

Data Science Course Group: Simulation Exercise

8th Session Lecture Materials

Instructor: Takeshi Kawasaki

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

Last update: August 22, 2024

Contents

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References

Contents

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References

1. Lecture Schedule

- This exercise will be conducted during the Spring 1st Term.
- Lecture materials will be uploaded by 11:00 AM on the scheduled day of each lecture.
- Schedule
 - 1 4/15: 1st Session
 - 2 4/22: 2nd Session
 - 3 4/30 (Tue): 3rd Session
 - 4 5/13: 4th Session (Midterm Report Assignment Released)
 - 5 5/20: 5th Session
 - 6 5/27: 6th Session (Midterm Report Submission Deadline)
 - 7 6/03: No Class
 - 8 6/10: 7th Session
 - 9 6/17: 8th Session (Final Report Assignment Released)
 - 10 6/24: 9th Session (Make-up Class)

1.1. Syllabus

The following topics are planned to be covered in this exercise (subject to change depending on progress):

1 Introduction

- Usage of C(C++) (mainly for numerical computation)
- Usage of Python (for data analysis and plotting)
- Principles of numerical computation
- Loss of significance
- Nondimensionalization in scientific computation

2 Numerical solutions of ordinary differential equations: Examples of damped oscillations and harmonic oscillators

- Numerical integration of differential equations
- Euler method

3 Brownian motion of a single particle

- Langevin equation (stochastic differential equation)
- Method of generating normal random numbers
- Euler-Maruyama method
- Time average and ensemble average
- Calculation of the diffusion coefficient

4 Brownian motion of multi-particle systems

- Method of calculating interaction forces
- Simulation of nonequilibrium systems: Example of phase separation

5 Molecular dynamics simulation of multi-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

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References

2. Explanation of the 7th Assignment

Assignment 7 Implementation of Molecular Dynamics Simulation (Classical MD)

Consider a 2D system confined within a periodic boundary of length $L = 40a$ on each side, consisting of identical disk particles with a number of particles $N = 1024$, diameter a , and mass m . The interparticle potential used here is a repulsive-only potential

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

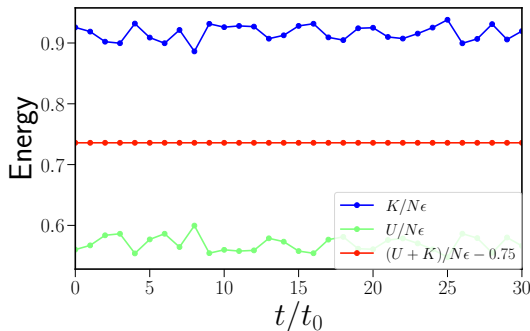
where the cutoff length is $a_{\text{cut}} = 3.0a$. Let the units of length be a , energy be ϵ , and time be $t_0 = \sqrt{ma^2/\epsilon}$. Answer the following questions:


- (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 thermally equilibrated state at a dimensionless temperature $T^* = k_B T / \epsilon = 0.9$.
- (2) Using the coordinates and velocities obtained in (1) as the initial conditions, execute a molecular dynamics simulation, and show that the mechanical energy $U + K$, where K is the kinetic energy and U is the potential energy, is conserved over time.
- (3) **(Advanced Assignment – Optimization) Accelerate the interaction calculation using the list (ledger) introduced in Assignment 5. The strategy is to create a list of particles within the radius $r_{\text{cut}} + r_{\text{skin}}$, and keep using this list until the maximum displacement of any particle exceeds $r_{\text{skin}}/2$.**

Explanation

2. Explanation of the 7th Assignment (2)

The sample program for Assignment 7 (which does not include the optimization using lists), "md.cpp", is shown in List 1 below and can be obtained from the GitHub repository at [\[Link\]](#).



 **1:** Example of the solution for Assignment 7(2). It can be seen that the mechanical energy is conserved. Here, the energy per particle is shown.

2. Explanation of the 7th Assignment (3)

リスト 1: Sample program of the 7th Assignment“md.cpp”. GitHub [\[Link\]](#).

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <iomanip>
5  #include <iostream>
6  #include <fstream>
7  #include <cfloat>
8  #include "BM.h"
9
10 #define Np 1024
11 #define L 40.0
12 #define teq 100
13 #define tmax 30
14 #define dtmd 0.001
15 #define dtbd 0.01
16 #define temp 0.9
17 #define dim 2
18 #define cut 3.0
19 #define polydispersity 0.0
20
21 void ini_coord_square(double (*x)[dim]){
22     int num_x = (int)sqrt(Np)+1;
23     int num_y = (int)sqrt(Np)+1;
24     int i,j,k=0;
25     for(j=0;j<num_y;j++){
26         for(i=0;i<num_x;i++){

```

2. Explanation of the 7th Assignment (4)

```

27     x[i+num_x*j][0] = i*L/(double)num_x;
28     x[i+num_x*j][1] = j*L/(double)num_y;
29     k++;
30     if(k>=Np)
31         break;
32     }
33     if(k>=Np)
34         break;
35     }
36 }
37
38 void set_diameter(double *a){
39     for(int i=0;i<Np;i++)
40         a[i]=1.0+polydispersity*gaussian_rand();
41 }
42
43 void p_boundary(double (*x)[dim]){
44     for(int i=0;i<Np;i++)
45         for(int j=0;j<dim;j++)
46             x[i][j]-=L*floor(x[i][j]/L);
47 }
48
49 void ini_array(double (*x)[dim]){
50     for(int i=0;i<Np;i++)
51         for(int j=0;j<dim;j++)
52             x[i][j]=0.0;
53 }

```

2. Explanation of the 7th Assignment (5)

```

54
55 void calc_force(double (*x)[dim],double (*f)[dim],double *a,double *U){
56     double dx,dy,dr2,dUr,w2,w6,w12,aij;
57     double Ucut=1./pow(cut,12);
58     ini_array(f);
59     *U=0;
60     for(int i=0;i<Np;i++)
61         for(int j=0;j<Np;j++){
62             if(i<j){
63                 dx=x[i][0]-x[j][0];
64                 dy=x[i][1]-x[j][1];
65                 dx-=L*floor((dx+0.5*L)/L);
66                 dy-=L*floor((dy+0.5*L)/L);
67                 dr2=dx*dx+dy*dy;
68                 if(dr2<cut*cut){
69                     aij=0.5*(a[i]+a[j]);
70                     w2=aij*aij/dr2;
71                     w6=w2*w2*w2;
72                     w12=w6*w6;
73                     dUr=-12.*w12/dr2;
74                     f[i][0]-=dUr*dx;
75                     f[j][0]+=dUr*dx;
76                     f[i][1]-=dUr*dy;
77                     f[j][1]+=dUr*dy;
78                     *U+=w12-Ucut;
79                 }
80             }

```

2. Explanation of the 7th Assignment (6)

```

81     }
82 }
83
84 void eom_langevin(double (*v)[dim], double (*x)[dim], double (*f)[dim], double *a, double *U, double dt, double
    temp0){
85     double zeta=1.0;
86     double fluc=sqrt(2.*zeta*temp0*dt);
87
88     calc_force(x, f, a, &(*U));
89     for(int i=0; i<Np; i++){
90         for(int j=0; j<dim; j++){
91             v[i][j]+=-zeta*v[i][j]*dt+f[i][j]*dt+fluc*gaussian_rand();
92             x[i][j]+=v[i][j]*dt;
93         }
94     p_boundary(x);
95 }
96
97 void eom_md(double (*v)[dim], double (*x)[dim], double (*f)[dim], double *a, double *U, double dt){
98     for(int i=0; i<Np; i++){
99         for(int j=0; j<dim; j++){
100             x[i][j]+=v[i][j]*dt+0.5*f[i][j]*dt*dt;
101             v[i][j]+=0.5*f[i][j]*dt;
102         }
103     calc_force(x, f, a, &(*U));
104     for(int i=0; i<Np; i++){
105         for(int j=0; j<dim; j++){
106             v[i][j]+=0.5*f[i][j]*dt;

```

2. Explanation of the 7th Assignment (7)

```

107     }
108     p_boundary(x);
109 }
110
111 void output(int k, double (*v)[dim], double U){
112     char filename[128];
113     double K=0.0;
114
115     std::ofstream file;
116     sprintf(filename, "energy.dat");
117     file.open(filename, std::ios::app); //append
118     for(int i=0; i<Np; i++)
119         for(int j=0; j<dim; j++)
120             K+=0.5*v[i][j]*v[i][j];
121
122     std::cout<< std::setprecision(6)<<k*dtmd<<"\t"<<K/Np<<"\t"<<U/Np<<"\t"<<(K+U)/Np<<std::endl;
123     file<< std::setprecision(6)<<k*dtmd<<"\t"<<K/Np<<"\t"<<U/Np<<"\t"<<(K+U)/Np<<std::endl;
124     file.close();
125 }
126
127 int main(){
128     double x[Np][dim], v[Np][dim], f[Np][dim], a[Np];
129     double tout=0.0, U;
130     int j=0;
131     set_diameter(a);
132     ini_coord_square(x);
133     ini_array(v);

```

2. Explanation of the 7th Assignment (8)

```
134
135 while(j*dtbd < 10.){
136     j++;
137     eom_langevin(v,x,f,a,&U,dtbd,5.0);
138 }
139
140 j=0;
141 while(j*dtbd < teq){
142     j++;
143     eom_langevin(v,x,f,a,&U,dtbd,temp);
144 }
145 j=0;
146 while(j*dtmd < tmax){
147     j++;
148     eom_md(v,x,f,a,&U,dtmd);
149     if(j*dtmd >= tout){
150         output(j,v,U);
151         tout+=1.;
152     }
153 }
154 return 0;
155 }
```

Contents

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems**
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References

3 Monte Carlo Method in Particle Systems

Monte Carlo Method in Particle Systems

- The Monte Carlo method (MC method) in particle systems is a method used to determine the distribution of particles in thermal equilibrium and the associated statistical quantities by using random numbers.
- In MD (Molecular Dynamics) calculations, the calculation of interparticle forces is crucial, while in the MC method, the calculation of energy is important.
- Except for this part, there are many overlapping aspects between the computational methods learned in Brownian dynamics (Langevin heat bath method) and molecular dynamics.
- The MC method primarily generates ensembles such as NVT , NPT , and μVT based on statistical mechanics.
- Below, the basic principles of the MC method are explained.

3.1 Markov Chain

Markov Chain [1, 2, 3]

- Suppose that in ℓ steps, the system transitions from state i to state k , and after another m steps, it reaches state j .
- If the **transition probability** between these states does **not depend on the history leading to that state**, this process is called a **Markov chain**.
- In particular, in the limit as m becomes large, the **probability of entering each state** becomes **independent of the initial state**. This is known as the **Markov chain's limit theorem**.

Markov Chain Limit Theorem and Stationary State

- In a Markov chain, the probability of transitioning from state i to state j in ℓ steps is denoted by $P_{ij}(\ell)$, and the following relationship holds (the **Chapman-Kolmogorov equation/Smoluchowski equation**):

$$P_{ij}(\ell + m) = \sum_{k=1}^M P_{ik}(\ell) P_{kj}(m) \quad (1)$$

3.1 Markov Chain (2)

- In particular, since a thermal equilibrium state is obtained after a long time (many steps), it does not depend on the initial state. This means that

$$P_{ij}(\infty) = \sum_{k=1}^M P_{ik}(\infty) P_{kj}(1) \quad (2)$$

Now, if we consider

$$P_{ij}(\infty) \rightarrow \Pi_j \quad (3)$$

$$P_{ik}(\infty) \rightarrow \Pi_k \quad (4)$$

then Π_j represents the probability of the system being in state j in the stationary state. In a Markov chain, $P_{kj}(1)$ does not depend on the previous state transition path, so it can be written as $P_{kj}(1) = P_{kj}$. Therefore,

$$\Pi_j = \sum_{k=1}^M \Pi_k P_{kj} \quad (5)$$

holds. Here,

$$\sum_{k=1}^M P_{jk} = 1 \quad (6)$$

3.1 Markov Chain (3)

Therefore, equation (5) shows that

$$\sum_{k=1}^M \Pi_j P_{jk} = \sum_{k=1}^M \Pi_k P_{kj} \quad (7)$$

indicating that the inflow and outflow of probability into state j are balanced.

- Moreover, this stationary state relationship corresponds to the master equation, which describes the time evolution of the probability of being in state j :

$$\frac{\partial \Pi_j}{\partial t} = \sum_{k=1}^M (\Pi_k P_{kj} - \Pi_j P_{jk}) \quad (8)$$

where $\frac{\partial \Pi_j}{\partial t} = 0$ in the stationary state.

3.2 Microscopic Reversibility (Detailed Balance) Equation

Microscopic Reversibility (Detailed Balance) Equation

- Next, we consider a special thermal equilibrium state within the stationary state. In thermal equilibrium, the detailed balance (microscopic reversibility) equation, which is the balance relation for all microscopic states, holds:

$$\Pi_i P_{ij} = \Pi_j P_{ji} \quad (9)$$

Of course, this detailed balance equation ensures the stationary state relationship discussed in the previous section.

- According to statistical mechanics, Π_i , which generates an NVT ensemble, is given by the canonical distribution:

$$\Pi_i = \frac{\exp(-\beta U_N(\mathbf{r}^N; i))}{Z_N} \quad (10)$$

where Z_N is the partition function:

$$Z_N = \sum_i \exp[-\beta \{U_N(\mathbf{r}^N; i)\}] \quad (11)$$

3.2 Microscopic Reversibility (Detailed Balance) Equation (2)

- In an NPT ensemble where pressure and temperature are constant, the probability Π_i is given by

$$\Pi_i = \frac{\exp[-\beta\{U_N(\mathbf{r}^N; i) + PV(\mathbf{r}^N; i)\}]}{Y_N} \quad (12)$$

where Y_N is the partition function for this case:

$$Y_N = \sum_i \exp[-\beta\{U_N(\mathbf{r}^N; i) + PV(\mathbf{r}^N; i)\}] \quad (13)$$

3.3 Metropolis Method

Metropolis Method [4]

The Metropolis method is a technique in Monte Carlo simulations that provides statistical mechanics criteria for accepting or rejecting new states generated by random numbers. This weighted sampling method allows for the approximate calculation of distribution functions.

- For various P_{ij} , it is possible to determine the probability Π_i . In this section, by acknowledging the principle of detailed balance, the transition probability P_{ij} that most simply achieves thermal equilibrium is given by

$$P_{ij} = \begin{cases} \frac{1}{M} & (\Pi_j > \Pi_i) \quad \text{accept} \\ \frac{1}{M} \frac{\Pi_j}{\Pi_i} & (\Pi_j \leq \Pi_i) \quad \text{accept} \\ \frac{1}{M} (1 - \frac{\Pi_j}{\Pi_i}) & (\Pi_j \leq \Pi_i) \quad \text{reject} \end{cases} , \quad (14)$$

This rule is known as the **Metropolis Method** [4].

3.3 Metropolis Method (2)

- In the Metropolis method, when $\Pi_j > \Pi_i$, as shown in Figure 2, the transition from a state with higher energy to a state with lower energy always occurs. That is,

$$P_{ij} = \frac{1}{M},$$

is used.

On the other hand, when $\Pi_j < \Pi_i$, the transition probability from a state j to a state i corresponding to a transition from a higher energy state to a lower energy state is

$$P_{ji} = \frac{1}{M},$$

Therefore, from the detailed balance equation, the transition probability P_{ij} is

$$P_{ij} = \frac{\Pi_j}{M\Pi_i},$$

This means that the transition from a lower energy state to a higher energy state $i \rightarrow j$ occurs with probability $\frac{\Pi_j}{\Pi_i}$ (it does not always occur). Conversely, the complementary event $1 - \frac{\Pi_j}{\Pi_i}$ results in the rejection of the transition from i to j .

3.3 Metropolis Method (3)

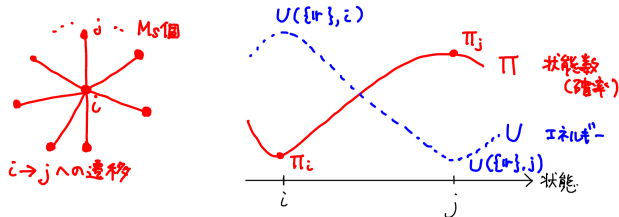


図 2: Transition from state i to state j . The number of possible transition states j (including i) is M . The transition $i \rightarrow i$ indicates staying in the same state. "Rejection" in the Metropolis method corresponds to this. (Right) The relationship between probability Π and potential energy U . The higher the U , the lower the probability.

3.3 Metropolis Method

Markov Chain Monte Carlo Method

- Here, we will implement a Markov chain using the Metropolis method. The implementation of the Metropolis method involves trials using uniformly distributed random numbers, which constitute the Monte Carlo simulation. In general, such simulations are referred to as **Markov Chain Monte Carlo (MCMC) Method**.
- Now, the probability that the state ν is generated under constant temperature and volume conditions, following the canonical distribution in statistical mechanics, is given by

$$\Pi_{\nu} \propto \exp(-\beta U_N(\mathbf{r}^N(\nu))), \quad (15)$$

- Next, consider the state transition from $\{\mathbf{r}^N(\nu)\}$ to $\{\mathbf{r}_{\text{trial}}^N\}$. Here, each particle is randomly displaced individually.
- Let k be the particle to be displaced, and the displaced state $\mathbf{r}_{\text{trial}}^k$

$$\mathbf{r}_{\text{trial}}^k = \mathbf{r}^k(\nu) + \Delta r \mathbf{R}, \quad (16)$$

where \mathbf{R} is a uniformly distributed random number in the range $[-1, 1]$.

- The value of Δr is empirically found to be $O(0.1)$.

3.3 Metropolis Method (2)

- Intuitively, if Δr is too large, many transitions will be rejected in the following Metropolis criteria, reducing computational efficiency. This value must be finely adjusted depending on the problem.
- Next, for the trial coordinates obtained by displacing particle k , perform a Metropolis criterion so that the canonical distribution is satisfied:

$$\begin{aligned}
 \frac{\Pi_{\text{trial}}}{\Pi_v} &= \frac{\exp[-\beta U_N(\{\mathbf{r}_{\text{trial}}^N\})]}{\exp[-\beta U_N(\{\mathbf{r}^N(v)\})]} \\
 &= \exp[-\beta(U_N(\{\mathbf{r}_{\text{trial}}^N\}) - U_N(\{\mathbf{r}^N(v)\}))] \\
 &= \exp[-\beta(\Delta U)]
 \end{aligned} \tag{17}$$

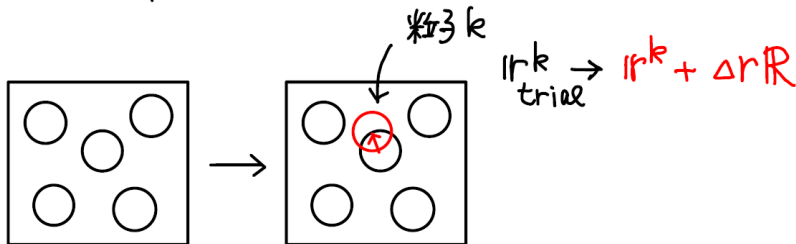
where ΔU represents the change in energy.

3.3 Metropolis Method (3)

- Using this probability, the following Metropolis criterion can be performed:

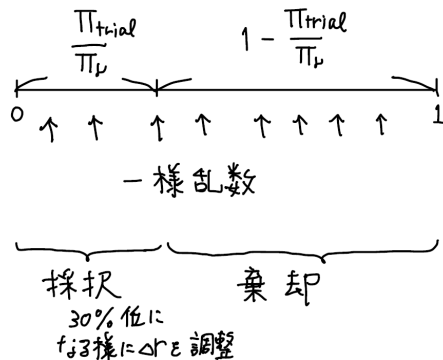
Metropolis Criterion

- If $\frac{\Pi_{\text{trial}}}{\Pi_{\nu}} > 1$: Always accept.
- If $\frac{\Pi_{\text{trial}}}{\Pi_{\nu}} < 1$: Accept with probability $\frac{\Pi_{\text{trial}}}{\Pi_{\nu}}$. Reject with probability $1 - \frac{\Pi_{\text{trial}}}{\Pi_{\nu}}$.



☒ 3: The random displacement of particle k is given by $\mathbf{r}_{\text{trial}}^k = \mathbf{r}^k(\nu) + \Delta r \mathbf{R}$. Here, \mathbf{R} is a uniformly distributed random number in the range $[-1, 1]$.

3.3 Metropolis Method (4)



☒ 4: Trial method for the Metropolis criterion $\frac{\pi_{\text{trial}}}{\pi_{\nu}} < 1$. Using a uniformly distributed random number R in the range $[0, 1]$, accept the trial if $0 < R < \frac{\pi_{\text{trial}}}{\pi_{\nu}}$ is satisfied.

3.3 Metropolis Method (5)

- Finally, we handle a special case. For a hard-sphere potential, the Metropolis criterion simplifies as follows:

Metropolis Criterion for Hard Spheres

- (1) If $\frac{\Pi_{\text{trial}}}{\Pi_V} = 1$: Accept.
- (2) If $\frac{\Pi_{\text{trial}}}{\Pi_V} < 1$: Reject.

Calculations for hard spheres, which are challenging in MD simulations learned previously, are relatively easy using the Monte Carlo method.

Contents

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References

4. 8th Assignment

8th Assignment Implementation of Markov Chain Monte Carlo Method

Reproduce the phase separation phenomenon from Assignment 6 using the Markov Chain Monte Carlo method.

The sample program for Assignment 8 (Monte Carlo method) "mc.cpp" (which does not include optimization using lists) is available in the GitHub repository at [\[Link\]](#).

Contents

- 1 Lecture Schedule
 - Syllabus
- 2 Explanation of the 7th Assignment
- 3 Monte Carlo Method in Particle Systems
 - Markov Chain
 - Microscopic Reversibility (Detailed Balance) Equation
 - Metropolis Method
- 4 8th Assignment
- 5 References**

References and Websites

- [1] 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).
- [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] Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E (1953) Equation of State Calculations
by Fast Computing Machines.
The Journal of Chemical Physics 21(6):1087–1092.