データサイエンス系科目群：シミュレーション実習
第 5 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: May 19, 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予定)
  7. **6/03**：休講
  8. 6/10：第 7 回 (期末レポート課題公開)
  9. 6/17：第 8 回
  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
   - 軌道の安定性と保存則
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
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
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

2.1. 正規乱数の発生方法：Box Muller 法 [1]

本節では，第 **4** 回で説明しきれなかった正規乱数を数値的に発生させる方法について紹介する．

- 分散 1 の正規乱数 $\mathbf{R}_G$ を計算で発生させる方法として Box Muller 法 [1] が有名である．

Box Muller 法 [1]

U_1 と U_2 を $[0,1]$ の範囲で分布する一様乱数とするとき，以下の X_1 と X_2 は独立な標準（分散 1 の）正規乱数となる．

$$X_1 = \sqrt{-2\log U_1}\cos 2\pi U_2 \tag{1}$$
$$X_2 = \sqrt{-2\log U_1}\sin 2\pi U_2 \tag{2}$$

(証明)

- 式 (1) and (2) を連立することで U_1, U_2 について解く．

2.1. 正規乱数の発生方法：Box Muller 法 [1] (2)

- まず X_1 と X_2 の 2 乗を取ることで以下の関係式

$$\log U_1 = -\frac{X_1^2 + X_2^2}{2} \tag{3}$$

$$U_1 = e^{-\frac{X_1^2+X_2^2}{2}} \tag{4}$$

を得る.

- 式 (1) を式 (2) で割ることで

$$\frac{X_2}{X_1} = \tan 2\pi U_2$$

$$U_2 = \frac{1}{2\pi} \arctan \frac{X_2}{X1} \tag{5}$$

を得る.

- これらを用いることで X_1 と X_2 が 独立なガウス過程 であることを示す.
- いま確率密度関数 $P(X_1, X_2)$ と $\tilde{P}(U_1, U_2)$ を導入する.

2.1. 正規乱数の発生方法：Box Muller 法 [1] (3)

- $P(X_1, X_2)$ と $\tilde{P}(U_1, U_2)$ の関係は，

$$P(X_1, X_2)\mathrm{d}X_1\mathrm{d}X_2 = \tilde{P}(U_1, U_2)\mathrm{d}U_1\mathrm{d}U_2 = \tilde{P}(U_1, U_2)\left|\frac{\partial(U_1, U_2)}{\partial(X_1, X_2)}\right|\mathrm{d}X_1\mathrm{d}X_2 \tag{6}$$

  となる．

- U_1 と U_2 は独立な $[0, 1]$ を閾値にもつ一様乱数であることから

$$\int_0^1 \mathrm{d}U_1 \int_0^1 \mathrm{d}U_2 \tilde{P}(U_1, U_2) = 1 \tag{7}$$

  となり $\tilde{P}(U_1, U_2) = 1$ を得る．したがって，

$$P(X_1, X_2) = \left|\frac{\partial(U_1, U_2)}{\partial(X_1, X_2)}\right| \tag{8}$$

  である．

- いま，ヤコビアン $\left|\frac{\partial(U_1, U_2)}{\partial(X_1, X_2)}\right|$ は

$$\left|\frac{\partial(U_1, U_2)}{\partial(X_1, X_2)}\right| = \begin{vmatrix} \frac{\partial U_1}{\partial X_1} & \frac{\partial U_2}{\partial X_1} \\ \frac{\partial U_1}{\partial X_2} & \frac{\partial U_2}{\partial X_2} \end{vmatrix} = \left|\frac{\partial U_1}{\partial X_1}\frac{\partial U_2}{\partial X_2} - \frac{\partial U_2}{\partial X_1}\frac{\partial U_1}{\partial X_2}\right| \tag{9}$$

2.1. 正規乱数の発生方法：Box Muller 法 [1] (4)

- となるので，適宜微分を実行すれば

$$\left|\frac{\partial(U_1,U_2)}{\partial(X_1,X_2)}\right| = \left| \frac{1}{2\pi}\frac{e^{-\frac{X_1^2+X_2^2}{2}}}{1+(\frac{X_2}{X_1})^2} - \frac{1}{2\pi}\frac{-X_2^2}{X_1^2}\frac{e^{-\frac{X_1^2+X_2^2}{2}}}{1+(\frac{X_2}{X_1})^2} \right| = \frac{1}{2\pi}e^{-\frac{X_1^2+X_2^2}{2}} = \frac{1}{\sqrt{2\pi}}e^{-\frac{X_1^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{X_2^2}{2}} \tag{10}$$

を得る．従って，

$$P(X_1,X_2) = \frac{1}{\sqrt{2\pi}}e^{-\frac{X_1^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{X_2^2}{2}} \tag{11}$$

である．$P(X_1,X_2) = p(X_1)p(X_2)$ であるので X_1 と X_2 は，独立なガウス過程であることが示された．

$y = \arctan x$ の微分

$y = \arctan x$ は，$x = \tan y$ である．これを y で微分すると $\frac{\mathrm{d}x}{\mathrm{d}y} = \frac{1}{\cos^2 y}$ である．つまり

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \cos^2 y = \frac{1}{1+\tan^2 y} = \frac{1}{1+x^2} \tag{12}$$

を得る．

2.2. Box Muller method の高速化（Marsaglia polar 法）

2.2 節の目的

つぎに，Marsaglia polar 法について解説する．

改良 Box Muller 法（Marsaglia polar 法） [2]

1. $[-1,1]$ を閾値にもつ一様乱数 u_1, u_2 を用いることで，2 次元ベクトル

$$\mathbf{R} = (u_1, u_2) \tag{13}$$

を導入する．ここでは，$\mathbf{R}$ を適当にふり，半径 1 の円の中に入った乱数のペアのみ抜き出す（外に出たものは棄却）．

2. すると，ここで得た $R^2 = u_1^2 + u_2^2$ は $[0,1]$ を閾値にもつ一様乱数となる (証明を下に載せた). そのため $U_1 = R^2$ とおくことができ，$\sqrt{-2\log U_1}$ を得る．

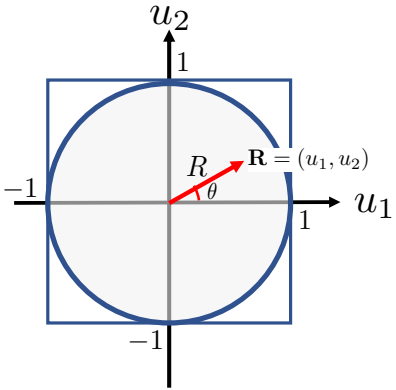
3. 次に $\mathbf{R}$ の偏角は $[0, 2\pi]$ の範囲を一様に分布することから $2\pi U_2$ と等価である．このことから，$2\pi U_2$ を引数にもつ三角関数は，間接的に

$$\frac{u_1}{R} = \cos 2\pi U_2 \tag{14}$$

$$\frac{u_2}{R} = \sin 2\pi U_2 \tag{15}$$

- Box Muller 法における，三角関数の計算は比較的重い計算であり，大量に乱数を発生させる際は大きなコストとなる．
- ここでは，計算方法を工夫することで，三角関数を直接計算せずに **Box Muller** 法と同等の結果を得る方法を紹介する（Marsaglia polar 法） [2].

2.2. Box Muller method の高速化（Marsaglia polar 法） (2)



$$U_1 = R^2$$

$$\cos 2\pi U_2 = \frac{u_1}{R}$$

$$\sin 2\pi U_2 = \frac{u_2}{R}$$

図 1: Marsaglia polar 法における一様乱数の取り方.

2.2. Box Muller method の高速化（Marsaglia polar 法） (3)

(2 の証明)
$X = R^2$ とすると確率密度関数 $f(X)$ を R に関する確率密度関数 $g(R)$ との関係は

$$f(X)\mathrm{d}X = g(R)\mathrm{d}R = g(R)\frac{\partial R}{\partial X}\mathrm{d}X \tag{16}$$

となる．いま，$g(R)$ は R に比例 [1] するので $g(R) = CR$ である（C は定数である）．従って以下の積分を実行することで

$$\int_0^1 g(R)\mathrm{d}R = \left[\frac{C}{2}R^2\right]_0^1 = \frac{C}{2} = 1 \tag{17}$$

となるので $C = 2$ を得る．一方，$X = R^2$ の関係から $\frac{\partial R}{\partial X} = \frac{1}{2R}$ であるので

$$f(X)\mathrm{d}X = \frac{2R}{2R}dX = 1dX \tag{18}$$

を得る．したがって，$f(X) = 1$ より，$X = R^2$ は $[0, 1]$ を閾値とする一様乱数であることが示された.

2.2. Box Muller method の高速化（Marsaglia polar 法） (4)

リスト 1: 正規乱数の発生アルゴリズム (Box-Muller 法)　以下のプログラム”BM.h”は以下の GitHub リポジトリより取得可能[リンク]. サブルーチンとして移植するかヘッダとして include して用いるとよい.

```
1  double unif_rand(double left, double right)
2  {
3    return left + (right - left)*rand()/RAND_MAX;
4  }
5  double gaussian_rand(void)
6  {
7    static double iset = 0;
8    static double gset;
9    double fac, rsq, v1, v2;
10
11   if (iset == 0) {
12     do {
13       v1 = unif_rand(-1, 1);
14       v2 = unif_rand(-1, 1);
15       rsq = v1*v1 + v2*v2;
16     } while (rsq >= 1.0 || rsq == 0.0);
17     fac = sqrt(-2.0*log(rsq)/rsq);
18
19     gset = v1*fac;
20     iset = 0.50;
21     return v2*fac;
```

2.2. Box Muller method の高速化（Marsaglia polar 法） (5)

```
22    } else {
23      iset = 0;
24      return gset;
25    }
26  }
```

[1]円環に分割した際，微小区間の面積は R に比例する

2.3. 様々な平均量

2.3 節の目的

数値計算結果は，統計誤差を含むため，解析の際，各種平均操作を行い，物理的に重要な要素を抜き出す必要がある．よく用いられる 2 つの平均手法を紹介する．

2.3.1. 時間平均

時間平均

定常状態の物理量について、各時刻の値に対して平均化する操作を時間平均という．

- この操作は定常状態における物理量 $X(t)$ に対して行われ，実際に

$$\langle X \rangle_{t_0} = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathrm{d}t_0 X(t_0) \tag{19}$$

  と計算される．
- 同様に，この操作を 2 時刻相関関数 $C(t, t_0) = X(t + t_0)X(t_0)$ に対して施す場合，

$$C(t) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathrm{d}t_0 C(t, t_0) \tag{20}$$

  となる．この時，$C(t) = \langle X(t + t_0)X(t_0)\rangle_{t_0}$，あるいは，より単純に $C(t) = \langle X(t)X(0)\rangle$ と書く．

2.3.1. 時間平均 (2)

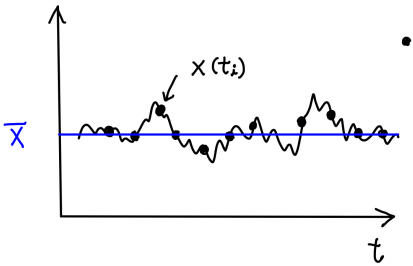


図 2: 時間平均の考え方.

2.3.2. アンサンブル平均

アンサンブル平均

独立かつ同等な試行（実験）をいくつも行いを加算平均することをアンサンブル平均という．

- 熱平衡状態にある α 番目のサンプルの物理量を A_i とするとき，アンサンブル平均は

$$\langle A \rangle_{\mathrm{ens}} = \frac{1}{N_{\mathrm{ens}}} \sum_{\alpha=1}^{N_{\mathrm{ens}}} A_\alpha \tag{21}$$

  となる．
- このことは以下の統計力学平均と同等である．

$$\langle A \rangle_{\mathrm{ens}} = \frac{\mathrm{Tr} A(\mathbf{q},\mathbf{p}) e^{-\beta \hat{H}(\mathbf{q},\mathbf{p})}}{Z} \tag{22}$$

  ここで，Z は分配関数: $Z = \mathrm{Tr} e^{-\beta \hat{H}(\mathbf{q},\mathbf{p})}$ である．

補足　熱平衡状態では，時間平均とアンサンブル平均が等しくなる．

→ エルゴード仮説

2.3.2. アンサンブル平均 (2)

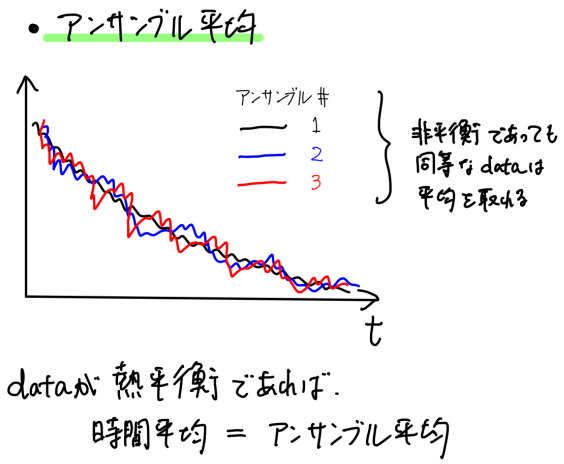


図 3: アンサンブル平均の考え方.

目次

3. 第 4 回自主課題解説

第 4 回自主課題　1 粒子ブラウン運動の実装

温度 T，摩擦係数が ζ である 3 次元溶媒中を熱揺動力による駆動される 1 粒子の運動を考える．この粒子の運動は Langevin 方程式 $m\dot{\mathbf{v}}(t) = -\zeta\mathbf{v}(t) + \mathbf{F}_{\mathrm{B}}(t)$ でモデル化できることが広く知られている．この Langevin 方程式を，長さ，時間の単位をそれぞれ a，$\frac{m}{\zeta}$ として無次元化すると，$T^* = \frac{mk_{\mathrm{B}}T}{a^2\zeta^2}$ がパラメータとなることを講義で扱った．そこで，この粒子の運動に関する以下の各問いに答えよ．なお，以下の $\langle\cdots\rangle$ は，アンサンブル平均や時間平均を十分とった量であることを表す．

(1) 平均二乗変位に関する解析解 $\left\langle \Delta\mathbf{r}(t)^2 \right\rangle = \frac{2dk_{\mathrm{B}}T}{\zeta}\left\{t + \frac{m}{\zeta}e^{-\zeta t/m} - \frac{m}{\zeta}\right\}$（第 4 回講義資料参照）を無次元化しパラメータ T^* を用いて表せ．

(2) 任意の T^* に対して，(1) の理論解と数値解が一致することを確認せよ．数値解は半陰 Euler・丸山法で求めよ．

(3) 粒子の速度相関関数 $C(t) = \langle \mathbf{v}(t)\cdot\mathbf{v}(0)\rangle$ を計算せよ．また，数値計算の結果と理論解の結果 $C(t) = \langle \mathbf{v}(t)\cdot\mathbf{v}(0)\rangle = \frac{dk_{\mathrm{B}}T}{m}e^{-\zeta t/m}$ を比較せよ．

3. 第 4 回自主課題解説 (2)

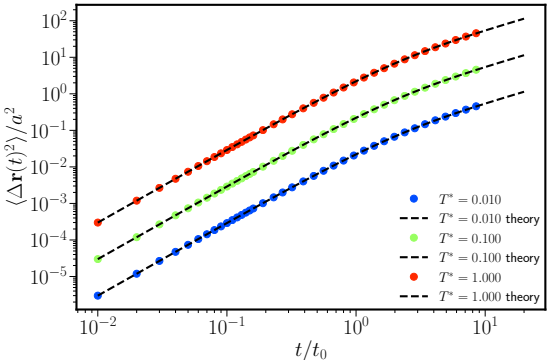


図 4: 平均二乗変位の数値結果．ここでは，パラメータ T^* を $0.01, 0.1, 1.0$ と変化させた時の様子を表す．点線は理論解 [式 (25)] を表す．数値解とよく合っている様子が見て取れる．

3. 第 4 回自主課題解説 (3)

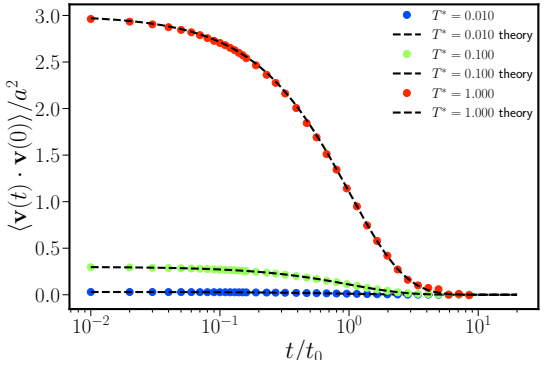


図 5: 速度相関関数の数値結果．パラメータ T^* を $0.01, 0.1, 1.0$ と変化させた時の様子を表す．点線は理論解 [式 (31)] を表す．数値解とそこそこ様子が見て取れる．

3. 第 4 回自主課題解説 (4)

解説

(1) 平均二乗変位の解析解 $\left\langle \Delta \mathbf{r}(t)^2 \right\rangle = \frac{2dk_{\mathrm{B}}T}{\zeta}\left\{t + \frac{m}{\zeta}e^{-\zeta t/m} - \frac{m}{\zeta}\right\}$ を無次元化する．長さの単位 a, 時間の単位 $t_0 = m/\zeta$ に注意して，

$$\begin{aligned} a^2\left\langle \Delta \tilde{\mathbf{r}}(\tilde{t})^2 \right\rangle &= \frac{2dk_{\mathrm{B}}T}{\zeta}[t_0\tilde{t} + t_0 e^{-\tilde{t}} - t_0] \\ &= \frac{2dmk_{\mathrm{B}}T}{\zeta^2}[\tilde{t} + e^{-\tilde{t}} - 1] \end{aligned} \tag{23}$$

しかるに，

$$\begin{aligned} \left\langle \Delta \tilde{\mathbf{r}}(\tilde{t})^2 \right\rangle &= \frac{2dmk_{\mathrm{B}}T}{\zeta^2 a^2}[\tilde{t} + e^{-\tilde{t}} - 1] \\ &= 2dT^*[\tilde{t} + e^{-\tilde{t}} - 1] \end{aligned} \tag{24}$$

つまり，３次元 ($d = 3$) であれば

$$\boxed{\left\langle \Delta \tilde{\mathbf{r}}(\tilde{t})^2 \right\rangle = 6T^*[\tilde{t} + e^{-\tilde{t}} - 1]} \tag{25}$$

を得る．

3. 第 4 回自主課題解説 (5)

補足

短時間極限では，$e^{-\tilde{t}} \sim 1 - \tilde{t} + \frac{1}{2}\tilde{t}^2$ と展開することで，

$$\boxed{\left\langle \Delta\tilde{\mathbf{r}}(\tilde{t})^2 \right\rangle \sim 3T^*\tilde{t}^2} \tag{26}$$

と表される弾道軌道を得る．また長時間極限では，$e^{-\tilde{t}} - 1$ の項が落ちるので

$$\boxed{\left\langle \Delta\tilde{\mathbf{r}}(\tilde{t})^2 \right\rangle \sim 6T^*\tilde{t}} \tag{27}$$

と表される拡散軌道を得る．つまり無次元化された拡散係数は，パラメータ T^* そのものの値となる．

(2) C 言語による数値計算のコーディング例をリスト 2 ,3 に示した．運動方程式を解く主計算プログラムは “langevin.cpp”，そこから吐き出される座標と速度のデータを用いて，平均二乗変位と速度相関関数の解析プログラムは “analyze.cpp”とした．平均二乗変位の数値結果を図 **4** に示した．ここでは，パラメータ T^* を $0.01, 0.1, 1.0$ と変化させた時の様子を表す．点線は理論解 [式 (25)] を表す．よく合っている様子が見て取れる．

3. 第 4 回自主課題解説 (6)

(3) 第 4 回講義で補遺で示した通り，ブラウン粒子の速度相関関数は $t \geq 0$ のとき，

$$C(t) = \langle \mathbf{v}(t) \cdot \mathbf{v}(0) \rangle = \frac{dk_B T}{m} e^{-\zeta t/m} \tag{28}$$

となる．これらを無次元化すると

$$\frac{a^2}{t_0^2} \langle \tilde{\mathbf{v}}(\tilde{t}) \cdot \tilde{\mathbf{v}}(0) \rangle = \frac{dk_B T}{m} e^{-\zeta t/m} \tag{29}$$

であるから

$$\langle \tilde{\mathbf{v}}(\tilde{t}) \cdot \tilde{\mathbf{v}}(0) \rangle = \frac{dmk_B T}{\zeta^2 a^2} e^{-\zeta t/m} \tag{30}$$

$$= 3T^* e^{-\tilde{t}} \tag{31}$$

となる．ここで $d = 3$ とした．この理論解と数値解の比較を図 **5** に記した．理論と数値解が平均二乗変位ほどではないがよく合っている様子が見て取れる．

3. 第 4 回自主課題解説 (7)

リスト 2: “langevin.cpp” 以下の GitHub リポジトリより取得可能[リンク]

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <iostream>
5  #include <fstream>
6  #include <cfloat>
7  #include "BM.h"
8
9  #define tmax 10
10 #define dt 0.01
11 #define temp 0.01 //parameter
12 #define ensemble 1000
13 #define dim 3
14 //using namespace std;
15
16 void ini_phase(double *x,double *v){
17   int i;
18   for(i=0;i<dim;i++){
19     x[i]=0.;
20     v[i]=0.;
21   }
22 }
```

3. 第 4 回自主課題解説 (8)

```
23
24 void ini_clock(int *j,double *tout){
25   *j=0;
26   *tout=1.e-2;
27 }
28
29 void eom(double *v,double *x){
30   int i;
31   for(i=0;i<dim;i++){
32     v[i]+=-v[i]*dt+sqrt(2.*temp*dt)*gaussian_rand();
33     x[i]+=v[i]*dt;
34   }
35 }
36
37 void output(double *x,double *v,int j){
38   char filename[128];
39   std::ofstream file;
40
41   sprintf(filename,"coord_dt%.3fT%.3f.dat",dt,temp);
42   file.open(filename,std::ios::app); //append
43   file <<j*dt<<"\t"<<x[0]<<"\t"<<x[1]<<"\t"<<x[2]<<std::endl;
44   //  std::cout<<j*dt<<"\t"<<x[0]<<"\t"<<x[1]<<"\t"<<x[2]<<std::endl;
45   file.close();
46
```

3. 第 4 回自主課題解説 (9)

```
47    sprintf(filename,"vel_dt%.3fT%.3f.dat",dt,temp);
48    file.open(filename,std::ios::app); //append
49    file <<j*dt<<"\t"<<v[0]<<"\t"<<v[1]<<"\t"<<v[2]<<std::endl;
50    file.close();
51
52  }
53
54  int main(){
55    double x[dim],v[dim],t,tout;
56    int i,j;
57    ini_phase(x,v);
58    for(i=0;i<ensemble;i++){
59      ini_clock(&j,&tout);
60      output(x,v,j);
61      while(j*dt < tmax){
62        j++;
63        eom(v,x);
64        if(j*dt >= tout){
65        output(x,v,j);
66        tout*=1.2;
67        }
68      }
69    }
70    return 0;
```

3. 第 4 回自主課題解説 (10)

```
71 }
```

リスト 3: “analyze.cpp” 以下の GitHub リポジトリより取得可能[リンク]

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <iostream>
5  #include <fstream>
6  #include <cfloat>
7
8  #define temp 0.01
9  #define dt 0.01
10 #define ensemble 1000
11 #define window 39
12 #define dim 3
13 //using namespace std;
14
15 void ini(double *dr2,double *corr){
16   for(int i=0;i<window;i++){
17     dr2[i]=0.0;
18     corr[i]=0.0;
19   }
```

3. 第 4 回自主課題解説 (11)

```
20  }
21
22  void input(double (*x)[dim],double (*v)[dim],double *t){
23    char filename[128];
24    std::ifstream file;
25    sprintf(filename,"coord_dt%.3fT%.3f.dat",dt,temp);
26    file.open(filename);
27    int asize=ensemble*window;
28    for(int i=0;i<asize;i++){
29      file >> t[i] >> x[i][0] >> x[i][1] >> x[i][2];
30    }
31    file.close();
32
33    sprintf(filename,"vel_dt%.3fT%.3f.dat",dt,temp);
34    file.open(filename);
35    for(int i=0;i<asize;i++){
36      file >> t[i] >> v[i][0] >> v[i][1] >> v[i][2];
37      // std::cout << t[i] <<"\t"<<v[0][i]<<"\t"<<v[1][i]<<"\t"<<v[2][i]<<std::endl;
38    }
39    file.close();
40  }
41
42  void output(double *t,double *dr2,double *corr){
43    char filename[128];
```

3. 第 4 回自主課題解説 (12)

```
44    std::ofstream file;
45    sprintf(filename,"msd_dt%.3fT%.3f.dat",dt,temp);
46    file.open(filename);
47    for(int i=1;i<window;i++)
48      file<<t[i]-t[0]<<"\t"<<dr2[i]<<std::endl;
49    file.close();
50
51    sprintf(filename,"corr_dt%.3fT%.3f.dat",dt,temp);
52    file.open(filename);
53    for(int i=1;i<window;i++)
54      file<<t[i]-t[0]<<"\t"<<corr[i]<<std::endl;
55    file.close();
56  }
57
58  void analyze(double (*x)[dim],double (*v)[dim],double *t,double *dr2,double *corr){
59    double dx[dim],corr_x[dim];
60    for(int i=0;i<ensemble;i++)
61      for(int j=0;j<window;j++){
62        for(int k=0;k<dim;k++){
63        dx[k]=(x[j+window*i][k]-x[window*i][k]);
64        corr_x[k]=v[j+window*i][k]*v[window*i][k];
65        dr2[j]+=(dx[k]*dx[k])/ensemble;
66        corr[j]+=(corr_x[k])/ensemble;
67        }
```

3. 第 4 回自主課題解説 (13)

```
68      }
69  }
70
71  int main(){
72    double t[ensemble*window],dr2[window],corr[window];
73    int i,j;
74    double  (*x)[dim] = new double[ensemble*window][dim];
75    double  (*v)[dim] = new double[ensemble*window][dim];
76
77    ini(dr2,corr);
78    input(x,v,t);
79    analyze(x,v,t,dr2,corr);
80    output(t,dr2,corr);
81    delete[] x;
82    delete[] v;
83    return 0;
84  }
```

目次

4. 第 5 回自主課題

第 5 回自主課題 多粒子計算の準備

2 次元平面中に，粒径（直径）1 の円板を一辺の長さ $L = 40$ の正方形の空間に 512 個均等に配置する．

(1) 粒子を正方格子で配置し，その結果を図示せよ．

(2) 粒子を六方格子で配置し，その結果を図示せよ．

(3) (発展) 正方形境界が周期境界である際，ある粒子 i と他の粒子 j の距離を測るアルゴリズムを考えよ．（力の計算等に必要）

(4) (発展)(3) の距離が（例えば）5 以下の “粒子番号”を格納する配列を計算するアルゴリズムを考えよ．（ベルレ帳簿法）

目次

5. 付録 1：スタックメモリと動的（ヒープ）メモリ確保の比較（C 言語）

リスト 4: スタックメモリを用いた変数・配列の確保 (C 言語の例)．スタックメモリ（スタック領域におけるメモリ）は，ラップトップ PC では 10MB 程度が上限（かなり小さい） [3]．

```
1 double x,y[10000],z[10000][10];
```

リスト 5: 動的メモリの確保 (C++の例)．変数と配列でメモリの解放のレトリックが異なる．メモリ（ヒープ領域）の上限はラップトップ PC で GB オーダまで確保可能 [3]． C 言語では malloc 関数を用いることにより同等のことができる．

```
1 double *x = new double;
2 double *y = new double[10000];
3 double (*z)[10] = new double[10000][10];
4 // 不要になったらメモリを解放
5 delete x;
6 delete [] y;
7 delete [] z;
```

目次

6. 付録 2：関数におけるポインタ渡し（C 言語）

ポインタ関連用語

- &: アドレス演算子　「変数のアドレスを返す演算子」　（例：変数 a に値が入っているとき，$\&a$ はそのアドレスを表す．）
- $*$: 間接演算子　「ポインタの指し示す値を返す演算子」 (例：$*(\&a)$ は a の値をさす．)
- $*b$: ポインタ変数. この時 b はアドレスを示す．(int $*b$ などと宣言：int 型ポインタ．)

リスト 6: C 言語におけるポインタ渡しのレトリック: 以下の型を常に参考にするとよい．

```
1
2  void function(double *x, double *y, double (*z)[10]){
3    *x=1.0;
4    for(int i=0;i<10000;i++)
5      y[i]=i;
6
7    for(int i=0;i<10000;i++)
8      for(int j=0;i<10;j++)
9        z[i][j]=i*j;
10 }
11
12 int main(){
```

6. 付録 2：関数におけるポインタ渡し（C 言語） (2)

```
13    double x, y[10000],z[10000][10];
14    function(&x,y,z);
15    return 0;
16  }
```

目次

参考文献・ウェブサイト

[1] Box GEP, Muller ME (1958) A Note on the Generation of Random Normal Deviates. The Annals of Mathematical Statistics 29(2):610–611.

[2] Marsaglia G, Bray TA (1964) A Convenient Method for Generating Normal Variables. SIAM Review 6(3):260–264.

[3] Lemniscater N (year?) C++のスタックメモリと動的メモリの上限値調査 (https://qiita.com/LemniscaterN/items/a3abfa143612cb928bde).