Manual for running the BD Codes

January 1, 2025

1 Instructions for running the codes stored in folder: t2_langevin_focker_planck

This section describe the evolution of particles distribution in time, where particles follow the Langevin stochastic dynamics. The relevant LAMMPS input file is given by the name `t2_langevin.in`.

Configuring system:

To configure the system there are system specific changes done in the script. For the detail please check Sec. 1 DPD LAMMPS manual.

1. **Creating the particles in a narrow region within a large simulation box:**

```
region my_box block 0 20 0 20 0 20
create_box 1 my_box

region my_region block 10 11 10 11 10 11

create_atoms 1 random 1000 8009 my_region
```

   A large simulation box is created and then particles are generated in a narrow region within the simulation box. So in practice, we are doing delta distribution of the particle at time $t = 0$.

2. **Switching off the pair coefficient:**

```
pair_style none
```

   You switch off the pair style to just make it ideal diffusing particles at all length scale, just to make the evolution smooth. In case if it switched then the particles will feel caging leading to the subdiffusive behaviour at smaller time scale.

3. **Implying an integrator and the thermostate:**

```
fix my_thermo_stat all langevin 1.0 (#start T) 1.0 (#end T) 0.1 89080

fix your_integrator all nve
```

You add Langevin part of the equation of motion by using the fix, which is in effective a thermostat, as the friction terms itself take care of divergence of velocity and keep the temperature constant. But thermostat is not an integrator, so you will use something to apply the movement of particle. In the example script we have used simplest integrator **nve**, which is enough to simulate the point particle (for example, if you have a particle with finite size then you will have to use some integrator which can integrate over rotational degree of freedom also e.g, **rigid/small molecule**). Since the particle is already thermostat-ed, so you can not use the **nvt** fix for the integration, where you have some builtin thermostats like Noose Hoover or Anderson thermostat to keep the temperature constant. So using **nvt** will lead to the two thermostats, because system is already thermostat-ed using Langevin itself.

In conclusion, not all kind of integrator can be used with different kind of thermo-stating and barrow-stating.

Steps to follow:

To perform the scaling of time, students can follow the following steps:

1. Run the code stored in the folder named **t2_langevin.in**.
2. Open the ovito and load the data file **simulation_data.lammpstrj**.
3. Open the Histogram filter and observe the position x evolution of the system.
4. While observing the whole system picture evolution obtained by solving the Focker-Plack equation stored in a GIF file named **fokker_planck_evolution.gif** which is generated by using the python code **t2_evolution_focker_planck.py**, you can compare with the LAMMPS system evolution.
5. You can observe that the Langevin dynamics which deals at the discrete level with individual particles produces the same whole picture as predicted by the continuum Focker-Planck equation, which indeed represents the evolution of the system as a whole.

2 Instructions for running the codes stored in folder: t2_patchy_particles

Now you are going to learn how to create a Brownian particle and decorate the same with the attractive patches for the purpose of the directional bonding and then thermostat both the translational and rotational degree of freedom using the Langevin thermostat and make it perform the Brownian dynamics simulation.

But first please read the below explanation of the system configuration, which focuses on some configurations in the LAMMPS script file `t2_patchy_particles.in`.

Configuring system:

To simulate a Brownian system you make the following changes in your LAMMPS script:

Note: For more details, please check the Sec. 1 of the DPD manual.

1. **Defining particle's attributes:**

```
units lj
atom_style  hybrid sphere molecular
boundary p p p
pair_style hybrid lj/cut 2.0 cosine/squared 0.12
```

Your brownian particle is solid sphere, which also have some angular momentum. Atom style `hybrid sphere molecular` makes particle to have attributes like angular momentum and can also have a rigid shape if merged with the patches using molecule part of the atom style, which are also particles.

2. **Making memory for two types of patches:**

```
region box block 0 30.0 0 30.0 0 30.0
create_box 2 box
```

This `create_box 2 box` commands ensures that you have memory to store two kinds of particles. One will be the heavy core particle at the center and the other type of particles will be patches.

3. **Loading molecule from a template:**

```
molecule patchy_part t2_patchy_molecule.mol
create_atoms 0 random 5000 87910 NULL mol patchy_part
... 454756 overlap 1.5 maxtry 50
```

A molecule can be imported from a molecule template files which contains the information like type id, position, particle id, and bonding information etc. So generally when you create the atoms from the template you assign atom type id 0, as it has already being supplied by the template file. So in the template file `t2_patchy_molecule.mol` you can change the position of the particles to change the patch configuration, shape and size of the molecule etc. You should avoid overlapping of the hard cores by using the command

`overlap 1.5 maxtry 50`. But if the LAMMPS would not be able to find a free space to fill the particle with 50 trial then that particle will not get generated. So, its not possible to generate high density system using this command. The best way is first create a low density system with high box size. Put some NPT ensemble to compress the system with a given pressure and then unfix the npt ensemble and apply the nve ensemble. In this case, we don't need to do this, as we are interested only in the low volume fraction (or density) system.

The molecule template stored in file `t2_patchy_molecule.mol` is given as:

```
#number of atoms
5 atoms

Coords
#particle id coordinate
1  1.0   1.0    1.0
2  1.2886  1.2886  1.2886
3  0.7113  0.7113  1.2886
4  0.7113  1.2886  0.7113
5  1.2886  0.7113  0.7113

Types
# particle id  particle type id
1   1
2   2
3   2
4   2
5   2
```

4. **Defining interaction between the core particles and the patches:**

```
pair_style hybrid lj/cut 2.0 cosine/squared 0.12
pair_coeff 1 1 lj/cut 0.01 1.5 2.0
pair_coeff 1 2 none
pair_coeff 2 2 cosine/squared 8 0.3 0.35 #square well attractive sites
```

We are defining the hard-core spherical particle using lj interaction. While patches are made attractive to each other. Patches do not interact with the cores. So in effective, we are creating a molecule with spherical shapes with some attractive sites such that whenever those sites (patches) come close to each other they attract.

5. **Defining the physical properties like mass and the diameter of the particles:**

```
set type 1 mass 1.0
set  type 2 mass 0.000001

set type 1 diameter 1.0
set type 2 diameter 0.0
```

We make the patchy particle as a point particle with almost negligible mass and size, as it is not an actual particle, but just an attractive interaction site, which have already been considered while defining the pair style (the command line `pair_coeff 2 2 cosine/squared 8 0.3 0.35`).

6. **Grouping particles:**

```
group core type 1
group patch type 2
group rigid_molecule type 1 2
```

Whenever we apply a fix, it is either applied on every particle by using the keyword `all`. Or on some group. You can not apply fixes directly calling the particular set of atoms by their atom id. So you first group a kind of atom in a particular set and then apply the fixes on that set by calling the set by the group names.

7. **Some command to speed up the simulation speed by switching off the unnecessary calculations:**

```
neigh_modify exclude molecule/intra rigid_molecule ...
...every 1 delay 0 check no
```

This command ignore the intra-particle interaction calculation. e,g, among the patches belonging to the same molecule. As there is no meaning of this calculation as the particles in the same body act like a rigid body.

8. **Fixing themostat and applying the integrator:**

```
fix thermo_stat  core langevin 1.0 1.0 0.1 428984  omega yes
fix rigid_thermo rigid_molecule rigid/small molecule
```

We fix the Langevin thermostat only over the cores. So there is no meaning of thermostating the patches as the patches are not actual particles but just the sites with almost massless. Here `omega yes` command is thermostat the rotational degree of freedom. But when we use the integrator like `rigid/small molecule`, it will consider a single molecule, e.g, the force exerted on the patches will be counted for the whole body including the core. Which is correct, as an interaction site (which is patchs in this case) can make the cores to attract and repel each other by exerting the force due to their potential.

9. **Doing some precautions while measuring the thermodynamic quantities:**

```
compute kinetic_core core ke
fix kinetic_output core ave/time 100 1 100  ...
...c_kinetic_core file kinetic.dat mode scalar
compute temp_core core temp/sphere
fix temp_output core ave/time 100 1 100   ...
...c_temp_core file temperature.dat mode scalar
```

However, when the LAMMPS will compute the temperature the patches will be computed as an actual particles, which in effectively are not as with such a less mass and the radius. So it will use the equipartition theorem and average the temperature over all the particles

including the patches and therefore will show the wrong results. As patches have not been thermostat-ed. Therefore we apply the calculation of these quantities only on the core group of particles, which is the correct representation of the thermodynamic state of the system.

Note: For more detail about the compute please check the Sec. 1 of the DPD manual

Steps to follow:

Please follow the following step to extract the ρ vs p (pressure) data. The mathematical relation between the two quantities are often referred as EOS (equation of state). Using the same data you can solve the related problems in your exercise.

1. Run the script file `t2_patchy_particles`.
2. Load the data in OVITO and visualize the self assembly of the patches.
3. Lower the temperature by using the command line `fix thermo_stat core langevin 1.0 1.0 0.1 428984 omega yes`. Please see the LAMMPS manual to find about this fix and how to set temperature. Generally you keep the start temperature and end temperature same e.g, `1.0 1.0` as in the given example, and the damping appropriate to make your system more like Brownian, like the value `0.1` in the given example. So, to create the high damping, we lower this value, and to lower the damping, we increase this parameter.
4. Observe the clustering of the Brownian molecules at low temperatures.

3 Instructions for running the codes stored in folder: t2_liquid_gas_coexistence

In this section, you will calculate the coexistence density between the gas and liquid phase of a Lennard-Jones fluid. You will first create a highly dense liquid without any bubble or nucleous of gas phase. And then merge with a gas phase and then calculate the density profile. For the purpose you will use multiple LAMMPS input files which will be configured like this.

System configuration:

To make a high density liquid regime, you will use the file named `t2_liquid_box_creator.in`. The description of a few key commands are given as below,

1. **Creating a large rectangular box:**

```
region my_box block 0 15 0 15 0 90
create_box 1 my_box
```

   All the details have already been explained.

2. **Applying NpT ensemble to compress the fluid and make it a liquid phase:**

```
fix my_pressure all npt temp 0.4 0.4 1 z 10 10 1.0 couple none
```

   After generating the box throughout the whole box you use a barostat along the z axis with some presssure. We use a guess pressure, where system can exist in liquid phase and apply that pressure. For example, in the given script you are apply the temperature `0.4 0.4`, and the pressure along the z axis as `10 10`. Applying along only z axis does not mean that in other directions the pressure will be different. It is just for the shape of the box, where box will be compressed along the z-axis to bring the system at the specified pressure.

3. **Printing the last configuration after the equillibration:**

```
write_data compressed_coordinate.data
```

   We use the command to print the last configuration. The NpT thermostat cum barostat have compressed the box along the z axis such that the box will located in the middle of the initial box. For example if box had initial extension beween $0-90$ then the now after compression it will be located between let us say $38-52$.

After we generate the liquid phase, we need to merge it with the gas phase. We again put out generated and equilibrated box with extension $38-52$ in a box of $0-90$ and fill the empty region $0-38$ and $52-90$ with the low density gas phase (just fills some appropriate number of particles randomly, if gas particle have less number of particle it will have lower chemical potential and some of the particles from liquid phase will evaporate and merge into the gas phase. If the gas particle greater then than required then it will have higher chemical potential and some of the particles from the gas phase will deposit into the liquid phase. This

will continue until the vapor pressure and chemical potential becomes equal to the equilibrium value at that temperature).

Now, to merge both the phase where we are putting liquid phase into a bigger box, which is bigger along the z-axis. So, the particle data (stored in the file named `compressed_coordinate.data`), which you printed by using the code `t2_liquid_box_creator.in`, are being putted in the larger box along z-axis, therefore you would need to remove the periodic neighbourhood along the z, as the particles which were counted within the interaction range are no more periodically connected as the wall has shifted away from the particle from the both the ends (e.g, at length 38 and 52).

To remove the periodicity along the z -axis, you use another script named `t2_intermediate.in` which load the data and rewrite the same data after running for 0 time steps.

1. **Removing periodicity along an axis:**

```
boundary p p s
```

As you can see the script does nothing but change the periodic condition in the system only for the z-axis. Using `s` means now the particles are not connected through the periodic boundary in nearest neighbor list (along z-axis) and it also ensure that the wall sat in the system encloses all the particles along z-axis, so that no particle left excluded.

Once you remove the periodicity you again write down the whole data, which is now loaded by the file named `t2_coexistence_simulator.in`. In this script, you load the system data and define a region on two sides of the liquid phase and then fill that region with particles.

(a) **Making the region and filling the particles:**

(b) **Removing periodicity along an axis:**

```
region gas_region_1 block 0 15 0 15 20 35
region gas_region_2 block 0 15 0 15 55 70
create_atoms 1 random 100 87910 gas_region_1 overlap 1. maxtry 50
create_atoms 1 random 100 87910 gas_region_2 overlap 1. maxtry 50
```

Here you define the two regions, but all within the simulation box range and then fill with the particles.

(c) **Binning the number of particles along the z axis:**

```
compute chunk_1 all chunk/atom bin/1d z lower 0.02 units reduced
compute myChunk1 all property/chunk chunk_1 count
fix 1 all ave/time 100 1 100 c_myChunk1 ...
...file bin_particles.lammpstrj mode vector
```

Using this command you print the number of particles in each spatial segment along the z-axis which is sliced in 1/0.02 number of bins in the unit of box length. For more detail see the LAMMPS or use the C H A T G P T.

Note: Rest of the description of the code is same as before.

Steps to follow:

To solve the relevant question students are advised to follow the following steps:

1. Run the LAMMPS script `t2_liquid_box_creator.in` after setting desired value of pressure, which can ensure the high density liquid phase. If you will not create high density liquid phase then during the simulation with the gas phase, the gas bubbles left in the system which is the nucleus of the gas phase will grow and you will get a fragmented region of liquid and gas.

2. Run the script `t2_intermediate.in`, which loads the file `compressed_coordinate.data`.

3. Now you have the file `compressed_coordinate.data`, with switched off periodic boundary condition along the z-axis. Open the data file using some text editor and then change the box length extending on the both sides of z axis. For example, if you have box length along z like `38 52 zlo zhi`, then extend it symmetrically and make it `20 70 zlo zhi`.

4. Save and close the file `compressed_coordinate.data`.

5. Run the script `t2_coexistence_simulator.in`, which will equilibrate the system and generate the number of particle data for each bins along the z-axis at different time and store in the file named `bin_particles.lammpstrj`. This data will be used later to calculate the density distribution.

6. Run the python code named `t2_data_rectifier.py`, which will load the binned data for number of particle distribution and average over time, and then print the density distribution z vs ρ in text file named `averaged_density_profile.txt`.

7. Open the DESMOS online graph plotter in your browser and type the function $tanh(x)$. See the nature of curve. Close to the gas-liquid interface you will observe similar kind of density distribution.

8. Run the python script named `t2_density_profile_fitter.py`, which will fit the ρ vs z data over a $tanh$ function. On terminal, you get the gas and liquid coexistence density. A plot with the simulated data and curve fit will also be exported in a PNG file named `density_profile_fit.png`.

9. Compare the result shown in the PNG file `reference.png`.

4 Instructions for running the codes stored in folder: e2_brownian_dynamics

In this folder you have codes which generates MSD for the particles to calculate the diffusivity. Following are the specification for the script `e2_brownian_dynamics.in`.

System configuration:

For the details see the previous sections.

The code generates the particles with homogeneous distribution along x and y axis and z axis. The particles interaction are switched off (student can look and guess which section does this, if they have read the previous instructions). Particles diffuse using Langevin dynamics and the script prints the MSD data.

1. **Setting up the temperature in the initial velocity distribution:**

```
# Assigning initial velocity distribution
velocity all create 4.0 12345 mom yes rot no
```

2. **Setting up the temperature in the Langevin thermostat:**

```
# Assigning initial velocity distribution
fix my_thermo_stat all langevin 4  4 .1 89080
```

In this section, we define the thermodynamic parameters like temperature (given as `4. 4.0` in example script) and the damping parameter (given as `0.1`), which is inverse of friction coefficient.

Caution: **Please keep the temperature for the intitial velocity distribution command as same as used in the thermostat.**

Steps to Follow:

1. Open the file named `et_brownian_dynamics.in` and set the desired temperature.
2. Run the script and wait for the output data.
3. Run the python script `e2_msd_data_rectifier_and_plotter.py` while will plot the MSD vs time data and also give the value of the diffusivity.
4. Change the mass of the particle and note down the the diffusivity, while keeping all the other parameters same and plot mass vs diffusivity by saving in `.txt` file, and loading and plotting in the python (or you can plot in your own way).
5. Change the friction of the system while keeping all the parameters same and plot friction vs diffusivity.
6. Change the temperature while keeping all the other parameters same and plot temperature vs diffusivity.

5 Instructions for running the codes stored in the folder: e2_binary_mixture

This system creates a binary mixture of Brownian particles using the script `e2_binary_mixture.in`. For the details of the command students are referred to the previous sections. However a few key configuration is given as,

System configuration:

The codes generate a binary mixture with the particles randomly and homogeneously distributed along the whole box. By changing the effective radius of the lj particles you can make the mixture additive and non additive. An additive mixture is designated as $d_{ij} = \frac{1}{2}(d_{ii} + d_{jj})$. All the other values of d_{ij} will be considered as non additive mixture.

1. **Setting the additive and non-additive binary mixture:**

```
pair_style lj/cut 2.4
pair_coeff  1  1  0.01  1.  1.5
pair_coeff  1  2  0.01  1.55  2.1
pair_coeff  2  2  0.01  2.  2.2
```

   In your description of the interparticle interaction, you are using the lj potential. The *sigma* value decide the size of the hard core. For the particle type id 1 and the type id 2 interaction, the example script have σ, as $\sigma = 1.55$, which is a little off than the additive condition given by $d_{ii} = \frac{1}{2} \cdot (1 + 2) = 1.5$.

2. **Setting the appropriate pressure to observe the phase separation:**

```
fix my_pressure all npt temp 1 1 1 iso 20 20 1.0
```

   In this scrip, you are using the *npt* ensemble with a defined temperature and pressure value, e.g, `iso 20 20 1.0` fixes pressure 20.0. The reason behind setting a pressure is that, the phase separation observed due to the non-additivity is purely entropic in nature. Not like DPD, where interaction is considered with different strength by using different force constant. As we increase the density, the system which is alway intended to maximize the entropy, will try to find more phase space for the particles to move. In a non-additive mixture, we get more phase space if particles are seperated from each other.

3. **Removing the NpT ensemble:**

```
unfix my_pressure
```

   Once the system equilibrate the system at a given pressure, we revoke the NpT ensemble and apply the dynamics in the system, and then let the system evolve and euillibriate.

Steps to follow:

1. Set the pair coefficient σ and pressure in the script `e2_binary_mixture.in`.
2. Run the code.

3. Load the simulation data file `simulation_data.lammpstrj` in OVITO.

4. Observe the phase separation, where in the case, you will observe clusters of a particular type of particles.

5. Change the pressure for the same inter particle pair coefficient σ once making it high as around $10 - 15$ and once low as $0.05 - 1$.

6. Observe the effect of pressure on the phase separation for that particular temperature and pair coefficients.

6 Instructions for running the codes stored in the folder: e2_patchy_particles

This folder contains a patchy particle template and the LAMMPS script to run the code. Students have already made familiar with all the command and functions used in the input script `e2_patchy_particle.in`.

However a few things are being clarified here.

System configuration:

1. Setting the molecular template:

```
5 atoms

Coords

1  1.0   1.0    1.0
2  1.2886  1.2886  1.2886
3  0.7113  0.7113  1.2886
4  0.7113  1.2886  0.7113
5  1.2886  0.7113  0.7113

Types

1   1
2   2
3   2
4   2
5   2
```

The template sets two type of particles with particle type id 1 and 2. The positions of the patches are set around the central core particle which have the particle type id 1 (while patches have particle type id 2). You can remove few patches by changing the atom number (`5 atoms`) and set `types`, and `Coords`, accordingly.

Steps to follow:

1. Customize the molecule template given with the file name `e2_patchy_molecule.mol`.
2. Run the script `e2_patchy_particle.in` and load the output simulation data in OVITO.
3. Find if the patches are located around the core, with the desired geometric configuration.

Appendix

Using OVITO for Visualization

1. Load the trajectory file: `File > Load File`.
2. Apply `Modifiers` to enhance visualization (e.g., `Color Coding`).
3. Adjust the perspective for better screenshots.

Using `matplotlib` for Plotting

1. Install `matplotlib` using `pip install matplotlib` if not already installed.
2. Run the provided plotting script, replacing placeholders with actual data.
3. Save the generated plot as a PNG file.