データサイエンス系科目群：シミュレーション実習
第 8 回 講義資料

担当：川﨑猛史

名古屋大学大学院理学研究科理学専攻物理科学系・非平衡物理研究室 (R 研)

Last update: June 17, 2024

目次

目次

1. 講義のスケジュール

- 当実習は春 1 期にて実施する．
- 講義資料は各講義予定日当日朝 **11** 時迄にアップロードする．
- スケジュール
  1. 4/15：第 1 回
  2. 4/22：第 2 回
  3. 4/30(火)：第 3 回
  4. 5/13：第 4 回 (中間レポート課題公開)
  5. 5/20：第 5 回
  6. 5/27：第 6 回 (中間レポート課題提出期限)
  7. 6/03：休講
  8. 6/10：第 7 回
  9. **6/17**：第 8 回 期末レポート課題公開
  10. 6/24：第 9 回（補講）

1.1. シラバス

当実習では以下の内容を扱う予定である（進捗に合わせ変更する可能性がある）.

1. 導入
   - C(C++) の使い方 (主に数値計算)
   - Python の使い方 (データ解析と作図)
   - 数値計算の理念
   - 桁落ち
   - 科学計算における無次元化
2. 常微分方程式の数値解法：減衰振動や調和振動子を例に
   - 微分方程式の数値積分
   - オイラー法
3. １粒子系のブラウン運動
   - ランジュバン方程式（確率微分方程式）
   - 正規乱数の生成法
   - オイラー・丸山法
   - 時間平均とアンサンブル平均
   - 拡散係数の計算
4. 多粒子系のブラウン運動
   - 相互作用力の計算方法
   - 非平衡系のシミュレーション：相分離現象を例に
5. 多粒子系の分子動力学シミュレーション
   - 位置ベルレ法と速度ベルレ法
   - 多粒子系における保存則（運動量・エネルギー・角運動量）
6. モンテカルロ法
   - 統計力学の復習
   - マルコフ連鎖モンテカルロ法
   - メトロポリス判定法

目次

2. 第 7 回自主課題解説

第 7 回自主課題 分子動力学シミュレーション（古典 MD) の実装

粒子数 $N = 1024$，直径 a，質量 m の同一円盤粒子からなる一辺の長さ $L = 40a$ の周期境界に閉じ込められた 2 次元系を考える．
今回用いる粒子間ポテンシャルは，斥力のみからなる

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

を採用する．ここでカットオフ長は $a_{\mathrm{cut}} = 3.0a$ とする．ここで，長さの単位を a, エネルギーの単位を ϵ, そして時間の単位を $t_0 = \sqrt{ma^2/\epsilon}$ とするとき以下の問いに答えよ．

(1) 自主課題 6 で構築した Langevin 熱浴を用いることにより，無次元温度 $T^* = k_B T/\epsilon = 0.9$ を取る熱平衡状態における各粒子の位置座標 $\{\mathbf{r}_j\}$ と速度 $\{\mathbf{v}_j\}$ を取得せよ．

(2) (1) で取得した座標と速度を初期条件として，分子動力学シミュレーションを実行し，この時，全粒子の運動エネルギーを K, ポテンシャルエネルギーを U とするとき，力学的エネルギー $U + K$ が時間に対して保存することを示せ．

(3) **(発展課題 – 高速化)** 第 5 回自主課題で紹介したリスト（帳簿を使って）相互作用計算を高速化せよ．（方針）リストを作る範囲を半径 $r_{cut} + r_{skin}$ の範囲にある粒子とし，一度作ったリストは，各粒子の変位の大きさの最大値が $r_{skin}/2$ を超えるまで存続させる．

解説

2. 第 7 回自主課題解説 (2)

自主課題 7 のサンプルプログラム（リストを用いた高速化はしていないもの）“md.cpp”は，以下のリスト 1 および GitHub リポジトリより取得可能である[リンク].

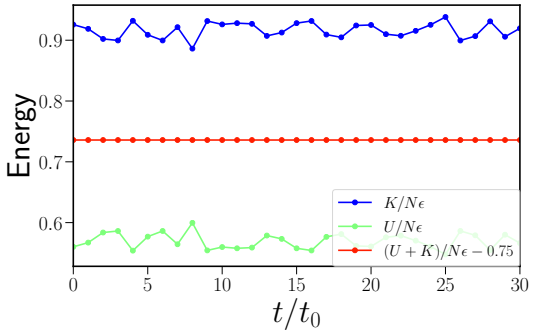

図 1: 自主課題 7(2) の解答例. 力学的エネルギーが保存している様子が見て取れる．なお，ここでは 1 粒子あたりのエネルギーを表示した．

2. 第 7 回自主課題解説 (3)

リスト 1: 自主課題 7 のサンプルプログラム “md.cpp”. 以下の GitHub リポジトリより取得可能[リンク].

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

#define Np 1024
#define L 40.0
#define teq 100
#define tmax 30
#define dtmd 0.001
#define dtbd 0.01
#define temp 0.9
#define dim 2
#define cut 3.0
#define polydispersity 0.0

void ini_coord_square(double (*x)[dim]){
  int num_x = (int)sqrt(Np)+1;
  int num_y = (int)sqrt(Np)+1;
  int i,j,k=0;
  for(j=0;j<num_y;j++){
    for(i=0;i<num_x;i++){
```

2. 第 7 回自主課題解説 (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. 第 7 回自主課題解説 (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. 第 7 回自主課題解説 (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. 第 7 回自主課題解説 (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. 第 7 回自主課題解説 (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 }
```

目次

3 粒子系におけるモンテカルロ法

粒子系におけるモンテカルロ法

- 粒子系におけるモンテカルロ法（MC 法）は、乱数を用いて熱平衡状態における粒子の分布とそれに伴う統計量を求める方法である。
- MD 計算では粒子間力の計算が重要であるが、MC 法ではエネルギー計算が重要になる。
- この部分を除けば、ブラウン動力学法 (Langevin 熱浴法) で学んだ計算手法と分子動力学法では重なる部分が多い。
- MC 法は基本的に統計力学に基づいて NVT, NPT, μVT などアンサンブルを生成する。
- 以下、MC 法の基本原理を説明する。

3.1 Markov 連鎖

Markov 連鎖 [1, 2, 3]

ここでは，連鎖的な M 個の状態からなる系を考える．

- 状態 i から ℓ ステップの遷移の際に，系が状態 k を取り，さらに m ステップ遷移した際に状態 j にあるとしよう．
- この状態間の **遷移確率** がそこに至るまでの履歴に依存しない場合，この過程を **Markov 連鎖** と呼ぶ．
- 特に，m が大きい極限では、**各状態に入る確率** は初期状態とは無関係に得られる．このことを **Markov 連鎖の極限定理** という。

Markov 連鎖極限定理と定常状態

- Markov 連鎖において ℓ ステップで $i \to j$ の状態へ遷移する確率を $P_{ij}(\ell)$ とすると，次の関係が成り立つ（**Chapman Kolmogorov 方程式/Smolkovsky 方程式**）：

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

3.1 Markov 連鎖 (2)

- 特に熱平衡状態は長い時間 (多段階) を経て得られるため，初期状態には依存しない．このことは，$\ell \to \infty$ と $m = 1$ とすると

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

であるので，いま

$$P_{ij}(\infty) \to \Pi_j \tag{3}$$

$$P_{ik}(\infty) \to \Pi_k \tag{4}$$

とすれば，Π_j は定常状態における状態 j を取る確率を考えることができる. Markov 連鎖では $P_{kj}(1)$ はそれまでの状態遷移経路に依存しないため $P_{kj}(1) = P_{kj}$ と表すことができる．すると，

$$\boxed{\Pi_j = \sum_{k=1}^{M} \Pi_k P_{kj}} \tag{5}$$

を得る．ここでは，

$$\sum_{k=1}^{M} P_{jk} = 1 \tag{6}$$

3.1 Markov 連鎖 (3)

であることから，式 (5) は，

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

となり，状態 j への確率の流入と流出が釣り合うことを表す．

- またこの定常状態の関係式は，状態 j を取る確率に関する時間発展方程式：マスター方程式

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

における $\frac{\partial \Pi_j}{\partial t} = 0$ に対応する．

3.2 微視的な可逆性（詳細釣り合い）の関係式

微視的な可逆性（詳細釣り合い）の関係式

- 前節の議論から，定常状態において状態 i を取る確率 Π_i は，すべての遷移確率 P_{jk} を定義することで一意に決まる．一方，ここでは，定常状態の中でも特別な熱平衡状態を考える．熱平衡状態においては，全ての微視的な状態の釣り合いの関係式である詳細釣り合い（微視的な可逆性）の関係式

$$\Pi_i P_{ij} = \Pi_j P_{ji} \tag{9}$$

が成立する．無論，この詳細釣り合い関係式は，前節で扱った定常状態の関係式を保証する．

- Π_i は，統計力学によれば，NVT アンサンブルを生成するカノニカル分布は

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

と表される．ここで Z_N は分配関数であり

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

である．

3.2 微視的な可逆性（詳細釣り合い）の関係式 (2)

- 圧力と温度が一定である NPT アンサンブルにおいては，確率 Π_i は

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

である．Υ_N は，この時の分配関数であり

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

となる．

3.3 Metropolis 法

Metropolis 法 [4]

メトロポリス法は、モンテカルロ・シミュレーションの乱数生成で生じた新しい状態を，受理と拒否の基準を統計力学的に与えるもので，この様な重みづけサンプリングにより分布関数を近似的に計算する方法である．

- 多様な P_{ij} に対して，確率 Π_i を求めることができるが，本節では，詳細釣り合いの原理を認めた上で，最も簡便に熱平衡状態を実現する遷移確率 P_{ij} を

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

  の様に与える．この様な規則を **Metropolis 法** [4] とよぶ.

3.3 Metropolis 法 (2)

- Metropolis 法では $\Pi_j > \Pi_i$ において，図 **2** の様にエネルギーの高い状態から低い状態への遷移に対応するため，$i \to j$ への遷移が必ず起こるとする．つまり

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

とする．
一方，$\Pi_j < \Pi_i$ では，エネルギーの高い状態から低い状態に対応する状態 $j \to i$ への遷移確率は

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

である．したがって，詳細釣り合いの関係式から，遷移確率 P_{ij} は

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

となる．つまりエネルギーの低い状態から高い状態 $i \to j$ への遷移は $\frac{\Pi_j}{\Pi_i}$ の確率で起こる（必ず起こるわけではない）．一方，これの余事象 $1 - \frac{\Pi_j}{\Pi_i}$ にて $i \to j$ への遷移が棄却される．

3.3 Metropolis 法 (3)

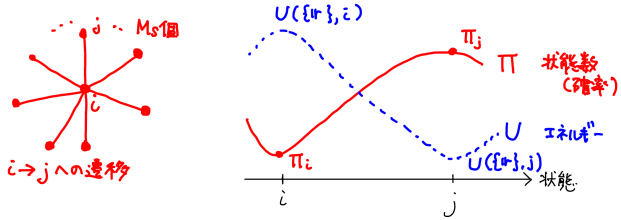

図 2: 状態 i から状態 j への遷移．ここでの遷移先の状態 j(i を含む) の場合の数は M 通りである．$i \to i$ の遷移は，同じ状態にとどまることを表す．Metlopolis 法における「棄却」がこれにあたる．(右) 確率 Π とポテンシャルエネルギー U の関係．U が高いほど確率が低くなる．

3.3 Metropolis 法

Markov 連鎖モンテカルロ法

- ここでは，Metlopolis 法を用いて，Markov 連鎖を実装しよう．Metlopolis 法の実装には，一様乱数を用いた試行，つまりモンテカルロシミュレーションを行う．つまり，この様なシミュレーションの総称として，**Markov 連鎖モンテカルロ法**とよぶ．
- いま，状態 ν が，定温，定積条件下で生成する確率は，統計力学におけるカノニカル分布を満たすため

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

  となる．
- 次に，状態遷移 $\{\mathbf{r}^N(\nu)\} \to \{\mathbf{r}^N_{\rm trial}\}$ を考える．ここでは 粒子を一つ一つ個別にランダムに変位させる．
- 変位させる粒子を k とすれば，変位後の状態 $\mathbf{r}^k_{\rm trial}$

$$\mathbf{r}^k_{\rm trial} = \mathbf{r}^k(\nu) + \Delta r \mathbf{R}, \tag{16}$$

  と表される．ここで，$\mathbf{R}$ は $[-1, 1]$ の範囲を分布する一様乱数である．
- Δr の大きさは経験的に $O(0.1)$ の値を与えればよいことがわかっている．

3.3 Metropolis 法 (2)

- 感覚的には，もし Δr が大き過ぎれば，以下の Metlopolis 判定の際，多くが棄却され計算の効率が下がる．この値は問題ごとに微調整する必要がある．

- 次に, 粒子 k を変位させた試行座標について，次のようなカノニカル分布を満たすようにメトロポリス判定を行う．

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

ここで，ΔU はエネルギーの変化量を表す．

3.3 Metropolis 法 (3)

- この確率を用いることで以下の Metropolis 判定を行うことができる

Metropolis 判定

(1) $\frac{\Pi_{\text{trial}}}{\Pi_\nu} > 1$: 常に採択
(2) $\frac{\Pi_{\text{trial}}}{\Pi_\nu} < 1$: with $\frac{\Pi_{\text{trial}}}{\Pi_\nu}$: 採択. With $1 - \frac{\Pi_{\text{trial}}}{\Pi_\nu}$: 棄却.

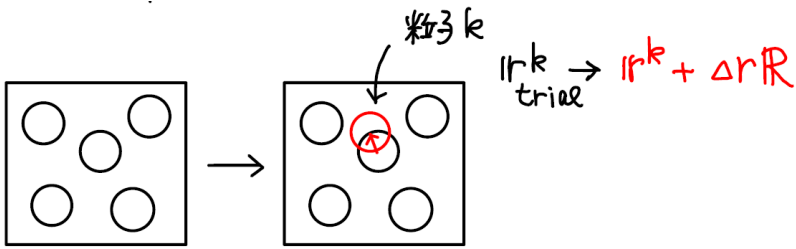

図 3: 粒子 k のランダムな変位を $\mathbf{r}^k_{\text{trial}} = \mathbf{r}^k(\nu) + \Delta r \mathbf{R}$ のように与える．ここで，$\mathbf{R}$ は $[-1, 1]$ の範囲を分布する一様乱数である．

3.3 Metropolis 法 (4)

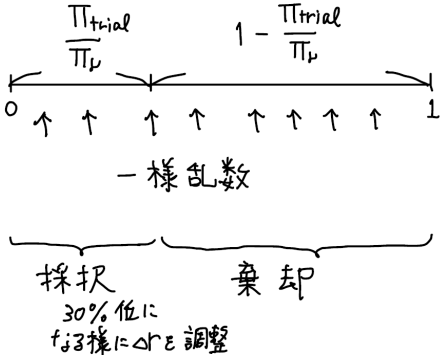

図 4: メトロポリス判定 $\frac{\Pi_{\text{trial}}}{\Pi_\nu} < 1$ の試行方法．$[0,1]$ の範囲分布する一様乱数 R を用いて，R が $0 < R < \frac{\Pi_{\text{trial}}}{\Pi_\nu}$ を満たすとき，その試行を受理すればよい．

3.3 Metropolis 法 (5)

- 最後に特別な場合を扱う．剛体球ポテンシャルに関しては以下の様に簡単になる．

剛体球における Metropolis 判定

(1) $\frac{\Pi_{\mathrm{trial}}}{\Pi_\nu} = 1$: accept
(2) $\frac{\Pi_{\mathrm{trial}}}{\Pi_\nu} < 1$： reject.

この様な剛体球の計算は，前回学んだ MD シミュレーションでは難しいが，モンテカルロ法では極めて容易である．

目次

4. 第 8 回自主課題

第 8 回自主課題 Markov 連鎖モンテカルロ法の実装

自主課題 6（相分離現象）を Markov 連鎖モンテカルロ法で再現せよ．

自主課題 8(モンテカルロ法) のサンプルプログラム（リストを用いた高速化はしていないもの）“mc.cpp”は，GitHub リポジトリより取得可能である[リンク].

目次

参考文献・ウェブサイト

[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.