データサイエンス系科目群：シミュレーション実習
第 6 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 22, 2023

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. 4/17：第 1 回
  2. 4/24：第 2 回
  3. 5/01：第 3 回
  4. 5/08：第 4 回 (週半ばに中間レポート課題公開)
  5. 5/15：第 5 回
  6. 5/20(土曜: 月曜授業日) 休講
  7. **5/22**：第 **6** 回 **(5/29**：中間レポート課題提出期限予定**)**
  8. 5/29：第 7 回
  9. 6/05：第 8 回 (期末レポート課題公開)
  10. 6/12：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 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．

解説

ここで扱うサンプルプログラムは GitHub リポジトリ (Lecture6 以下) より取得可能[リンク]

(1) 任意の大きさ L の正方形の箱（2 次元）上に正方格子状に粒子を配置するプログラム **“lattice.cpp”**におけるサブルーチン ini_square(x) で生成 (リスト 1)．作図は **“draw_particle.py”**(リスト 2) で行った．

2. 第 5 回自主課題 (2)

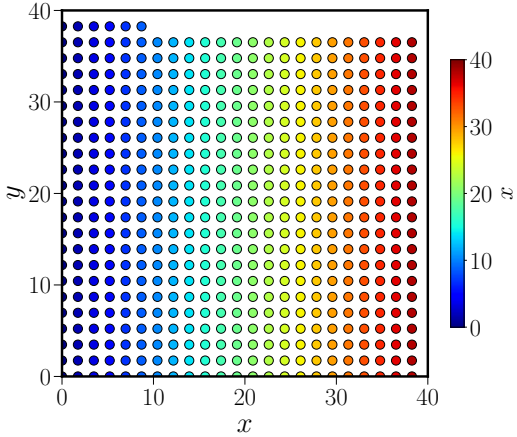

図 1: リスト 1 のプログラムを用いて生成した正方格子.

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

2. 第 5 回自主課題 (4)

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

2. 第 5 回自主課題 (5)

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

2. 第 5 回自主課題 (6)

```
78 }
```

粒子座標の描画には以下の matplotlib.patches（[参考資料リンク]）を用いたサンプルプログラム “draw_particles.py”をリスト 2 に示す．

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. 第 5 回自主課題 (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. 第 5 回自主課題 (8)

```
46 ax.spines['top'].set_linewidth(3)
47 ax.spines['bottom'].set_linewidth(3)
48 ax.spines['left'].set_linewidth(3)
49 ax.spines['right'].set_linewidth(3)
50 plt.tick_params(which='major',width = 1, length = 10)
51 plt.tick_params(which='minor',width = 1, length = 5)
52 plt.xticks(color='k', size=30)
53 plt.yticks(color='k', size=30)
54 plt.xlabel(r"$x$",color='k', size=35)
55 plt.ylabel(r"$y$",color='k', size=35)
56
57 ax.set_aspect('equal')
58
59 plt.savefig('./Lecture6/square.pdf',bbox_inches="tight")
60 plt.show()
```

(2) リスト 1 に示す通り，任意の大きさ L の正方形の箱（2 次元）上に六方格子状に粒子を配置するプログラム **“lattice.cpp”**におけるサブルーチン ini_hex(x) で生成．作図は **“draw_particle.py”**で行った（ファイル名は適宜変更せよ）．

2. 第 5 回自主課題 (9)

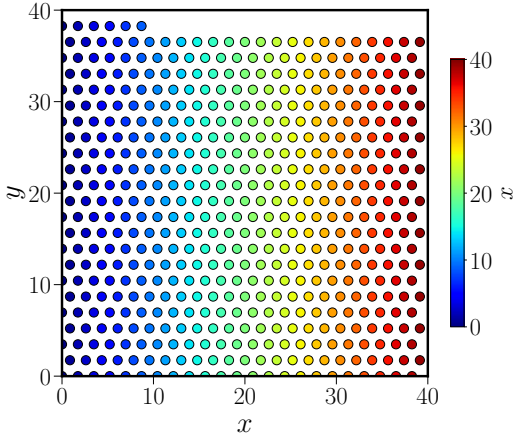

図 2: リスト 1 のプログラムを用いて生成した正方格子.

目次

3. 多粒子系のブラウン運動（Langevin 熱浴中の多粒子運動）

3 章の目的

- これまで Langevin 熱浴中を運動する 1 粒子の運動方程式やその運動（ブラウン運動）について議論した [1].
- ここからは，Langevin 熱浴中の多粒子運動を扱う．
- 多粒子が織りなす系の物性に対して，粒子間相互作用が重要な役割を担う．
  - 例えば，気液転移は，相互作用に引力が存在する場合にのみ起こる．
- 今回は分子シミュレーションで用いられる代表的な相互作用ポテンシャルと相互作用力の計算方法を説明する．
- 次回は，系の体積を定義する上で不可欠な（周期的）境界条件を導入する．
- 具体的な多体シミュレーションを実装する．

3.1. 相互作用ポテンシャルの一例

相互作用ポテンシャルは実に多様であるが，分子シミュレーションで用いるポテンシャルのごく一例を紹介する．

(1) Lennard-Jones ポテンシャル [2, 3, 4](図 **3** 参照)

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

- 変数の意味：
  - $a_{ij} = \frac{a_i + a_j}{2}$ （平均粒径:粒子接触を特徴づける距離）．
  - ϵ は物質固有のエネルギー．（LJ エネルギー単位ともいう）
  - C_{jk} はカットオフエネルギーであり，カットオフ長 a_{cut} において，$U(a_{\mathrm{cut}}) = 0$ になるようにおく．
- 性質：
  - 斥力部分と引力部分から成る．
  - モデル分子として広く用いられる（汎用性が高い）．
  - 斥力の起源は，原子同士の接近の際，パウリの排他律による生じる反発力（冪の 12 乗自体には物理的根拠はないが広く実験を再現する）．
  - 引力の起源は，原子間の双極子．双極子間の相互作用（これで冪の 6 乗が出る）．

3.1. 相互作用ポテンシャルの一例 (2)

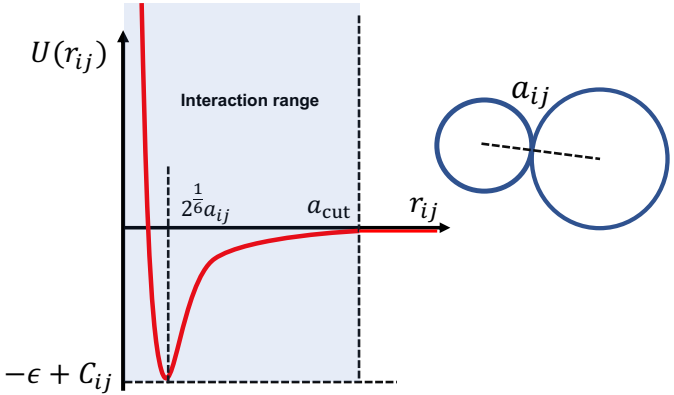

図 3: Lennard-Jones ポテンシャルの概形．粒子接触長 a_{ij} に関する絵解き図．

3.1. 相互作用ポテンシャルの一例 (3)

(2) Coulomb ポテンシャル [3, 4, 5, 6]

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

- ε：誘電率，C:エネルギー変換定数.
- $q_{i(j)}$:粒子 $i(j)$ の電荷.
- 相互作用レンジが長いため，Ewald 法など計算の工夫が必要になる.

3.2. 相互作用力の計算方法

ポテンシャルエネルギーの位置微分から，粒子 j にかかる力 $\mathbf{F}_j^{\mathrm{I}}$ が以下の通り計算できる．

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

3.3. 周期境界条件

周期境界条件とは

- 多数の粒子の運動を扱う場合，粒子の密度は系の物理的特性を決定する重要な役割を果たす。
- 密度を定義する為には、境界（壁など）を設定する必要がある。
- しかし、壁を設置すると、壁近傍の粒子のダイナミクスや構造がバルクと異なってしまう。
- これに対して，周期境界はそのような影響を防ぐためによく導入される。

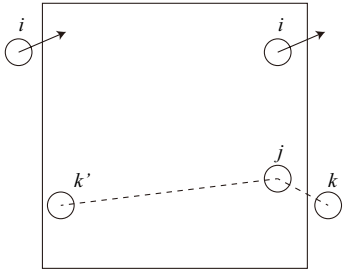

図 4: 周期的な境界条件．右側の境界を跨ぐ粒子 i は左の壁から出てくる．粒子 k と k' は周期境界条件下では同一粒子である．粒子 j と相互作用するのは粒子 k（近い方）である．k' とは相互作用してはいけない．

- 以下，周期境界条件下で用いるアルゴリズムについて解説する．
- 最も基本的なものは，境界からはみ出してしまった粒子を境界内に格納するものである (リスト 3)．

3.3. 周期境界条件 (2)

リスト 3: 周期境界条件のアルゴリズム（座標配置）

```
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 }
```

- 次に，2 体ポテンシャルからの力の計算の際，2 粒子間の距離の計算を行う必要がある．これを周期境界条件下で行うアルゴリズムを考えよう．
- 図 **4** で議論した様に，粒子 k と k' が周期境界条件下では同一粒子であるとき，粒子 j と相互作用するのは粒子 k（近い方）である．特に重要な箇所を/////で囲った．

3.3. 周期境界条件 (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;
```

- floor 関数を用いて，よりスマートに書いたものが以下のリスト 4 である．

3.3. 周期境界条件 (4)

リスト 5: 周期境界条件下での粒子間距離の計算アルゴリズム (2)

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

3.4. 周期境界条件下での力の計算

- 以下，周期境界条件下における相互作用計算のアルゴリズムを示す．
- ここでは Lennard-Jones ポテンシャル（カットオフ距離は 2.5a）を用いた．
- Newton 第３法則（作用・反作用則）から，粒子 ij 間に働く力を，粒子 i,j それぞれに用いた．
- これによりループの計算回数を半分に抑えることができる．
- dUr と表した変数は，微分 $\frac{\mathrm{d}U(r_{ij})}{\mathrm{d}r_{ij}}$ の解析計算の結果を用いている．ポテンシャルを変える場合は，ここを変える必要がある．

リスト 6: 周期境界条件下での粒子間相互作用計算

```
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;
12
```

3.4. 周期境界条件下での力の計算 (2)

```
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          }
36  }
```

3.4. 周期境界条件下での力の計算 (3)

- 今回は **2** 体の全組み合わせを考えたが，遠くに存在する粒子同士の相互作用は計算する必要はないので，この様な粒子の寄与を除くことで計算コストを大幅に削減させることができる（帳簿計算）.

3.5. 任意温度における初期構造の作り方

本節では任意温度における初期構造の作り方について説明する．

- 不適当な配置（例えば完全ランダム配置）からシミュレーションを行うと，粒子のオーバーラップ（重なり）が生じ，莫大な力が粒子に働くことで計算が数値的に破綻する．これを防ぐために，最初期配置は粒子間のオーバーラップが極力少ない配置にするとよい **(図 6)**．(第５回自主課題で扱った結晶的配置が好ましい)
- そして，先の配置から高温状態を取れば，粒子は程よく混ざるため，独立なサンプルを生成することができる（多数の独立なランダム配置を作ることができる）(リスト１参照)．
- その後，目的の温度に下げて平衡化させる．

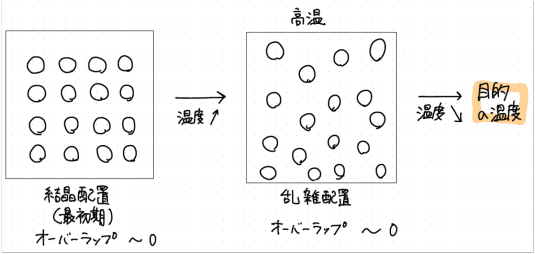

図 **5**: 特定の温度の初期配置の作り方．

3.6. Langevin 方程式の無次元化と離散化

本節では，Langevin 熱浴中の多粒子の運動方程式（Langevin 方程式）を考える．これを数値計算するために，無次元化を施す．

- 粒子 j の運動は次の Langevin 方程式で駆動される．

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

- ここで，$\mathbf{F}_j^{\mathrm{I}}(t)$ は粒子 j に作用する相互作用力（ポテンシャル力）である．
- 以下，Langevin 方程式を，　a は粒子の直径とし，時間を $t = t_0\tilde{t}$，速さを $\mathbf{v} = \frac{a}{t_0}\tilde{\mathbf{v}}$ と無次元化すると

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

を得る．

3.6. Langevin 方程式の無次元化と離散化 (2)

- [1] の係数で方程式の両辺を割ることで，無次元化された方程式

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

  を得る．
- ここで各項の無次元化された係数から，特徴的な時間スケールを抽出する．
- 様々なバリエーションがあるが今回は，

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

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

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

3.6. Langevin 方程式の無次元化と離散化 (3)

を用いることで

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

を得る．この様に表すと全ての係数が無次元化されていることがわかる．

- ここで [3] の係数を 1 とするために，単位時間 t_0 を

$$\boxed{t_0 = t_\mathrm{v}\left(= \sqrt{\frac{ma^2}{\epsilon}}\right)} \tag{9}$$

と置く．

3.6. Langevin 方程式の無次元化と離散化 (4)

- すると，運動方程式は

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\frac{t_\mathrm{v}}{t_\mathrm{d}}\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_\mathrm{v}^3}{t_\mathrm{B}t_\mathrm{d}^2}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{10}$$

  となる．ここで [2] と [4] にパラメータが現れる．

- いま，[2] の係数 $\frac{t_\mathrm{v}}{t_\mathrm{d}} = \zeta\sqrt{\frac{a^2}{m\epsilon}}$ は相互作用の強さに対する摩擦係数の強さの比であるので ζ^* と置くことにする．
- 次に，ζ^* を用いて [4] の係数を表すと

$$\underbrace{\sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_\mathrm{v}^3}{t_\mathrm{B}t_\mathrm{d}^2}}\mathbf{R}_\mathrm{G}}_{[4]} = \sqrt{\frac{2}{\Delta\tilde{t}}\frac{t_v^2}{t_B t_d}\frac{t_\mathrm{v}}{t_\mathrm{d}}}\mathbf{R}_\mathrm{G} \tag{11}$$

$$= \sqrt{\frac{2}{\Delta\tilde{t}}T^*\zeta^*}\mathbf{R}_\mathrm{G} \tag{12}$$

3.6. Langevin 方程式の無次元化と離散化 (5)

ここで，$\boxed{\frac{t_v^2}{t_B t_d} = \frac{k_B T}{\epsilon} = T^*}$(無次元化温度：相互作用と熱エネルギーの比) となる．

- 以上をまとめると，

$$\underbrace{\frac{\mathrm{d}\tilde{\mathbf{v}}_j(\tilde{t})}{\mathrm{d}\tilde{t}}}_{[1]} = \underbrace{-\zeta^*\tilde{\mathbf{v}}_j(\tilde{t})}_{[2]} + \underbrace{\tilde{\mathbf{F}}_j}_{[3]} + \underbrace{\sqrt{\frac{2\zeta^* T^*}{\Delta\tilde{t}}}\mathbf{R}_\mathrm{G}}_{[4]} \tag{13}$$

を得る．ここでのパラメータは ζ^* と T^* である．特に，T^* を変化させると，系の相が変化することがある．

- 最後にこれを，半陰的 Euler・丸山法で離散化すれば，

$$\boxed{\tilde{\mathbf{v}}_j(\tilde{t}+\Delta\tilde{t}) = \tilde{\mathbf{v}}_j(\tilde{t}) - \zeta^*\tilde{\mathbf{v}}_j(\tilde{t})\Delta\tilde{t} + \tilde{\mathbf{F}}_j\Delta\tilde{t} + \sqrt{2\zeta^* T^*\Delta\tilde{t}}\mathbf{R}_\mathrm{G}} \tag{14}$$

$$\boxed{\tilde{\mathbf{r}}_j(\tilde{t}+\Delta\tilde{t}) = \tilde{\mathbf{r}}_j(\tilde{t}) + \tilde{\mathbf{v}}_j(\tilde{t}+\Delta\tilde{t})\Delta\tilde{t}} \tag{15}$$

となり，これを計算機に載せて計算すればよい．

目次

4. 第 6 回自主課題

第 6 回自主課題 Langevin 熱浴中の多粒子シミュレーションの実装（相分離現象）

直径 a の 1024 個の円板を周期境界をもつ 1 辺の長さが $L = 40a$ の正方形の平面に分散させる．この時，円盤 j の運動は，Langevin 方程式 $m\frac{\mathrm{d}\mathbf{v}_j(t)}{\mathrm{d}t} = -\zeta\mathbf{v}_j(t) + \mathbf{F}^{\mathrm{I}}_j(t) + \mathbf{F}^{\mathrm{B}}_j(t)$ により駆動されるとする．ここで $\mathbf{F}^{\mathrm{B}}_j(t)$ は，熱揺動力であり，以下の揺動散逸定理 $\langle \mathbf{F}_{\mathrm{B}}(t)\mathbf{F}_{\mathrm{B}}(t')\rangle = 2k_BT\zeta\delta(t-t')\mathbf{1}$ を満たすものとする．$\mathbf{F}^{\mathrm{I}}_j(t)$ は相互作用力であり，以下の Lennard-Jones ポテンシャル

$$U(r_{jk}) = 4\epsilon\left[\left(\frac{a_{jk}}{r_{jk}}\right)^{12} - \left(\frac{a_{jk}}{r_{jk}}\right)^{6}\right] + C_{jk} \quad (r_{jk} < a_{\mathrm{cut}}) \tag{16}$$

により与えられ，カットオフ長は $a_{\mathrm{cut}} = 2.5a$ とする. いま，時間の単位を $t_0 = \sqrt{ma^2/\epsilon}$, 長さの単位を a，摩擦係数を $\zeta = \sqrt{m\epsilon/a^2}$ とする．この時，無次元温度 k_BT/ϵ を様々に変化させた際の，相分離の有無を数値的に観察せよ．（発展課題）さらに粒子の大きさが分散をもつ場合を考える．現実問題としてまさにコロイド粒子は大きさに分散があることからも，コロイドのモデル計算ではよく粒径分散を考慮する．例えば粒径は平均 a, 標準偏差 $\mathbf{0.15}a$ の正規分布を満たす場合について，上記と同様のシミュレーションを実装せよ．

4. 第 6 回自主課題 (2)

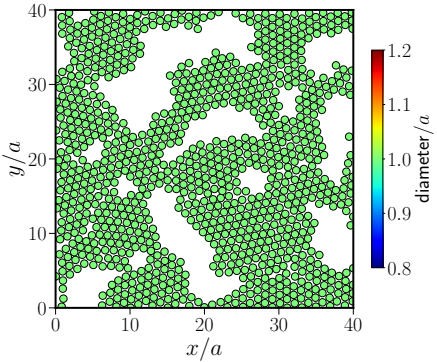

図 6: 粒子直径がすべて a の場合，無次元温度 $T^* = 0.2$ ではこの様な絵が描ける．気体と液体（固体）の相分離が起こっている．

補足：**Github** リポジトリにアップした **“langevin.cpp”**はこのシミュレーションのサンプルプログラムである．適宜参照せよ．

目次

参考文献・ウェブサイト

[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.