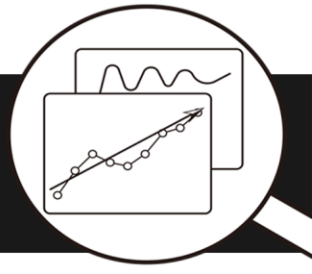
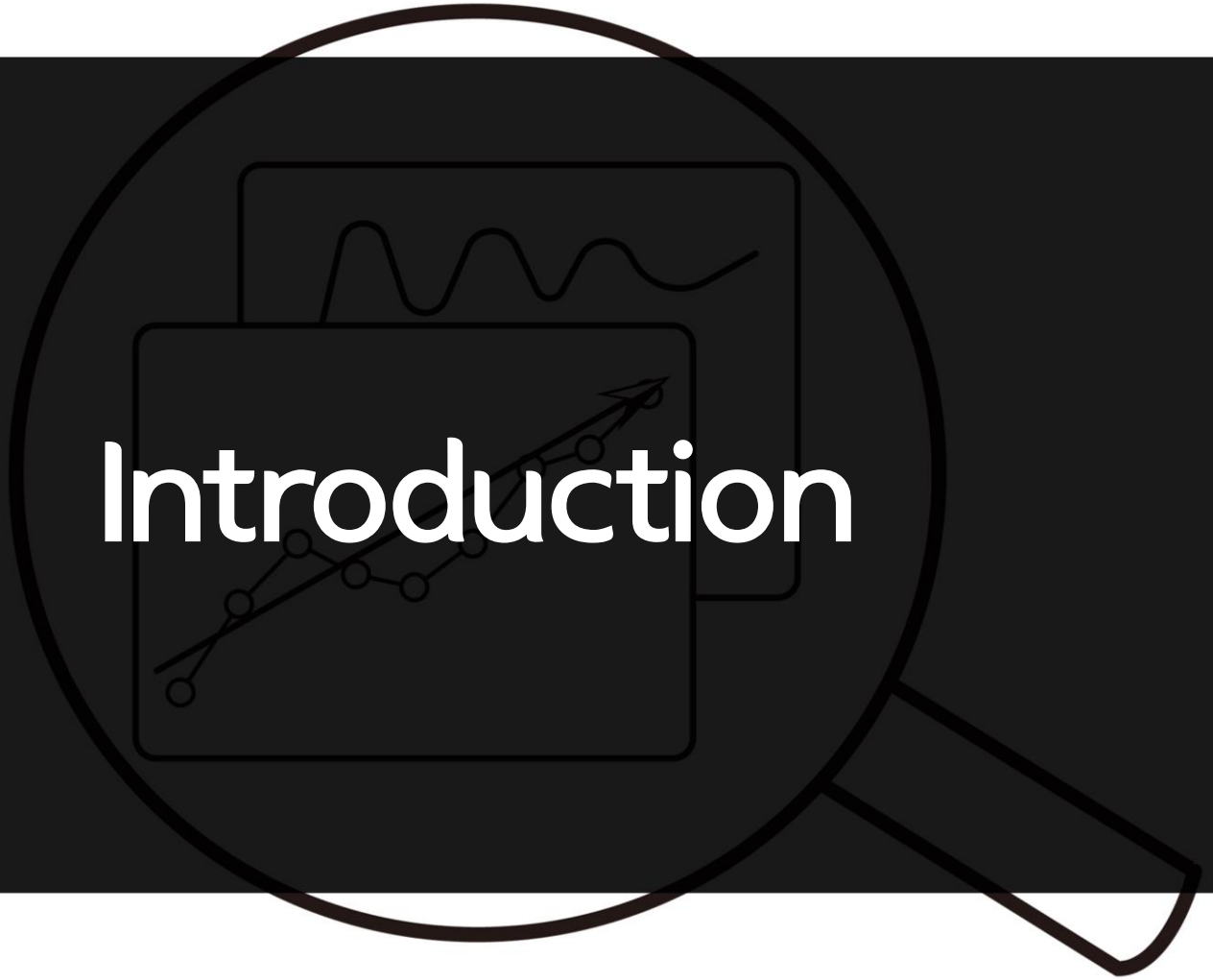
Moldy Manual for Beginners

2020-06-27 Intensive Seminar

SDlab, Jungmin Kim

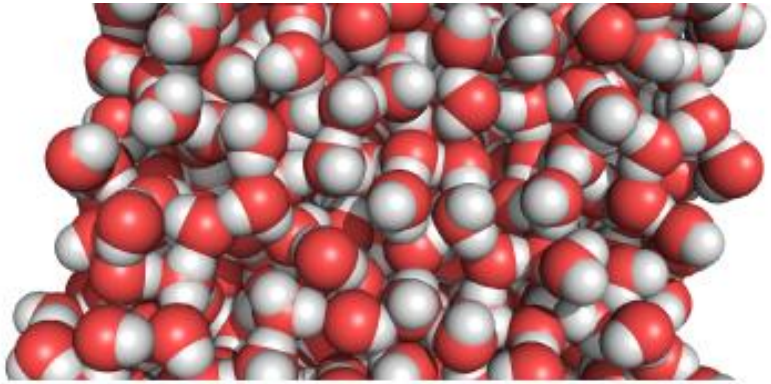


Introduction

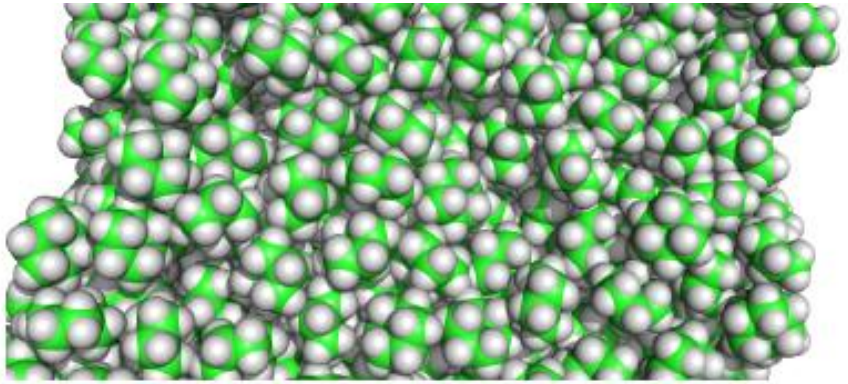


Introduction
Molecular dynamics

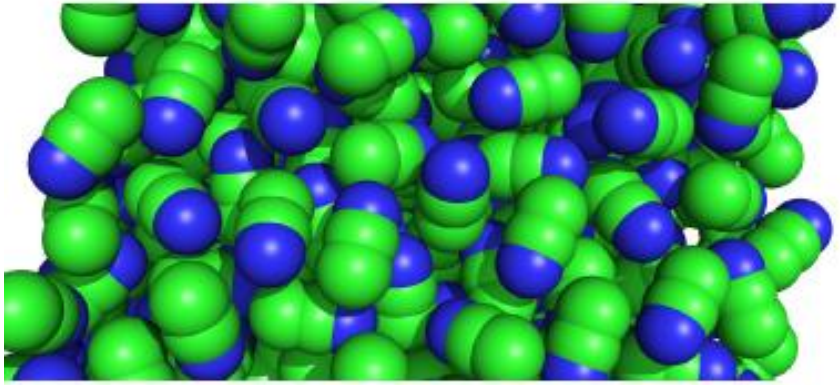
- Molecular dynamics (MD) is a **computer simulation** method for analyzing the **physical movements** of **molecules.**



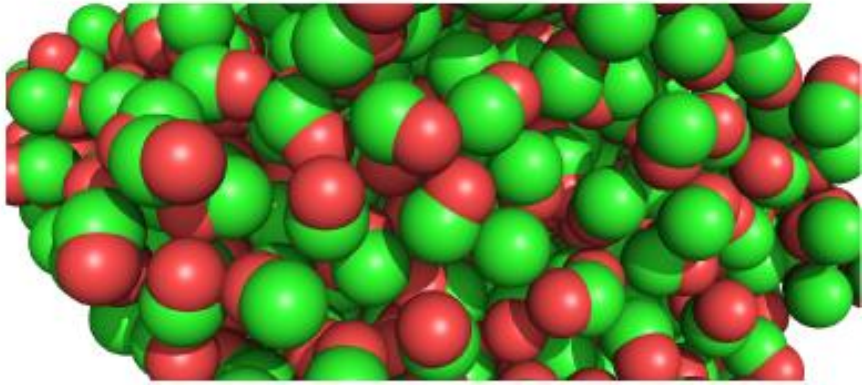
Water



Cyclohexane



Acetonitrile

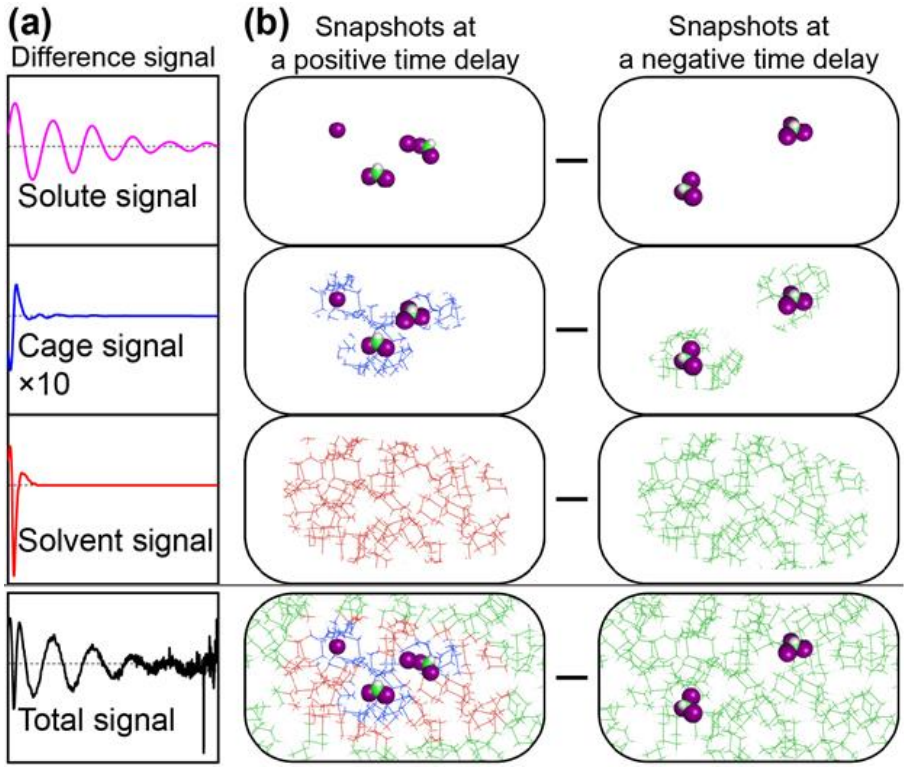


Methanol

https://en.wikipedia.org/wiki/Molecular_dynamics

Introduction

Why MD? 1. Cage signals



Debye equation

Molecular dynamics

Heating experiments

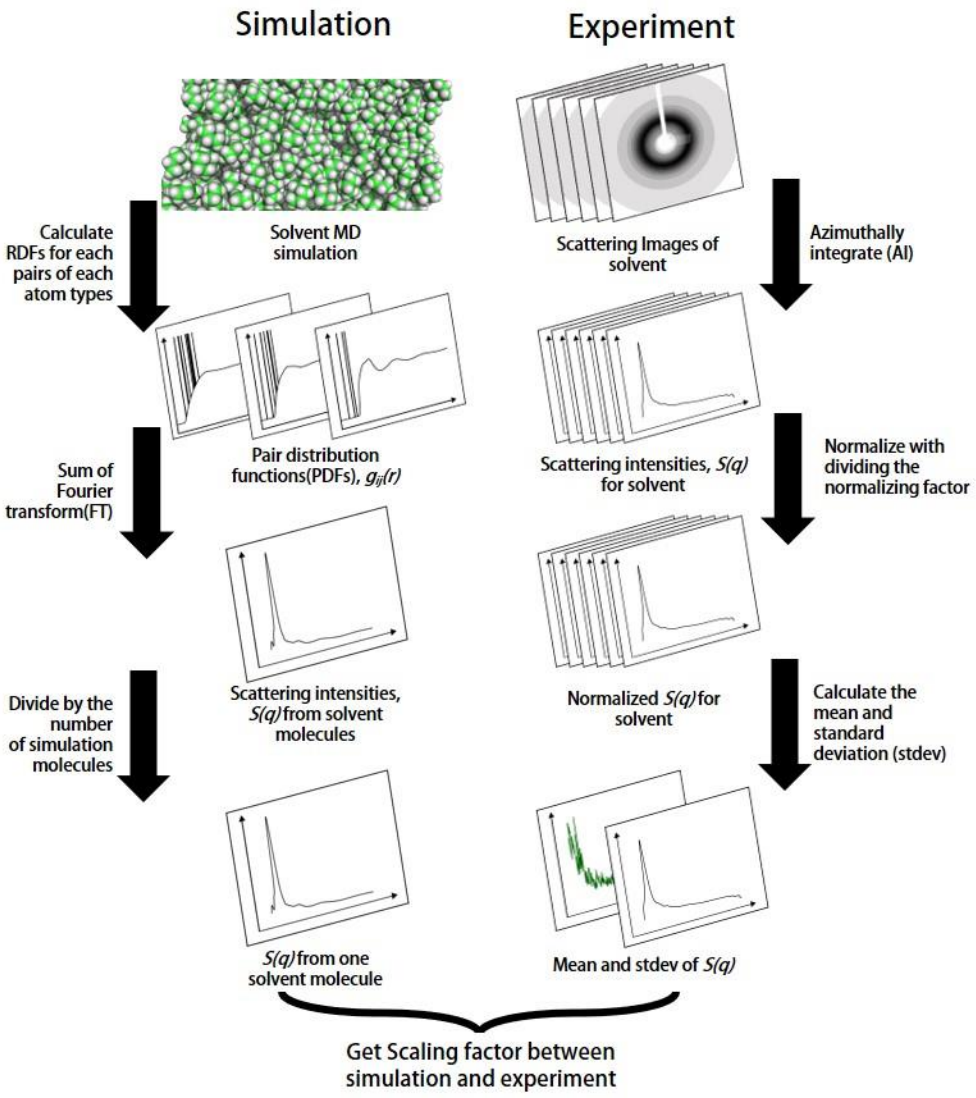
The TRXL data can be understood to consist of solute, solvent, and cage signals.

The cage signals can be obtained through MD simulations.

J. Synchrotron Rad. (2020). 27, 633–645

Introduction

Why MD? 2. Experimental data scaling



Experimental solvent static scattering data needs to be scaled in scientific units because it is a detector unit that has no scientific meaning.

The experimental data can be expressed in electron units (e.u.) per solvent molecule by scaling the experimental data with the solvent scattering data obtained from MD.

J. Synchrotron Rad. (2020). 27, 633–645

Introduction

Why MD? 3. Heating data scaling

$$\left.\frac{\partial S}{\partial T}\right|_{\rho} \Delta T(100\ \text{ps}) = \Delta S(100\ \text{ps})$$

$$\left.\frac{\partial S}{\partial \rho}\right|_{T} \Delta\rho(1\ \mu\text{s}) = \Delta S(1\ \mu\text{s}) - \frac{C_V}{C_P}\Delta S(100\ \text{ps})$$

MD set 1
(N =512, T=300 K, ρ=1 Kg $\cdot m^{-3}$)

MD set 2
(N =512, T=330 K, ρ=1 Kg $\cdot m^{-3}$)

MD set 3
(N =512, T=300 K, ρ=0.9 Kg $\cdot m^{-3}$)

Solvent heating experiments usually give better $\frac{\partial S}{\partial T}$ and $\frac{\partial S}{\partial \rho}$ than calculated from MD.

However, since it is difficult to measure ΔT in a heating experiment, the shape of the heating signal is obtained through solvent heating experiments and then scaled with MD results.

Get scattering differences, which are $\frac{\partial S}{\partial T}$ (e.u. / solvent molecule / K) and $\frac{\partial S}{\partial \rho}$ (e.u./solvent molecule / Kg / m^{-3})

Phys. Chem. Chem. Phys., 2013, 15, 15003–15016

Introduction
Ensemble

- Microcanonical ensemble
  number of molecules (N), volume (V),
  and energy(E)

- Canonical ensemble ← What we need !
  N, V, T(temperature)

- Grand canonical ensemble
  μ (chemical potential), V, T

Introduction

Molecular dynamics

The molecules are allowed to **interact** for a fixed period of time, giving a view of the dynamic **"evolution" of the system.**

The trajectories of molecules are determined by numerically solving **Newton's equations** of motion for a system of interacting particles, where forces between the particles and their potential energies are often calculated using interatomic potentials or molecular mechanics **force fields.**

https://en.wikipedia.org/wiki/Molecular_dynamics

Moldy is a computer program for performing molecular dynamics simulations of condensed matter. It can handle any assembly of rigid polyatomic molecules, atoms or ions and any mixture thereof. It uses the 'link cell' method to calculate short-range forces and the Ewald sum technique to handle long-range electrostatic forces. Simulations may be performed either in the usual NVE ensemble or in NVT, $N\sigma H$ or $N\sigma T$ ensembles using Nosé-Hoover thermostat and Parrinello and Rahman constant-stress methods. As the MD cell need not be cubic, the program is equally suitable for simulations of solids and liquids.

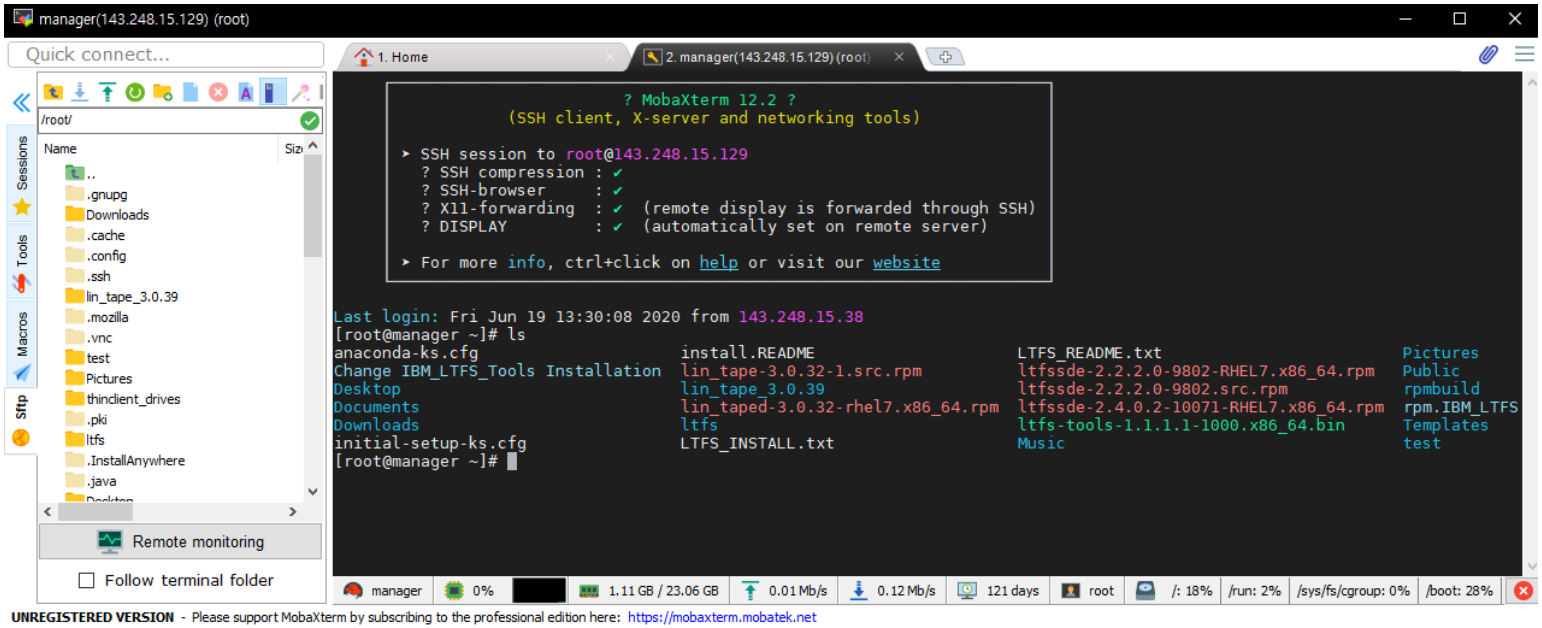
Most existing MD programs are limited in their capabilities, for example to one kind of potential function, or molecular symmetry, or to some restricted number of molecules. *Moldy* is (as far as possible) free from such arbitrary constraints. The system is specified at the beginning of each run and its size is only limited by the amount of memory available to the program: if a system is too large to handle, the solution is to buy some more memory. The system may contain a mixture of an arbitrary number of molecular species, each with an arbitrary number of atoms and an arbitrary number of molecules of each. Molecules or ions may be monatomic or polyatomic, linear or three dimensional in any combination. The potential functions may be of the Lennard-Jones, Buckingham (including Born-Mayer) or MCY types, and other potential types may be easily added. Such flexibility is possible because *Moldy* is written in the 'C' language which permits dynamic memory allocation.

Moldy is written to be highly portable and has been tested on a wide range of computers and operating systems, including VAX/VMS, MS-DOS and Unix[1] (both BSD and system V varieties). It should be straightforward to move it to any other machine with a good 'C' compiler.

To be of real use a simulation program must run efficiently on modern high-speed computers, which are increasingly of vector or parallel architectures. *Moldy* is written so as to be highly vectorizable and has been tested on a range of vector machines from manufacturers including Cray, Convex, Stardent and Alliant. On the cray XMP-48 its performance can exceed 100 MFlop/sec (on a suitably large system). *Moldy* is also able to run on a parallel computer of either shared or distributed memory architectures, and has been tested on multiprocessors from Stardent, Convex, Cray Research, SGI and IBM SP1 and Cray Research T3D massively parallel machines.

Introduction

Linux system



Linux is open source Unix-like operating system (OS) based on the Linux kernel.

There are over 50 Linux servers in the lab.

These Linux servers are available through the SSH (Secure Shell Protocol).

The SSH application mainly used in our lab is mobaXterm.

Introduction

Linux command

```
[root@manager ~]# ls
anaconda-ks.cfg                  install.README                  LTFS_README.txt                          Pictures
Change IBM_LTFS_Tools Installation  lin_tape-3.0.32-1.src.rpm    ltfssde-2.2.2.0-9802-RHEL7.x86_64.rpm    Public
Desktop                          lin_tape_3.0.39                 ltfssde-2.2.2.0-9802.src.rpm             rpmbuild
Documents                        lin_taped-3.0.32-rhel7.x86_64.rpm  ltfssde-2.4.0.2-10071-RHEL7.x86_64.rpm  rpm.IBM_LTFS
Downloads                        ltfs                            ltfs-tools-1.1.1.1-1000.x86_64.bin       Templates
initial-setup-ks.cfg             LTFS_INSTALL.txt                Music                                    test
```

ls : print current directory contents

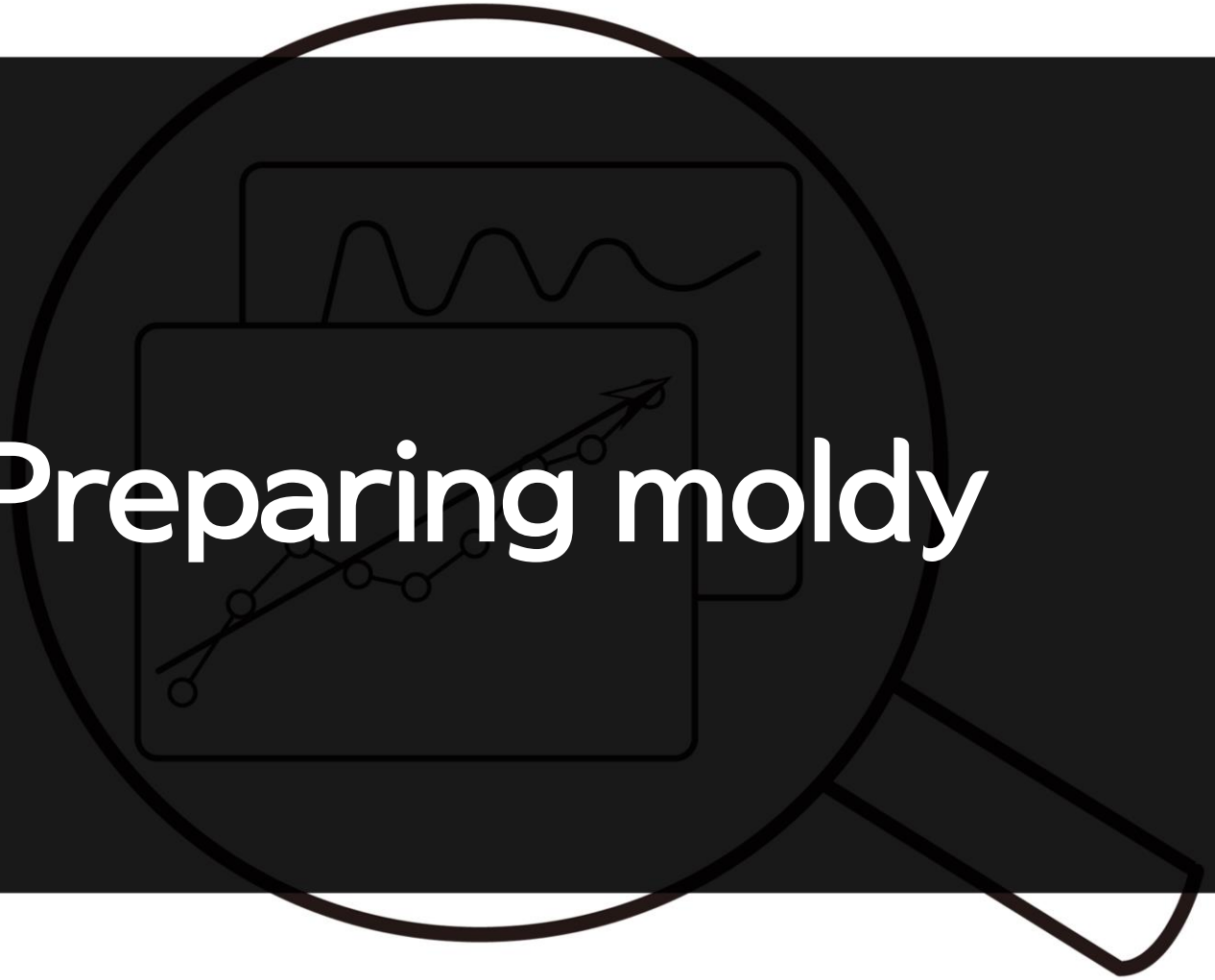
```
[root@manager ~]# ls
anaconda-ks.cfg                  install.README
Change IBM_LTFS_Tools Installation  lin_tape-3.0.32-1.src.rpm
Desktop                          lin_tape_3.0.39
Documents                        lin_taped-3.0.32-rhel7.x86_64.rpm
Downloads                        ltfs
initial-setup-ks.cfg             LTFS_INSTALL.txt
[root@manager ~]# cd ltfs/
[root@manager ltfs]# ls
ltfssde-2.2.2.0  ltfssde_2220_9802.tar.gz
```

cd (target directory): change directory to (target directory)

vi (target text file): edit text file

I : insert mode
esc : mode exit
:wq = Write the file and quit vi
:q = Quit vi, unless the buffer has been changed.
:q! =Quit vi without writing.

https://cs.stanford.edu/people/miles/vi.html



Preparing moldy

Preparing moldy

Linux command

What we need for moldy?

→Moldy execution file

→The control file

→The system specification file

→The restart file

Preparing moldy

Moldy execution file

→ Moldy execution file

Executable compiled files from source code based on c

→ PATH setting

PATH is an environmental variable in Linux operating systems that tells the shell which directories to search for executable in response to commands issued by a user.

Simply, uploading the moldy execution file to path (/usr/bin) makes it easy to execute.

→ So, how to MD?

Command : moldy < control file > output file
Example : moldy < pre_therm_Azo_c_S0 > pre_therm_Azo_c_S0.out
with the control, system specification files and restart file (optionally, automatically generated from previous moldy run) in current directory.

Preparing moldy

The control file

Open

100ps_Azo_c_S0
~/molecule_dynamics/Azo_DiBrAzo/Azo_c_S0

```
# Input system files
sys-spec-file=Azo_c_S0.in
restart-file=50ps_Azo_c_S0.mds

# Characteristics
density=0.779
temperature=300

# timesteps
const-temp=1
nsteps=100000
step=0.0005

# when to calculate averages and print
print-interval=100
roll-interval=100
average-interval=100
begin-average=0

# values
#alpha=-1
subcell=2.5
strict-cutoff=1
cutoff=11.86
surface-dipole=0

# rdf
begin-rdf=0
rdf-limit=20
rdf-out=400
nbins=800

# dump files
dump-file=100ps_Azo_c_S0.dump.%d
dump-level=1
dump-interval=20
ndumps=1000

save-file=100ps_Azo_c_S0.mds
```

sys-spec-file : Name of system specification file
restart-file : Name of restart file (Optional)

density : solvent density
temperature : solvent temperature

dump-file : Name of dump file
save-file : Name of save-file

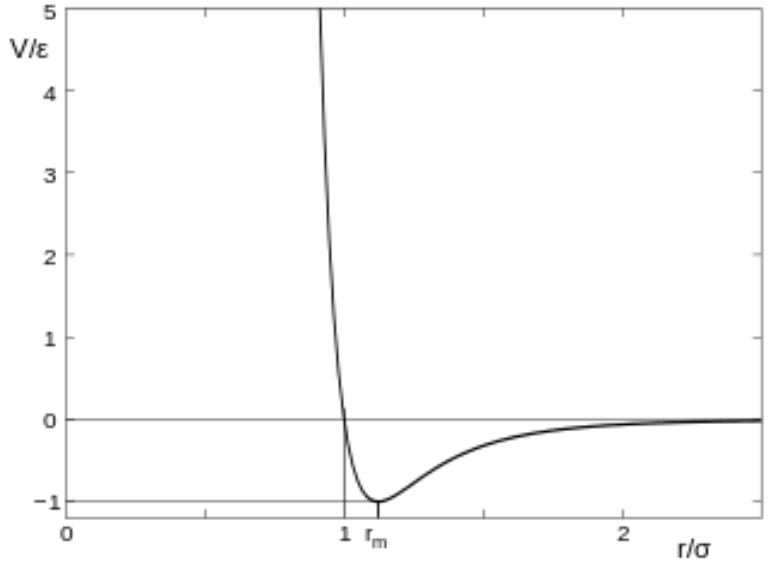
Preparing moldy

The system specification file

```
#simulation of HgI2 in methanol     (remark)
Methanol 512  (Name of molecules, Number of molecules)
1       0       0       0       15.999  -0.683 O (X position // Y posi. // Z posi. // weigth // charge // name of atom for 1st atom type)
2       -1.337  0       0.447   12.011  0.145   C
3       0       0       -0.945  1.008   0.418   H_O
4       -1.356  0       1.537   1.008   0.040   H_C
4       -1.845  -0.89   0.075 (When is empty parameter, is same lately atom )
4       -1.845  0.89    0.075
HgI2 1   (Name of molecules, Number of molecules)
5       0.000000000  0.000000000  0.000000000  200.59  1.054   Hg
6       0.000000000  0.000000000  2.7433710    126.90040       -0.527  I_1
7       0.000000000  0.000000000  -2.7433710   126.90040       -0.527  I_2
end
Lennard-Jones   (Name of molecules, Number of molecules)
1 1     2.845   3.210  (1st atom type //  -1st atom type // 4*ε (KJ/mol) // σ (Å))
2 2     1.105   3.500
4 4     0.502   2.500
5 5     24.943  2.969
6 6     5.644   3.610
7 7     5.644   3.610
1 2     1.773   3.355
1 4     1.195   2.855
1 5     8.424   3.090
1 6     4.007   3.410
1 7     4.007   3.410
2 4     0.745   3.000
2 5     5.249   3.235
2 6     2.497   3.555
2 7     2.497   3.555
4 5     3.539   2.735
4 6     1.683   3.055
4 7     1.683   3.055
5 6     11.865  3.290
5 7     11.865  3.290
6 7     5.644   3.610
end
```

Preparing moldy

Lennard-jones potential



The Lennard-Jones potential (also termed the L-J potential, 6-12 potential, or 12-6 potential) is a mathematically simple model that approximates the intermolecular potential energy between a pair of molecules.

$$r_m = 2^{1/6}\sigma \approx 1.122\sigma.$$

$$V_{LJ} = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right] = \varepsilon\left[\left(\frac{r_m}{r}\right)^{12} - 2\left(\frac{r_m}{r}\right)^{6}\right]$$

writing the potential parameters to the output. **N.B.** These may vary slightly from the original input parameters depending on the exact value of the time-unit used in the control file to specify energy units other than kJ mol^{-1}. See section 3.2.1. The parameters for the species being introduced can be read in from a potential parameter file (using option -v), otherwise the parameters between the dopant and

Lennard-Jones $\phi(r) = \epsilon((\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^{6})$ 4*ε (KJ/mol) || σ (Å))

Moldy User's Manual

en.wikipedia.org/wiki/Lennard-Jones_potential

Preparing moldy

The restart file

Pre_therm nsteps=5000~1000000 step= 0.0000001 ~ 0.0000000005 Time evolution = 0.0005 ps	**therm1** nsteps= 10000 step= 0.000001 Time evolution = 0.01 ps	**therm2** nsteps= 10000 step= 0.000001 Time evolution = 0.01 ps	**therm3** nsteps= 20000 step= 0.0005 Time evolution = 10 ps	**therm4** nsteps= 20000 step= 0.0005 Time evolution = 10 ps
50 ps nsteps=100000 step= 0.0005 Time evolution = 50 ps	**100 ps** nsteps=100000 step= 0.0005 Time evolution = 50 ps	**200 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**300 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**400 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**500 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps
600 ps nsteps=200000 step= 0.0005 Time evolution = 100 ps	**700 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**800 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**900 ps** nsteps=200000 step= 0.0005 Time evolution = 100 ps	**1 ns** nsteps=200000 step= 0.0005 Time evolution = 100 ps

The purpose of pre_therm and therms phase: When MD starts, it is arranged in a lattice shape and has a high potential. Therefore, it is a process of releasing the potential in a relatively short step.

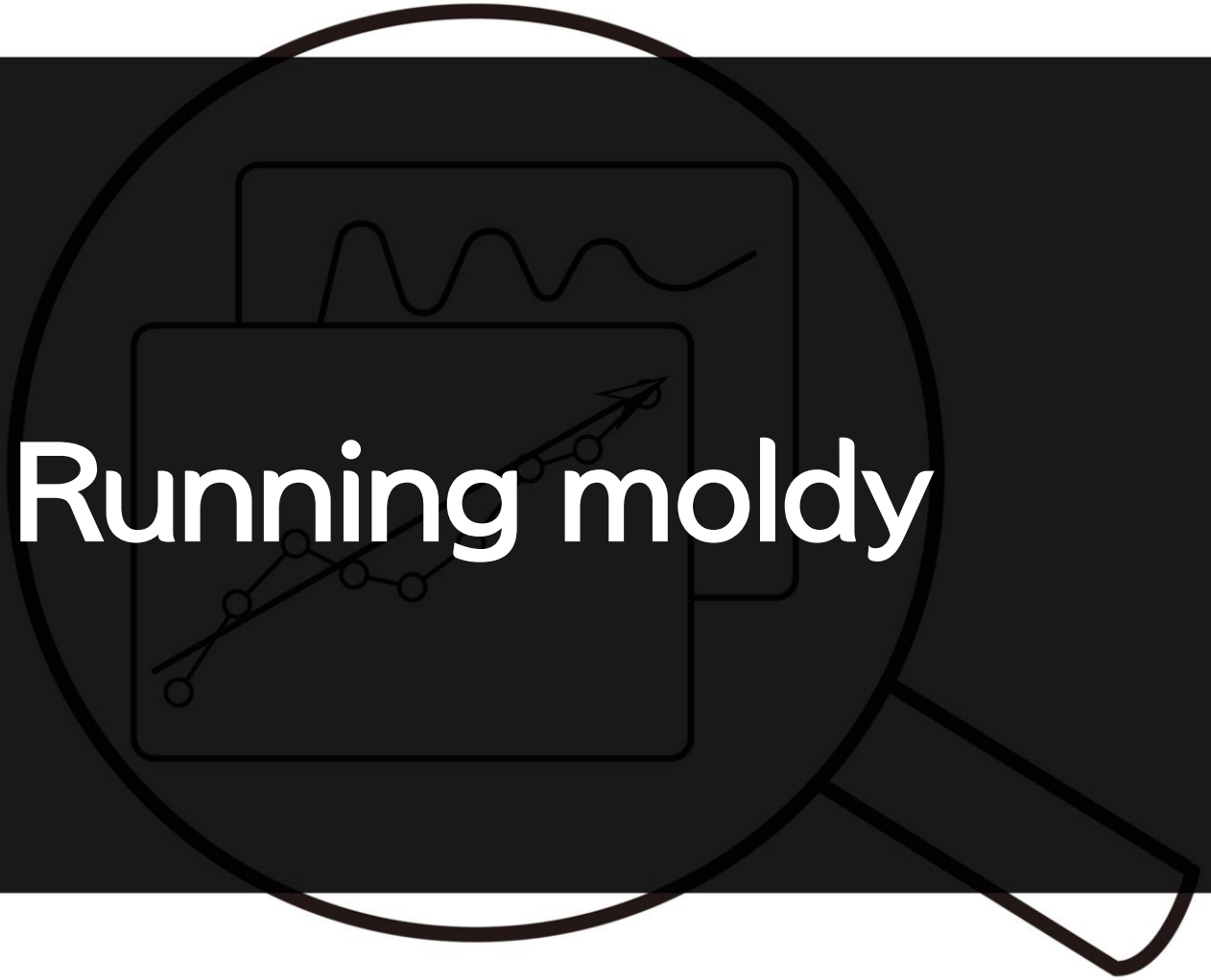
The most errors occur during the pre_therm phase. This can be solved by taking a short time step of the pre_therm step because the potential is too high and the next snapshot is an abnormal situation.

MD is performed from 50 ps to 1 ns, snapshots are taken, and results are collected.

For this sequential process, it is necessary to restart from the previous MD result, and a restart file is required.

Moldy User's Manual
en.wikipedia.org/wiki/Lennard-Jones_potential

Running moldy



Why you should automate the MD process as much as possible?

→In most cases, MD is not a strong point in our study

→However, MD takes a long time to calculate and has a lot to do.

→There is a need to achieve results with minimal effort.

Create file.cpp

→ C based code which output the 10 control files and 1 execution_script file

→ By execute the execution_script file, conduct MD step by step from pre_therm to 1 ns output

Running moldy

Modify the create file.cpp

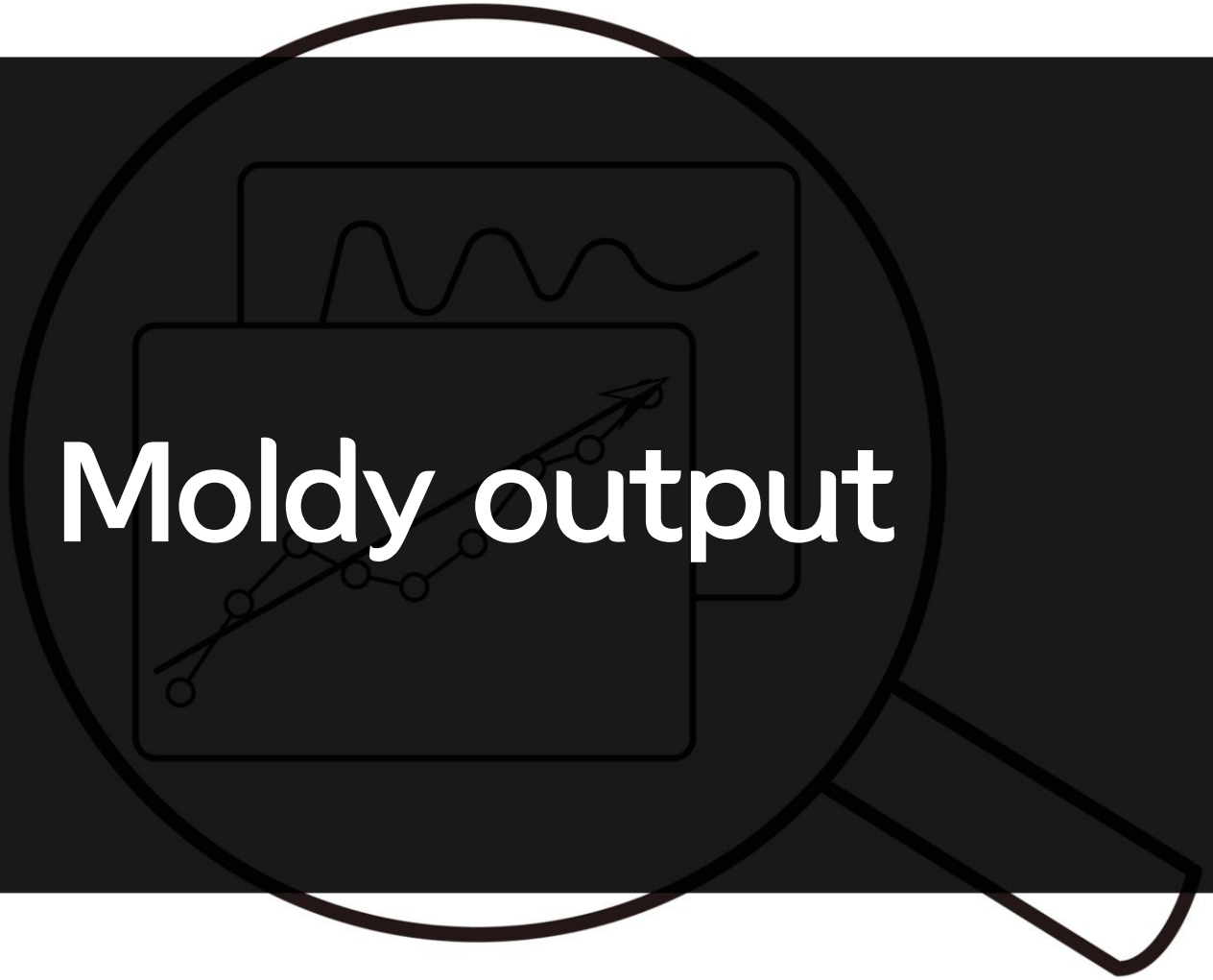
```
int main() {
        ofstream fout;

        string molecule_name = "Azo_c_S0";    (Change to target sample name)
        string input_name = "Azo_c_S0.in";    (Name of a control file)
        ----------------------  ... ----------------------
                fout << "# Characteristics" << endl
                        << "density=" << "0.779" << endl     (Change to target density)
                        << "temperature=" << "300" << endl    (Change to target temperature)
                        << endl;

                fout << "# timesteps" << endl;
                if(i==0) {
                        // pre_therm
                        fout << "nsteps=" << "1000000" << endl       (When get error, increase nsteps values)
                                << "step=" << "0.0000000005" << endl;    (When get error, decrease step values)
----------------------  ... ----------------------
// Saves execution record on ah_debug
        fout.open("execution_script");
        for(int i=0; i<16; i++) {
                fout << "date >> ah_debug" << endl
                        << "echo Start : " << file_name[i] << " on `hostname` >> ah_debug" << endl
                        << "moldy < " << (file_name[i]+"_"+molecule_name) << " > "
<< (file_name[i]+"_"+molecule_name+".out") << endl
                        << "chmod 777 " << (file_name[i]+"_"+molecule_name+".out") << endl
                        << "echo Finish : " << file_name[i] << " >> ah_debug" << endl
                        << "date >> ah_debug" << endl
                        << "echo ================================== >> ah_debug" << endl;
        }
        fout.close();
```

Running moldy

Run moldy with the create file.cpp and execution script

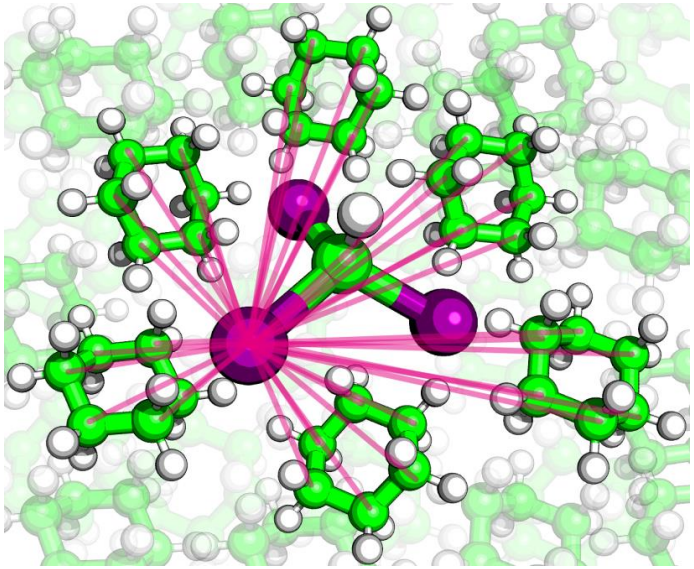
```
g++ create file.cpp
./a.out
chmod 755 execution_script
./execution_script &
disown -a
```

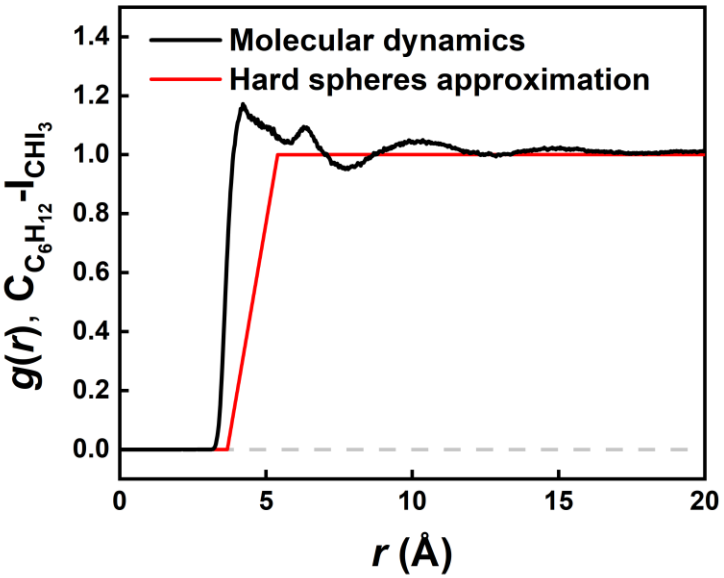
Moldy output



Moldy output

Rdf, pdf





Radial distribution function (RDF)
Pair distribution function (PDF)

PDF – sine Fourier transform –> S(q)

Moldy output

Rdf, pdf

$$S_{cage}(q) = \sum_{i,j} f_i(q) f_j(q) \frac{N_i N_j}{V} 4\pi \left(\int_0^{R_{box}} g_{i,j}(r) \frac{sin(qr)}{qr} r^2 dr \right)$$

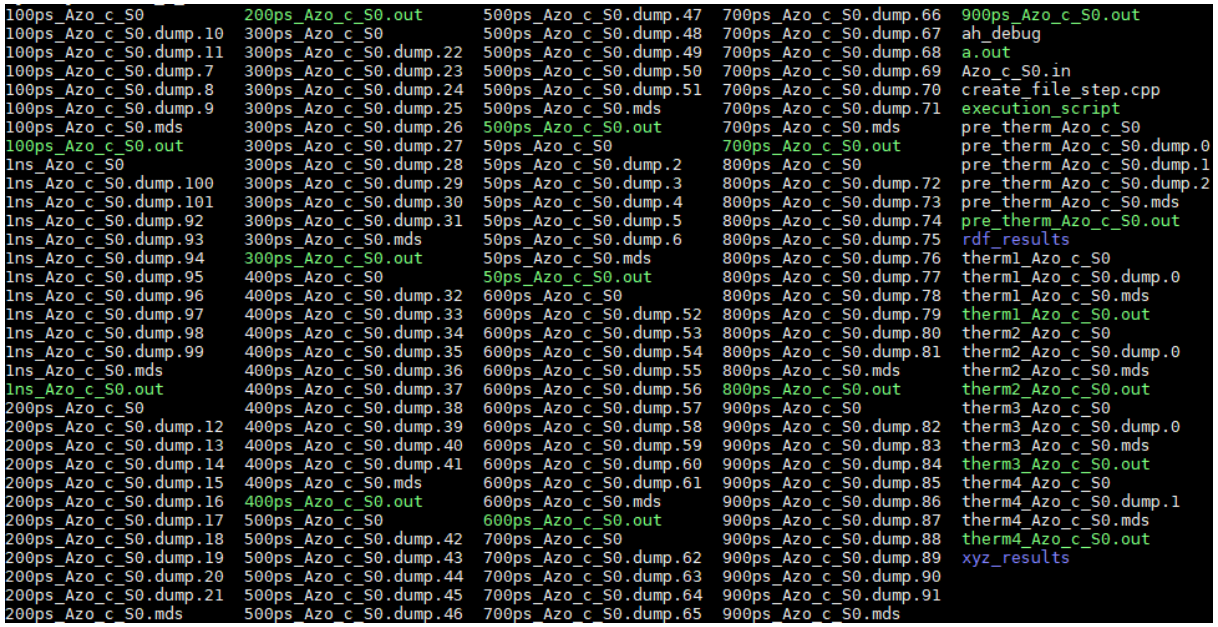
where i and j are indices of atom types in solute and solvent molecules, respectively, f_i and f_j are atomic form factors for atom−types $_i$ and $_j$, N_i and N_j are the numbers of atoms for atom−type $_i$ and $_j$, and V is the box volume for MD simulations.

Every pair between types of atom $_i$ in solute and types of atom $_j$ in solvent.

J. Synchrotron Rad. (2020). 27, 633–645

Moldy output

*.out, *.mds



[snapshotname].out → [rdfname].rdf

plotrdf –a –np [snapshotname].out > [rdfname].rdf

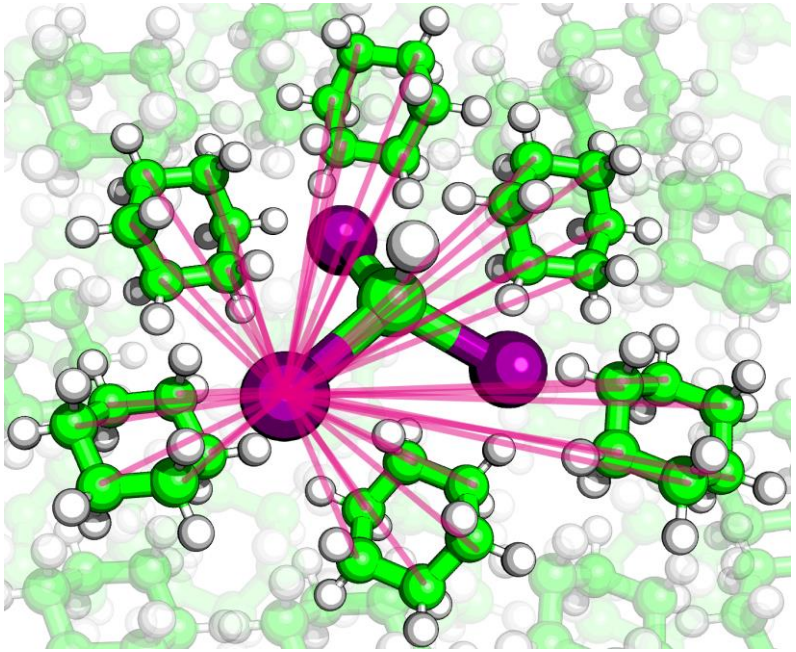
[snapshotname].mds → [xyzname].xyz

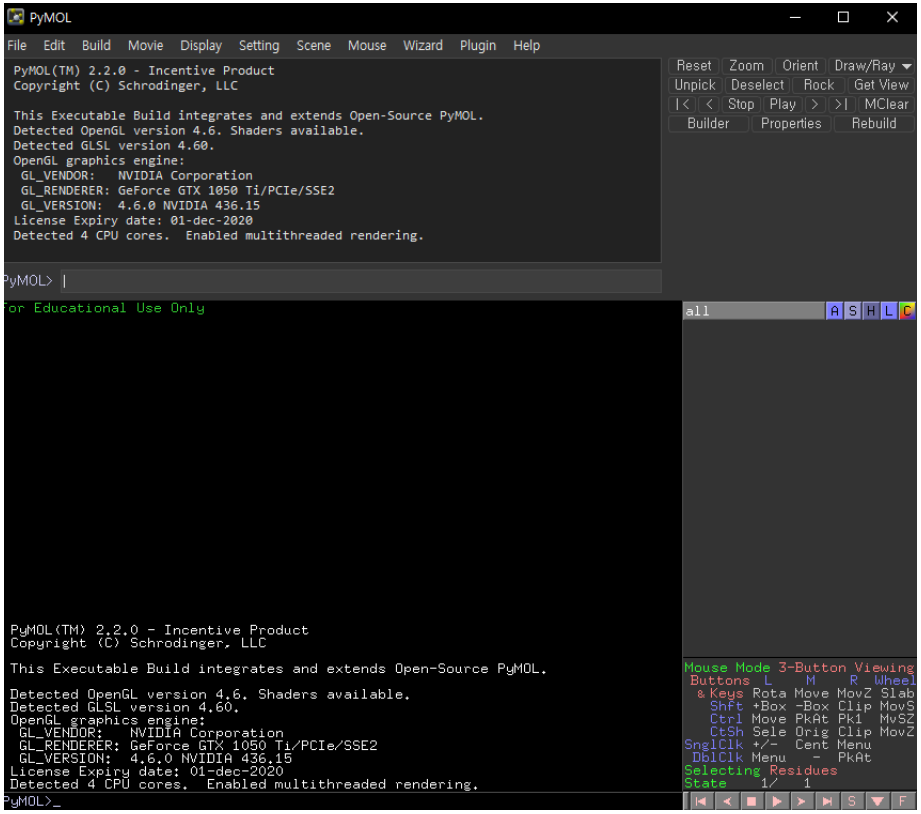
mdshak –r [snapshotname].mds –f xyz –o [xyzname].xyz

The xyz file can be viewed through programs such as pymol or vmd.

Moldy output

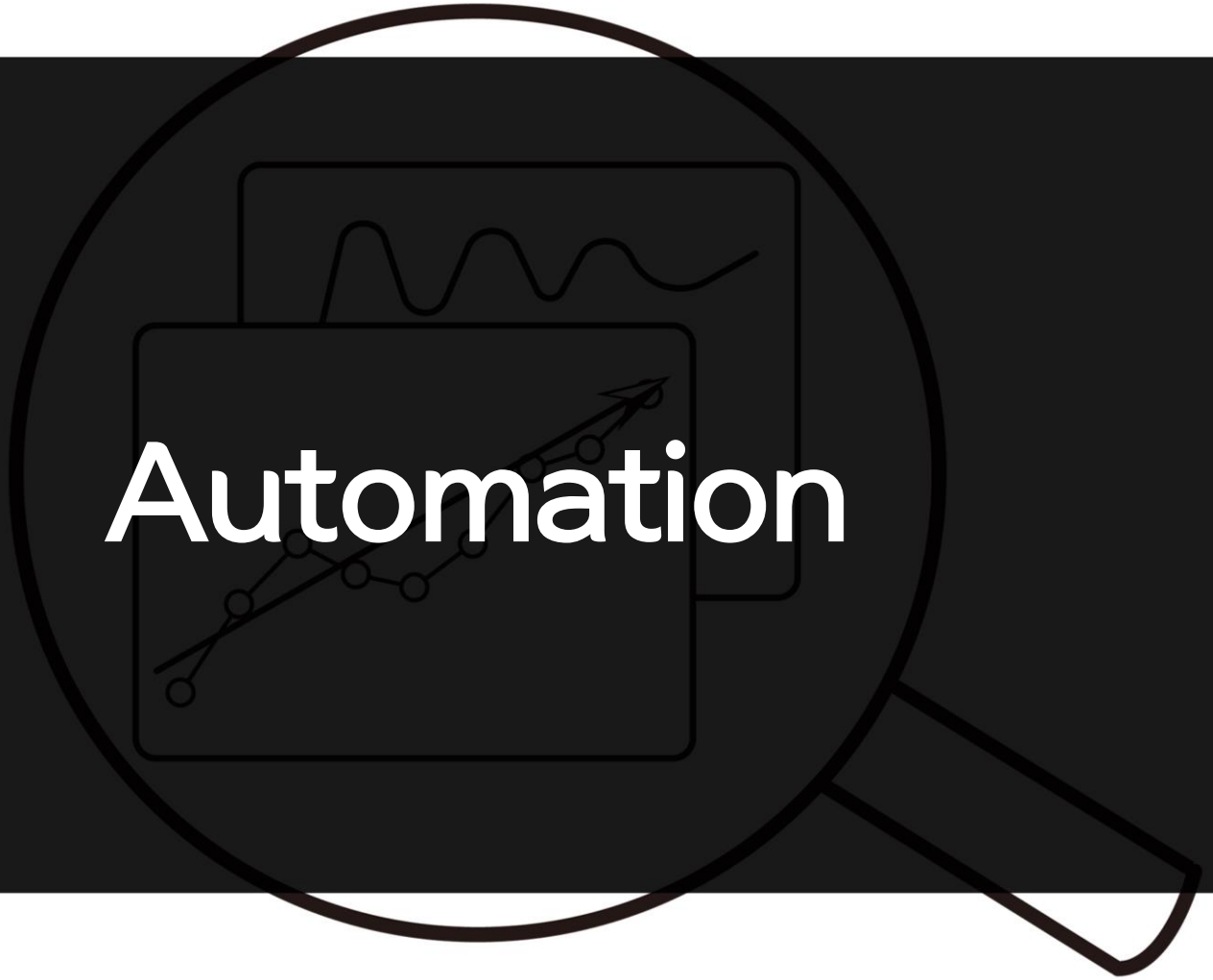
pymol





Pymol is a python based molecular structure viewer.
The easiest and most beautiful tool to draw molecules

Automation



Automation

motivation

16*5 = 80 times md running *!!*

The need for automation

Automation

MD_inputmaker.m

```
numofsolv=512;

solvent_name={
   'Cyclohexane'
    };

solvent_list={
   'input/Cyclohexane_md.xyz'
   };

coord_list={
  'input/Os3CO12_DFT_md.xyz'
  'input/Os3CO11axi_DFT_md.xyz'
};

solute_names={
   'Os3CO12'
   'Os3CO11_axi'
}
charge{1}=[
   -1.33468
   -1.33435
   -1.33435
...
   -0.42897
   -0.42892
   ];
charge{2}=[
   -1.33468
   -1.33435
   -1.33435
...
   -0.42897
   -0.42892
   ];

label_change{1}=[
   "Hsolvent"
   "Csolvent"
   ];

label_change{2}=[
   "H_99"
   "C_99"
];
```

Cyclohexane_md.xyz

1	Csolvent	0.00000000	0.00000000	0.00000000
2	Csolvent	0.00000000	0.00000000	1.52600000
3	Csolvent	1.43847092	0.00000000	2.03538927
4	Csolvent	2.15872810	1.24575236	1.52744571
5	Csolvent	2.16026142	1.24663720	0.00144674
6	Csolvent	0.71923546	1.24575236	-0.50938927
7	Hsolvent	0.71923546	1.24575236	-1.59938927
8	Hsolvent	0.20586075	2.13620750	-0.14657244
9	Hsolvent	2.67400181	0.35681454	-0.36240273
10	Hsolvent	2.67473084	2.13646031	-0.36137008
11	Hsolvent	1.64535391	2.13620750	1.89026327
12	Hsolvent	3.18620733	1.24575236	1.89129519
13	Hsolvent	1.95294034	-0.88982311	1.67257244
14	Hsolvent	1.43847092	-0.00000000	3.12538927
15	Hsolvent	-0.51373961	0.88982311	1.88984948
16	Hsolvent	-0.51373961	-0.88982311	1.88984948
17	Hsolvent	0.51373961	-0.88982311	-0.36384948
18	Hsolvent	-1.02747923	-0.00000000	-0.36384948

Os3CO12_DFT_md.xyz

1	Os_1	-0.041000	-1.693000	0.000000
2	Os_2	1.478000	0.810000	0.000000
3	Os_3	-1.438000	0.880000	0.000000
4	C_1	-0.040000	-1.669000	1.947000
5	C_2	-0.040000	-1.669000	-1.947000
6	C_3	-1.543000	-2.861000	0.000000
7	C_4	1.405000	-2.931000	0.000000
8	C_5	1.463000	0.797000	1.947000
9	C_6	1.463000	0.797000	-1.947000
10	C_7	3.246000	0.101000	0.000000
11	C_8	1.810000	2.684000	0.000000
12	C_9	-1.423000	0.867000	1.947000
13	C_10	-1.423000	0.867000	-1.947000
14	C_11	-1.679000	2.768000	0.000000
15	C_12	-3.238000	0.258000	0.000000
16	O_1	-0.043000	-1.760000	3.081000
17	O_2	-0.043000	-1.760000	-3.081000
18	O_3	-2.431000	-3.578000	0.000000
19	O_4	2.258000	-3.688000	0.000000
20	O_5	1.549000	0.842000	3.080000
21	O_6	1.549000	0.842000	-3.080000
22	O_7	4.312000	-0.305000	0.000000
23	O_8	2.026000	3.804000	0.000000
24	O_9	-1.506000	0.915000	3.081000
25	O_10	-1.506000	0.915000	-3.081000
26	O_11	-1.841000	3.897000	0.000000
27	O_12	-4.322000	-0.094000	0.000000

MD_inputmaker.m is a script written based on matlab.

Automation

constant.mat

변수 - constant

constant

1x1 struct 9개 필드 포함

필드	값
Xfactor	*106x14 double*
elements	*1x106 cell*
masses	*1x106 double*
ha	6.6261e-34
speed_l	299792000
eV	1.6020e-19
Hartree_to_J	4.3600e-18
Na	6.0221e+23
LJ	*106x2 double*

constant.LJ

	1	2
1	0.7364	2.5711
2	0.9372	2.1043
3	0.4184	2.1836
4	1.4226	2.4455
5	3.0125	3.6375
6	1.7573	3.4309
7	1.1548	3.2607
8	1.0042	3.1181
9	0.8368	2.9970
10	0.7029	2.8892
11	0.5021	2.6576
12	1.8577	2.6914
13	8.4517	4.0082
14	6.7279	3.8264
15	5.1045	3.6946
16	4.5857	3.5948
17	3.7991	3.5164
18	3.0962	3.4460
19	0.5858	3.3961
20	3.9832	3.0282
21	0.3180	2.9355

constant.mat is a MATLAB variable file commonly used in jkim code.
LJ Potential is written based on UFF (universal force field).

UFF, a Full Periodic Table Force Field for Molecular Mechanics and Molecular Dynamics Simulations

A. K. Rappé,* C. J. Casewit,‡ K. S. Colwell, W. A. Goddard III,# and W. M. Skiff†

Contribution from the Department of Chemistry, Colorado State University, Fort Collins, Colorado 80523. *Received March 23, 1992*

Abstract: A new molecular mechanics force field, the Universal force field (UFF), is described wherein the force field parameters are estimated using general rules based only on the element, its hybridization, and its connectivity. The force field functional forms, parameters, and generating formulas for the full periodic table are presented.

Table I. Atomic Data

	valence		nonbond			effective		valence		nonbond			effective
atom type	bond r_I[a]	angle θ_0[b]	distance x_I[a]	energy D_I[c]	scale ζ	charge Z_I*[d]	atom type	bond r_I[a]	angle θ_0[b]	distance x_I[a]	energy D_I[c]	scale ζ	charge Z_I*[d]
H_	0.354	180.0	2.886	0.044	12.0	0.712	Ru6+2	1.478	90.0	2.963	0.056	12.0	3.40
H_b	0.460	83.5	2.886	0.044	12.0	0.712	Rh6+3	1.332	90.0	2.929	0.053	12.0	3.508
He4+4	0.849	90.0	2.362	0.056	15.24	0.098	Pd4+2	1.338	90.0	2.899	0.048	12.0	3.21
Li	1.336	180.0	2.451	0.025	12.0	1.026	Ag1+1	1.386	180.0	3.148	0.036	12.0	1.956
Be3+2	1.074	109.47	2.745	0.085	12.0	1.565	Cd3+2	1.403	109.47	2.848	0.228	12.0	1.65
B_3	0.838	109.47	4.083	0.180	12.052	1.755	In3+3	1.459	109.47	4.463	0.599	11.0	2.07
B_2	0.828	120.0	4.083	0.180	12.052	1.755	Sn3	1.398	109.47	4.392	0.567	12.0	2.961
C_3	0.757	109.47	3.851	0.105	12.73	1.912	Sb3+3	1.407	91.6	4.420	0.449	13.0	2.704
C_R	0.729	120.0	3.851	0.105	12.73	1.912	Te3+2	1.386	90.25	4.470	0.398	14.0	2.882
C_2	0.732	120.0	3.851	0.105	12.73	1.912	I_	1.382	180.0	4.50	0.339	15.0	2.65
C_1	0.706	180.0	3.851	0.105	12.73	1.912	Xe4+4	1.267	90.0	4.404	0.332	12.0	0.556

$$x_{IJ} = \tfrac{1}{2}(x_I + x_J) \qquad D_{IJ} = (D_I D_J)^{1/2}$$

Cyclohexane−Benzene Mixtures: Thermodynamics and Structure from Atomistic Simulations

Giuseppe Milano*,†,‡ and Florian Müller-Plathe‡

Dipartimento di Chimica, Universita di Salerno, I-84081 Baronissi (SA), Italy, and International University of Bremen, School of Engineering and Science, D-28759 Bremen, Germany

Received: February 7, 2004; In Final Form: March 17, 2004

TABLE 2: Geometry and Force Field Parameters of Cyclohexane and Benzene[a]

parameter		C_6H_{12}	C_6H_6
Atomic Masses [u]			
H	M	1.00787	1.00787
C	M	12.01	12.01
Bond Length Constraints [nm]			
C−C		0.1526	0.1390
C−H		0.1090	0.1080
Angle Potential ϕ_0 [deg], k_ϕ [kJ/(mol rad^2)]			
C−C−C	ϕ_0	109.5	120
	K_ϕ	335	376.6
C−C−H	ϕ_0	109.5	120
	K_ϕ	420	418.8
H−C−H	ϕ_0	109.5	
	K_ϕ	290	
Torsions (Periodicity p =3), τ_0 [deg], k_τ [kJ/(mol rad^2)]			
C−C−C−C	τ_0	180	
	k_τ	10	
Harmonic Dihedrals δ_0 [deg], k_δ [kJ(mol rad^2)]			
C−C−C−C	δ_0		0
	k_δ		167.4
C_2−C_3−C_1−H [on C2]	δ_0		0
	k_δ		167.4
Lennard−Jones ϵ [kJ/mol], σ [nm]			
C	ϵ	0.299	0.294
	σ	0.328	0.335
H	ϵ	0.189	0.126
	σ	0.258	0.242

J. Phys. Chem. B 2004, 108, 7415−7423

Automation

MD_inputmaker.m

```
label_change{1}=[
    "Hsolvent"
    "Csolvent"
    ];

label_change{2}=[
    "H_99"
    "C_99"
];
```

Cyclohexane_md.xyz

1	Csolvent	0.00000000	0.00000000	0.00000000
2	Csolvent	0.00000000	0.00000000	1.52600000
3	Csolvent	1.43847092	0.00000000	2.03538927
4	Csolvent	2.15872810	1.24575236	1.52744571
5	Csolvent	2.16026142	1.24663720	0.00144674
6	Csolvent	0.71923546	1.24575236	-0.50938927
7	Hsolvent	0.71923546	1.24575236	-1.59938927
8	Hsolvent	0.20586075	2.13620750	-0.14657244
9	Hsolvent	2.67400181	0.35681454	-0.36240273
10	Hsolvent	2.67473084	2.13646031	-0.36137008
11	Hsolvent	1.64535391	2.13620750	1.89026327
12	Hsolvent	3.18620733	1.24575236	1.89129519
13	Hsolvent	1.95294034	-0.88982311	1.67257244
14	Hsolvent	1.43847092	-0.00000000	3.12538927
15	Hsolvent	-0.51373961	0.88982311	1.88984948
16	Hsolvent	-0.51373961	-0.88982311	1.88984948
17	Hsolvent	0.51373961	-0.88982311	-0.36384948
18	Hsolvent	-1.02747923	-0.00000000	-0.36384948

Os3CO12_DFT_md.xyz

1	Os_1 -0.041000 -1.693000 0.000000
2	Os_2 1.478000 0.810000 0.000000
3	Os_3 -1.438000 0.880000 0.000000
4	C_1 -0.040000 -1.669000 1.947000
5	C_2 -0.040000 -1.669000 -1.947000
6	C_3 -1.543000 -2.861000 0.000000
7	C_4 1.405000 -2.931000 0.000000
8	C_5 1.463000 0.797000 1.947000
9	C_6 1.463000 0.797000 -1.947000
10	C_7 3.246000 0.101000 0.000000
11	C_8 1.810000 2.684000 0.000000
12	C_9 -1.423000 0.867000 1.947000
13	C_10 -1.423000 0.867000 -1.947000
14	C_11 -1.679000 2.768000 0.000000
15	C_12 -3.238000 0.258000 0.000000
16	O_1 -0.043000 -1.760000 3.081000
17	O_2 -0.043000 -1.760000 -3.081000
18	O_3 -2.431000 -3.578000 0.000000
19	O_4 2.258000 -3.688000 0.000000
20	O_5 1.549000 0.842000 3.080000
21	O_6 1.549000 0.842000 -3.080000
22	O_7 4.312000 -0.305000 0.000000
23	O_8 2.026000 3.804000 0.000000
24	O_9 -1.506000 0.915000 3.081000
25	O_10 -1.506000 0.915000 -3.081000
26	O_11 -1.841000 3.897000 0.000000
27	O_12 -4.322000 -0.094000 0.000000

변수 - constant

constant

1x1 struct 9개 필드 포함

필드	값
Xfactor	*106x14 double*
elements	*1x106 cell*
masses	*1x106 double*
ha	6.6261e-34
speed_l	299792000
eV	1.6020e-19
Hartree_to_J	4.3600e-18
Na	6.0221e+23
LJ	*106x2 double*

constant.LJ

	1	2
1	0.7364	2.5711
2	0.9372	2.1043
3	0.4184	2.1836
4	1.4226	2.4455
5	3.0125	3.6375
6	1.7573	3.4309
7	1.1548	3.2607
8	1.0042	3.1181
9	0.8368	2.9970
10	0.7029	2.8892
11	0.5021	2.6576
12	1.8577	2.6914
13	8.4517	4.0082
14	6.7279	3.8264
15	5.1045	3.6946
16	4.5857	3.5948
17	3.7991	3.5164
18	3.0962	3.4460
19	0.5858	3.3961
20	3.9832	3.0282
21	0.3180	2.9355

constant.elements

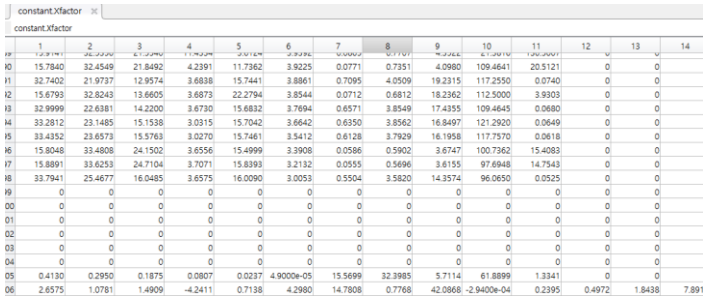
constant.elements

97	98	99	100	101	102	103	104	105	106
Bk	Cf	Es	Fm	Md	No	Lr	X	Hsolvent	Csolvent

constant.masses

constant.masses

97	98	99	100	101	102	103	104	105	106
247.1000	252.1000	252.1000	257.1000	256.1000	259.1000	260.1000	0	1.0079	12.0100



Automation

MD_inputmaker.m

CO	2020-06-08 오후 5:38	파일 폴더
Os3CO10_equequ	2020-06-08 오후 5:38	파일 폴더
Os3CO11_axi	2020-06-08 오후 5:38	파일 폴더
Os3CO11_axiaxi	2020-06-08 오후 5:38	파일 폴더
Os3CO11_axiequ	2020-06-08 오후 5:38	파일 폴더
Os3CO11_bri	2020-06-08 오후 5:38	파일 폴더
Os3CO11_equ	2020-06-08 오후 5:38	파일 폴더
Os3CO12	2020-06-08 오후 5:38	파일 폴더

HDD1 (G:) › SDlab › Program › MD_inputmaker_v2 › results › CO

이름	수정한 날짜	유형
CO.in	2020-06-08 오후 12:00	IN 파일

DD1 (G:) › SDlab › Program › MD_inputmaker_v2 › results › CO

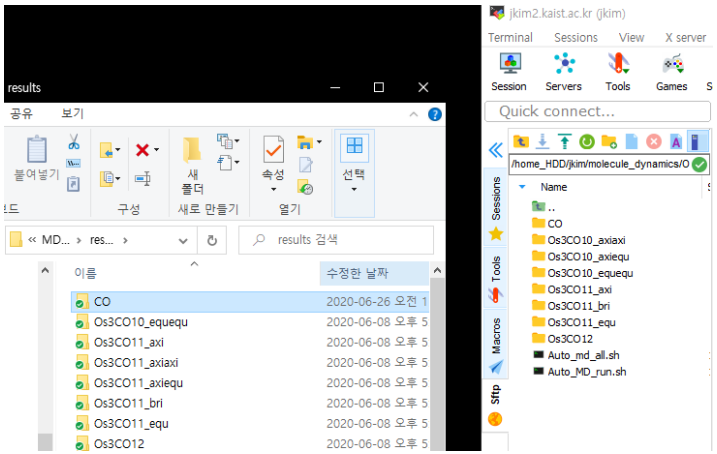
이름	수정한 날짜	유
CO.in	2020-06-08 오후 12:00	IN
create_file_step.cpp	2019-11-04 오후 11:02	CP

*G:₩SDlab₩Program₩MD_inputmaker_v2₩results₩CO₩cr

파일(F) 편집(E) 찾기(S) 보기(V) 인코딩(N) 언어(L)

create_file_step.cpp

```cpp
#include <iostream>
#include <fstream>
#include <string>
using namespace std;

int main() {
    ofstream fout;

    string molecule_name = "CO";
    string input_name = "CO.in";
```




```
[jkim@jkim2 Os3CO12_20200529]$ ls
CO  Os3CO10_axiaxi  Os3CO10_axiequ  Os3CO10_equequ  Os3CO11_axi  Os3CO11_bri  Os3CO11_equ  Os3CO12
```

Automation

Md_autorun_depth1.m

```
#!/bin/sh
echo Run the moldy autorun script.
echo Made by Jungmin Kim, 20200620
echo ----------------------------------------
for index in $(find $PWD -mindepth 1 -maxdepth 1 -type d)
do
    cd $index
        pwd
    rm -f MDBCK*
    rm -f ah_debug
    rm -f *dump*
    g++ *.cpp -o create_file_step.out
    ./create_file_step.out
    chmod 755 execution_script
    ./execution_script &
    echo ----------------------------------------
done
sleep 5
jobs
echo ----------------------------------------
echo The moldy autorun script done.
```

```
[jkim@jkim2 Os3CO12_20200529]$ ls
CO  Os3CO10_axiaxi  Os3CO10_axiequ  Os3CO10_equequ  Os3CO11_axi  Os3CO11_bri  Os3CO11_equ  Os3CO12
[jkim@jkim2 Os3CO12_20200529]$ md_autorun_depth1.sh
```

A few weeks later...

Automation

Md_autorun_depth1.m

top

Shift+t

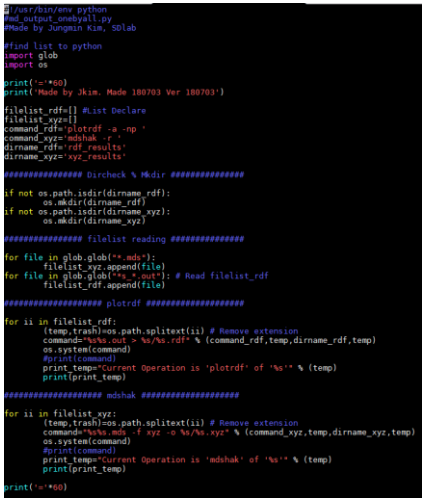
```
top - 12:57:47 up 47 days, 14:50,  9 users,  load average: 8.04, 8.06, 8.05
Tasks: 339 total,   1 running, 335 sleeping,   3 stopped,   0 zombie
%Cpu(s): 56.3 us,  0.8 sy,  0.0 ni, 43.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 73726448 total, 58344952 free, 11584440 used,  3797052 buff/cache
KiB Swap: 12189692 total,  6193100 free,  5996592 used. 61302724 avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 21917 horang2   20   0 5541628 542840 140836 S 100.0  0.7   1365:08 MATLAB
 21919 horang2   20   0 5541628 532960 140860 S 100.0  0.7   1365:07 MATLAB
 21926 horang2   20   0 5541628 516584 140808 S 100.0  0.7   1365:07 MATLAB
105824 root      20   0 15.216g 5.528g 155152 S 100.0  7.9  57:08.76 MATLAB
 21915 horang2   20   0 5541632 529180 140832 S  93.8  0.7   1365:07 MATLAB
 21922 horang2   20   0 5541628 531880 140840 S  93.8  0.7   1365:06 MATLAB
 21924 horang2   20   0 5541628 522912 140932 S  93.8  0.7   1365:07 MATLAB
 21928 horang2   20   0 5541628 517672 140820 S  93.8  0.7   1365:06 MATLAB
 21930 horang2   20   0 5541628 514884 140824 S  93.8  0.7   1365:07 MATLAB
 36617 jkim      20   0  158024   2424   1516 R   6.2  0.0   0:00.01 top
```

```
[jkim@jkim2 CO]$ ls
100ps_CO           1ns_CO.dump.96    200ps_CO.dump.20  300ps_CO.mds      500ps_CO          50ps_CO.dump.3   600ps_CO.dump.60  700ps_CO.mds      900ps_CO          create_file_step.cpp  therm2_CO.dump.0
100ps_CO.dump.10   1ns_CO.dump.97    200ps_CO.dump.21  300ps_CO.out      500ps_CO.dump.42  50ps_CO.dump.4   600ps_CO.dump.61  700ps_CO.out      900ps_CO.dump.82  create_file_step.out  therm2_CO.mds
100ps_CO.dump.11   1ns_CO.dump.98    200ps_CO.mds      400ps_CO          500ps_CO.dump.43  50ps_CO.dump.5   600ps_CO.mds      800ps_CO          900ps_CO.dump.83  execution_script      therm2_CO.out
100ps_CO.dump.7    1ns_CO.dump.99    200ps_CO.out      400ps_CO.dump.32  500ps_CO.dump.44  50ps_CO.dump.6   600ps_CO.out      800ps_CO.dump.72  900ps_CO.dump.84  pre_therm_CO          therm3_CO
100ps_CO.dump.8    1ns_CO.mds        300ps_CO          400ps_CO.dump.33  500ps_CO.dump.45  50ps_CO.mds      700ps_CO          800ps_CO.dump.73  900ps_CO.dump.85  pre_therm_CO.dump.0   therm3_CO.dump.0
100ps_CO.dump.9    1ns_CO.out        300ps_CO.dump.22  400ps_CO.dump.34  500ps_CO.dump.46  50ps_CO.out      700ps_CO.dump.62  800ps_CO.dump.74  900ps_CO.dump.86  pre_therm_CO.dump.1   therm3_CO.mds
100ps_CO.mds       200ps_CO          300ps_CO.dump.23  400ps_CO.dump.35  500ps_CO.dump.47  600ps_CO         700ps_CO.dump.63  800ps_CO.dump.75  900ps_CO.dump.87  pre_therm_CO.dump.2   therm3_CO.out
100ps_CO.out       200ps_CO.dump.12  300ps_CO.dump.24  400ps_CO.dump.36  500ps_CO.dump.48  600ps_CO.dump.52  700ps_CO.dump.64  800ps_CO.dump.76  900ps_CO.dump.88  pre_therm_CO.mds      therm4_CO
1ns_CO             200ps_CO.dump.13  300ps_CO.dump.25  400ps_CO.dump.37  500ps_CO.dump.49  600ps_CO.dump.53  700ps_CO.dump.65  800ps_CO.dump.77  900ps_CO.dump.89  pre_therm_CO.out      therm4_CO.dump.1
1ns_CO.dump.100    200ps_CO.dump.14  300ps_CO.dump.26  400ps_CO.dump.38  500ps_CO.dump.50  600ps_CO.dump.54  700ps_CO.dump.66  800ps_CO.dump.78  900ps_CO.dump.90  rdf_results           therm4_CO.mds
1ns_CO.dump.101    200ps_CO.dump.15  300ps_CO.dump.27  400ps_CO.dump.39  500ps_CO.dump.51  600ps_CO.dump.55  700ps_CO.dump.67  800ps_CO.dump.79  900ps_CO.dump.91  therm1_CO             therm4_CO.out
1ns_CO.dump.92     200ps_CO.dump.16  300ps_CO.dump.28  400ps_CO.dump.40  500ps_CO.mds      600ps_CO.dump.56  700ps_CO.dump.68  800ps_CO.dump.80  900ps_CO.mds      therm1_CO.dump.0      xyz_results
1ns_CO.dump.93     200ps_CO.dump.17  300ps_CO.dump.29  400ps_CO.dump.41  500ps_CO.out      600ps_CO.dump.57  700ps_CO.dump.69  800ps_CO.dump.81  900ps_CO.out      therm1_CO.mds
1ns_CO.dump.94     200ps_CO.dump.18  300ps_CO.dump.30  400ps_CO.mds      50ps_CO           600ps_CO.dump.58  700ps_CO.dump.70  800ps_CO.mds      ah_debug          therm1_CO.out
1ns_CO.dump.95     200ps_CO.dump.19  300ps_CO.dump.31  400ps_CO.out      50ps_CO.dump.2    600ps_CO.dump.59  700ps_CO.dump.71  800ps_CO.out      CO.in             therm2_CO
[jkim@jkim2 CO]$ cat ah_debug
```

Automation

Md_output_depth1.m

```
#!/bin/sh
echo Run the moldy rdf,xyz script.
echo Made by Jungmin Kim, 20200620

pwd_int=$PWD

for index in $(find $pwd_int -mindepth 1 -maxdepth 1 -type d)
do
    cd $index
    pwd
    md_output_onebyall.py > /dev/null &
    echo moldy in operation ...
    echo ------------------------------------------
done
```




```
[jkim@jkim2 Os3CO12_20200529]$ ls
CO  Os3CO10_axiaxi  Os3CO10_axiequ  Os3CO10_equequ  Os3CO11_axi  Os3CO11_bri  Os3CO11_equ  Os3CO12
[jkim@jkim2 Os3CO12_20200529]$ md_output_depth1.sh
```

```
8  pre_therm_CO.mds       therm4_CO
9  pre_therm_CO.out       therm4_CO.dump.1
0  rdf_results            therm4_CO.mds
1  therm1_CO              therm4_CO.out
   therm1_CO.dump.0       xyz_results
   therm1_CO.mds
```

```
[jkim@jkim2 rdf_results]$ ls
100ps_Azo_c_S0.rdf  200ps_Azo_c_S0.rdf  400ps_Azo_c_S0.rdf  50ps_Azo_c_S0.rdf   700ps_Azo_c_S0.rdf  900ps_Azo_c_S0.rdf    Average_std_Azo_c_S0.rdf
1ns_Azo_c_S0.rdf    300ps_Azo_c_S0.rdf  500ps_Azo_c_S0.rdf  600ps_Azo_c_S0.rdf  800ps_Azo_c_S0.rdf  Average_Azo_c_S0.rdf
```

```
[jkim@jkim2 xyz_results]$ ls
100ps_Azo_c_S0.xyz  200ps_Azo_c_S0.xyz  400ps_Azo_c_S0.xyz  50ps_Azo_c_S0.xyz   700ps_Azo_c_S0.xyz  900ps_Azo_c_S0.xyz      therm1_Azo_c_S0.xyz  therm3_Azo_c_S0.xyz
1ns_Azo_c_S0.xyz    300ps_Azo_c_S0.xyz  500ps_Azo_c_S0.xyz  600ps_Azo_c_S0.xyz  800ps_Azo_c_S0.xyz  pre_therm_Azo_c_S0.xyz  therm2_Azo_c_S0.xyz  therm4_Azo_c_S0.xyz
```

Automation

Intensity.m

$$S_{cage}(q) = \sum_{i,j} f_i(q) f_j(q) \frac{N_i N_j}{V} 4\pi \left(\int_0^{R_{box}} g_{i,j}(r) \frac{sin(qr)}{qr} r^2 dr \right)$$

Intensity.cpp

```
%% User configuration
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global atom
atom={};
atom(1,:)={'C',256};
atom(2,:)={'H_2',256};
atom(3,:)={'H_1',256};
q=[0.005:0.01:10]'; %#ok<NBRAK>

name='900ps_t2rho2_v5.rdf';
Box_length=35.0762; %% Check in moldy '~.out' file.
solvent={'O_99','C_99'}; %% For cage term calculation.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Automation

Intensity.m

```
[jkim@jkim2 Azo_c_S0]$ ls
100ps_Azo_c_S0          200ps_Azo_c_S0          300ps_Azo_c_S0.dump.29  500ps_Azo_c_S0.dump.44  600ps_Azo_c_S0.dump.54  700ps_Azo_c_S0.mds      900ps_Azo_c_S0.dump.87  therm1_Azo_c_S0.mds
100ps_Azo_c_S0.dump.10  200ps_Azo_c_S0.dump.12  300ps_Azo_c_S0.dump.30  500ps_Azo_c_S0.dump.45  600ps_Azo_c_S0.dump.55  700ps_Azo_c_S0.out      900ps_Azo_c_S0.dump.88  therm1_Azo_c_S0.out
100ps_Azo_c_S0.dump.11  200ps_Azo_c_S0.dump.13  300ps_Azo_c_S0.dump.31  500ps_Azo_c_S0.dump.46  600ps_Azo_c_S0.dump.56  800ps_Azo_c_S0          900ps_Azo_c_S0.dump.89  therm2_Azo_c_S0
100ps_Azo_c_S0.dump.7   200ps_Azo_c_S0.dump.14  300ps_Azo_c_S0.mds      500ps_Azo_c_S0.dump.47  600ps_Azo_c_S0.dump.57  800ps_Azo_c_S0.dump.72  900ps_Azo_c_S0.dump.90  therm2_Azo_c_S0.dump.0
100ps_Azo_c_S0.dump.8   200ps_Azo_c_S0.dump.15  300ps_Azo_c_S0.out      500ps_Azo_c_S0.dump.48  600ps_Azo_c_S0.dump.58  800ps_Azo_c_S0.dump.73  900ps_Azo_c_S0.dump.91  therm2_Azo_c_S0.mds
100ps_Azo_c_S0.dump.9   200ps_Azo_c_S0.dump.16  400ps_Azo_c_S0          500ps_Azo_c_S0.dump.49  600ps_Azo_c_S0.dump.59  800ps_Azo_c_S0.dump.74  900ps_Azo_c_S0.mds      therm2_Azo_c_S0.out
100ps_Azo_c_S0.mds      200ps_Azo_c_S0.dump.17  400ps_Azo_c_S0.dump.32  500ps_Azo_c_S0.dump.50  600ps_Azo_c_S0.dump.60  800ps_Azo_c_S0.dump.75  900ps_Azo_c_S0.out      therm3_Azo_c_S0
100ps_Azo_c_S0.out      200ps_Azo_c_S0.dump.18  400ps_Azo_c_S0.dump.33  500ps_Azo_c_S0.dump.51  600ps_Azo_c_S0.dump.61  800ps_Azo_c_S0.dump.76  ah_debug                therm3_Azo_c_S0.dump.0
1ns_Azo_c_S0            200ps_Azo_c_S0.dump.19  400ps_Azo_c_S0.dump.34  500ps_Azo_c_S0.mds      600ps_Azo_c_S0.mds      800ps_Azo_c_S0.dump.77  a.out                   therm3_Azo_c_S0.mds
1ns_Azo_c_S0.dump.100   200ps_Azo_c_S0.dump.20  400ps_Azo_c_S0.dump.35  500ps_Azo_c_S0.out      600ps_Azo_c_S0.out      800ps_Azo_c_S0.dump.78  Azo_c_S0.in             therm3_Azo_c_S0.out
1ns_Azo_c_S0.dump.101   200ps_Azo_c_S0.dump.21  400ps_Azo_c_S0.dump.36  50ps_Azo_c_S0           700ps_Azo_c_S0          800ps_Azo_c_S0.dump.79  create_file_step.cpp    therm4_Azo_c_S0
1ns_Azo_c_S0.dump.92    200ps_Azo_c_S0.mds      400ps_Azo_c_S0.dump.37  50ps_Azo_c_S0.dump.2    700ps_Azo_c_S0.dump.62  800ps_Azo_c_S0.dump.80  execution_script        therm4_Azo_c_S0.dump.1
1ns_Azo_c_S0.dump.93    200ps_Azo_c_S0.out      400ps_Azo_c_S0.dump.38  50ps_Azo_c_S0.dump.3    700ps_Azo_c_S0.dump.63  800ps_Azo_c_S0.dump.81  pre_therm_Azo_c_S0      therm4_Azo_c_S0.mds
1ns_Azo_c_S0.dump.94    300ps_Azo_c_S0          400ps_Azo_c_S0.dump.39  50ps_Azo_c_S0.dump.4    700ps_Azo_c_S0.dump.64  800ps_Azo_c_S0.mds      pre_therm_Azo_c_S0.dump.0  therm4_Azo_c_S0.out
1ns_Azo_c_S0.dump.95    300ps_Azo_c_S0.dump.22  400ps_Azo_c_S0.dump.40  50ps_Azo_c_S0.dump.5    700ps_Azo_c_S0.dump.65  800ps_Azo_c_S0.out      pre_therm_Azo_c_S0.dump.1  xyz_results
1ns_Azo_c_S0.dump.96    300ps_Azo_c_S0.dump.23  400ps_Azo_c_S0.dump.41  50ps_Azo_c_S0.dump.6    700ps_Azo_c_S0.dump.66  900ps_Azo_c_S0          pre_therm_Azo_c_S0.dump.2
1ns_Azo_c_S0.dump.97    300ps_Azo_c_S0.dump.24  400ps_Azo_c_S0.mds      50ps_Azo_c_S0.mds       700ps_Azo_c_S0.dump.67  900ps_Azo_c_S0.dump.82  pre_therm_Azo_c_S0.mds
1ns_Azo_c_S0.dump.98    300ps_Azo_c_S0.dump.25  400ps_Azo_c_S0.out      50ps_Azo_c_S0.out       700ps_Azo_c_S0.dump.68  900ps_Azo_c_S0.dump.83  pre_therm_Azo_c_S0.out
1ns_Azo_c_S0.dump.99    300ps_Azo_c_S0.dump.26  500ps_Azo_c_S0          600ps_Azo_c_S0          700ps_Azo_c_S0.dump.69  900ps_Azo_c_S0.dump.84  rdf_results
1ns_Azo_c_S0.mds        300ps_Azo_c_S0.dump.27  500ps_Azo_c_S0.dump.42  600ps_Azo_c_S0.dump.52  700ps_Azo_c_S0.dump.70  900ps_Azo_c_S0.dump.85  therm1_Azo_c_S0
1ns_Azo_c_S0.out        300ps_Azo_c_S0.dump.28  500ps_Azo_c_S0.dump.43  600ps_Azo_c_S0.dump.53  700ps_Azo_c_S0.dump.71  900ps_Azo_c_S0.dump.86  therm1_Azo_c_S0.dump.0
```

```
[jkim@jkim2 Azo_c_S0]$ vi 100ps_Azo_c_S0.out
```

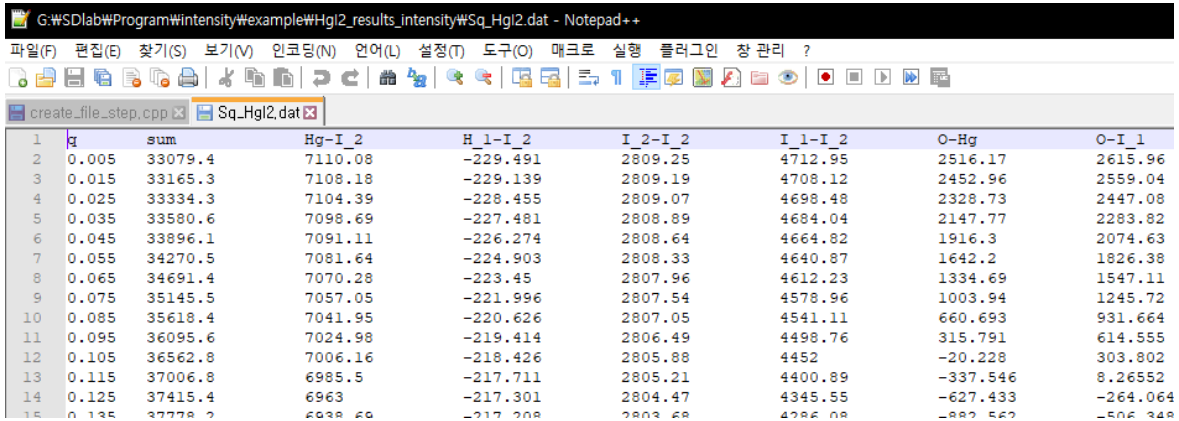
```
 *I* control file read in successfully
^L      Thu Feb 20 00:07:06 2020       Test Simulation Page 1
 *I* control file read in successfully
 *I* restart file "50ps_Azo_c_S0.mds" successfully read in
^L      Thu Feb 20 00:07:06 2020       Test Simulation Page 2


              #     # ####### #      ######  #     #
              ##   ## #     # #      #     # #  #  #
              # # # # #     # #      #     # #  # #
              #  #  # #     # #      #     #  ##  #
              #     # #     # #      #     #  ##  #
              #     # #     # #      #     # #  # #
              #     # ####### ####### ######  #     #

              Version Release2_16e  (Exp ) 2000/12/11 12:33:24

              Keith Refson
              Department of Earth Sciences
              Parks Road, Oxford OX1 3PR
              keith@earth.ox.ac.uk

              Moldy Copyright (C) Keith Refson 1988, 1992, 1993
              Moldy comes with ABSOLUTELY NO WARRANTY:
              This is free software and you are welcome to
              redistribute it under certain conditions.
              For details see file COPYING included with source.

System specification read in from restart file 50ps_Azo_c_S0.mds
Cyclohexane
       Number of molecules             = 256
       Number of sites                 = 18
       Mass                            = 84.156 amu
       Electric Charge                 = -1.90627e-16 Qe
       Dipole moment                   = 1.83002e-16 D
       Moments of inertia              = 116.111 116.111 202.695 amuA**2
Azo_c_S0
       Number of molecules             = 1
       Number of sites                 = 24
       Mass                            = 182.22 amu
       Electric Charge                 = 1e-05 Qe
       Dipole moment                   = 0.0220209 D
       Moments of inertia              = 1729.63 185.194 1914.81 amuA**2

MD cell vectors
       a                               = 35.9113 0 0 A
       b                               = 0 35.9113 0 A
       c                               = 0 0 35.9113 A
```

Automation

Intensity.m

Hgl2_results_intensity	2020-01-21 오후 8:07	파일 폴더	
Hgl2.rdf	2018-06-28 오전 12:24	RDF 파일	205KB

Sq_Hgl2.dat	2019-11-18 오후 4:52	DAT 파일	376KB
Sq_Hgl2_cage.dat	2019-11-18 오후 4:52	DAT 파일	20KB
Sq_Hgl2_cage_elements.dat	2019-11-18 오후 4:52	DAT 파일	160KB



q	sum	Hg-I_2	H_1-I_2	I_2-I_2	I_1-I_2	O-Hg	O-I_1
0.005	33079.4	7110.08	-229.491	2809.25	4712.95	2516.17	2615.96
0.015	33165.3	7108.18	-229.139	2809.19	4708.12	2452.96	2559.04
0.025	33334.3	7104.39	-228.455	2809.07	4698.48	2328.73	2447.08
0.035	33580.6	7098.69	-227.481	2808.89	4684.04	2147.77	2283.82
0.045	33896.1	7091.11	-226.274	2808.64	4664.82	1916.3	2074.63
0.055	34270.5	7081.64	-224.903	2808.33	4640.87	1642.2	1826.38
0.065	34691.4	7070.28	-223.45	2807.96	4612.23	1334.69	1547.11
0.075	35145.5	7057.05	-221.996	2807.54	4578.96	1003.94	1245.72
0.085	35618.4	7041.95	-220.626	2807.05	4541.11	660.693	931.664
0.095	36095.6	7024.98	-219.414	2806.49	4498.76	315.791	614.555
0.105	36562.8	7006.16	-218.426	2805.88	4452	-20.228	303.802
0.115	37006.8	6985.5	-217.711	2805.21	4400.89	-337.546	8.26552
0.125	37415.4	6963	-217.301	2804.47	4345.55	-627.433	-264.064

q	H_1-I_2	H_2-Hg	C-I_1	H_1-I_1	C-I_2	O-H_2	H_2-I_1	O-H_1
0.005	-229.491	-1.76197	288.875	1379.51	330.551	-493.88	-1.03233	-1713.
0.015	-229.139	-2.54044	279.639	1349.98	319.866	-483.532	-1.62092	-1671.
0.025	-228.455	-4.0972	261.403	1291.9	298.791	-463.145	-2.79115	-1589.
0.035	-227.481	-6.43143	234.629	1207.25	267.905	-433.322	-4.52914	-1468.
0.045	-226.274	-9.54073	199.991	1098.85	228.046	-394.942	-6.8142	-1313.
0.055	-224.903	-13.4198	158.355	970.312	180.285	-349.125	-9.61895	-1127.
0.065	-223.45	-18.059	110.746	825.853	125.885	-297.19	-12.9096	-917.2
0.075	-221.996	-23.4422	58.32	670.156	66.2523	-240.61	-16.646	-687.2

q	cage
0.005	-3513.83
0.015	-3448.01
0.025	-3318.66
0.035	-3130.29
0.045	-2889.38
0.055	-2604.18
0.065	-2284.31
0.075	-1940.38
0.085	-1583.56
0.095	-1225.11
0.105	-875.928
0.115	-546.155
0.125	-244.754
0.135	20.7901
0.145	244.729
0.155	423.217
0.165	554.394
0.175	638.375
0.185	677.156
0.195	674.438
0.205	635.388
0.215	566.34
0.225	474.455
0.235	367.369
0.245	252.824
0.255	138.326
0.265	30.818
0.275	-63.5901
0.285	-139.845
0.295	-194.109
0.305	-223.846
0.315	-227.848
0.325	-206.184
0.335	-160.098

Automation

Intensity_batch.m

```
q=[0.005:0.01:10]'; %%ok<NBRAK>
Box_length=35.9113; %% Check in moldy '~.out' file.
solvent={'C_100','H_100','H_200'}; %% For off cage calculate;
numofatom_solv=[256*6,256*6,256*6];
coord_list={
    'MD_input/coord/Azo_t_S0.xyz'
    'MD_input/coord/Azo_t_T1.xyz'
    'MD_input/coord/Azo_t_S1.xyz'
    'MD_input/coord/Azo_c_S0.xyz'
    'MD_input/coord/Azo_c_T1.xyz'
    'MD_input/coord/Azo_c_S1.xyz'

    'MD_input/coord/DiBrAzo_t_S0.xyz'
    'MD_input/coord/DiBrAzo_t_T1.xyz'
    'MD_input/coord/DiBrAzo_t_S1.xyz'
    'MD_input/coord/DiBrAzo_c_S0.xyz'
    'MD_input/coord/DiBrAzo_c_T1.xyz'
    'MD_input/coord/DiBrAzo_c_S1.xyz'
    };

input_filenames={
    'input/Azo_c_S0/rdf_results/Average_Azo_c_S0.rdf'
    'input/Azo_c_S1/rdf_results/Average_Azo_c_S1.rdf'
    'input/Azo_c_T1/rdf_results/Average_Azo_c_T1.rdf'
    'input/Azo_t_S0/rdf_results/Average_Azo_t_S0.rdf'
    'input/Azo_t_S1/rdf_results/Average_Azo_t_S1.rdf'
    'input/Azo_t_T1/rdf_results/Average_Azo_t_T1.rdf'
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    'input/DiBrAzo_c_S0/rdf_results/Average_DiBrAzo_c_S0.rdf'
    'input/DiBrAzo_c_S1/rdf_results/Average_DiBrAzo_c_S1.rdf'
    'input/DiBrAzo_c_T1/rdf_results/Average_DiBrAzo_c_T1.rdf'
    'input/DiBrAzo_t_S0/rdf_results/Average_DiBrAzo_t_S0.rdf'
    'input/DiBrAzo_t_S1/rdf_results/Average_DiBrAzo_t_S1.rdf'
    'input/DiBrAzo_t_T1/rdf_results/Average_DiBrAzo_t_T1.rdf'
    };
```

Github

github.com/Jkim9486/moldy_automation

Thank you for your attention!