Data Science Subject Group: Simulation Tutorials
6th Lecture Lecture Materials

Instructor: Takeshi Kawasaki

Nagoya University Graduate School of Science, Department of Physics, Non-equilibrium Physics Laboratory (R Lab)

Last update: August 23, 2024

Table of Contents

Contents

1. Lecture Schedule

- This tutorial lecture will be conducted during the first spring term.
- Lecture materials will be uploaded by 11:00 AM on the day of the scheduled lecture.
- Schedule
  1. 4/15: 1st Lecture
  2. 4/22: 2nd Lecture
  3. 4/30 (Tue): 3rd Lecture
  4. 5/13: 4th Lecture (Midterm Report Assignment Release)
  5. 5/20: 5th Lecture
  6. 5/27: 6th Lecture (Midterm Report Assignment Due)
  7. 6/03: No Lecture
  8. 6/10: 7th Lecture (Final Report Assignment Release)
  9. 6/17: 8th Lecture
  10. 6/24: 9th Lecture (Makeup Class)

1.1. Syllabus

This practice Lecture is planned to cover the following topics (subject to change depending on progress):

1. Introduction
   - How to use C (C++) (mainly for numerical calculations)
   - How to use Python (for data analysis and plotting)
   - Principles of numerical computation
   - Loss of precision (catastrophic cancellation)
   - Non-dimensionalization in scientific computing
2. Numerical solutions of ordinary differential equations: Using examples of damped and harmonic oscillators
   - Numerical integration of differential equations
   - Euler's method
3. Brownian motion of a single particle
   - Langevin equation (stochastic differential equations)
   - Method for generating normal random numbers
   - Euler-Maruyama method
   - Time average and ensemble average
   - Calculation of the diffusion coefficient
4. Brownian motion in a multi-particle system
   - Method for calculating interaction forces
   - Simulation of non-equilibrium systems: Using phase separation phenomena as an example
5. Molecular dynamics simulation for multi-particle systems
   - Position Verlet method and Velocity Verlet method
   - Conservation laws in multi-particle systems (momentum, energy, angular momentum)
6. Monte Carlo methods
   - Review of statistical mechanics
   - Markov Chain Monte Carlo method

Contents

2. Assignment 5

Assignment 5 Preparation for Multi-Particle Calculation

In a two-dimensional plane, arrange 512 disks with a particle size (diameter) of 1 in a square space with a side length of $L = 40$.

(1) Arrange the particles in a square lattice and plot the result.

(2) Arrange the particles in a hexagonal lattice and plot the result.

Explanation

The sample program discussed here can be obtained from the GitHub repository (under Lecture6) **[Link]**.

(1) The subroutine ini_square(x) in the program **"lattice.cpp"** generates particles arranged in a square lattice within a square box of arbitrary size L (2D) (see list 1). The plot was created using **"draw_particle.py"** (list 2).

2. Assignment 5 (2)

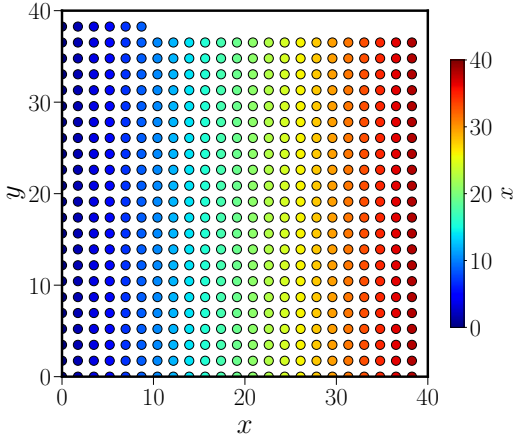

Fig. 1: Square lattice generated using the program in list 1.

2. Assignment 5 (3)

リスト 1: サイズ L の正方系領域内に Np 個の点を正方格子状に均等に配置するアルゴリズム.“lattice.cpp”におけるサブルーチン ini_square(x) は正方格子，ini_ hex(x) は六方格子を生成する．

```
1
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <math.h>
5  #include <iostream>
6  #include <fstream>
7
8  #define L 40.0
9  #define a 1.0
10 #define Np 512
11 #define dim 2
12
13 void ini_square(double (*x)[dim]){
14   int num_x = (int)sqrt(Np)+1;
15   int num_y = (int)sqrt(Np)+1;
16   int i,j,k=0;
17   for(j=0;j<num_y;j++){
18     for(i=0;i<num_x;i++){
19       x[i+num_x*j][0] = i*L/(double)num_x;
20       x[i+num_x*j][1] = j*L/(double)num_y;
21       k++;
22       if(k==Np){
23         break;
```

2. Assignment 5 (4)

```
24        }
25      }
26      if(k==Np){
27          break;
28      }
29    }
30  }
31
32  void ini_hex(double (*x)[dim]){
33    int num_x = (int)sqrt(Np)+1;
34    int num_y = (int)sqrt(Np)+1;
35    int i,j,k=0;
36    double shift;
37    for(j=0;j<num_y;j++){
38      for(i=0;i<num_x;i++){
39        shift=(double)j*0.5-j/2;
40        x[i+num_x*j][0] = (shift+i)*L/(double)num_x;
41        x[i+num_x*j][1] = j*L/(double)num_y;
42        k++;
43        if(k==Np){
44          break;
45        }
46      }
47      if(k==Np){
48          break;
49      }
50    }
```

2. Assignment 5 (5)

```
51  }
52
53
54  void output(double (*x1)[dim],double (*x2)[dim]){
55    char filename[128];
56    std::ofstream file;
57    sprintf(filename,"coord_square_L%.1fN%d.dat",L,Np);
58    file.open(filename);
59    for(int i=0;i<Np;i++)
60      file<<x1[i][0]<<"\t"<<x1[i][1]<<"\t"<< a <<std::endl;
61    file.close();
62
63    sprintf(filename,"coord_hex_L%.1fN%d.dat",L,Np);
64    file.open(filename);
65    for(int i=0;i<Np;i++)
66      file<<x2[i][0]<<"\t"<<x2[i][1]<<"\t"<< a <<std::endl;
67    file.close();
68  }
69
70  int main(){
71    double  (*x1)[dim] = new double[Np][dim];
72    double  (*x2)[dim] = new double[Np][dim];
73    ini_square(x1);
74    ini_hex(x2);
75    output(x1,x2);
76    delete[] x1;
77    delete[] x2;
```

2. Assignment 5 (6)

```
78    return 0;
79  }
```

For drawing particle coordinates, a sample program ”draw_particles.py” that utilizes matplotlib.patches (see **[Reference Link]**) is shown in list 2.

リスト 2: 粒子配置を作図するサンプルプログラム “draw_particles.py”.

```
1  #第５回自主課題
2  %matplotlib inline
3  import math
4  import matplotlib
5  %config InlineBackend.figure_format = 'retina'
6  import matplotlib.cm as cm  # colormap
7  import matplotlib.pyplot as plt
8  import numpy as np
9  import matplotlib.patches as mpatches
10
11 plt.rcParams["text.usetex"] =True
12 plt.rcParams["font.size"] = 30
13
14 from matplotlib.collections import PatchCollection
15 from matplotlib.patches import Circle
16 import numpy as np
17
18 resolution = 100 # the number of vertices
```

2. Assignment 5 (7)

```
19 Np=512
20 L=40.0
21 patches = []
22
23 fig = plt.figure(figsize=(10,10))
24 ax = fig.add_subplot(111)
25
26 x, y,r = np.loadtxt("./Lecture6/coord_square_L40.0N512.dat", comments='#', unpack=True)
27
28 for i in range(Np):
29     circle = mpatches.Ellipse((x[i],y[i]), r[i], r[i])  # 楕円の中心座標，長軸　短軸　（今回は真円）
30     patches.append(circle)
31
32 plt.xlim(0, L)
33 plt.ylim(0, L)
34
35 colors = x
36 p = PatchCollection(patches, cmap=matplotlib.cm.jet, alpha=1.0,ec='k')
37 p.set_array(colors)
38 ####color range ####
39 p.set_clim(0,L)
40 ####
41 ax.add_collection(p)
42
43 C=plt.colorbar(p,shrink=0.6) # shirink:  controlling the size of color bar: 1.0 is maximum
44 C.set_label(r"$x$", fontsize=30) # color bar label
45
```

2. Assignment 5 (8)

```
ax.spines['top'].set_linewidth(3)
ax.spines['bottom'].set_linewidth(3)
ax.spines['left'].set_linewidth(3)
ax.spines['right'].set_linewidth(3)
plt.tick_params(which='major',width = 1, length = 10)
plt.tick_params(which='minor',width = 1, length = 5)
plt.xticks(color='k', size=30)
plt.yticks(color='k', size=30)
plt.xlabel(r"$x$",color='k', size=35)
plt.ylabel(r"$y$",color='k', size=35)

ax.set_aspect('equal')

plt.savefig('./Lecture6/square.pdf',bbox_inches="tight")
plt.show()
```

(2) As shown in Listing 1, the subroutine ini_hex(x) in the program **"lattice.cpp"** generates particles arranged in a hexagonal lattice within a square box (2D) of arbitrary size L. The visualization was done using **"draw_particle.py"** (adjust the file name as needed).

2. Assignment 5 (9)

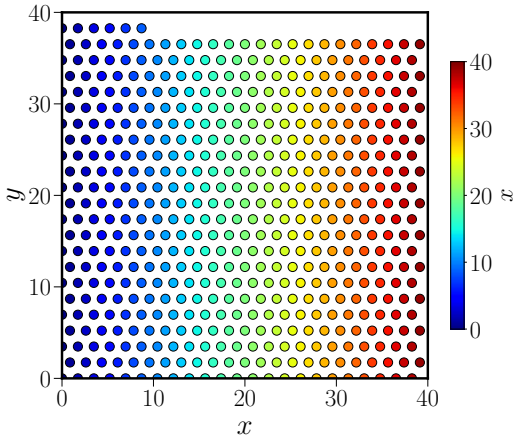

Fig. 2: Square lattice generated using the program in Listing 1.

Contents

3.Brownian Motion in Many-Particle Systems (Motion of Multiple Particles in a Langevin Heat Bath)

Objectives of Chapter 3

- Up to this point, we have discussed the equation of motion and the behavior (Brownian motion) of a single particle in a Langevin heat bath [1].
- From here on, we will address the motion of multiple particles in a Langevin heat bath.
- Interactions between particles play a crucial role in determining the properties of the system formed by many particles.
  - For example, gas-liquid phase transitions occur only when attractive interactions are present.
- This time, we will explain the representative interaction potentials used in molecular simulations and how to calculate interaction forces.
- In the next Lecture, we will introduce (periodic) boundary conditions that are essential for defining the system's volume.
- We will implement a concrete many-body simulation.

3.1. An Example of Interaction Potentials

There are a wide variety of interaction potentials, but we will introduce just a few examples commonly used in molecular simulations.

(1) Lennard-Jones Potential [2, 3, 4] (See Fig. **3**)

$$U(r_{jk}) = 4\epsilon \left\{ \left(\frac{a_{jk}}{r_{jk}} \right)^{12} - \left(\frac{a_{jk}}{r_{jk}} \right)^{6} \right\} + C_{jk} \quad (r_{jk} < a_{\mathrm{cut}} \sim 2.5a) \tag{1}$$

- Explanation of Variables:
  - $a_{ij} = \frac{a_i + a_j}{2}$ (mean particle diameter: **the distance characterizing particle contact**).
  - ϵ is the energy specific to the material (also known as the LJ energy unit).
  - C_{jk} is the cutoff energy, set so that $U(a_{\mathrm{cut}}) = 0$ at the cutoff length a_{cut}.
- Properties:
  - Consists of repulsive and attractive parts.
  - Widely used as a model molecule due to its high versatility.
  - The origin of the repulsive force is the repulsion caused by the Pauli exclusion principle when atoms approach each other (although the 12th power itself has no physical basis, it reproduces experiments well).
  - The origin of the attractive force is the dipole interaction between atoms (which gives rise to the 6th power).

3.1. An Example of Interaction Potentials (2)

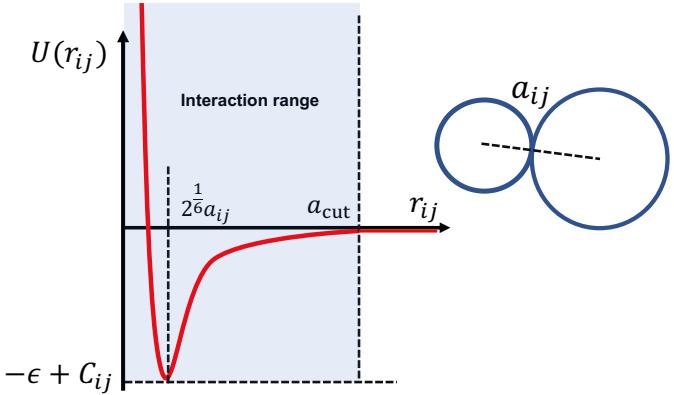

Fig. 3: Schematic of the Lennard-Jones potential. Diagram explaining the particle contact length a_{ij}.

3.1. An Example of Interaction Potentials (3)

(2) Coulomb Potential [3, 4, 5, 6]

$$U(r_{jk}) = \frac{Cq_iq_j}{\varepsilon r} \quad (r_{jk} < a_{\mathrm{cut}}) \tag{2}$$

- ε: Dielectric constant, C: Energy conversion constant.
- $q_{i(j)}$: Charge of particle $i(j)$.
- Due to the long interaction range, methods like the Ewald summation are needed to efficiently compute interactions.

3.2. Calculation of Interaction Forces

The force $\mathbf{F}_j^{\mathrm{I}}$ acting on particle j can be calculated as follows by taking the spatial derivative of the potential energy:

$$
\begin{aligned}
\mathbf{F}_j^{\mathrm{I}} = -\sum_{k\neq j} \frac{\partial U(\mathbf{r}_{jk})}{\partial \mathbf{r}_j} &= -\sum_{k\neq j} \frac{\partial r_{jk}}{\partial \mathbf{r}_j} \frac{\partial U(\mathbf{r}_{jk})}{\partial r_{jk}} \\
&= -\sum_{k\neq j} \begin{pmatrix} \frac{\partial r_{jk}}{\partial x_j} \\ \frac{\partial r_{jk}}{\partial y_j} \\ \frac{\partial r_{jk}}{\partial z_j} \end{pmatrix} \frac{\partial U(\mathbf{r}_{jk})}{\partial r_{jk}} \\
&= -\sum_{k\neq j} \begin{pmatrix} \frac{x_{jk}}{r_{jk}} \\ \frac{y_{jk}}{r_{jk}} \\ \frac{z_{jk}}{r_{jk}} \end{pmatrix} \frac{\partial U(\mathbf{r}_{jk})}{\partial r_{jk}} \\
&= -\sum_{k\neq j} \frac{\mathbf{r}_{jk}}{r_{jk}} \underbrace{\frac{\partial U(\mathbf{r}_{jk})}{\partial r_{jk}}}_{\text{Calculated analytically}}
\end{aligned}
$$

3.3. Periodic Boundary Conditions

What are Periodic Boundary Conditions?

- When dealing with the motion of many particles, **particle density** plays an important role in determining the physical properties of the system.
- To define density, it is necessary to set boundaries (such as walls).
- However, setting up walls causes the dynamics and structure of particles near the walls to differ from those in the bulk.
- In contrast, **periodic boundaries** are often introduced to prevent such effects.

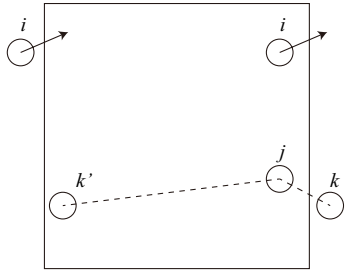

Fig. 4: Periodic boundary conditions. A particle i that crosses the right boundary reappears from the left wall. Particles k and k' are the same particle under periodic boundary conditions. The particle interacting with particle j is particle k **(the closer one)**. It should not interact with k'.

- Next, let's discuss the algorithms used under periodic boundary conditions.

3.3. Periodic Boundary Conditions (2)

- The most basic one is to store particles that have crossed the boundary back into the boundary (Listing 3).

リスト 3: Algorithm for periodic boundary conditions (coordinate placement)

```
1 void p_boundary(double (*x)[dim]){
2   for(int i=0;i<Np;i++)
3     for(int j=0;j<dim;j++)
4       x[i][j]-=L*floor(x[i][j]/L);
5 }
```

- Next, when calculating forces from the two-body potential, it is necessary to compute the distance between two particles. Let's consider an algorithm to do this under periodic boundary conditions.
- As discussed in Fig. **4**, when particles k and k' are the same particle under periodic boundary conditions, the particle interacting with particle j is particle k **(the closer one)**. The parts of particular importance are enclosed by /////.

3.3. Periodic Boundary Conditions (3)

リスト 4: 周期境界条件下での粒子間距離の計算アルゴリズム (1)

```
1 dx=x[i][0]-x[j][0];
2 dy=x[i][1]-x[j][1];
3 ////////////////////
4 if(dx>0.5*L) dx-=L;
5 if(dx<-0.5*L) dx+=L;
6 if(dy>0.5*L) dy-=L;
7 if(dy<-0.5*L) dy+=L;
8 ////////////////////
9 dr2=dx*dx+dy*dy;
```

- The more streamlined version, using the floor function, is shown in the following list 4.

3.3. Periodic Boundary Conditions (4)

リスト 5: Algorithm for calculating particle distances under periodic boundary conditions (2)

```
1
2
3
4  dx=x[i][0]-x[j][0];
5  dy=x[i][1]-x[j][1];
6  ////////////////////
7  dx-=L*floor((dx+0.5*L)/L);
8  dy-=L*floor((dy+0.5*L)/L);
9  ////////////////////
10 dr2=dx*dx+dy*dy;
```

3.4. Force Calculation under Periodic Boundary Conditions

- Below is the algorithm for calculating interactions under periodic boundary conditions.
- Here, the Lennard-Jones potential (with a cutoff distance of $2.5a$) is used.
- Using Newton's Third Law (the action-reaction principle), the force between particles ij is applied to both particles i and j.
- This reduces the number of loop calculations by half.
- The variable denoted as dUr uses the analytical result of the derivative $\frac{\partial U(r_{ij})}{\partial r_{ij}}$. If the potential is changed, this part needs to be modified.

リスト 6: Interaction calculation between particles under periodic boundary conditions

```
1  #define Np 1024
2  #define L 40.0
3  #define dim 2
4  #define cut 2.5
5  void ini_array(double (*x)[dim]){
6    for(int i=0;i<Np;i++)
7      for(int j=0;j<dim;j++)
8        x[i][j]=0.0;
9  }
10 void calc_force(double (*x)[dim],double (*f)[dim],double *a){
11   double dx,dy,dr2,dUr,w2,w6,w12,aij;
```

3.4. Force Calculation under Periodic Boundary Conditions (2)

```
12
13    ini_array(f);
14
15    for(int i=0;i<Np;i++)
16      for(int j=0;j<Np;j++){
17        if(i<j){
18             dx=x[i][0]-x[j][0];
19            dy=x[i][1]-x[j][1];
20            dx-=L*floor((dx+0.5*L)/L);
21            dy-=L*floor((dy+0.5*L)/L);
22            dr2=dx*dx+dy*dy;
23            if(dr2<cut*cut){
24                aij=0.5*(a[i]+a[j]);
25                w2=aij*aij/dr2;
26                w6=w2*w2*w2;
27                w12=w6*w6;
28                dUr=-48.*w12/dr2+24.*w6/dr2;
29                f[i][0]-=dUr*dx;
30                f[j][0]+=dUr*dx;
31                f[i][1]-=dUr*dy;
32                f[j][1]+=dUr*dy;
33             }
34            }
35          }
```

3.4. Force Calculation under Periodic Boundary Conditions (3)

```
36 }
```

- **In this case, all pairs of particles are considered, but the contribution of interactions between particles that are far apart does not need to be calculated. By excluding such contributions, the computational cost can be significantly reduced (bookkeeping calculation).**

3.5. How to Create an Initial Structure at Arbitrary Temperatures

This section explains how to create an initial structure at arbitrary temperatures.

- If the simulation starts with an inappropriate configuration (e.g., completely random configuration), particle overlap may occur, causing huge forces to act on the particles and leading to numerical instability in the calculations. To prevent this, **it is recommended that the initial configuration is such that particle overlap is minimized as much as possible (see Figure 6)**. (The crystalline configuration used in the fifth assignment is preferable.)
- Then, by starting from this configuration at a high temperature, the particles **mix well enough to generate independent samples** (see **many independent random configurations can be created**) (refer to Listing 1).
- After that, **lower the temperature to the target value and equilibrate the system**.

3.5. How to Create an Initial Structure at Arbitrary Temperatures (2)

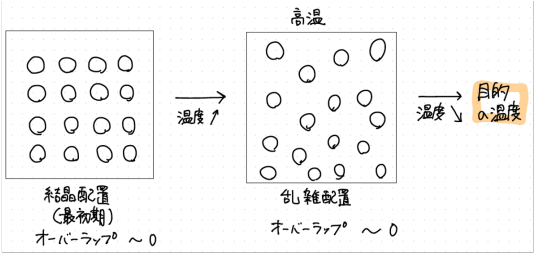

Fig. 5: How to create an initial configuration at a specific temperature.

3.6. Nondimensionalization and Discretization of the Langevin Equation

This section considers the equation of motion (Langevin equation) of multiple particles in a Langevin thermal bath. To numerically compute this, the equation is nondimensionalized.

- The motion of particle j is driven by the following Langevin equation:

$$m\frac{\mathrm{d}\mathbf{v}_j(t)}{\mathrm{d}t} = -\zeta\mathbf{v}_j(t) + \mathbf{F}_j^{\mathrm{I}}(t) + \mathbf{F}_j^{\mathrm{B}}(t) \tag{3}$$

- Here, $\mathbf{F}_j^{\mathrm{I}}(t)$ is the interaction force (potential force) acting on particle j.
- By nondimensionalizing the Langevin equation using the particle diameter a, time $t = t_0\tilde{t}$, and velocity $\mathbf{v} = \frac{a}{t_0}\tilde{\mathbf{v}}$, we obtain:

$$\underbrace{m\frac{a}{t_0^2}\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\zeta\frac{a}{t_0}\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\frac{\epsilon}{a}\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{2k_{\mathrm{B}}T\zeta\frac{1}{t_0\Delta\tilde{t}}}\mathbf{R}_{\mathrm{G}}}_{[4]} \tag{4}$$

3.6. Nondimensionalization and Discretization of the Langevin Equation (2)

- By dividing both sides of the equation by the coefficient of [1], the nondimensionalized equation is obtained:

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\frac{\zeta t_0}{m}\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\frac{t_0^2\epsilon}{ma^2}\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{2k_\mathrm{B}T\zeta\frac{t_0^3}{m^2a^2\Delta\tilde{t}}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{5}$$

- Here, we extract characteristic time scales from the nondimensionalized coefficients of each term.
- Although there are various options, we will use:

$$\boxed{t_\mathrm{d} = \frac{m}{\zeta}} \tag{6}$$

$$\boxed{t_\mathrm{B} = \frac{a^2\zeta}{k_B T}} \tag{7}$$

$$\boxed{t_\mathrm{v} = \sqrt{\frac{ma^2}{\epsilon}}} \tag{8}$$

3.6. Nondimensionalization and Discretization of the Langevin Equation (3)

to express the equation as:

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\frac{t_0}{t_\mathrm{d}}\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\frac{t_0^2}{t_\mathrm{v}^2}\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_0^3}{t_\mathrm{B}t_\mathrm{d}^2}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{9}$$

It can be seen that all coefficients are nondimensionalized.

- To set the coefficient of [3] to 1, set the unit time t_0 as:

$$t_0 = t_\mathrm{v}\left(= \sqrt{\frac{ma^2}{\epsilon}}\right) \tag{10}$$

- Then, the equation of motion becomes:

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\frac{t_\mathrm{v}}{t_\mathrm{d}}\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_\mathrm{v}^3}{t_\mathrm{B}t_\mathrm{d}^2}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{11}$$

Here, parameters appear in [2] and [4].

3.6. Nondimensionalization and Discretization of the Langevin Equation (4)

- Now, the coefficient $\frac{t_\mathrm{v}}{t_\mathrm{d}} = \zeta\sqrt{\frac{a^2}{m\epsilon}}$ in [2] represents the ratio of friction coefficient to interaction strength, and is denoted as ζ^*.
- Next, by expressing the coefficient in [4] using ζ^*:

$$\underbrace{\sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_\mathrm{v}^3}{t_\mathrm{B}t_\mathrm{d}^2}}\mathbf{R}_\mathrm{G}}_{[4]} = \sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_v^2}{t_B t_d}\frac{t_\mathrm{v}}{t_\mathrm{d}}}\mathbf{R}_\mathrm{G} \tag{12}$$

$$= \sqrt{\frac{2}{\Delta\tilde{t}}T^*\zeta^*}\mathbf{R}_\mathrm{G} \tag{13}$$

where $\boxed{\frac{t_v^2}{t_B t_d} = \frac{k_B T}{\epsilon} = T^*}$ (nondimensional temperature: the ratio of interaction to thermal energy).

3.6. Nondimensionalization and Discretization of the Langevin Equation (5)

- In summary,

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\zeta^*\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{\frac{2\zeta^* T^*}{\Delta\tilde{t}}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{14}$$

  is obtained. The parameters here are ζ^* and T^*. Notably, changing T^* can sometimes lead to changes in the phase of the system.

- Finally, by discretizing this using the semi-implicit Euler-Maruyama method, we get:

$$\tilde{\mathbf{v}}_j(\tilde{t}+\Delta\tilde{t}) = \tilde{\mathbf{v}}_j(\tilde{t}) - \zeta^*\tilde{\mathbf{v}}_j(\tilde{t})\Delta\tilde{t} + \tilde{\mathbf{F}}_j\Delta\tilde{t} + \sqrt{2\zeta^* T^*\Delta\tilde{t}}\mathbf{R}_\mathrm{G} \tag{15}$$

$$\tilde{\mathbf{r}}_j(\tilde{t}+\Delta\tilde{t}) = \tilde{\mathbf{r}}_j(\tilde{t}) + \tilde{\mathbf{v}}_j(\tilde{t}+\Delta\tilde{t})\Delta\tilde{t} \tag{16}$$

  and this can be computed on a computer.

Contents

4. Assignment 6

Assignment 6 Implementation of Multi-Particle Simulation in a Langevin Heat Bath (Phase Separation Phenomenon)

4. Assignment 6 (2)

Disperse 1024 circular disks with a diameter of a in a square plane with a side length of $L = 40a$ under periodic boundary conditions. The motion of disk j is driven by the Langevin equation $m\frac{\mathrm{d}\mathbf{v}_j(t)}{\mathrm{d}t} = -\zeta\mathbf{v}_j(t) + \mathbf{F}_j^{\mathrm{I}}(t) + \mathbf{F}_j^{\mathrm{B}}(t)$. Here, $\mathbf{F}_j^{\mathrm{B}}(t)$ is the thermal fluctuation force, which satisfies the fluctuation-dissipation theorem $\langle \mathbf{F}_{\mathrm{B}}(t)\mathbf{F}_{\mathrm{B}}(t')\rangle = 2k_BT\zeta\delta(t-t')\mathbf{1}$. $\mathbf{F}_j^{\mathrm{I}}(t)$ is the interaction force given by the following Lennard-Jones potential:

$$U(r_{jk}) = 4\epsilon\left[\left(\frac{a_{jk}}{r_{jk}}\right)^{12} - \left(\frac{a_{jk}}{r_{jk}}\right)^{6}\right] + C_{jk} \quad (r_{jk} < a_{\mathrm{cut}}), \tag{17}$$

with a cutoff length $a_{\mathrm{cut}} = 2.5a$. Now, set the time unit as $t_0 = \sqrt{ma^2/\epsilon}$, the length unit as a, and the friction coefficient as $\zeta = \sqrt{m\epsilon/a^2}$. Observe numerically the presence or absence of phase separation when varying the dimensionless temperature k_BT/ϵ. **(Advanced Task) Consider the case where the particle sizes have a distribution. In real-world scenarios, colloidal particles often have size distributions, so this is commonly considered in colloid model calculations. Implement the above simulation for the case where the particle sizes follow a normal distribution with an average of a and a standard deviation of $0.15a$.**

4. Assignment 6 (3)

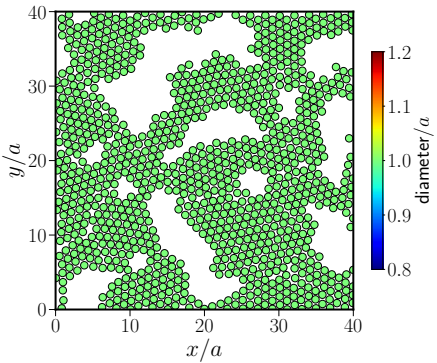

Fig. 6: This illustration shows the result when all particle diameters are a at a dimensionless temperature of $T^* = 0.2$. Phase separation between gas and liquid (solid) occurs.

Supplement: The "langevin_many.cpp" uploaded to the Github repository is a sample program for this simulation. Please refer to it as needed.

Contents

References and Websites

[1] Zwanzig R (2001) Nonequilibrium Statistical Mechanics.
(Oxford University Press, Oxford ; New York).

[2] Verlet L (1967) Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules.
Physical Review 159(1):98–103.

[3] Allen MP, Tildesley DJ (2017) Computer Simulation of Liquids: Second Edition.
(Oxford University Press).

[4] Frenkel D, Smit B (2001) Understanding Molecular Simulation: From Algorithms to Applications.
(Elsevier).

[5] Kolafa J, Perram JW (1992) Cutoff Errors in the Ewald Summation Formulae for Point Charge Systems.
Molecular Simulation 9(5):351–368.

[6] Di Pierro M, Elber R, Leimkuhler B (2015) A Stochastic Algorithm for the Isobaric–Isothermal Ensemble with Ewald Summations for All Long Range Forces.
Journal of Chemical Theory and Computation 11(12):5624–5637.