卒業論文

Processingによるブラウン運動の視覚化

関西学院大学理工学部

情報科学科 西谷研究室

2554 藤本　宏樹

2016年3月

概 要

ブラウン運動をおこなう粒子の視覚化を Processing 言語を用いておこない，原子の存在を仮定してブラウン運動をおこなう粒子の理論的挙動を示したアインシュタインの論文内容と比較，考察を目的とした．粒子が不規則な運動をおこなう原因となる，媒質中の分子と粒子の衝突を理解しやすくするため，単純な 2 物体の衝突を示すプログラムも作成した．このプログラムの試行，詳細検討から，衝突のアルゴリズムや効果的な表示法を考案した. 衝突の軌跡をチャート紙のように表現することで視覚的に理解できるようになった．これらのプログラムからブラウン運動への理解を深め，アインシュタイン論文内容を深く理解する考察をおこなった.

目 次

第1章　序論

1.1　背景

1.1.1　ブラウン運動

ブラウン運動は1828年にイギリスの植物学者であったロバート・ブラウンにより発見された現象である．植物学者であったブラウンは植物の花粉を顕微鏡で観察する際に水面に浮かんだ花粉の微粒子が不規則な運動をしていることを発見した．当初ブラウンは他の種類の花粉についても調べたところ同様の運動が見られたためにブラウン運動は生き物の運動であるとし，雄性の生殖細胞の特異性によるものであると仮説をたてた．その後ブラウンは花の咲かない植物である，ある種の苔の種子や胚珠を材料に実験をおこなったが同様の運動をおこなうことを発見した．また多くの有機物でも同様にテストしており木材の一部，ガラスや鉱物などでもテストをおこなった結果，同じ運動をおこなうことを発見した．ブラウンは最初に自身がたてた仮説を否定したが結論がどうなったかは述べていない [1]．1905年にアインシュタインがブラウン運動に関係する論文を発表し，原子，分子が存在するという仮説をたてた．その後，ペランは実際にブラウン運動を観察し，そして1908年にアインシュタインの導きだした公式が正しいことを証明した [1]．この分子が実在することの証明という功績によりペランは，1926年のノーベル賞を「物質の不連続的構造に関する研究、特に沈殿平衡についての発見」で受賞している [2].

ブラウン運動の研究

アインシュタインが論文を発表する以前にもブラウン運動についての研究はおこなわれており，研究の結果をジャン・ペランの総合報告「ブラウン運動の実在性」および単行本「原子」をもとに「アインシュタイン選集1」の中で井上健 (1971) は以下のように述べている [1].

最初に組織的実験を行い，液体分子の熱運動論に説明を求めたのは，アインシュタインの論文にも引用されているギイ (Gouy) の研究 (Jour.de.Phys.(2), 7, 561, 1888) である．彼の結果と他の人々の結果とをとりまとめたペランの報告を引用すれば，それらは次のような 7 項目に集約される．

1. 運動は極めて不規則で，並進および回転運動から構成されており，その経路は接線をもたないように見える．
2. 2 個の粒子が，その直径よりも小さな距離内に近づいた場合においても，それぞれは独立に運動しているように見える．
3. 運動は粒子が小さければ小さいほど活発である．
4. 粒子の組成および密度による効果は見られない．
5. 運動は液体の粘性が低いほど活発である．
6. 温度が高くなれば運動は活発になる．
7. 運動は決して停止することはない．

以上がアインシュタインが論文を発表するまでに分かったことだった．これらの事象を説明するには当時存在がまだ証明されていなかった分子運動論が適していた．

1.2 目的

ブラウン運動は気体や液体中などの媒質中にある微粒子が不規則な運動をおこなうことであるが，文章で説明を受けてもどのような運動かイメージしにくい．実際にブラウン運動の観察をおこなえれば視覚的に理解できるが，観察には機材の準備が必要であり，実際に観察するには相当な労力が必要となる．そこで，本研究ではブラウン運動の視覚化をおこない，コンピュータ上でブラウン運動の観察ができるようにするのが目的である．また，アインシュタインが原子の存在を仮定して，ブラウン運動を説明する論文を発表したが，その中で出てきた式に沿った計算とプログラム上の数値を比較し，このブラウン運動のデモンストレーションプログラムがどの程度現実を再現しているかについて考察をおこなった．

第2章　方法

2.1　開発環境

今回のプログラムは Processing 言語で作成した．Processing は柔軟なソフトウェアスケッチブックであり，ビジュアルアートの文脈の中でコーディングを学ぶ言語である [3]．またオープンソースのプログラミング言語であり，PDE(Processing Development Environment) と呼ばれるスケッチブックが用意されており簡単に図の描写が可能である [4]．

2.2　アインシュタインの理論

1905 年にアインシュタインは「熱の分子論から要求される静止液中の懸濁粒子の運動について」という論文を発表した [1]．アインシュタインはこの論文で，液体中の懸濁粒子の無秩序な運動について述べており，これはブラウン運動のことであると考えられる．論文中でアインシュタインは時刻を t，x 方向への変位を x，拡散係数を D とすると，x 方向への粒子の平均変位 λ_x に対して，

$$\lambda_x = \sqrt{x^2} = \sqrt{2Dt} \tag{2.1}$$

となることを導いている．またある液体中で，半径 P の小さな球状をした懸濁粒子の拡散係数 D について，気体定数を R，絶対温度を T，アボガドロ定数を N, 液体の粘性係数を κ，とすると

$$D = \frac{RT}{N}\frac{1}{6\pi\kappa P} \tag{2.2}$$

となり，式 (2.1)，(2.2) より，

$$\lambda_x = \sqrt{t}\sqrt{\frac{RT}{N}\frac{1}{3\pi\kappa P}} \tag{2.3}$$

とx軸方向への変位の平均値を導いている．この導きだされた式を N についての式にすると

$$N = \frac{t}{\lambda_x^2} \frac{RT}{3\pi\kappa P} \tag{2.4}$$

の形になる．これによりブラウン運動を観測し，時間 t と粒子の変位値 λ_x が分かればをアボガドロ定数が計算できることになる．

第3章　結果

本章では実際に作成したプログラムの実行結果とプログラムの解説をおこなう.

3.1　2物体の衝突の視覚化

ブラウン運動の原因は粒子に媒質中の分子が衝突するためであり，それによって粒子が不規則な運動をおこなうことが分かっている．そこで，衝突の計算に間違いがないことや，衝突した際の物体の動きなどをイメージしやすくするために基本となる2物体の弾性衝突のシミュレーションを作成した.

3.1.1　実行結果

図3.1は赤い球の質量M=10，青い球の質量m=10とし，2つの球の質量比が1:1となるように設定した際の実行結果である．単純化するために2つの球はx軸方向のみにしか移動しないとする．図の左側に表示されているものは，物体の質量M,m，物体のx軸方向への速度V_MとV_m，物体の運動量量MV,mv，2物体の運動量の合計であるtotal，2物体の運動エネルギーの合計であるE_total，反発係数eを表示している．図の中でチャート紙のように描かれている部分は球の衝突の軌跡を追いかけている．チャートの縦軸はプログラムのフレーム数を表している．横軸は球の座標であり，50ピクセル毎に線が入っている．質量が同じのため衝突が起きる度に速度の交換が起きている.

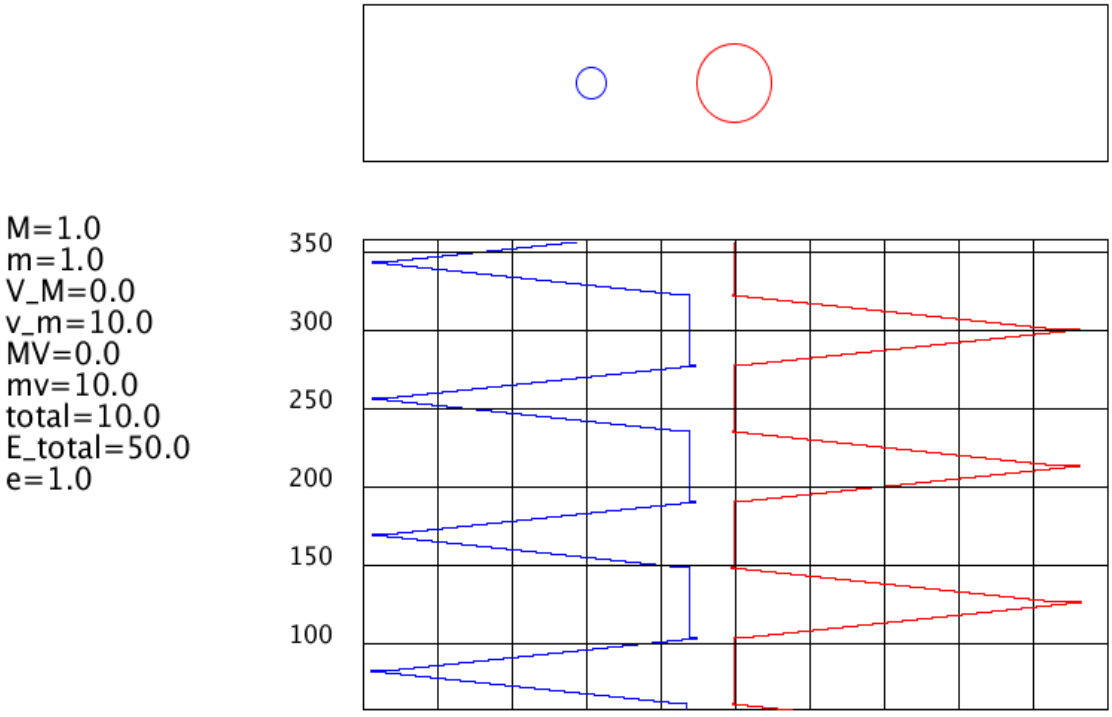

図 3.1: 2 物体の衝突視覚化プログラム.

図 3.2 は赤い球の質量 M=10，青い球の質量を m=1 に変更し，2 物体の質量比を 1:10 とした場合である．小さい方の球が壁との間で激しく振動する様子や，衝突に伴う運動量の移行に伴い速度が微妙に変化する様子が，表示されている．

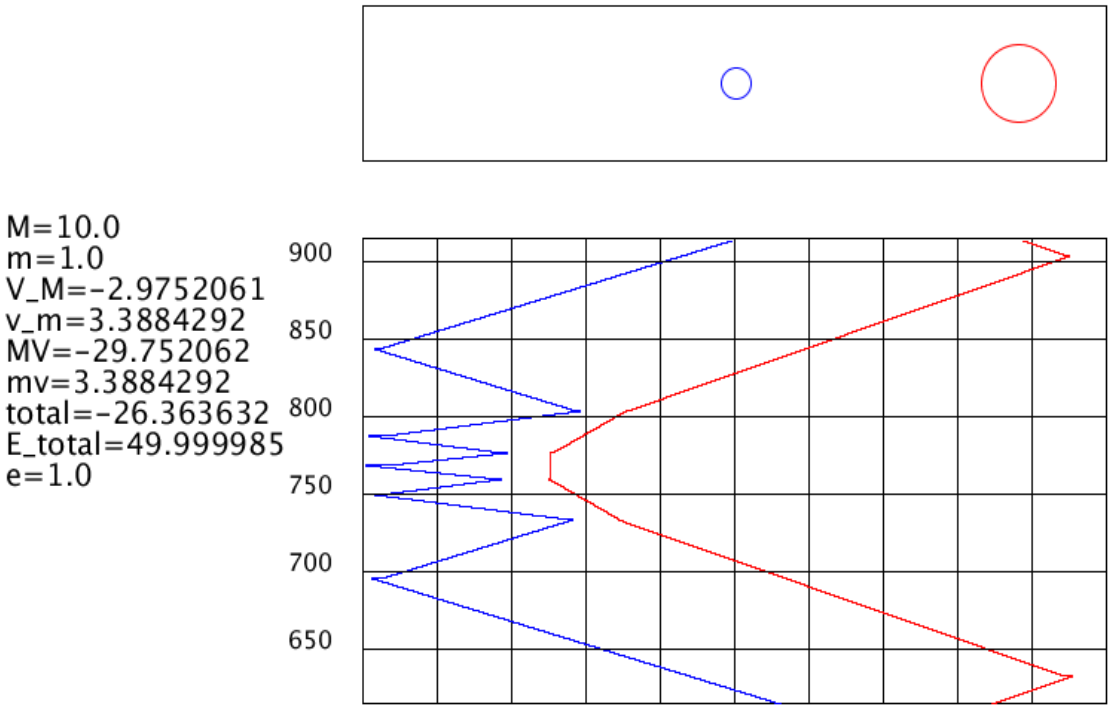

図 3.2: 質量比 1:10 のプログラム.

3.1.2 衝突時の速度計算

以下では 2 物体の衝突が起こった際の物体の速度計算式を示す.

運動量保存則

物体の質量を m, M とし，物体の速度を v, V とする．また衝突後の物体の速度を v', V' とすると

$$mv + MV = mv' + MV' \tag{3.1}$$

となる.

反発係数

反発係数とは物体の衝突において衝突前の互いの速度と衝突後の互いの速度の比のことである．物体の速度を v, V とする．衝突後の速度を v', V' とすると反発係数 e は

$$e = \frac{|v' - V'|}{|v - V|} = -\frac{v' - V'}{v - V} \tag{3.2}$$

と表される．

上記の 2 つの式を連立させて衝突後の x 方向，y 方向それぞれの方向への速度を計算する．今回は弾性衝突の場合を考えるため $e = 1$ として考える．青い球の質量，x 軸方向への速度，y 軸方向への速度を m, v_x, v_y とし，赤い球の質量，x 軸方向への速度，y 軸方向への速度を M, V_x, V_y とする．また衝突後の速度をそれぞれ v'_x, v'_y, V'_x, V'_y として方程式を計算するとそれぞれ

$$V'_x = \frac{MV_x - V_x m + 2mv_x}{M + m} \tag{3.3}$$

$$V'_y = \frac{MV_y - V_y m + 2mv_y}{M + m} \tag{3.4}$$

$$v'_x = \frac{2MV_x - M + mv_x}{M + m} \tag{3.5}$$

$$v'_y = \frac{2MV_y - M + mv_y}{M + m} \tag{3.6}$$

となる．導出されたこの式を利用して物体が衝突した際の衝突後の速度を計算する．

3.1.3 衝突時の物体のめり込みの解消

2 つの球の衝突判定で最も単純なプログラムは青い球の x 座標 small_x，y 座標 small_y，半径 small_r，x 軸方向の速度 v，赤い球の x 座標 large_x，y 座標 large_y，半径を large_r，x 軸方向の速度 V とすると

```
if(dist(small_x,small_y,large_x,large_y) < small_r + large_r) {
    pre_v = v;
    pre_V = V;
    v = (2*M*pre_V-M*pre_v+m*pre_v)/(M+m);
    V = (M*pre_V-pre_V*m+2*m*pre_v)/(M+m);
}
```

となる．

ここで pre_v,pre_V に関して，衝突時の small_v,large_V を一時的に保存するための変数である．上記のプログラムは球の中心の距離をとり，その値が 2 つの球の半径の合計以下になると衝突したと判定するものである．時分割の刻みが大きいため，衝突判定時の座標をそのまま利用すると，2 物体がめり込んだ状態から反発し始めることになる．そこで，以下のような変更を施した．

```
if(dist(small_x,small_y,large_x,large_y) < small_r + large_r) {
   small_x = small_x - ((small_r + large_r)
  -dist(small_x, small_y, large_x, large_y)) / 2;
  large_x = large_x + ((small_r + large_r)
  -dist(small_x, small_y, large_x, large_y)) / 2;
  pre_v = v;
  pre_V = V;
  v = (2*M*pre_V-M*pre_v+m*pre_v)/(M+m);
  V = (M*pre_V-pre_V*m+2*m*pre_v)/(M+m);
}
```

このプログラムでは 2 つの球の半径の和と 2 つの球の中心の距離の差が球のめり込んでいる距離になるので，それを 2 等分し，衝突時の small_x からはマイナスし,large_x にはプラスしている．今回のプログラムでは青い球は常に赤い球の左側に存在すると考えているためである．こちらも簡易的な方法ではあるが，めり込みは回避されている．壁と衝突をした際にも同じように球の中心から壁までの距離と球の半径の差の分だけ球の座標をずらしている．

3.1.4 軌跡の表示

衝突の視覚化をおこなうにあたって物体の移動を表示できれば衝突の様子が分かりやすい．数値を出力し，それを図にプロットするよりもプログラム上で表示できるようにした．実際のプログラムを以下に示す.

```
data[rear % N] = small_x;
data1[rear % N] = large_x;
if(rear % 50==0){
  ytics[rear%N] = rear;
}
else{
  ytics[rear%N] = 0;
}
if(rear > N){
  front = rear % N + 1;
}
locus();
```

data,data1 という float 型の配列にその時点での青い球の x 座標 small_x と赤い球の座標 x 座標 large_x を格納している．メモリサイズが無限にあれば大きく配列の長さを指定すれば良いがそれは不可能なので長さ N を宣言しその長さで，配列が循環するように工夫した．関数 locus() の内容は以下であり data,data1 に格納されている座標を参照し線でつなぐ関数である.

```
void locus(){
  int j = 0;
  for(int i = front; ; i++, j++) {
    i = i % N;
    j = j % N;
    println("jlen=", j);
    if(i == rear % N) {
      break;
    }
    stroke(0, 0, 255);
    line(data[i],500-j,data[(i+1)%N],500-j);
    stroke(255, 0, 0);
    line(data1[i],500-j,data1[(i+1)%N],500-j);
    if(ytics[i]!=0){
      stroke(0, 0, 0);
      line(250,500-j,750,500-j);
      fill(0);
      text(ytics[i], 200, 500-j);
      fill(255);
    }
    if(rear < 300){
      fill(0);
      text(0, 200, 500);
      fill(255);
    }
  }
  rear++;
}
```

3.1.5 エネルギーの計算

運動エネルギーの計算にはエネルギー保存則の式を利用する.

エネルギー保存則

物体の衝突前のエネルギーと衝突後のエネルギーは同じであるため，物体の質量をm, Mとし，物体の速度をv, Vとする．衝突後の速度をv', V'とすると

$$\frac{1}{2}mv^2 + \frac{1}{2}MV^2 = \frac{1}{2}m(v')^2 + \frac{1}{2}M(V')^2 \tag{3.7}$$

のようなエネルギー保存の法則が成り立つ.

3.2 ブラウン運動の視覚化

本研究の目的である，ブラウン運動の視覚化をおこなう．2 物体の衝突のプログラムから衝突時の計算式は問題ないと判断し分子と粒子の衝突が起こった際の計算に同じ式 (3.5) を使用する.

3.2.1 実行結果

図 3.3 は実際の動作中の図である．赤い球がブラウン運動をおこなう粒子を表しており，水色の小さな球が媒質の分子を表している．図の下側と右側に存在するチャート紙は青い線が x 軸，赤い線が y 軸での赤い球の座標の動きを示している．右下に表示されている数値は E がエネルギ保存の式 (3.7) を利用して計算している．$\mathrm{dx}^2, \mathrm{dy}^2$ は粒子の x 軸方向への移動距離 dx，y 軸方向への移動距離 dy を 2 乗している．$\mathrm{sqrt}(\mathrm{dx}^2+\mathrm{dy}^2)$ は粒子の移動距離を表している．N はアインシュタインの公式にプログラム中の変位値を当てはめた時のアボガドロ定数を表している.

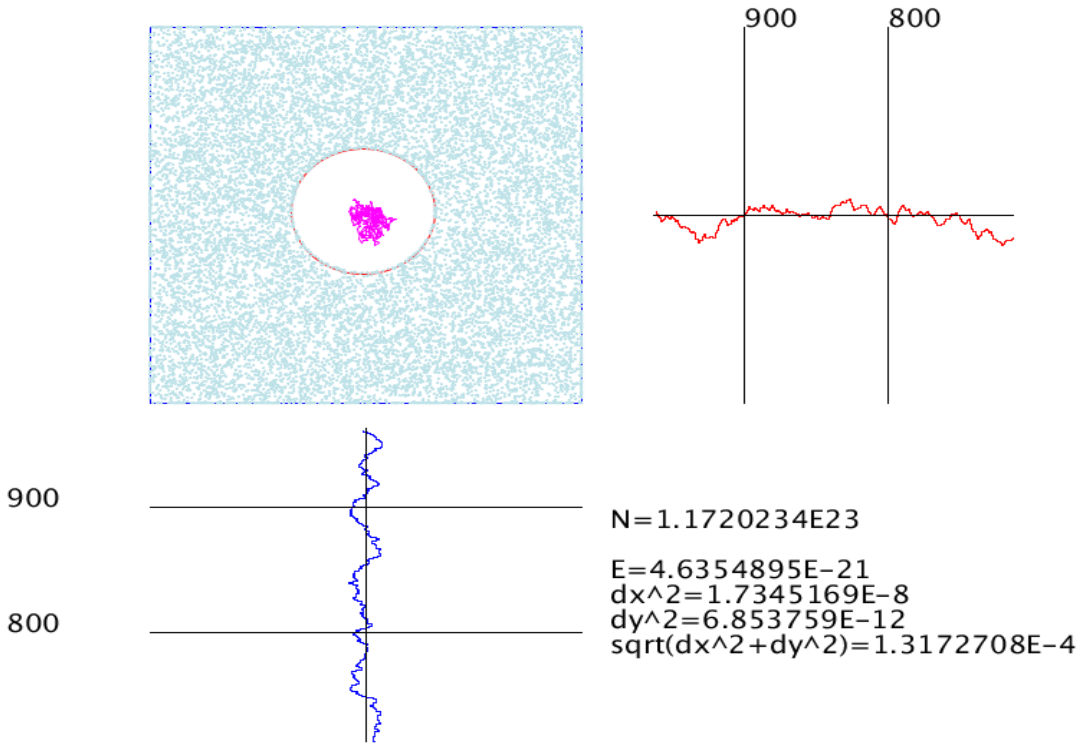

図 3.3: ブラウン運動の視覚化プログラム.

図 3.4 は分子の表示を消した場合である．プログラム実行時は分子は表示されているがキーボード'space' キーを押すことで分子の表示を切り替えることができる．分子を表示するよりも紫の線で表示される粒子の軌跡が分かりやすい.

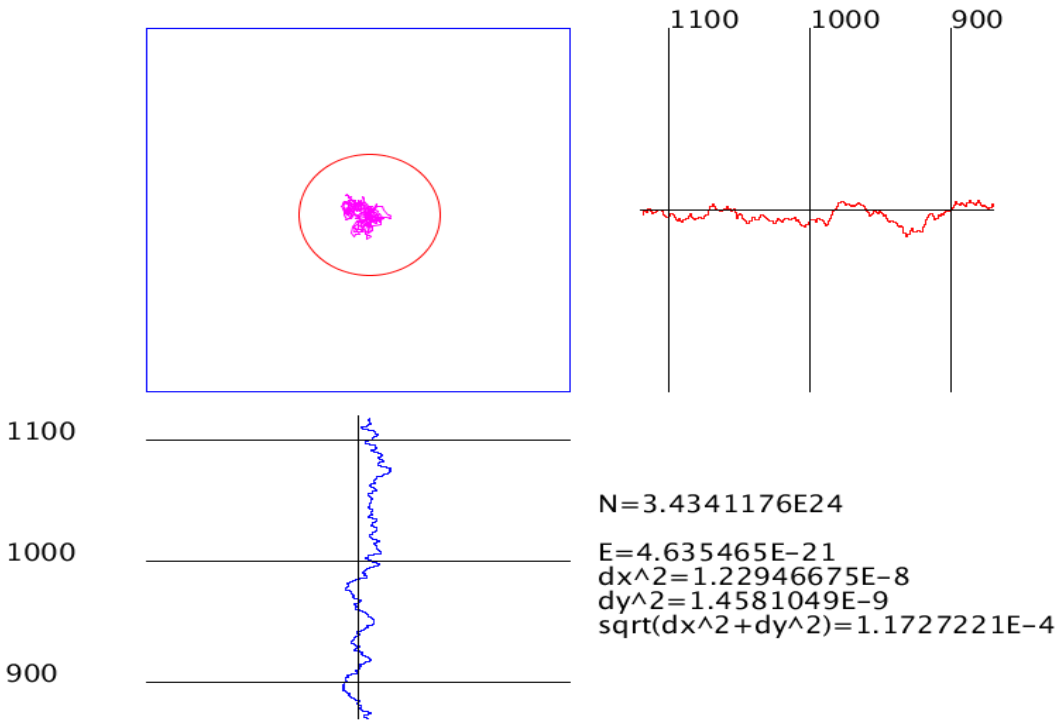

図 3.4: 分子の表示を消した場合.

3.2.2 初期速度の設定

視覚化をおこなうにあたって分子の初期速度を決定する必要がある．粒子の移動が分子の衝突により起こっていることから分子の初期速度によって粒子の動き方に違いが出てくると考えられるからである．動作の問題はエネルギーが保存されているかどうかを確認する．最初は初期速度の値を全て 1 に設定した．これにより，エネルギーが保存されているか確認しやすくするためである．実際に作成したプログラムの計算部分は以下である．

```
for (int i=0; i < X; i++){
 radian = random(TWO_PI);
 vx_pa[i] = cos(radian);
 vy_pa[i] = sin(radian);
 };
```

ここで分子数 X 個,x 軸方向への速度 vx_pa，y 軸方向への速度 vy_pa に対してそれぞれ 2π

までの乱数 radian を作成し vx_pa,vy_pa それぞれ cos(radian),sin(radian) で速度を決定している．実際には分子の速さはマクスウェル分布に従った速度になる．

3.2.3 マクスウェル分布

熱力学的平衡状態において系の分子の速度はマクスウェル分布に従った分布になる．粒子の質量を m, 系の温度を T, ボルツマン定数を k とおき速度が v_i〜$(v_i + dv_i)$ の範囲内の値を有する確率を $f(v_i)dv_i$ とすれば,

$$f(v_i) = \left(\frac{m}{2kT}\right)^{\frac{3}{2}} \exp\left\{-\frac{m}{2kT}(v_{ix}^2 + v_{iy}^2 + v_{iy}^2)\right\} \tag{3.8}$$

と 3 次元の系では上記の確率分布に従う [5].
今回は 2 次元の系で視覚化を行うので，2 個の正規乱数 (平均値 0，標準偏差:$(\frac{kT}{m}^{\frac{1}{2}})$)$w_j$ と w_{j+1} を 2 個の一様乱数 u_j と u_{j+1} より式を作成すると

$$w_j = \left(\frac{kT}{m}\right)^{\frac{1}{2}} (-2\ln u_j) \tag{3.9}$$

$$w_{j+1} = \left(\frac{kT}{m}\right)^{\frac{1}{2}} (-2\ln u_{j+1}) \tag{3.10}$$

であり，原子 a の x 軸方向への速度 v_x^a と y 軸方向への速度 v_y^a に対して

$$v_x^a = w_j \tag{3.11}$$

$$v_y^a = w_{j+1} \tag{3.12}$$

となる [6].

今回マクスウェル分布に従うように作成した初期速度を決定するプログラムは以下である．

```
    for (int i = 0; i < X; i++) {
    float rand1=random(1);
    float rand2=random(1);
    vx_pa[i] = sqrt(k*T/particle_m)*sqrt(-2*log(rand1))*cos(2*PI*rand2);
    vy_pa[i] = sqrt(k*T/particle_m)*sqrt(-2*log(rand1))*sin(2*PI*rand2);
  }
```

X 個の分子に対して，分子の質量を particle_m，x 軸方向への速度 vx_pa，y 軸方向への速度 vy_pa として，ボルツマン定数 k，絶対温度 T，0 から 1 までの乱数 rand1,rand2 を用意し，2 次元でのマクスウェル分布の式 (3.11)，(3.12) に従うように計算をおこなっている．

3.2.4 衝突時のめり込みの処理

2 物体の衝突と同じようにブラウン運動の場合も衝突時のめり込みを防ぐ処理をおこなう必要がある．2 物体の衝突の場合と異なり，x 軸 y 軸の 2 次元上での衝突を考える必要がある．よって物体が衝突した場合に衝突前の座標と衝突後の座標を通る直線と，粒子の大きさの円との交点に座標を移動させる手法をとっている．a, b を定数として直線と円の方程式はそれぞれ

$$\begin{cases} y = ax + b \\ x^2 + y^2 = r^2 \end{cases} \tag{3.13}$$

となり，これを解いて交点の x 座標，y 座標の決定をする．x は

$$x = \frac{-ab + \sqrt{a^2r^2 - b^2 + r^2}}{a^2 + 1} \tag{3.14}$$

$$x = -\frac{ab + \sqrt{a^2r^2 - b^2 + r^2}}{a^2 + 1} \tag{3.15}$$

となり，直線が円の接線でない限り交点は二つあることになる．衝突時の座標を判別するために衝突前の x 座標と，式 (3.14)，(3.15) それぞれの値との差を求め，差が小さい方の座標へ移動させる．

3.2.5 キー入力

'space' キーを押せば分子の表示を切り替えることができる．'c' キーを押せばその時点でのスクリーンショットを保存することができる．スクリーンショットはプログラムのある同じフォルダに'capture-####.png'(####の部分にはフレーム数) が保存される．'l' キーを押せばチャート紙の表示の切り替え，'p' キーを押せばプログラムの一時停止ができるようになっている．コマンド部分のプログラムは以下である．

```
void keyPressed() {
  if (key == 'c') {
    saveFrame("capture-####.png");
  }
  if (key == ' ') {
    if (hide == true) {
      hide = false;
    } else {
      hide = true;
    }
  }
    if (key == 'l') {
    if (locus_hide == true) {
      locus_hide = false;
    } else {
      locus_hide = true;
    }
  }
    if (key == 'p') {
    if (pause == true) {
      noLoop();
      pause = false;
    } else {
      loop();
      pause = true;
    } }  }
```

第4章　考察

本章では作成したプログラムの動作について，動作中の値や数値の理想的な値との比較から考察を問題点や工夫すべき点などの考察をおこなう．

4.1　2物体の衝突のプログラムについて

4.1.1　運動量保存則について

図4.1は2物体の衝突のプログラムで質量比を1:10に設定した際の運動量の合計mv+MVを表したものである．縦軸がmv+MVの値で横軸は動作中のフレーム数である．図から分かる通りmv+MVの値は周期的に変動している．プログラム作成当初これをみて計算方法やプログラムの条件付けに間違いを疑った．なぜなら運動量保存則，式(3.1)に従うのであれば図の数値は常に一定でなければおかしいと考えたからだ．壁まで含めた全ての系の運動量を考えると保存則は常に成り立つはずである．しかし，今は，壁の質量を無限大だと考えて，壁に反射した際の速度変化を-1をかけることで表現している．したがってそこでは運動量保存則が成り立たず，壁の衝突事象が起こるとその前後で2つの粒子の運動量は変化する．言い直すと，時系列をとった時に2つの粒子の運動量は保存されないということになる．その場合でも，運動エネルギーは保存される．

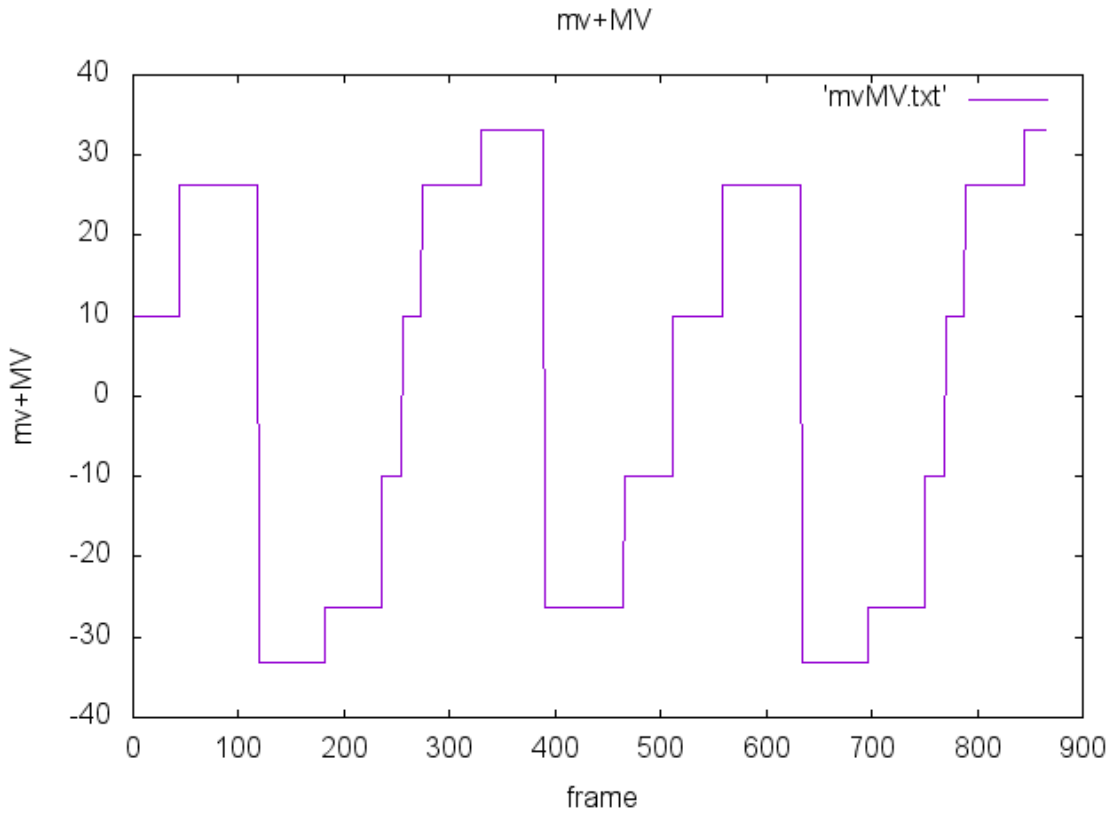

図 4.1: 質量比 1:10 の mv+MV.

4.2 ブラウン運動のプログラムについて

4.2.1 考察の際の注意点

ブラウン運動のプログラムを作成するあたって初期値をどのように設定するかが問題となる．分子数を実際の数を表示して演算することが困難なためである．今回作成したプログラムでは媒質は水だと考えて作成したので初期値として設定すべき κ は水の粘性係数と考え 8.55×10^{-4}(Pa/s) より 8.55×10^{-5}(g/m s) と設定する．粒子の密度を調整する為に水の密度 $\rho = 1.0 \times 10^{6}$(g/m^3) だと考えるが表示する粒子数は 15000 個なので，調整するための定数 scale を考えて水分子の重さ $18/(6.02 \times 10^{23})$(g) として，密度と表示する粒子数の差を合わせるために表示する範囲は 300×300 ピクセルで深さを 1 ピクセルであると

考えて，

$$1.0 \times 10^{-6} = \frac{\frac{18}{6.02 \times 10^{23}} \times 15000}{300 \times 300 \times 1 \times \mathrm{scale}^3} \tag{4.1}$$

という計算をおこない，scale$= 1.708 \times 10^{-10}$(m/pixel) と設定した．ここで設定した数値を利用し，作成したプログラムの考察をおこなう．

4.2.2 マクスウェル分布

式 (3.11)，(3.12) に従って，15000 回の試行をおこなった度数分布が図 4.2 である．横軸は分子の速度 $\mathrm{v} = \sqrt{\mathrm{v_x}^2 + \mathrm{v_y}^2}$ であり，縦軸は速度の度数を表している．ボルツマン定数 $k = 1.38 \times 10^{-23}$(J/K) とし，質量 $m = 18/(6.02 \times 10^{23})$(g) とする．また，温度 T の値を 100K,300K,500K と変更した度数分布が図 4.3 である．図から分かる通り，温度が低くなれば速度は遅くなり，速度の分布の幅が狭くなる．また，温度が高くなると速度は速くなり，速度の分布の幅が広くなる．

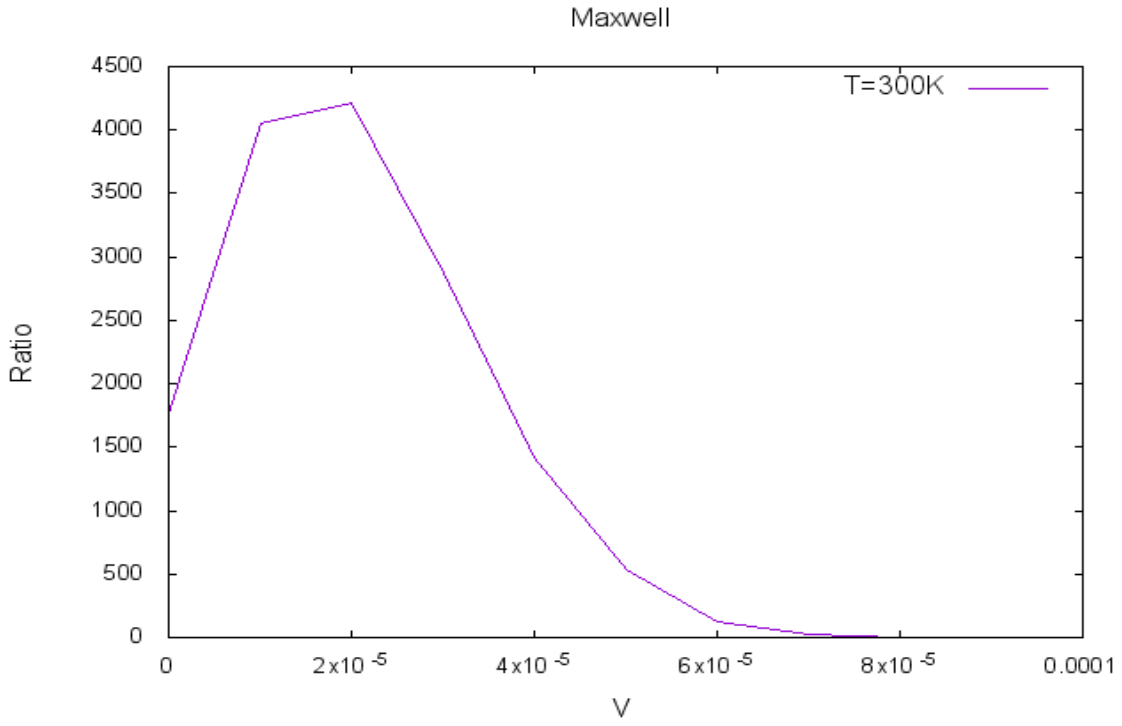

図 4.2: 式 (3.9)，(3.10) に従った場合の度数分布．

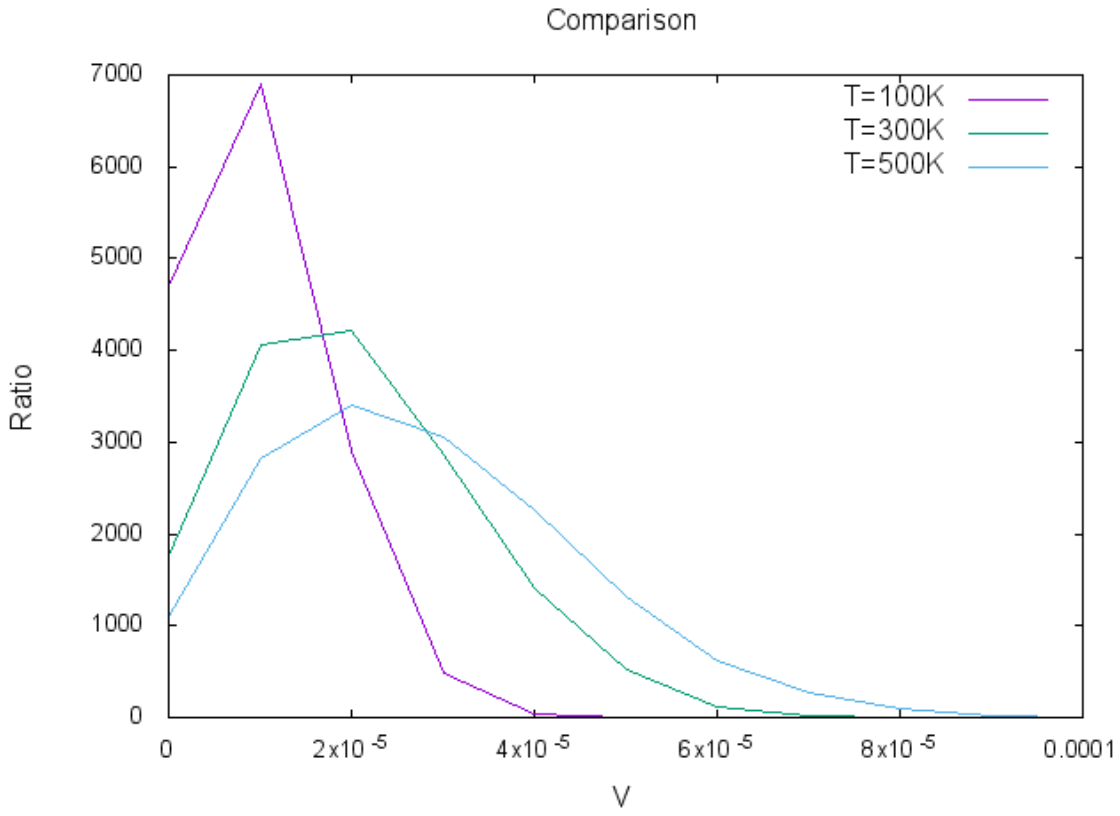

図 4.3: 絶対温度 T の値を変えた場合の度数分布.

4.2.3 移動距離

図 4.4 は時間の経過と粒子の総移動距離の関係を表したものである．縦軸は粒子の移動距離，横軸はフレーム数を表している．移動距離は，そのフレーム時点までの粒子の x 軸への変位値の合計．y 軸方向への変位値の合計を 2 乗して足し，その平方根をとっている．グラフから分かるようにフレーム数に比例するように移動距離は増加している．また，グラフの傾きの違いから温度が高くなれば粒子の移動距離は大きくなっていることがわかる．式 (2.1) より正しい動作をしていると考えられる．

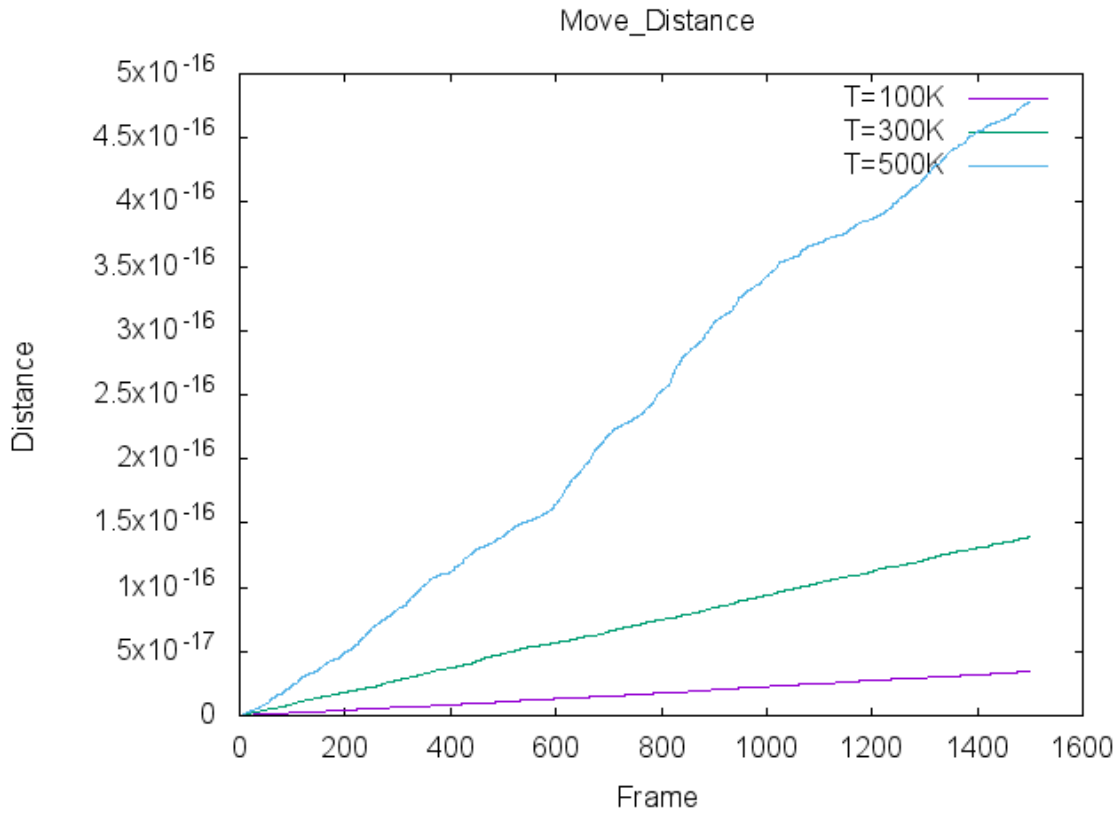

図 4.4: 粒子の移動距離を表したグラフ.

4.2.4 アボガドロ数

アインシュタインの式に従って計算おこなう．式 (2.4) で

$$\frac{RT}{3\pi\kappa P} \tag{4.2}$$

の部分は気体定数 $R = 8.31(\mathrm{J/molK})$, 絶対温度 $T = 300(\mathrm{K})$, ブラウン運動する粒子の半径 $P = 50(\mathrm{pixel})$，粘性係数 κ を水として $8.55 \times 10^{-4}(\mathrm{Pa/s})$ より $8.55 \times 10^{-7}(\mathrm{m^2/s})$ と設定する．未知の部分は

$$\frac{t}{\lambda_x^2} \tag{4.3}$$

の部分となる．変位値 λ_x は毎フレームごとの x 軸方向への粒子の変位 vx_pol(m/s) と y 軸方向への変位 vy_pol(m/s) それぞれに scale をかけて 2 乗して足す

$$\lambda_x = \sqrt{(\mathrm{scale} \times \mathrm{vx\_pol})^2 + (\mathrm{scale} \times \mathrm{vy\_pol})^2} \tag{4.4}$$

という式で表す．プログラム中では 100 フレーム毎の粒子の変位をとり，その値を利用して (4.3) の部分の値を計算している．アボガドロ数の値は初期値として設定した粒子の半径や，媒質の粘性係数の変更によって変化するものである．今回は媒質を水，粒子を花粉と考えて大きさや粘性係数を設定している．しかし，水分子の大きさと花粉の粒子の大きさを実際に考えた際にはプログラム上で表示されているような大きさの比率にはならない．また，分子数もプログラム上で表示されている数は実際の数には全く足りてはいない．これらの点からアボガドロ数は正確な値にはならない．また 1600 フレーム分プログラムを実行し続けた際に一定のフレーム数で，アボガドロ数の計算結果が大きく変化することが分かった．これに関しても粒子の初期位置をランダムで決定していることが問題だと予想するが確認はできていない．このプログラムを実際に使用する際にはブラウン運動をおこなう粒子や媒質を何と考えるかなど，初期値を適した値に設定する必要がある．

第5章　総括

本研究では，ブラウン運動をおこなう粒子の視覚化，また2物体の衝突のプログラムを作成した．それにより得られた成果を以下に示す．

1. 2物体の衝突の視覚化をおこなったことで質量が異なる際の物体の動きを見ることができるので直感的に理解しやすくなった．

2. ブラウン運動をおこない粒子の視覚化により，ブラウン運動の原因である分子の動きを理解できるようになった．またブラウン運動自体の動きがどういったものであるかも粒子の動きを見ることですぐに理解できる．

3. ブラウン運動のプログラムで得られる数値を追うアインシュタインの論文がどういった内容であるか，またペランがおこなった実験がどういった手法であったかを体験できると考えられる．

謝辞

本研究をおこなうにあたり，多大なるご指導，ご鞭撻をいただいた西谷滋人教授に深く御礼申し上げます．また研究の進行にあたり，様々な助力をして頂きました西谷研究室所属の先輩方，同輩に感謝の意を示します．本当にありがとうございました．

参考文献

[1] 湯川秀樹監修, 井上健, 谷川安孝, 中村誠太郎訳,「アインシュタイン選集１-相対性理論・量子論・ブラウン運動-」 (共立出版,1971).

[2] ノーベル物理学賞-Wikipedia https://ja.wikipedia.org/wiki/ノーベル物理学賞#1920.E5.B9.B4.E4.BB.A3(2 月 13 日時点閲覧)

[3] Processing.org https://processing.org/(2 月 1 日時点閲覧).

[4] Environment(IDE)/Processing.org https://processing.org/reference/environment/(2 月 1 日時点閲覧).

[5] 神山新一, 佐藤明著 「分子動力学シミュレーション」 (朝倉書店,1997)

[6] 北川浩, 北村隆行, 澁谷陽二, 中谷彰宏編著 「初心者のための分子動力学法」 (養賢堂,1997)

付 録A　ソースコード

ここでは作成したプログラムのソースコードを示す.

A.1　ブラウン運動

```
int X=15000;
float particle[][] = new float[X][2];
float pre_particle[][] = new float[X][2];
float pollen_x = 450;
float pollen_y = 250;
float pollen_r = 50;
float pre_pollen_x;
float pre_pollen_y;
float [] test_x = new float [X];
float [] test_y = new float [X];
float [] vx_pa = new float [X];
float [] vy_pa = new float [X];
float vx_pol = 0, vy_pol = 0;
float pollen_M = 1.0e-16, particle_m = 18/(6.02*pow(10, 23));
float radian;
float Px, Py, px, py;
float Px1, Py1, px1, py1;
float alpha, par_slope, pol_slope, plus, minus;
float pre_vx_pol, pre_vy_pol;
float E_total;
float sumx=0, sumy=0;
```

```
float k = 1.38e-23;
float T = 300;

int NUM = 500;
int L = 250;
float datax[] = new float[NUM];
float datay[] = new float[NUM];
int xtics[] = new int[NUM];
int ytics[] = new int[NUM];
float data_x[] = new float[L];
float data_y[] = new float[L];
int front = 0;
int rear = 0;
int first = 0;
int last = 0;

boolean hide = true;
boolean locus_hide = true;
boolean pause =true;



float N = 0;
float R = 8.31;
float u = 8.55e-5;
float t = 0;
float scale = 1.708e-6;

float sigmax,sigmay,sigmaxy;
PrintWriter output,output1,output2,output3;




void move() {
  for (int i=0; i < X; i++)
```

```
{
  particle[i][0] = particle[i][0] + vx_pa[i];
  particle[i][1] = particle[i][1] + vy_pa[i];
  pre_particle[i][0] = particle[i][0] - vx_pa[i];
  pre_particle[i][1] = particle[i][1] - vy_pa[i];
  pollen_x = pollen_x + vx_pol;
  pollen_y = pollen_y + vy_pol;
  pre_pollen_x = pollen_x - vx_pol;
  pre_pollen_y = pollen_y - vy_pol;

  par_slope = (particle[i][1] - pre_particle[i][1]) / (particle[i][0]
  - pre_particle[i][0]);

  if (particle[i][0] >= 600)
  {
    particle[i][0] = 600;
    particle[i][1] = (par_slope * 600) - (par_slope * pre_particle[i][0])
    + pre_particle[i][1];
    vx_pa[i] = vx_pa[i] * -1;
  }
  if (particle[i][0] <= 300)
  {
    particle[i][0] = 300;
    particle[i][1] = (par_slope * 300) - (par_slope * pre_particle[i][0])
    + pre_particle[i][1];
    vx_pa[i] = vx_pa[i] * -1;
  }
  if (particle[i][1] >= 400)
  {
    particle[i][1] = 400;
    particle[i][0] = (400-pre_particle[i][1]+par_slope*pre_particle[i][0])
    / par_slope;
    vy_pa[i] = vy_pa[i] * -1;
  }
```

```
if (particle[i][1] <= 100)
{
  particle[i][1] = 100;
  particle[i][0] = (100-pre_particle[i][1]+par_slope*pre_particle[i][0])
  / par_slope;
  vy_pa[i] = vy_pa[i] * -1;
}

// pollen collision
if (pollen_x + pollen_r >= 600)
{
  vx_pol = vx_pol * -1;
}
if (pollen_x - pollen_r <= 300)
{
  vx_pol = vx_pol * -1;
}
if (pollen_y + pollen_r >= 400)
{
  vy_pol = vy_pol * -1;
}
if (pollen_y - pollen_r <= 100)
{
  vy_pol = vy_pol * -1;
}

if (dist(particle[i][0], particle[i][1], pollen_x, pollen_y) <= pollen_r)
{
  pre_vx_pol = vx_pol;
  pre_vy_pol = vy_pol;
  vx_pol = (pollen_M*pre_vx_pol-pre_vx_pol
  *particle_m+2*particle_m*vx_pa[i])/(pollen_M+particle_m);
  vy_pol = (pollen_M*pre_vy_pol-pre_vy_pol*particle_m+2*particle_m
  *vy_pa[i])/(pollen_M+particle_m);
```

```
plus  = (par_slope*par_slope*pre_particle[i][0]+par_slope*pollen_y
-par_slope*pre_particle[i][1]+sqrt(par_slope*par_slope*pollen_r
*pollen_r-par_slope*par_slope*pollen_x*pollen_x+2*par_slope
*par_slope*pollen_x*pre_particle[i][0]-par_slope*par_slope
*pre_particle[i][0]*pre_particle[i[0]+2*par_slope*pollen_x
*pollen_y-2*par_slope*pollen_x*pre_particle[i][1]
-2*par_slope*pollen_y*pre_particle[i][0]+2*par_slope
*pre_particle[i][0]*pre_particle[i][1]+pollen_r*pollen_r
-pollen_y*pollen_y+2*pollen_y*pre_particle[i][1]
-pre_particle[i][1]*pre_particle[i][1])+pollen_x)
/(par_slope*par_slope+1);
minus = -(-par_slope*par_slope*pre_particle[i][0]-par_slope
*pollen_y+par_slope*pre_particle[i][1]+sqrt(par_slope*par_slope
*pollen_r*pollen_r-par_slope*par_slope*pollen_x
*pollen_x+2*par_slope*par_slope*pollen_x*pre_particle[i][0]
-par_slope*par_slope*pre_particle[i][0]
*pre_particle[i][0]+2*par_slope*pollen_x*pollen_y-2*par_slope*pollen_x
*pre_particle[i][1]-2*par_slope*pollen_y*pre_particle[i][0]
+2*par_slope*pre_particle[i][0]*pre_particle[i][1]+pollen_r*pollen_r
-pollen_y*pollen_y+2*pollen_y*pre_particle[i][1]-pre_particle[i][1]
*pre_particle[i][1])-pollen_x)/(par_slope*par_slope+1);

if (abs(particle[i][0]-plus) < abs(particle[i][0]-minus) ||
abs(particle[i][0]-plus) == abs(particle[i][0]-minus))
{
  particle[i][0] = plus;
  particle[i][1] = (par_slope * plus)-(par_slope * pre_particle[i][0])
  + pre_particle[i][1];
}
if (abs(particle[i][0]-plus) > abs(particle[i][0]-minus))
{
  particle[i][0] = minus;
  particle[i][1] = (par_slope * minus)-(par_slope * pre_particle[i][0])
  + pre_particle[i][1];
```

```
      }
      vx_pa[i]=(2*pollen_M*pre_vx_pol-pollen_M*vx_pa[i]+particle_m*vx_pa[i])
      /(pollen_M+particle_m);
      vy_pa[i]=(2*pollen_M*pre_vy_pol-pollen_M*vy_pa[i]+particle_m*vy_pa[i])
      /(pollen_M+particle_m);
    }
    E_total = pollen_M*(vx_pol*vx_pol)+pollen_M*(vy_pol*vy_pol)
    +particle_m*(vx_pa[i]*vx_pa[i])+particle_m*(vy_pa[i]*vy_pa[i]);
    if (hide==false) {
      stroke(188, 226, 232);
      ellipse(particle[i][0], particle[i][1], 1, 1);
    }
  }
}

void locus() {
  int num = 0;
  for (int i = front;; i++, num++) {
    i = i % NUM;
    num = num % NUM;
    if (i == rear % NUM) {
      break;
    }
    stroke(255, 0, 255);
    line(datax[i], datay[i], datax[(i+1)%NUM], datay[(i+1)%NUM]);
  }
  num =0;
  for (int i = first;; i++, num++) {
    i = i % L;
    num = num % L;
    if (i == last % L) {
      break;
    }
    stroke(0, 0, 255);
```

```
    line(data_x[i], 670-num, data_x[(i+1)%L], 670-num);
    stroke(255, 0, 0);
    line(900-num, data_y[i], 900-num, data_y[(i+1)%L]);
    stroke(0, 0, 0);
    line(450, 670, 450, 420);
    line(650, 250, 900, 250);
    if (xtics[i] != 0)
    {
      fill(0);
      text(xtics[i], 200, 670-num);
      text(ytics[i], 900-num, 100);
      fill(255);
      stroke(0, 0, 0);
      line(300, 670-num, 600, 670-num);
      line(900-num, 100, 900-num, 400);
    }
  }
}

void sum() {
  float tmpx,tmpy;
  if(last % 100 ==0){
    sumx=0;
    sumy=0;
    tmpx = vx_pol*scale * vx_pol*scale;
    sumx = sumx + tmpx;
    tmpy = vy_pol*scale * vy_pol*scale;
    sumy = sumy + tmpy;
    calc(sqrt(sumx+sumy));
    output3.println(N);
  } else {
    tmpx = vx_pol*scale * vx_pol*scale;
    sumx = sumx + tmpx;
    tmpy = vy_pol*scale * vy_pol*scale;
```

```
    sumy = sumy + tmpy;
  }
  sigmax = sigmax + tmpx;
  sigmay = sigmay + tmpy;
  sigmaxy = sigmaxy + tmpx + tmpy;

}

void calc(float x) {
   println("************");
   println(x);
   t = 0.1;
   N=(t/(x*x))*((R*T)/(3*PI*u*pollen_r));
}

void keyPressed() {
  if (key == 'c') {
    saveFrame("capture-####.png");
  }
  if (key == ' ') {
    if (hide == true) {
      hide = false;
    } else {
      hide = true;
    }
  }
    if (key == 'l') {
    if (locus_hide == true) {
      locus_hide = false;
    } else {
      locus_hide = true;
    }
  }
    if (key == 'p') {
```

```
    if (pause == true) {
      noLoop();
      pause = false;
    } else {
      loop();
      pause = true;
    }
  }
}



void setup() {
  output = createWriter("sigmax.txt");
  output1 = createWriter("sigmay.txt");
  output2 = createWriter("sigmaxy.txt");
  output3 = createWriter("N.txt");

  size(1000, 700);
  background(255);
  stroke(0, 0, 256);
  frameRate(60);
  rect(300, 100, 300, 300);
  for (int i=0; i<X; i++) {
    particle[i][0] = random(300, 600);
    particle[i][1] = random(100, 400);
    while (dist(particle[i][0], particle[i][1], pollen_x, pollen_y)
    <= pollen_r)
    {
      particle[i][0] = random(300, 600);
      particle[i][1] = random(100, 400);
    }
    stroke(188, 226, 232);
    ellipse(particle[i][0], particle[i][1], 1, 1);
  }
```

```
  /*for (int i=0; i < X; i++){
   radian = random(TWO_PI);
   vx_pa[i] = cos(radian);
   vy_pa[i] = sin(radian);
   }*/
  for (int i = 0; i < X; i++) {
    float rand1=random(1);
    float rand2=random(1);
    vx_pa[i] = sqrt(k*T/particle_m)*sqrt(-2*log(rand1))*cos(2*PI*rand2);
    vy_pa[i] = sqrt(k*T/particle_m)*sqrt(-2*log(rand1))*sin(2*PI*rand2);
  }
}

void draw() {
  background(255);
  stroke(0, 0, 256);
  println(E_total);
  rect(300, 100, 300, 300);
  stroke(256, 0, 0);
  ellipse(pollen_x, pollen_y, pollen_r*2, pollen_r*2);
  datax[rear%NUM] = pollen_x;
  datay[rear%NUM] = pollen_y;
  data_x[last%L] = pollen_x;
  data_y[last%L] = pollen_y;
  if (rear > NUM)
  {
    front = rear % NUM + 1;
    println("front=", front);
    println("rear=", rear);
  }
  if (last > L)
  {
    first = last % L + 1;
    println("first=", first);
```

```
  println("last=", last);
}
if (last % 100 == 0)
{
  xtics[last%L] = last;
  ytics[last%L] = last;
} else
{
  xtics[last%L] = 0;
  ytics[last%L] = 0;
}
if(locus_hide==false){
  locus();
}
sum();
output.println(sigmax);
output1.println(sigmay);
output2.println(sigmaxy);
if(rear == 1500){
output.flush();
output.close();
output1.flush();
output1.close();
output2.flush();
output2.close();
output3.flush();
output3.close();
}
rear++;
last++;
move();
fill(0);
textSize(20);
text("E="+E_total, 620, 540);
```

```
  text("N="+N, 620, 500);
  text("dx^2="+(vx_pol*vx_pol), 620, 560);
  text("dy^2="+(vy_pol*vy_pol), 620, 580);
  text("sqrt(dx^2+dy^2)="+sqrt(vx_pol*vx_pol+vy_pol*vy_pol), 620, 600);
  fill(255);
}
```

A.2　2物体の衝突

```
float large_x = 500;
float large_y = 100;
float large_r =25;
float M = 1;

float small_x = 300;
float small_y = 100;
float small_r = 10;
float m = 1;

float V = 0;
float v = 10;
float pre_v, pre_V;
float total, e_total, e;

int N = 300;
float data[] = new float[N];
int front = 0;
int rear = 0;
float data1[] = new float[N];
int ytics[] = new int[N];
int frame[] = {0,50,100,150,250};
int k = 0;
```

```
int k_front=0;
int k_rear=4;

boolean pause = true;

void keyPressed() {
    if (key == 'c'){
    saveFrame("capture-####.png");
  }
    if (key == 'p') {
    if (pause == true) {
      noLoop();
      pause = false;
    } else {
      loop();
      pause = true;
    }
  }
}

void move() {
  data[rear % N] = small_x;
  data1[rear % N] = large_x;
  if(rear % 50==0){
    ytics[rear%N] = rear;
  }
  else{
    ytics[rear%N] = 0;
  }
  if(rear > N)
  {
    front = rear % N + 1;
    println("front=", front);
    println("rear=", rear);
```

```
  }
  locus();
//  front = front + 1;

  pre_v = v;
  pre_V = V;
  if (large_x - large_r < 250)
  {
    large_x = 250 + large_r + (large_r - (large_x - 250));
    V = V * -1;
  }
  if (large_x + large_r > 750)
  {
    large_x = 750 - large_r - (large_r - (750-large_x));
    V = V * -1;
  }
  if (small_x - small_r < 250)
  {
    small_x = 250 + small_r + (small_r - (small_x - 250));
    v = v * -1;
  }
  if (small_x + small_r > 750)
  {
    small_x = 750 - small_r - (small_r - (750-small_x));
    v = v * -1;
  }
  if (dist(small_x, small_y, large_x, large_y) < small_r + large_r)
  {
    small_x = small_x - ((small_r + large_r)
    -dist(small_x, small_y, large_x, large_y)) / 2;
    large_x = large_x + ((small_r + large_r)
    -dist(small_x, small_y, large_x, large_y)) / 2;

    pre_v = v;
```

```
    pre_V = V;
    v = (2*M*pre_V-M*pre_v+m*pre_v)/(M+m);
    V = (M*pre_V-pre_V*m+2*m*pre_v)/(M+m);

    e_total = 0.5*M*V*V + 0.5*m*v*v;

  }

  e = abs(V - v)/abs(pre_V - pre_v);
  large_x = large_x + V;
  small_x = small_x + v;
}



void locus()
{
  int j = 0;
  for(int i = front; ; i++, j++)
  {
    i = i % N;
    j = j % N;
    println("jlen=", j);
    if(i == rear % N)
    {
      break;
    }

//    println("data=", data[1]);
    stroke(0, 0, 255);
    line(data[i],500-j,data[(i+1)%N],500-j);
    stroke(255, 0, 0);
    line(data1[i],500-j,data1[(i+1)%N],500-j);
    if(ytics[i]!=0){
```

```
      stroke(0, 0, 0);
      line(250,500-j,750,500-j);
      fill(0);
      textSize(16);
      text(ytics[i], 200, 500-j);
      fill(255);
    }
    if(rear < 300){
      fill(0);
      textSize(16);
      text(0, 200, 500);
      fill(255);
    }
  }
  rear++;
  println("rear=", rear);
}


void setup() {

  size(800, 600);
  background(255);
  frameRate(60);
  stroke(0, 0, 0);
  rect(250, 50, 500, 100);
  rect(250, 200, 500, 300);
  stroke(0, 0, 255);
  ellipse(small_x, small_y, small_r*2, small_r*2);
  stroke(255, 0, 0);
  ellipse(large_x, large_y, large_r*2, large_r*2);
}

void draw() {
```

```
  background(255);
  stroke(0, 0, 0);
  rect(250, 50, 500, 100);
  rect(250, 200, 500, 300);
  for(int i = 0; i < 10;i++)
  {
    line(50*i+250,200,50*i+250,500);
  }

  //  float V = 0;
  //float v = 10;
  fill(0);
  textSize(20);
  text("M="+M, 10, 200);
  text("m="+m, 10, 220);
  text("V_M="+V, 10, 240);
  text("v_m="+v, 10, 260);
  text("MV="+(M*V), 10, 280);
  text("mv="+(m*v), 10, 300);
  total = M * V + m * v;
  text("total="+total, 10, 320);
  text("E_total="+e_total, 10, 340);
  text("e="+e, 10, 360);
  fill(255);
  //
  move();
  stroke(0, 0, 255);
  ellipse(small_x, small_y, small_r*2, small_r*2);
  stroke(255, 0, 0);
  ellipse(large_x, large_y, large_r*2, large_r*2);
}
```