EMD Package

EMD (acronym for Ellipsoidal Molecular Dynamics) is a free open-source software written in C++ (tested under LINUX OS) under the GPL license for molecular dynamics simulation of coarse-grained ellipsoids. This software is currently capable of simulating multiple ellipsoid particles, their interaction with surfaces and spheres through EES force field. The MD algorithm is based on "symplectic integration of quaternion rigid-body coordinates" [1] and supports NVE, NVT and NPT ensembles .
The following force fields are currently supported by the program : RE-Squared [2], Bi-axial Gay-Berne , EES [3] and EES Spherical [4].
The neighbor list algorithm used in this program is "hybrid Cell Decomposition / Verlet neighbor list" [5] using hash tables.
The Nose-Hoover Chain (NHC) thermostat and barostat are used in the simulation.

---

1. *J. Chem. Phys. 122, 224114 (2005)*
2. *J. Chem. Phys. 124, 174708 (2006) , J. Comp. Phys. , 219 , 2 (2006)*
3. *EPL 77 , 23002 (2007)*
4. EES Spherical is a force field between an ellipsoid and a sphere. If we assume that the sphere is much larger than the ellipsoid we can approximate it by a surface and use EES potential.
5. *Comp. Phys. Comm. ,161, 1-2 , p. 27-35 (2004)*

Installation

Installation of EMD is through automake tools in a straightforward manner similar to any make project. To install the package do the following :

1. Open a terminal .
2. **cd** to the folder where you have copied EMD files.
3. Type **./configure** and hit enter.
4. Type **make** and hit enter.

In case there is a missing library error check the following section.

Required Libraries

This software uses the following libraries which should be installed on your system:

- gsl
- boost filesystem
- boost graph

Main Parts

The program is consisted of some job files and several executable files as follows (we will come back to these definitions later in more detail):

Job Files:

- **param.txt :** This file is used for keeping important simulation parameters which will remain constant throughout the simulation (i.e. ellipsoids' radii or interaction energy constants)
- **base.dyn :** This file is used to keep track of the important simulation information like the integrator type or time-step.
- ***.uni :** These files are the universe files which contain the information about each ellipsoid coordinates.

NOTE : The names and extensions used for job files are only a matter of naming for easier reference and the program is not sensitive to them.

Executables:

- **emd-cli :**
  This is the main part of the program and the executable that does all the computational processes and input parsing.
- **config-manip :**
  This part of the program can change the content of the job files or add/ remove new fields to/from them.
- **emd-tools :**
  This program can be used for manipulating the input universe (i.e. juxtaposing two universes ,adding noise to particles' velocities ,etc.)
- **emd-analyzer :**
  This is the program for analyzing the outputs of the simulation and extracting useful data from them (i.e. radial distribution function, structure factor ,etc.)

Shell Scripts:

There are three scripts for automation of certain tasks which will be discussed in more detail later:

- run_auto.sh
- analyze_auto.sh
- resume_auto.sh

Job Files

The job files are like scripts with a simple language that the program can easily understand. Every job file contains different fields which are separated by one or several white-space characters (space , tab or enter) .The program's parser is sensitive to certain keywords and wherever it encounters those predefined keywords in the file it will search for the information supplied by the user in the next fields as the keyword's arguments (each keyword is followed by a certain number of arguments that come directly after the keyword separated by white-space characters). The # character is used for comments which means that wherever the program encounters this character it will ignore the remaining text and continues reading the next line. There are certain keywords which if not supplied by the user will be determined by the program's default parameters. There are also certain keywords which are exclusive so that the inclusion of one of them causes the program to ignore the other ones; For example in an NVT ensemble the program requires the "goal temperature" and if the user supplies the "goal pressure" the program will simply ignore it. The opposite is true for an NPT ensemble.
NOTE : The program's parser is not case-sensitive.

There are several keywords that the user must choose based on the problem he/she is dealing with .For simplicity the following two symbols are used in this documentation to determine the optional and mandatory parts of the syntax:

<PARAM>	required
{PARAM}	optional

besides for variable types the following abbreviations are used:

INT	=	integer
FLT	=	float

STR = string

BOOL = Boolean (it's either 0 or 1 , keywords like *true* and *false* are not recognizable by the program)

---

base.dyn

This file can contain the following fields:

- **universe_input_file <STR** *filename***>:**
  The path and name of the input uni file (the initial universe)

- **log_file <STR** *filename***>:**
  The path and name of the log file

- **uni_snapshot_seq_file <STR** *filename***>:**
  The path and the word that the name of snapshots should begin with. The program produces a sequence of output *.uni* files which are snapshots of the system at different times . These files' names are created by merging *pov_snapshot_seq_file* with a number at its end which is incremented each time the program takes a new snapshot.

- **pov_snapshot_seq_file <STR** *filename***>:**
  This field is similar to the above field just mentioned except that instead of *.uni* files it saves *.pov* files . The user is discouraged to use this feature because it results in an abundant volume of data that can be created with *emd-tools* whenever required.

- **universe_output_file <STR** *filename***>:**
  This field determines the path and name of the output .uni file (The last snapshot of the system before the simulation is finished)

- **fix_cm <BOOL>:**
  This field if set to 1 keeps the whole system's center of mass constant , if the value is 0 the center of mass' position may change through time (the default value is set to 1)

- **save_universe_snapshot <BOOL>:**
  If set to 1 the program will save snapshots of the system in uni files . If set to 0 the snapshots won't be saved.(default value : 0)

- **save_povray_snapshot <BOOL>:**
  This is similar to the above field except that it determines whether or not the program should save povray files .(as mentioned before the user is discouraged to use this feature)

- **integrator_type <INT>:**
  This field determines which type of integrator should be used in MD algorithm. 0 for NVE , 1 for NVT and 2 for NPT . (all the three cases use Miller Symplectic method)

- **time_step <FLT>:**
  This field determines the time step of the simulation (all the calculations are in reduced dimension system)

- **ignore_steps <INT>:**
  The number of starting simulation steps in which no output is saved .Note that ignore steps are included in 'steps' (refer to the next field) and aren't extra steps .

- **steps <INT>:**
  The total number of simulation steps .

- **goal_temperature <FLT>:**
  The ultimate temperature we expect the system to be in when it reaches equilibrium. This field is ignored in the NVE ensemble.

- **goal_pressure <FLT>:**
  The ultimate pressure we expect the system to be in when it reaches equilibrium . This field is ignored in the NVE and NVT ensemble.

- **nl_update_interval <INT>:**
  Number of steps to pass before the program upgrades the neighbor list . (default 10)

- **snapshot_freq <INT>:**
  Number of steps to pass before the program saves a snapshot of the system .

- **log_to_file <BOOL>:**
  If set to 1 means that throughout the simulation the program will log certain important information to a file .This file has the same content as what can be viewed on the screen . It includes the following data in columns : time , potential , energy , temperature , pressure ,order parameter ,Hamiltonian

- **log_to_stdout <BOOL>:**
  If set to 1 causes the program to print the output to std-out during the run-time(in this case showing the output in a terminal with the same columns as the above field)

- **log_force_on_surfs <BOOL> :**
  If set to 1 will log the force exerted on the surfaces.

- **log_file_force_on_surfs <STR> :**
  The path and filename to which the forces are logged to.

- **re2_cutoff <FLT>:**
  Determines the cutoff distance for the RE-squared force field.

- **re2_skin <FLT>:**
  Determines the skin thickness for the RE-squared force filed.

- **gb_cutoff <FLT>:**
  Determines the cutoff distance for the Gay-Berne force field.

- **gb_skin <FLT>:**
  Determines the skin thickness for the Gay-Berne force filed.

- **ees_cutoff <FLT>:**
  Determines the cutoff distance for the EES force field.

- **ees_skin <FLT>:**
  Determines the skin thickness for the EES force filed.

- **save_output <BOOL>:**
  If set to 1 means that the last snapshot of the system will be saved to *universe_output_file*.

- **stats_file <STR *filename*>:**
  The path and file name of the output file where some of the statistics are stored (e.g mean and std of temperature)

- **log_to_stats_file <BOOL>:**
  If set to 1 causes the statistics ti be saved to *stats_file.*

- **stats_interval <INT>:**
  Number of steps to be passed before a new statistics data is saved.

- **NHC_file <STR>:**
  The path and filename where the Nose-Hoover Chain information will be saved.

- **save_NHC_snapshot <BOOL>:**
  If set to 1 the Nose-Hoover Chain data will be saved to a file with *.nhc* extension.

- **NHC_T_length <INT>:**
  Nose-Hoover Chain thermostat length[default : 5]

- **NHC_P_length <INT>:**
  Nose-Hoover Chain barostat length [default : 5]

- **NHC_T_freq <FLT>:**
  Nose-Hoover Chain thermostat frequency [default : 0.01]

- **NHC_P_freq <FLT>:**
  Nose-Hoover Chain barostat frequency [default : 0.0001]

- **NHC_T_ksi_t :**
  Nose-Hoover Chain theromostat velocity (translational)

- **NHC_T_ksi_r :**
  Nose-Hoover Chain theromostat velocity (rotational)

- **NHC_T_s_t :**
  Nose-Hoover Chain theromostat state variable (translational)

- **NHC_T_s_r :**
  Nose-Hoover Chain theromostat state variable (rotational)

- **NHC_P_eps :**

Nose-Hoover barostat state (volume)

- **NHC_P_ksi_eps :**
  Nose-Hoover barostat momentum (volume)

- **NHC_P_eta :**
  Nose-Hoover barostat chain state (chain)

- **NHC_P_ksi_eta :**
  Nose-Hoover barostat momenta (chain)

- **log_temperature_drift <FLT** *init_temp***> <FLT** *final_temp***>:**
  If supplied the temperature changes logarithmically untill it reaches a final quantity.
  init_temp is the temperature the system starts with.
  final_conc is the temperature the system is supposed to reach logarithmically within the simulation time.

- **lin_temperature_drift <FLT** *init_temp***> <FLT** *final_temp***>:**
  If supplied the temperature changes linearly untill it reaches a final quantity.
  init_temp is the temperature the system starts with.
  final_conc is the temperature the system is supposed to reach linearly within the simulation time.

- **lin_concentration_drift <FLT** *final_conc***>:**
  If supplied changes the concentration linearly . *final_conc* is the concentration the system is supposed to reach linearly within the simulation time.

- **lin_sph_radius_drift <INT** *idx***><FLT** *init_rad***><FLT** *final_rad***>:**
  This option adds a sphere labeled as *idx* with radius *init_rad* and inflates it linearly to a final radius of *final_rad* within the simulation time. This option is useful when the system is almost packed with particles and putting a large sphere in it will superpose with the particles already in the system.

- **save_restart <BOOL>:**
  If set to 1 makes the system to save a snapshot to save_restart_freq ,even during the ignore steps (can be used to resume the program in case of a crash).

- **save_restart_freq <INT>:**
  Number of steps passed before a new restart file is saved.

- **restart_file <STR** *filename***>:**
  The path and name of the restart file.

here is an example of *base.dyn*

```
## sample dynamics job input file ##


universe_input_file          /home/MyUser/emd/automated.re2/base.uni
universe_output_file         /home/MyUser/emd/automated.re2/out.uni
log_file                     /home/MyUser/emd/automated.re2/out.log
stats_file                   /home/MyUser/emd/automated.re2/out_stats.log
pov_snapshot_seq_file        /home/MyUser/emd/automated.re2/povray/out
uni_snapshot_seq_file        /home/MyUser/emd/automated.re2/snapshot/out
restart_file                 /home/MyUser/emd/automated.re2/output/
restart.uni


fix_cm                       1
save_universe_snapshot       1
save_povray_snapshot         0
save_NHC_snapshot            0
save_output                  1
log_to_file                  1
log_to_stdout                1
log_to_stats_file            1
save_restart                 1


integrator_type              1
time_step                0.005
ignore_steps                 0
steps                     1000
goal_temperature           0.9
goal_pressure              2.0

NHC_T_length                 5
NHC_P_length                 5
NHC_T_freq                 3.0
NHC_P_freq                 3.0

re2_cutoff                 3.0
re2_skin                   0.5
gb_cutoff                  3.0
gb_skin                    0.5
ees_cutoff                 3.0
ees_skin                   1.0

nl_update_interval           7
snapshot_freq               50
stats_interval              50
save_restart_freq          200
```

Param.txt

This file contains the important parameters of the system (not the simulation) which remain constant throughout the simulation . These constants are the radii, mass and principle moments of inertia for each ellipsoid and the force field constants which differ for each kind of force field chosen.

The syntaxes are as follows (Note that the newline character is just for display reasons and can conveniently be replaced by any other white-space character or a combination of them):

rigid_class <**INT** *class_id*>* <**STR** *name*>
<**FLT** *mass*>
<**FLT** *I_body_x*>
<**FLT** *I_body_y*>
<**FLT** *I_body_z*>

class_id is an integer that is used for the labeling of the ellipsoids . If u have multiple ellipsoids this command should be repeated for each particle , each one with a distinct *class_id* than the others.

name is a string that determines a label (in words) for each ellipsoid.

mass is the particle's mass.

I_body_x , I_body_x , I_body_x are the principal moments of inertia for each ellipsoid.

The user can adopt as many surfaces as he wants but should determine a *class_id* for each surface.

int_surf_class <**INT** *class_id*>

For force fields the following syntax must be used :
NOTE: The word *homo* is used when the two particles are similar(of the same *class_id*) and the word *hetero* is used for dissimilar particles (two different *class_id*'s)

RE-Squared Potential:

re2_homo_int <**INT** *class*>
<**INT** *h_type*>*
<**FLT** *hamaker*> <**FLT** *sigma*>
<**FLT** *a*> <**FLT** *b*> <**FLT** *c*>
<**FLT** *e_a*> <**FLT** *e_b*> <**FLT** *e_c*>

re2_hetero_int <**INT** *class_1*> <**INT** *class_2*>
<**INT** *h_type*>*
<**FLT** *hamaker*> <**FLT** *sigma*>
{**FLT** *a_1* } {**FLT** *b_1*} {**FLT** *c_1*}
{**FLT** *e_a1* } {**FLT** *e_b1*} {**FLT** *e_c1*}
{**FLT** *a_2* } {**FLT** *b_2*} {**FLT** *c_2*}
{**FLT** *e_a2* } {**FLT** *e_b2*} {**FLT** *e_c2*}

* the allowed types are:

H_FORMULA_GB	*0*
H_FORMULA_SPH	*1 (NOT IMPLEMENTED YET)*

Gay-Berne potential :

gb_homo_int <**INT** *class*>
<**FLT** *epsilon_0*> <**FLT** *sigma_0*>
<**INT** *mu*> <**INT** *nu*>
<**FLT** *a*> <**FLT** *b*> <**FLT** *c*>
<**FLT** *e_a*> <**FLT** *e_b*> <**FLT** *e_c*>

gb_hetero_int <**INT** *class_1*> <**INT** *class_2*>
<**FLT** *epsilon_0*> <**FLT** *sigma_0*>
<**INT** *mu*> <**INT** *nu*>
{**FLT** *a_1* } {**FLT** *b_1* } {**FLT** *c_1* }
{**FLT** *e_a1*} {**FLT** *e_b1*} {**FLT** *e_c1*}

{**FLT** *a_2* } {**FLT** *b_2* } {**FLT** *c_2* }
{**FLT** *e_a2*} {**FLT** *e_b2*} {**FLT** *e_c2*}

EES Potential:

ees_int <**INT** *class_id_surf*>

<**INT** *class_id_rigid_mol*>

<**FLT** *A*> <**FLT** *sigma_c*>

<**FLT** *a*> <**FLT** *b*> <**FLT** *c*>

<**FLT** *e11*> <**FLT** *e22*> <**FLT** *e33*>

<**FLT** *e12*> <**FLT** *e23*> <**FLT** *e31*>

Here is an example of *param.txt*

```
##################################################

rigid_class     1          Needle1
             1.0
             0.85      0.85      0.10

rigid_class     2          Needle2
             1.0
             0.356     0.356     0.676

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

re2_homo_int     1
                  0
                100        1.0
                0.5        0.5         2.0
                0.5        0.5         8.0

re2_homo_int   2
                0
                100            1.0
                1.3            1.3         0.3
                3.3333      3.3333       0.1775

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

re2_hetero_int   1          2
                  0
                100        1
```

*.uni files

uni files or *universe files* are script files with the *uni extension* that the program uses to keep a snapshot of the universe. Note that the *uni extension* is just for the matter of convenience that the author has adopted and any other extension can be used with no difference .

Each *uni file* contains several fields that determine the coordinates of every particle , it also contains an overhead that keeps certain information about the system (e.g. box size or surface coordinates). Below are the keywords used for this type of script:

- **param_file <STR** *filename***>:**
  The path and name of the file *param.txt* which was discussed before.

- **rigid_body_num <INT>:**
  Total number of ellipsoids used .

- **flat_int_surf_num <INT>:**
  Total number of flat surfaces .

- **sph_int_surf_num <INT>:**
  Total number of spheres .

- **rigid_body_dipole <FLT** *Px***> <FLT** *Py***> <FLT** *Pz***>**:
  Attributes a dipole moment to ellipsoids with components *px , py , pz .*

- **flat_int_surf < INT** *class_id***> <FLT** *d***>**
  < FLT *n_x* **> < FLT** *n_y* **> < FLT** *n_z* **>**
  class_id is the class the flat surface belongs to and its interaction parameters should be supplied in param.txt .
  n_x , n_y , n_z determine the normal vector of the surface.
  d is the distance of the surface from the origin.

- **sph_int_surf < INT** *class_id***>**
  <FLT *r***>**
  < FLT *r_x* **> < FLT** *r_y* **> < FLT r**_*z* **>**
  class_id is the class the spherical surface belongs to and its interaction parameters should be supplied in param.txt .
  r is the sphere's radius.
  r_x , r_y , r_z determine the sphere's center.

- **x_periodic <FLT** *x_begin***> <FLT** *x_end***>:**
  The *begin* and *end* numbers determine the limits of the cuboid.

- **y_periodic <FLT** *y_begin***> <FLT** *y_end***>:**
  The *begin* and *end* numbers determine the limits of the cuboid.

- **z_periodic <FLT** *z_begin***> <FLT** *z_end***>:**
  The *begin* and *end* numbers determine the limits of the cuboid.

- **interior_point <FLT** *x***> <FLT** *y***> <FLT** *z***> :**
  This three numbers determine the coordinates of one point within the simulation volume. It is required for the program to function correctly.

- **global_electrical_field <FLT** *E_x***> <FLT** *E_y***> <FLT** *E_z***> :**
  Sets a global electric field with *E_x, E_y , E_z* as its components which can interact with particles' dipole moment .

- **time <FLT> :**
  Simulation time (determines at which time the universe snapshot has been created)

- **rigid_body <INT** *class_id***> <BOOL** *mobility***> <BOOL** *mask***>**
  <FLT *x* **> <FLT** *y* **> <FLT** *z* **>**
  <FLT *theta* **> <FLT** *phi* **> <FLT** *psi* **>**
  <FLT *v_x* **> <FLT** *v_y* **> <FLT** *v_z* **>**
  <FLT *w_x* **> <FLT** *w_y* **> <FLT** *w_z* **>**

  This keyword is the most important keyword which when used , the next 12 fields are expected by the program to be its arguments .

  class_id is an integer tag used as the label of the particle which must match with the one supplied in *param.txt* by the user.
  mobility is a Boolean number (either 0 or 1) which determines whether the particles can move or if they are fixed.
  mask is an integer which determines what types of force fields can be used for the particle. This number is simply the sum of certain integers each of which determines one type of force field as follows:

RE2	= 2
EES	= 4
EES Spherical	= 8
GB(Gay-Berne)	= 16
Dipole	= 32

  As an example if you have an ellipsoid which can interact with other ellipsoids through RE-Squared potential , and can have interaction with a surface through EES potential and with a sphere through EES Spherical , its mask will be as follows:

  MASK = RE2 + EES + EES Spherical = 2 + 4 + 8 = 14

  x , y , z determine the particle's center of mass.
  theta , phi , psi are the Euler angles which determine the ellipsoid's orientation.
  v_x , v_y , v_z determine the particle's center of mass' velocity.
  w_x , w_y , w_z are the componenets of angular velocity of the ellipsoid.

Here is an example of *base.uni*

```
## saved universe snapshot ##


param_file          /home/MyUser/emd/automated.re2/param.txt

time                         112
rigid_body_num               6

x_periodic          -5.676               5.676
y_periodic          -5.676               5.676
z_periodic          -5.676               5.676
interior_point       0.000    0.000    0.000

# 0
rigid_body      2                1                14
            +3.93133167e+00   +5.28928192e+00   +1.99596034e+00
            -6.11094499e+01   +8.25362535e+01   -2.60647409e+01
            +1.405380376+00   +1.66472678e+00   +8.7004902e-02
            +7.91009646e-01   -3.25052642e-01   +2.32581885e-04


# +1
rigid_body      2                1                14
            +2.36686954e+00   -1.45580462e+00   -2.85155821e+00
            +7.94130383e+01   +1.76867761e+02   +1.5836280e+02
            +2.78997235e-01   -8.08982319e-02   -1.99467331e+00
            -2.06607492e+00   +1.59523513e+00   -1.28880671e-01


# +2
rigid_body      2                1                14
            +1.31708280e+00   +5.00147864e+00   -2.09178369e+00
            +7.35279532e+00   +7.48197937e+01   -7.93667281e+01
            -1.28094567e+00   -9.41605600e-01   +1.61832350e-01
            +7.85816987e-01   -1.00769325e+00   +4.13692631e-02


# +3
rigid_body      2                1                14
            +2.30977002e+00   -5.82593178e-01   +1.14665083e+00
            +1.82329842e+01   -6.54483559e+01   +6.87796218e+01
             -3.89202907e-01   +1.51045764e+00   -5.67134307e-01
            +5.34546086e-01   +1.32936956e+00   -2.09808772e-01


# +4
rigid_body      2                1                14
            +3.03285363e+00   -3.82372713e+00   -1.55045659e-01
```

```
               -1.05921152e+01   +7.08213867e+01   -1.01896003e+02
               -1.04098238e+00   -6.64371391e-01   -8.41338695e-02
               +4.65038185e+00   +2.42802436e+00   +1.35220439e-01


# +5
rigid_body      2                    1                    14
               -6.99868634e-01   -4.02644506e+00   -4.66998537e+00
               -8.46141356e+01   -1.15562673e+02   -6.09696586e+01
               +3.97794958e-01   +1.43455662e+00   -4.18928168e-01
               +2.18024507e+00   +4.35518516e-02   -8.34677727e-02
```

Executables:

Executables are the main files that should either be used as a command in a terminal or inside a script. Each executable can take some arguments which will be discussed bellow:

- ***emd-cli <-i dyn_file>***

This binary is the core of the program where all the simulation takes place. *dyn_file* is the input *.dyn* file which stores the information about simulation parameters.

- ***config-manip {***

```
    < -i  <input_file>   >
   <-o  <output_file> >
  {-a <key> <value>}
  {-c <key> <new_value>}
   }
```

This binary takes a text file (usually a *dyn* file) as its input and makes certain changes to it according to its arguments . This program can be used to automatically change *dyn* files in the scripts .

input_file is the text file which the program is about to change.

output_file is the file where the changes will be saved

-a **:** If this argument is used a new line will be added to the end of the the file with the format: *key value* (Note that *dyn* files have the format *key value* so this will in fact add a new command to the *dyn* file)

-c : If this argument is used the program will search for the keyword "*key"* in the input file and will replace the value that comes after it with *value*

- ***emd-tools***

```
  {< -d <input_universe> <output_universe> <num> >
{-param <input_universe> }
{-add_sphere <input_universe> <output_universe> <r_x> <r_y>
<r_z> <rad> <pad> <class_id>}
{-pert_vel <input_universe> <output_universe> <v_amp>
<omega_amp>}
{-concat_uni <input_universe_1> <input_universe_2>
<output_universe> <axis_idx>}
{-pov <input_universe> }}
```

-d

Duplicates the *input_universe, num* times along each side of a larger cube.

-param

Change *param* file

-add_sphere
Add a spherical surface of type *class_id* to the universe at <*r_x, r_y, r_z*> with a radius *rad.* also, remove the intersecting rigid bodies with padding *pad.*

-pert_vel
Perturb the linear and angular velocities by a linear random variable of half width *v_amp* and *omega_amp.*

-concat_uni
Concatenate *input_universe_1* and *input_universe_2* along *axis_idx.*

-pov
Can convert the *uni* files to *povray* files which can be viewed in 3D with *povray software*.

- ***emd-analyzer***

-l <*log_file_path*>

The path of the log file

-s <*snapshot_dir_path*>

The path of the directory of universe snapshots

-o <*output_dir_path*>

The path of the output directory

-eval_3d_rdf

Evaluate scalar, para. and perp. RDFs

-eval_2d_rdf

Evaluate in-plane 2D RDF

-eval_tilt

Evaluate the director-plane tilt angle

-axis_idx <*val*>

Axis index for P_2 (0, 1, 2) [default: 2]

-max_rad <*val*>

Maximum range of RDFs [default: 8.0]

-divs <*val*>

Number of division points in the range of RDFs [default: 200]

-min_wl <*val*>

Minimum possible wavelength [default: 1.0]

-iporder <*val*>

In-plane RDF order [default: 0]

-snap_skip_rate <*val*>

Snapshot skipping rates [default: 0]

-dd_penalty_coeff <*val*>

Director deviation penalty coefficient [default: 1.0]

-eval_smectic_config

Configuration of a smectic plane

-eval_bond_order

Bond orientational order

-bond_order_r_min

Bond order min cutoff [default: 2.0]

-bond_order_r_max

Bond order max cutoff [default: 3.0]

-eval_plane_ave <*normal_idx*> <*center*> <*width*> <*divs_1*> <*divs_2*>

Evaluate averages (density, OP) in a plane

-DFT

Evaluate the DFT of time averaged RDFs

-uni <*uni_file_path*>

The path of the uni file

-sf

Evaluate structure factor and rdf for a single snapshot

-class_id *<val>*

Class(label) of the particle to be analyzed

-count_regions

Counts the number of regions created by particle <class_id> after spinodal decomposition

-neighbor_dist *<val>*

The size of the cell for determination of regions

-param *<param_file_name>*

If this field is supplied the param file will be read from param_file_name rather than the location universe files point to

-density_fourier

Calculates the Fourier transformation of the density

-freq_divs *<val>*

Number of division points in the range of frequencies [default: 20]

-structure_factor

Calculates the structure factors for number density and concentration

-order_param

Calculates the order parameter for particle *<class_id>*

-rel_dir *<class_id1>* *<class_id2>*

Calculate the angle between director of particle *<class_id1>* and *<class_id2>*

Shell Scripts

There are three scripts which automate certain tasks:

- **run_auto.sh** starts from an initial density , gradually increases the density and equilibriates the system for a certain number of steps. It then increases the density a little and does the simulation for a certain equilibrium time . In this way we can investigate the behavior of the system under gradual density increase. Each density's output is placed in a folder with the name of the density number.

- **resume_auto.sh** is a script that can be used once *run_auto.sh* has been executed and halted later on for any reason (e.g. power loss) to resume the simulation.

- **analyze_auto.sh** is a script that automates the process of analyzing the the output of *run_auto.sh* which is in different folders.