

MDSPASS manual

Y. Umeno

February 4, 2012

1 Tutorial

1. Download mdspass package from <http://www.cmsm.iis.u-tokyo.ac.jp/code/>
 `md.XXXX.tar.gz` : Full package (with sample input files and scripts, potential data, etc.)
 `src.XXXX.tar.gz` : Fortran source codes only
2. Then extract files and compile.

```
$ tar xzvf md.XXXX.tar.gz
$ cd md.XXXX
$ ./build
```

3. Set configuration and setting files.

```
$ cp config.sample/CONFIG.Sn333 CONFIG
$ cp setdat.sample/SETDAT.typical SETDAT
```

4. Run.

```
$ ./mdspass
```

♠ Note

When compile errors due to lack of lapack and lblas appear, install them. You may use yum package manager if available:

```
# yum install lapack*
# yum install lblas*
```

2 Input/output files

2.1 Input

- SETDAT : Settings of simulation conditions.
- CONFIG : Atom positions, atom species and potential function.
- CONT : This file is read when `continue_md` is set to `yes` to run MD continuing from a preceeding run.
- CONT.VAR : Information relevant to FIRE algorithm is stored. This file is read when `continue_md` is set to `yes`.

2.2 Output

- CONFIG.END / CONFIG.END.FRA : Final atom configuration written in the same format as CONFIG. In CONFIG.END.FRA, atom position are written in fractional coordinates.
- energy.d : Potential energy, kinetic energy and total energy at each step. [eV/atom]
- stress.d : Stress tensor ($\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \tau_{xy}, \tau_{xz}, \tau_{yx}, \tau_{yz}, \tau_{zx}, \tau_{zy}$) at each step. [GPa]
- cell.d : Simulation cell size ($h1x, h1y, h1z, h2x, h2y, h2z, h3x, h3y, h3z$) at each step. [Å]
- conf.xyz/conf.xsf/conf.cfg : Atom configuration in formats for viewers. (xyz : rasmol or xcrysden, xsf : xcrysden, cfg : atomeye)
- LAST : Final atom positions, velocity, etc. This file should be renamed to CONT to run a sequential calculation (do not forget to set `continue_md yes` in SETDAT).

2.3 Potential files

When you use EAM or MEAM potential, EAMPARAM* or meam-*.ref.d should be put in `pot/` directory, respectively. (Please do NOT touch files in `pot/`)

3 Frequently used settings in SETDAT file

3.1 Molecular dynamics

3.1.1 Standard calculation

Basic settings for an MD calculation.

```
#
# SETTING FILE
#
# For real number input, use double precision form
# unit should be in MKS (e.g. m, K, Pa)
# For logical input, not only .true./.false.
# but YES/Yes/yes/NO/No/no are acceptable.
#
# For more detail, please check 'readcondition.F'
#

# Time step
dt 1.0d-15

# Setting cut-off radius. (useless except for MEAM Sn)
# rc 8.0d-10
# Setting cut-off radius for book-keeping.
frc 8.0d-10
# Length of bookkeeping list table can be set here
nbook 12000000
# Interval of bookkeeping
nbk 20
```

```

# Region partitioning (for accelerating bookkeeping)
rpart no
npcx 2
npcy 2
npcz 8

# Iteration number (Number of MD step calculated in one mdspass run)
step_ite 100

# Tolerance (energy(eV/atom) or max.force(eV/ang)
# negative value or comment out to ignore
# Energy tolerance
#tolene 1.0e-18
# Force tolerance
#tolfor 1.0e-5
# Stress tolerance
#tolst 1.0e1

# Velocity scaling (nonzero means interval steps)
vscale 1

# Atom damper for relaxation (1=GLOC,2=FIRE)
damper 0

# Constraint of cell-deformation (works for cell-changing algorithm)
cellconstx no
cellconsty no
cellconstz no

# Algorithm number
#      ! 0=Velocity Verlet, NVE (Allen & Tildesley)
#      ! 1=Gear's P-C, NVE (Allen & Tildesley)
#      ! 2=Gear, NVT (Nose thermostat)
#      ! 3=Gear, NPH (Andersen pressure control)
#      ! 4=Gear, NPT (Nose and Andersen)
#      ! 5=Gear, NPH-PR (Parrinello-Rahman)
#      ! 6=Gear, NPT-PR (Nose and Parrinello-Rahman)
#      ! 7=Stress relaxation by annealing, no atom motion
#      ! 8=Stress relaxation by annealing, atom motion by verlet
#      ! 107=Gear, Stress relaxation by NPH-PR & breaking
#      ! 100=Atom relaxation by quasi Newton-Raphson (BFGS), cell size fixed
algorithm 0

# Factor for initial relaxation step of algorithm = 100
relax_factor 3.0

# Weight of piston/thermostat (positive value)
# (1.0d0 is default. large value for slow response.)
stress_weight 5.0d1
temp_weight 1.0d-1

# Periodic boundary

```

```

pbcx yes
pbcy yes
pbcz yes

# Target temperature
temp_set 500.0d0

# Target stress
# For Andersen method, use sgm_set
sgm_set 0.0d6
# For Parrinello-Rahman method, use following values
sxx 0.0d6
syy 0.0d6
szz 0.0d6
syz 0.0d6
szx 0.0d6
sxy 0.0d6

# Continue? If yes, CONT is used. (should be copied from LAST)
# Otherwise CONFIG is used. (Remember: last conf is saved to CONFIG.END)
continue_md no

# Whether initial velocity is given (does not work when contine_md)
ini_velocity yes

# Infinitesimal strain for numerical evaluation of elastic constants
# If the value is positive, elastic constants are calculated at the end of MD
# Cubic symmetry is assumed
econstd -1.0d-3

# Instability check
inst no

# Voronoi polyhedron analysis: radius (ang), zero or negative means no analysis
voronoi -6.0

# Diffusion simulation
# difout_int is an interval steps of making displacement file
diffusion no
difout_int 100

```

3.1.2 MD calculation under constant stress

To perform an MD calculation under constant stress, some lines in SETDAT should be changed as follows. For example, to apply hydrostatic stress of 100 MPa, set the folloing tags.

```

sxx 100.0d6      # target stress value are set here.
syy 100.0d6
szz 100.0d6
syz 0.0d6
szx 0.0d6
sxy 0.0d6
algorithm 5      # Parrinello-Rahman method
stress_weight 5.0d # Weight of piston for PR method

```

♠ Note

stress_weight tag should be chosen (maybe by trial and error) so that the fluctuation in stress values has a reasonable cycle and amplitude.

3.1.3 Md calculation for diffusion

To perform a diffusion simulation, you have only to change some tags in SETDAT:

```
diffusion yes
difout_int 100
```

diffusion yes switches on the diffusion analysis mode, where the displacements of atoms from the initial sites are stored during an MD run. Mean square displacement is calculated based on the stored information of displacements and is written to **msd.d** file at every time step.

difout_int determines the number of interval steps to write the displacement in **displacementXXX.d** (XXX is the file number). **displacementXXX.d** has the format of

```
δxi=1, δyi=1, δzi=1
δxi=2, δyi=2, δzi=2
δxi=3, δyi=3, δzi=3
...
```

where δx^i , δy^i and δz^i indicate the displacement of the atom i in the x , y and z directions, respectively.

At the end of an MD run, **LAST.DIF** file storing the displacements is created, which has to be renamed to **CONT.DIF** to continue MD with **continue_md yes**.

3.2 Atom relaxation

If you want to do an atomic relaxation, set the following values.

Convergence condition can be chosen from energy (**tolene**) or max.force (**tolfor**).

```
tolfor 1.0e-12    # max.force (eV/ang)
#tolene 1.0e-18   # energy      (eV/atom)
```

```
damper 2          # 1: GLOC, 2: FIRE
```

The FIRE algorithm should be more efficient than GLOC in most of the cases. It should be noted that the time step (**dt**) changes in FIRE.

3.3 Constant stress MD at finite temperature (Parrinello-Rahman with damper)

For an MD under constant stress at a nonzero temperature, you may want to reduce the cell-size oscillation of the Parrinello-Rahman stress control method. For this purpose, the 'Parrinello-Rahman with damper' algorithm (**algo 50**) is provided. A sample SETDAT file can be found in **setdat.sample/** directory (**SETDAT.PR_with_damper**), where explanations for related tags are interspersed.

It should be needed to adjust the three tags: **average_span**, **stress_weight**, and **prdamper**. It may be recommended to set **prdamper** to 0 first (switching off the PR damper) and adjust **stress_weight** so that you have cell-size change oscillating with an appropriate cycle (probably, around 100 fs). See **cell.d** to check this. Then, you keep the value of **stress_weight** and set **prdamper** to a positive value (*e.g.*, 1.0). If you are lucky you will find the oscillation is reduced. You can control the strength of the damper by adjusting **prdamper**.

By setting **stress_weight** to extremely large value (*e.g.*, 1.0d50), you can fix the cell-size and see how much the stress oscillation by thermal vibration of atoms is. The oscillation, which should be larger at higher temperatures, cannot be avoided because it is the nature of the crystal at finite temperatures.

average_span controls the time span over which the stress is averaged. When it is 100 for example, at each time step the stress values in the last 100 steps are averaged and are written in 'stressave.d'. (Note: **average_span** does not affect the behavior of MD, though. It is just for the display of the stress data.)

The minimum and maximum values for `average_span` are set to 10 and 1000, respectively. If you set the value over (below) the maximum (minimum) value, it will be replaced with 1000 (10). The default value of `average_span` is 100.

Yet another algorithm for P-R damper A new tag `prdamper2` has been added. When it is positive, yet another scheme of P-R damper is activated. This new scheme may be more efficient in reducing P-R fluctuation, i.e. the amplitude of fluctuation in cell matrix components can be smaller. But the difference may not be remarkable.

prlimit tag to avoid divergence

The `prlimit` parameter is available to avoid divergence. It sets the limit of cell change (“velocity” of cell matrix); when reaching the limit the cell matrix “velocity” (1st derivative of cell matrix) is halved and higher derivatives are nullified to apply “break” to the cell matrix fluctuation.

A recommended value for `prlimit` is around 100.0, but it is not well-tested. Please find a good value by trial and error.

3.4 Cell and atom relaxation

The followings are sample settings for cell (stress) and atom relaxation, but this is not well-tested.

```
tolst 1.0e1      # stress convergence condition (MPa)
vscale 0         # no velocity scaling
damper 2         # FIRE algorithm
algorithm 50      # Gear, Stress relaxation by NPH-PR & breaking
temp_set 0.0d0   # temperature 0 K
```

4 CONFIG file

In this section some sample CONFIG files are shown.

4.1 Sample of unaly system: Copper unit cell

```
Cu      <-- Atom species.
EMT_JSN <-- Potential function. For details see section "Potential function".
      4  <-- Number of atoms. N
3.58736 <-- Lattice constant a. []
1.0 0.0 0.0 <-- Lattice vector 1. Multiplied by a.
0.0 1.0 0.0 <-- Lattice vector 2. Multiplied by a
0.0 0.0 1.0 <-- Lattice vector 3. Multiplied by a
fra     <-- Atom position are given in fractional coordination.
0.0 0.0 0.0 <-- Atom position for N atoms.
0.0 0.5 0.5
0.5 0.0 0.5
0.5 0.5 0.0
```

4.1.1 Atom constraint

To constrain atom motion, add T(move) or F(fix) after atom position. For example,

```
0.0 0.0 0.0 F F F <-- completely fixed
0.0 0.5 0.5 T T F <-- only motion in z direction is constrained
0.5 0.0 0.5      <-- free
0.5 0.5 0.0      <-- free
```

♠ Note

If you apply atom constraint, T or F tag should be set for every components.
The following lines are not acceptable.

```
0.0 0.0 0.0 F
0.0 0.5 0.5 T F
0.5 0.0 0.5 F   F
0.5 0.5 0.0 T T
```

4.2 Sample of binary system: SiC

When simulation cell contains many species, set CONFIG file as follows.

```
Multi
Tersoff M
      8
4.37
1.0 0.0 0.0
0.0 1.0 0.0
0.0 0.0 1.0
fra
0.00 0.00 0.00 Si
0.50 0.50 0.00 Si
0.00 0.50 0.50 Si
0.50 0.00 0.50 Si
0.25 0.25 0.25 C
0.75 0.75 0.25 C
0.25 0.75 0.75 C
0.75 0.25 0.75 C
```

To constrain atom motion, change lines as follows,

```
0.00 0.00 0.00 Si F F F
```

4.3 Potential function

Potential functions available in MDSPASS are given in Table 4.3. To specify potential function in CONFIG, use keyword.

5 Modifications and additional remarks

5.1 Bookkeeping method

5.1.1 Region partitioning method to accelerate the bookkeeping

5.1.2 Size of bookkeeping list table

5.2 Directory structure

5.3 Multiplication of simulation cell

You may want to double (triple, ...) the size of the cell. When `multicellx 2` is included in SETDAT, the cell is doubled in the x direction and the number of the atoms becomes two times as many, immediately after CONFIG is read. `multicelly` and `multicellz` tags are also available. These tags are valid only when the configuration is read from CONFIG (i.e. `continue_md no`).

Keyword	Potential	Reference
EMT_ST1	EMT for Au	J.Chem.Phys. 92 (1990) p.6306
EMT_HM1	EMT for Cu	J.Phys.: Condens. Matter 1 (1989) p.9765
EMT_JSN	EMT for Cu, Ag, Au, Ni, Pd, Pt	Surface Science 366 (1996) p.394
EAM_MIS	Mishin potential for Al, Cu, Ni	PRB 59 (1999) p.3393
EAM_Mishin	(Synonym of above)	
EAM_FS1	Finnis-Sinclair potential for Fe	Phil. Mag. A 50 (1984) p.45
EAM_NiAl	EAM for NiAl by Mishin	PRB 65 (2002) art.224114
GEAM_PRB	Generalized EAM for 16 species of metal	PRB 69 (2004) art.144113 ^{*1}
GEAM_AM	Generalized EAM for 16 species of metal	Acta Mater. 49 (2001) p.4005
EAM_GEAM	(Synonym of above)	
EAM_TiAl	EAM for TiAl by Zope	PRB 68 (2003) art.024102
EAM_ZopeAl	EAM for Al by Zope ^{*2}	PRB 68 (2003) art.024102
Tersoff_B	Tersoff potential (B, surface structure)	PRB 38-14(1988)p.9902
Tersoff_C	Tersoff potential (C, elastic property)	PRB 38-14(1988)p.9902
Tersoff_B2	The same as B except for cutoff ($R = 2.75, D = 0.1$)	PRB 37-12(1988)p.6991
Tersoff_M	Tersoff multisystem (Si-C, Si-Ge)	PRB 39-8(1989)p.5566
TersoffNM	Tersoff for small cells (see caveat in the text)	
Tersoff_Sn	Tersoff potential for Sn diffusion	J.Negami (2011) unpublished
StilWeb	Stillinger-Weber potential for Si	PRB 31, p.5262 (1985)
Shellmodel_PT0	Shell Model for PbTiO ₃	
Morse	Morse potential	Phys. Rev. 114 (1959) p.687
MEAM_SiC	Modified EAM (MEAM) for SiC by Huang et al.	MSMSE 3, pp.615-627 (1995)
MEAM_Sn	Modified EAM (MEAM) for Sn by Ravelo et al.	PRB 79, pp.2482-2485 (1997)
Brenner	Brenner potential for C/H	

^{*1} Gives a strange lattice constant for Al bulk.

^{*2} Uses parameters for Al single-atom system available at www.ctcms.nist.gov/potentials/Al.

5.4 Random number generation for initial velocity

With the modification on Jan. 31, 2011, random numbers are generated using the information of current time so that you'll get different distribution of initial velocity each time you run a new MD.

If you don't like this setting, you may set `ran_seed XXX` in `SETDAT` so that the specified number (XXX) is used as the "seed" for the (pseudo)random number generation. I.e. MD runs with the same seed number will produce exactly the same results. Note that XXX should be a positive integer, otherwise the present-time-dependent random generation scheme is used.

5.5 Caveat for MEAM Sn

Cut-off radius setting

For MEAM Sn, `rc` tag in `SETDAT` is valid. Please give an appropriate value for `rc`.

Speed-up

The MEAM potential is relatively time-consuming. It may be improved by reducing the frequency of stress calculation, if it is not necessary to calculate stress (local atomistic stress and global cell stress) at each time step. You can do so by setting `meam_stress_frequency` tag in `SETDAT`. For example, `meam_stress_frequency 10` means the stress is calculated every 10 steps (step=1, 11, 21, ...). When `meam_stress_frequency` is zero or negative, it is ignored.

This may cause problem when a stress-control algorithm (such as Parrinello-Rahman) is used (not tested well).

5.6 Restriction of cell rotation

With `fixrotation yes` in `SETUP`, you can fix the cell rotation during MD. This fixes the plane by the 1st and 2nd cell vectors (5th and 6th lines in `CONFIG`) to the $x - y$ plane.

5.7 Source of errors

6 Instability analysis

7 Shell Model

7.1 CONFIG file

A sample `CONFIG` file for PbTiO_3 (the cubic unit cell) is shown below.

```
Multi
Shellmodel PT0
      10
      3.8910
      1.0000000000000000 0.0000000000000000 0.0000000000000000
      0.0000000000000000 1.0000000000000000 0.0000000000000000
      0.0000000000000000 0.0000000000000000 1.0000000000000000
fra
      0.0000000000000000 0.0000000000000000 0.0000000000000000 Pb
      0.5000000000000000 0.5000000000000000 0.5000000000000000 Ti
      0.5000000000000000 0.5000000000000000 0.0000000000000000 O
      0.5000000000000000 0.0000000000000000 0.5000000000000000 O
      0.0000000000000000 0.5000000000000000 0.5000000000000000 O
      0.0000000000000000 0.0000000000000000 0.0000000000000000 Pb
      0.5000000000000000 0.5000000000000000 0.5000000000000000 Ti
      0.5000000000000000 0.5000000000000000 0.0000000000000000 O
      0.5000000000000000 0.0000000000000000 0.5000000000000000 O
      0.0000000000000000 0.5000000000000000 0.5000000000000000 O
```

7.2 Note for settings

8 Useful script

8.1 Calculate average lattice constant

8.1.1 CONFIG.TMP

Prepare `CONFIG` file for ideal crystal, then copy on `CONFIG.TMP`.

8.1.2 SETDAT

Set following parameters in `SETDAT` file,

```
frc 3.0d-10
step_ite 1
pbcx yes
pbcy yes
pbcz yes
algorithm 0
temp_set 0.0d0
ini_velocity no
```

8.1.3 Script

Run following script. The result will be output in SUMMARY.

```
#!/usr/bin/perl
open (S, ">>SUMMARY");
$a=3.00;
$ainc=0.1;
while ($a<=4.50) {

    open(C, "<CONFIG.TMP");
    open(F, ">CONFIG");
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;print F "$a\n";
    while ($line=<C>) {
        chomp($line);
        print F "$line\n";
    }
    close(C);
    close(F);
    system("./mdspass");

    # Extract energy output (last line) from energy.d
    open (T,"<energy.d");
    while ($line=<T>) {
        chomp($line);
        @splitline=split(/\s+/, $line);
        @result[1]= "$splitline[1]";
        @result[2]= "$splitline[2]";
        @result[3]= "$splitline[3]";
        @result[4]= "$splitline[4]";
    }
    close (T);
    $ene=$result[2];
    print S "$a $ene\n";

    $a=$a+$ainc;
}
close(S);
```

8.2 Uniaxial loading

Compression of Si.

8.2.1 CONFIG.TMP

```
Si
 Tersoff B2
      64
5.4310
1.41421356 0.0 0.0
0.0 1.41421356 0.0
0.0 0.0 4.0
```

```

fra
0.0000000000000000 0.0000000000000000 0
0.2500000000000000 0.2500000000000000 0.125
0.2500000000000000 0.0000000000000000 0.0625
0.0000000000000000 0.2500000000000000 0.1875
...

```

8.2.2 SETDAT

```

dt 1.0d-15
frc 3.5d-10
nbc 20
step_ite 1
algorithm 0
pbcx yes
pbcy yes
pbcz yes
temp_set 0.0d0
continue_md no
ini_velocity no

```

8.2.3 Script

```

#!/usr/bin/perl
open (S, ">SUMMARY2");
$lz=1.00;          # lz
$lzinc=-0.01;      #
while ($lz>=0.6) { #
    $lx=1.00;      # lx
    $lxinc=0.01;   #
    while ($lx<1.2) { #
        $lzz=$lz*2.0;
        open(C, "<CONFIG.TMP");
        open(F, ">CONFIG");
        $line=<C>;chomp($line);print F "$line\n";
        $line=<C>;chomp($line);print F "$line\n";
        $line=<C>;chomp($line);print F "$line\n";
        $line=<C>;chomp($line);print F "$line\n";
        $line=<C>;print F " $lx $lx 0.0\n";
        $line=<C>;print F " -$lx $lx 0.0\n";
        $line=<C>;print F " 0.0 0.0 $lzz\n";
        while ($line=<C>) {
            chomp($line);
            print F "$line\n";
        }
        close(C);
        close(F);
        system("./mdspass");
    }
    # Extract energy output (last line) from energy.d
    open (T,"<energy.d");
    while ($line=<T>) {
        chomp($line);
        @splitline=split(/\s+/, $line);
        @result[1]= "$splitline[1]";
    }
}

```

```

@result[2]= "$splitline[2]";
@result[3]= "$splitline[3]";
@result[4]= "$splitline[4]";
}
close (T);
$ene=$result[2];
# Extract stress output (last line) from stress.d
open (T,"<stress.d");
while ($line=<T>) {
chomp($line);
@splitline=split(/\s+/, $line);
@result[1]= "$splitline[1]";
@result[2]= "$splitline[2]";
@result[3]= "$splitline[3]";
@result[4]= "$splitline[4]";
}
close (T);
$sxx=$result[2];$syy=$result[3];$szz=$result[4];
print S "$lx $lz $ene $sxx $syy $szz\n";
    $lx=$lx+$lxinc;
}
print S "\n";
$lz=$lz+$lzinc;
}
close(S);

```

8.3 x,y direction : stress controll, z direction : compression

8.3.1 SETDAT

```

dt 1.0d-15
frc 3.5d-10
nbk 20
step_ite 1000
tolst 1.0e-3      # Stop calculation when changes in stress is smaller than tolst
cellconstx no
cellconsty no
cellconstz yes    # Allow cell size change in z direction.
algorithm 7
stress_weight 10.0d-1
pbcx yes
pbcy yes
pbcz yes
temp_set 0.0d0
# Set transverse stress here:
sxx 5000.0d6
syy 5000.0d6
continue_md no
ini_velocity no

```

8.3.2 Script

```

#!/usr/bin/perl
open (S, ">SUMMARY");

```

```

$lz=0.45;
$lzinc=-0.01;
while ($lz>=0.4) {

    $lzz=$lz*4.0;
    $ba1x=1.0;
    $ba1y=0.0;
    $ba1z=0.0;
    $ba2x=0.0;
    $ba2y=1.0;
    $ba2z=0.0;
    $ba3x=0.0;
    $ba3y=0.0;
    $ba3z=1.0;

    open(C, "<CONFIG.TMP");
    open(F, ">CONFIG");
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;chomp($line);print F "$line\n";
    $line=<C>;print F " 0.0 0.0 $lzz\n";
    while ($line=<C>) {
        chomp($line);
        print F "$line\n";
    }
    close(C);
    close(F);
    system("./mdspass");

# Extract energy output (last line) from energy.d
open (T,"<energy.d");
while ($line=<T>) {
    chomp($line);
    @splitline=split(/\s+/, $line);
    @result[1]= "$splitline[1]";
    @result[2]= "$splitline[2]";
    @result[3]= "$splitline[3]";
    @result[4]= "$splitline[4]";
}
close (T);
$ene=$result[2];
# Extract stress output (last line) from stress.d
open (T,"<stress.d");
while ($line=<T>) {
    chomp($line);
    @splitline=split(/\s+/, $line);
    @result[1]= "$splitline[1]";
    @result[2]= "$splitline[2]";
    @result[3]= "$splitline[3]";
    @result[4]= "$splitline[4]";
}

```

```

}
close (T);
$sxx=$result[2];$syy=$result[3];$szz=$result[4];

# Extract cell-size output (last line) from cell.d
open (T,"<cell.d");
while ($line=<T>) {
chomp($line);
@splitline=split(/\s+/, $line);
@result[1]= "$splitline[2]";
@result[2]= "$splitline[3]";
@result[3]= "$splitline[4]";
@result[4]= "$splitline[5]";
@result[5]= "$splitline[6]";
@result[6]= "$splitline[7]";
@result[7]= "$splitline[8]";
@result[8]= "$splitline[9]";
@result[9]= "$splitline[10]";
}
close (T);
    $l1=sqrt($result[1]**2+$result[2]**2+$result[3]**2);
    $l2=sqrt($result[4]**2+$result[5]**2+$result[6]**2);
    $l3=sqrt($result[7]**2+$result[8]**2+$result[9]**2);

print S "$lz $ene $sxx $syy $szz $l1 $l2 $l3\n";

$lz=$lz+$lzinc;
}
close(S);

```

8.4 Output sequential picture

Script to execute sequential calculation. For example, set `step_int 100` in SETDAT.

8.4.1 Script

```

#!/usr/bin/perl
system ("rm -rf conf*cfg");
for ($count=1; $count<=500; $count++) {
    $num=sprintf("%03d", $count);
    if ($count==1) {
        $contvalue="no";
    } else {
        $contvalue="yes";
    }
}

# Read SETDAT and change "continue_md" key
open(C, "<SETDAT");
open(F, ">SETDAT.TMP");
while ($line=<C>) {
    chomp($line);
    @splitline=split(/\s+/, $line);
    if ($splitline[0] eq 'continue_md') {

```

```

print F "continue_md $contvalue\n";
    } else {
print F "$line\n";
    }
}
close(C);
close(F);
system ("rm SETDAT"); system ("mv SETDAT.TMP SETDAT");
system ("./mdspass");
system ("cp conf.cfg conf$num.cfg");
system ("rm CONT");
system ("cp LAST CONT");
}

```

Calculation will be executed sequentially for 500 times.
 (100 step/execution \times 500 times = 50000 steps in total)
 Atom position will be output every 100 steps in conf*.cfg.

8.4.2 Make sequential picture

Make `scr_anim` file containing following script

```

90                                <-- Jpg quality
conf001.cfg 001.jpg
conf002.cfg 002.jpg
...
conf500.cfg 500.jpg

```

Launch `atomeye`
`$ atomeye conf001.cfg`
 press 'y', then 'enter'.

For details see <http://mt.seas.upenn.edu/Archive/Graphics/A/>