Data Science Course Group: Simulation Practice
Session 7 Lecture Materials

Instructor: TatsuSaki Kawasaki

Department of Physics, Graduate School of Science, Nagoya University, Nonequilibrium Physics Laboratory (R Lab)

Last update: August 22, 2024

Contents

Contents

1. Lecture Schedule

- This practice session will be conducted during the first semester of spring.
- Lecture materials will be uploaded by 11:00 AM on the day of each lecture.
- Schedule
  1. 4/15: Session 1
  2. 4/22: Session 2
  3. 4/30 (Tue): Session 3
  4. 5/13: Session 4 (Midterm report assignment release)
  5. 5/20: Session 5
  6. 5/27: Session 6 (Midterm report submission deadline)
  7. 6/03: No class
  8. **6/10: Session 7**
  9. 6/17: Session 8 **Final report assignment release**
  10. 6/24: Session 9 (Makeup class)

1.1. Syllabus

The following topics are planned for this practice session (subject to change based on progress).

1. Introduction
   - How to use C (C++) (mainly for numerical computation)
   - How to use Python (for data analysis and plotting)
   - Principles of numerical computation
   - Round-off errors
   - Dimensionless quantities in scientific computation
2. Numerical solutions to ordinary differential equations: Examples of damped oscillations and harmonic oscillators
   - Numerical integration of differential equations
   - Euler's method
3. Brownian motion of a single particle
   - Langevin equation (stochastic differential equation)
   - Generation of normal random numbers
   - Euler-Maruyama method
   - Time averaging and ensemble averaging
   - Calculation of diffusion coefficients
4. Brownian motion of multiple particles
   - Calculation methods for interaction forces
   - Simulations of nonequilibrium systems: Example of phase separation phenomena
5. Molecular dynamics simulations of multiple particle systems
   - Position Verlet method and velocity Verlet method
   - Conservation laws in multi-particle systems (momentum, energy, angular momentum)
6. Monte Carlo method
   - Review of statistical mechanics
   - Markov chain Monte Carlo method

Contents

2. Assignment 6

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

Distribute 1024 disks with a diameter a on 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, satisfying the fluctuation-dissipation theorem: $\langle \mathbf{F}_j^{\mathrm{B}}(t)\mathbf{F}_k^{\mathrm{B}}(t')\rangle = 2k_BT\zeta\delta(t-t')\delta_{jk}\mathbf{1}$. $\mathbf{F}_j^{\mathrm{I}}(t)$ is the interaction force, given by the Lennard-Jones potential:

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

The cutoff length is set to $a_{\mathrm{cut}} = 2.5a$. Let the unit of time be $t_0 = \sqrt{ma^2/\epsilon}$, the unit of length be a, and the friction coefficient be $\zeta = \sqrt{m\epsilon/a^2}$. Numerically observe the presence or absence of phase separation as the dimensionless temperature k_BT/ϵ is varied.

Explanation

The sample program for Assignment 6, "langevin_many.cpp", is shown in Listing 1 for reference. Here, a crystalline configuration is randomly mixed at a dimensionless temperature $T^* = 5.0$ and then cooled to the

2. Assignment 6 (2)

target T^* (specified by the parameter *temp*). The behavior at different target temperatures, $T^* = 0.2, 0.4, 0.6, 1.0$, is illustrated in Fig. **1**.

リスト 1: 自主課題 6 のサンプルプログラム “langevin_many.cpp”. 以下の GitHub リポジトリより取得可能[リンク].

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <fstream>
#include <cfloat>
#include "BM.h"

#define Np 1024
#define L 40.0
#define tmax 100
#define dt 0.01
#define temp 0.2
#define dim 2
#define cut 2.5
#define polydispersity 0.0

void ini_coord_square(double (*x)[dim]){
```

2. Assignment 6 (3)

```
19    int num_x = (int)sqrt(Np)+1;
20    int num_y = (int)sqrt(Np)+1;
21    int i,j,k=0;
22    double shift;
23    for(j=0;j<num_y;j++){
24      for(i=0;i<num_x;i++){
25        x[i+num_x*j][0] = i*L/(double)num_x;
26        x[i+num_x*j][1] = j*L/(double)num_y;
27        k++;
28        if(k==Np)
29          break;
30      }
31         if(k==Np)
32          break;
33    }
34  }
35
36  void set_diameter(double *a){
37    for(int i=0;i<Np;i++)
38      a[i]=1.0+polydispersity*gaussian_rand();
39  }
40
41  void p_boundary(double (*x)[dim]){
42    for(int i=0;i<Np;i++)
```

2. Assignment 6 (4)

```
43      for(int j=0;j<dim;j++)
44        x[i][j]-=L*floor(x[i][j]/L);
45  }
46
47  void ini_array(double (*x)[dim]){
48    for(int i=0;i<Np;i++)
49      for(int j=0;j<dim;j++)
50        x[i][j]=0.0;
51  }
52
53  void calc_force(double (*x)[dim],double (*f)[dim],double *a){
54    double dx,dy,dr2,dUr,w2,w6,w12,aij;
55
56    ini_array(f);
57
58    for(int i=0;i<Np;i++)
59      for(int j=0;j<Np;j++){
60        if(i<j){
61          dx=x[i][0]-x[j][0];
62          dy=x[i][1]-x[j][1];
63          dx-=L*floor((dx+0.5*L)/L);
64          dy-=L*floor((dy+0.5*L)/L);
65          dr2=dx*dx+dy*dy;
66          if(dr2<cut*cut){
```

2. Assignment 6 (5)

```
67            aij=0.5*(a[i]+a[j]);
68            w2=aij/dr2;
69            w6=w2*w2*w2;
70            w12=w6*w6;
71            dUr=-48.*w12/dr2+24.*w6/dr2;
72            f[i][0]-=dUr*dx;
73            f[j][0]+=dUr*dx;
74            f[i][1]-=dUr*dy;
75            f[j][1]+=dUr*dy;
76            }
77          }
78      }
79  }
80
81  void eom(double (*v)[dim],double (*x)[dim],double (*f)[dim],double temp0){
82    double zeta=1.0;
83    double fluc=sqrt(2.*zeta*temp0*dt);
84    for(int i=0;i<Np;i++)
85      for(int j=0;j<dim;j++){
86        v[i][j]+=-zeta*v[i][j]*dt+f[i][j]*dt+fluc*gaussian_rand();
87        x[i][j]+=v[i][j]*dt;
88      }
89  }
90
```

2. Assignment 6 (6)

```
91  void output(double (*x)[dim],double *a){
92    char filename[128];
93    std::ofstream file;
94    static int j=0;
95    sprintf(filename,"coord_T%.3f_%d.dat",temp,j);
96    file.open(filename);
97    for(int i=0;i<Np;i++)
98      file <<x[i][0]<<"\t"<<x[i][1]<<"\t"<<a[i]<<std::endl;
99    file.close();
100   j++;
101 }
102
103 int main(){
104   double x[Np][dim],v[Np][dim],f[Np][dim],a[Np];
105   double tout=0.0;
106   int j=0;
107   set_diameter(a);
108   ini_coord_square(x);
109   ini_array(v);
110
111   while(j*dt < 10.0){
112     j++;
113     calc_force(x,f,a);
114     eom(v,x,f,5.0);
```

2. Assignment 6 (7)

```
115     }
116     j=0;
117     while(j*dt < tmax){
118       j++;
119       calc_force(x,f,a);
120       eom(v,x,f,temp);
121       p_boundary(x);
122       if(j*dt >= tout){
123         output(x,a);
124         tout+=10.;
125       }
126     }
127     return 0;
128   }
```

2. Assignment 6 (8)

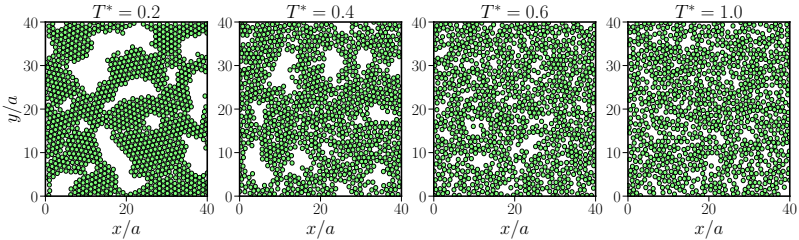

Fig. 1: Particle distribution when the dimensionless temperature T^* is varied, with the length of one side of the periodic boundary being $L = 40a$ and all particle diameters being a. Phase separation can be observed, especially at low temperatures. Additionally, phase separation seems to start occurring around $T^* = 1$. **[Note]** $T^* = 1$ corresponds to $\epsilon = k_\mathrm{B}T$, where the depth of the potential well is comparable to the thermal energy.

Contents

3 Molecular Dynamics Simulation

Molecular Dynamics Simulation (Classical MD)

- In this chapter, we will cover simulations of systems where atoms and molecules are driven by Newton's equations of motion: **Molecular Dynamics (MD) Simulation**.
- Such systems, where quantities like momentum, energy, and angular momentum are conserved, are referred to as Hamiltonian systems.
- Therefore, fluctuating forces and dissipation are not explicitly included, and the temperature does not remain constant (the system generally reaches a steady state).
- The statistical ensemble for such a system is called the NVE ensemble, or the microcanonical ensemble.

3.1 Equations of Motion and Their Nondimensionalization

Equations of Motion and Their Nondimensionalization

- The equation of motion for a classical particle j with mass m interacting with other particles through an interparticle potential U is given by,

$$m\ddot{\mathbf{r}}_j = -\frac{\partial U(\{\mathbf{r}\})}{\partial \mathbf{r}_j} \tag{2}$$

- The equation of motion itself is much simpler compared to the Langevin equation, and nondimensionalization for computational implementation is relatively straightforward.
- Specifically, by choosing the unit of length as a, the unit of energy as ϵ, and the unit of time as t_0, the equation of motion can be rewritten in nondimensionalized variables, denoted with tildes, as

$$m\frac{a}{t_0^2}\ddot{\tilde{\mathbf{r}}}_j = -\frac{\epsilon}{a}\frac{\partial \tilde{U}(\{\tilde{\mathbf{r}}\})}{\partial \tilde{\mathbf{r}}_j} \tag{3}$$

- Dividing both sides by $m\frac{a}{t_0^2}$ gives

$$\ddot{\tilde{\mathbf{r}}}_j = -\frac{\epsilon t_0^2}{ma^2}\frac{\partial \tilde{U}(\{\tilde{\mathbf{r}}\})}{\partial \tilde{\mathbf{r}}_j} \tag{4}$$

3.1 Equations of Motion and Their Nondimensionalization (2)

- By choosing the unit of time as

$$t_0 = \sqrt{\frac{ma^2}{\epsilon}} \tag{5}$$

the equation of motion becomes

$$\ddot{\tilde{\mathbf{r}}}_j = -\frac{\partial \tilde{U}(\{\tilde{\mathbf{r}}\})}{\partial \tilde{\mathbf{r}}_j} \tag{6}$$

which results in a parameter-free equation.

- As mentioned earlier, various physical quantities are conserved in this system. Therefore, if the discretization is not handled carefully, the accumulation of errors can lead to the violation of conservation laws, resulting in incorrect computational results.
- The following sections introduce a symplectic and highly accurate discretization method (numerical integration) to prevent error accumulation.

3.2 Position Verlet Method

Position Verlet Method

In this section, we introduce the Verlet method, a highly accurate discretization technique. Specifically, the method described below is referred to as the **Position Verlet Method** (original paper [1], reference books [2, 3, 4]).

- The Verlet method is a discretization of the equation of motion using the central difference method.
- In the central difference method, performing the forward difference and backward difference at time t yields:

$$\mathbf{r}_j(t+\Delta t) = \mathbf{r}_j(t) + \dot{\mathbf{r}}_j(t)\Delta t + \frac{1}{2!}\ddot{\mathbf{r}}_j(t)(\Delta t)^2 + \frac{1}{3!}\dddot{\mathbf{r}}_j(t)(\Delta t)^3 + O((\Delta t)^4) \tag{7}$$

$$\mathbf{r}_j(t-\Delta t) = \mathbf{r}_j(t) - \dot{\mathbf{r}}_j(t)\Delta t + \frac{1}{2!}\ddot{\mathbf{r}}_j(t)(\Delta t)^2 - \frac{1}{3!}\dddot{\mathbf{r}}_j(t)(\Delta t)^3 + O((\Delta t)^4). \tag{8}$$

- Adding Eq. (7) and Eq. (8) gives:

$$\begin{aligned} \mathbf{r}_j(t+\Delta t) &= 2\mathbf{r}_j(t) - \mathbf{r}_j(t-\Delta t) + \ddot{\mathbf{r}}_j(t)(\Delta t)^2 + O((\Delta t)^4), \\ &= 2\mathbf{r}_j(t) - \mathbf{r}_j(t-\Delta t) + \underbrace{\frac{\mathbf{F}_j(t)}{m}(\Delta t)^2}_{(*)\text{very small}} + O((\Delta t)^4) \end{aligned} \tag{9}$$

3.2 Position Verlet Method (2)

- Simultaneously, subtracting Eq. (8) from Eq. (7) gives the velocity relation:

$$\dot{\mathbf{r}}_j(t) = \frac{\mathbf{r}_j(t+\Delta t) - \mathbf{r}_j(t-\Delta t)}{2\Delta t} + O((\Delta t)^2) \tag{10}$$

- Solving this, we obtain a precision of $O((\Delta t)^3)$ for position and $O(\Delta t)$ for velocity. Although the precision of velocity is not high, it does not lead to error accumulation. The higher precision of the position equation ensures that error accumulation is minimized. This type of numerical integration method is known as the **Position Verlet Method**.
- **However, the Position Verlet Method has numerical issues.**
- In Eq. (9), the term marked (*) is extremely small compared to the other terms.
- This leads to risks of significant rounding errors, so the Position Verlet Method is generally not used in molecular dynamics calculations.
- While the mathematical structure remains the same, a method called the **Velocity Verlet Method** is used to avoid these risks. The Velocity Verlet Method is explained below.

3.3 Velocity Verlet Method

Velocity Verlet Method [2, 3, 4]

- Here, we explain the Verlet method that avoids significant rounding errors (Velocity Verlet Method).
- First, the equation from Eq. (9) can be rearranged as follows:

$$
\begin{aligned}
\mathbf{r}_j(t+\Delta t) &= 2\mathbf{r}_j(t) - \mathbf{r}_j(t-\Delta t) + \frac{\mathbf{F}_j(t)}{m}(\Delta t)^2 + O((\Delta t)^4) \\
&= \mathbf{r}_j(t) + \frac{\mathbf{r}_j(t)}{2} + \frac{\mathbf{r}_j(t)}{2} - \mathbf{r}_j(t-\Delta t) + \frac{\mathbf{F}_j(t)}{m}(\Delta t)^2 + O((\Delta t)^4) \\
&= \mathbf{r}_j(t) + \frac{1}{2}\left[2\mathbf{r}_j(t-\Delta t) - \mathbf{r}_j(t-2\Delta t) + \frac{\mathbf{F}_j(t-\Delta t)}{m}(\Delta t)^2\right] \\
&+ \frac{1}{2}\mathbf{r}_j(t) - \mathbf{r}_j(t-\Delta t) + \frac{\mathbf{F}_j(t)}{m}(\Delta t)^2 + O((\Delta t)^4) \\
&= \mathbf{r}_j(t) + \boxed{\mathbf{v}_j(t-\Delta t)\Delta t} + \frac{(\Delta t)^2}{2m}[\mathbf{F}_j(t) + \mathbf{F}_j(t-\Delta t)] + \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t) + \boxed{\frac{(\Delta t)^3}{3!}\dddot{\mathbf{r}}_j(t)} + O((\Delta t)^4) \\
&= \mathbf{r}_j(t) + \mathbf{v}_j(t)\Delta t + \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t) + \frac{(\Delta t)^3}{3!}\dddot{\mathbf{r}}_j(t) + O((\Delta t)^4)
\end{aligned}
\tag{11}
$$

3.3 Velocity Verlet Method (2)

- Here,

$$\frac{1}{2}\left[\mathbf{r}_j(t) - \mathbf{r}_j(t-2\Delta t)\right] - \frac{(\Delta t)^3}{3!}\dddot{\mathbf{r}}_j(t) = \mathbf{v}_j(t-\Delta t)\Delta t + O((\Delta t)^4) \tag{12}$$

  is used.

- This gives the following relation:

$$\mathbf{v}_j(t) = \mathbf{v}_j(t-\Delta t) + \frac{\Delta t}{2m}[\mathbf{F}_j(t) + \mathbf{F}_j(t-\Delta t)] + O((\Delta t)^3) \tag{13}$$

- The main computation for time evolution is performed with respect to the velocity, thereby avoiding significant rounding errors.
- Additionally, the precision of the velocity here is $O((\Delta t)^2)$, which is quite high.
- Below are the key equations for the Velocity Verlet method:

3.3 Velocity Verlet Method (3)

(Summary) Velocity Verlet Method

$$\mathbf{v}_j(t+\Delta t) = \mathbf{v}_j(t) + \frac{\Delta t}{2m}\{\mathbf{F}_j(t+\Delta t) + \mathbf{F}_j(t)\} \tag{14}$$

$$\mathbf{r}_j(t+\Delta t) = \mathbf{r}_j(t) + \mathbf{v}_j(t)\Delta t + \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t) \tag{15}$$

However, **terms of third order and below are omitted in the position update due to potential rounding errors.** Although personally, I am curious to see if including these third-order terms would change the results, I have not yet investigated it.

3.3 Velocity Verlet Method (4)

When coding, you can perform the calculations extremely efficiently by following these steps:

Calculation Steps in the Velocity Verlet Method

(1) $\mathbf{r}_j(t+\Delta t) = \mathbf{r}_j(t) + \mathbf{v}_j(t)\Delta t + \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t)$

(2) $\mathbf{v}'_j(t+\Delta t) = \mathbf{v}_j(t) + \frac{\Delta t}{2m}\mathbf{F}_j(t)$

(3) Using $\mathbf{r}_j(t+\Delta t)$, compute $\mathbf{F}_j(t+\Delta t)$.

(4) $\mathbf{v}_j(t+\Delta t) = \mathbf{v}'_j(t+\Delta t) + \frac{\Delta t}{2m}\mathbf{F}_j(t+\Delta t)$

(5) Return to (1) with updated time.

3.3 Velocity Verlet Method (5)

Time-Reversal Symmetry

Starting from the discretized equations of the Velocity Verlet method (Equation 15), it is possible to trace back the same trajectory by reversing time from $t + \Delta t \to t$. First, by rearranging the terms for velocity, we have:

$$\mathbf{v}_j(t) = \mathbf{v}_j(t + \Delta t) + \frac{-\Delta t}{2m}\{\mathbf{F}_j(t + \Delta t) + \mathbf{F}_j(t)\} \tag{16}$$

Next, for position:

$$\begin{aligned}
\mathbf{r}_j(t) &= \mathbf{r}_j(t + \Delta t) - \mathbf{v}_j(t)\Delta t - \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t) \\
&= \mathbf{r}_j(t + \Delta t) - \mathbf{v}_j(t + \Delta t)\Delta t + \frac{(\Delta t)^2}{2m}[\mathbf{F}_j(t + \Delta t) + \mathbf{F}_j(t)] - \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t) \\
&= \mathbf{r}_j(t + \Delta t) - \mathbf{v}_j(t + \Delta t)\Delta t + \frac{(\Delta t)^2}{2m}\mathbf{F}_j(t + \Delta t)
\end{aligned} \tag{17}$$

which allows for reversing the trajectory.

3.4 Seventh Assignment

Seventh Assignment | Implementation of Molecular Dynamics Simulation (Classical MD)

Consider a two-dimensional system confined within a periodic boundary with side length $L = 40a$, consisting of $N = 1024$ identical circular particles with diameter a. The inter-particle potential used here is repulsive only:

$$U(r_{jk}) = \epsilon \left(\frac{a_{jk}}{r_{jk}} \right)^{12} + C_{jk} \quad (r_{jk} < a_{\mathrm{cut}})$$

where the cutoff length is set to $a_{\mathrm{cut}} = 3.0a$. Using a as the unit of length, ϵ as the unit of energy, and $t_0 = \sqrt{ma^2/\epsilon}$ as the unit of time, answer the following questions:

(1) Using the Langevin heat bath constructed in Assignment 6, obtain the position coordinates $\{\mathbf{r}_j\}$ and velocities $\{\mathbf{v}_j\}$ of each particle in the thermal equilibrium state at a dimensionless temperature $T^* = k_B T/\epsilon = 0.9$.

(2) Using the coordinates and velocities obtained in (1) as initial conditions, perform a molecular dynamics simulation. Show that the mechanical energy $U + K$ (where K is the kinetic energy and U is the potential energy) is conserved over time.

The sample program for the seventh assignment (without list-based optimization), “md.cpp”, can be obtained from the GitHub repository **[Link]**. Please refer to and study it as needed.

3.4 Seventh Assignment (2)

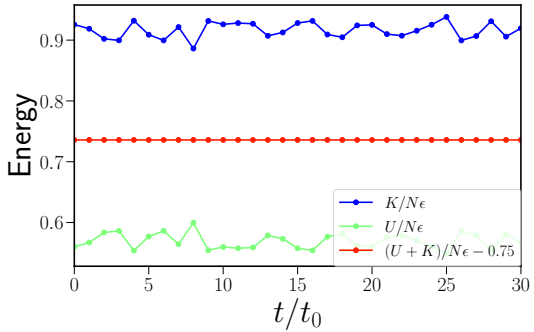

Fig. 2: Example solution for Assignment 7(2). It can be observed that the mechanical energy is conserved. Note that the energy per particle is displayed here.

Contents

Reference

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

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

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

[4] Okazaki S, Yoshii N (2011) コンピュータ・シミュレーションの基礎（第 2 版） - 株式会社 化学同人.
(化学同人).