently, the configuration output file has a fixed file name, “atomeye.cfg”. You can change the file name by inseting this line, `command = “mv atomeye.cfg filenameyouwant.cfg”` `runcommand` in the script file, if you want to.