シミュレーション レポート

第11回～第15回

提出日　2021年1月25日

組番号　408
学籍番号　17406

氏名　金澤雄大

1　目的

シミュレーションの授業で学習したアルゴリズムを実装し, 数値計算を行うことを目的とする.

2　実験環境

実験環境を表 1 に示す. gcc は Windows 上の WSL(Windows Subsystem for Linux) で動作するものを用いる.

表 1: 実験環境

CPU	AMD Ryzen 5 3600 6-Core Processor
メモリ	16.0GB DDR4
OS	Microsoft Windows 10 Home
gcc	(Ubuntu 9.3.0-17ubuntu1〜20.04) 9.3.0
Make	GNU Make 4.2.1

3　課題 11

本章では課題 11 のプログラム, 実行結果, 手計算の 3 つについて述べる.

3.1　プログラム

リスト 1 に課題 11 のコードを示す.

リスト 1: 課題 11 のコード

```
#include<stdio.h>
#define N 4

double x[N]={0,0,0,0};

double A[N][N]={{1,2,1,5},
                {8,1,3,1},
                {1,7,1,1},
                {1,1,6,1}};
double b[N]={20.5,14.5,18.5,9.0};

void disp(double A[N][N],double b[N]){
    int i,j;
    printf("--------------------\n");
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            printf("%.2lf ",A[i][j]);
        }
        printf("| %.2lf\n",b[i]);
    }
    printf("--------------------\n");
}

void forwardElimination(){
    int i,j,k;
    double m;

```

```
    for(k=0;k<N-1;k++){
        for(i=k+1;i<N;i++){
            m=A[i][k]/A[k][k];
            //printf("m = %lf\n",m);
            A[i][k]=0;
            for(j=k+1;j<N;j++){
                A[i][j] = A[i][j]-A[k][j]*m;
                //printf("A[%d][%d] = %lf\n",i,j,A[i][j]);
            }
            b[i]=b[i] -b[k]*m;
            //printf("b[%d] = %lf\n\n",i,b[i]);
        }
        //disp(A,b);
    }
}
void dispAns(){
    int i;
    printf("Answer\n");
    for(i=0;i<N;i++){
        printf("x[%d] = %lf\n",i+1,x[i]);
    }
}

void backwordSubstitution(void){
    int k,j;
    x[N-1]=b[N-1]/A[N-1][N-1];
    for(k=N-2;k>=0;k--){
        x[k]=b[k];
        for(j=k+1;j<=N;j++){
            x[k]-=A[k][j]*x[j];
        }
        x[k]/=A[k][k];
    }
}

void gaussElimination(void){
    disp(A,b);
    forwardElimination();
    disp(A,b);
    backwordSubstitution();
    dispAns();
}

int main(void){
    gaussElimination();
    return 0;
}
```

3.2 実行結果

リスト 2 に課題 11 の実行結果を示す. リスト 2 では計算する行列, 前進消去の計算過程, 計算結果の 3 つを表示している. リスト 2 から,$(x_1, x_2, x_3, x_4) = (1, \frac{1}{2}, 3, 2)$ であることがわかる.

リスト 2: 課題 11 の実行結果

```
--------------------
1.00 2.00 1.00 5.00 | 20.50
8.00 1.00 3.00 1.00 | 14.50
1.00 7.00 1.00 1.00 | 18.50
1.00 1.00 6.00 1.00 | 9.00
--------------------
--------------------
1.00 2.00 1.00 5.00 | 20.50
0.00 -15.00 -5.00 -39.00 | -149.50
0.00 5.00 0.00 -4.00 | -2.00
0.00 -1.00 5.00 -4.00 | -11.50
```

```
--------------------
--------------------
1.00 2.00 1.00 5.00 | 20.50
0.00 -15.00 -5.00 -39.00 | -149.50
0.00 0.00 -1.67 -17.00 | -51.83
0.00 0.00 5.33 -1.40 | -1.53
--------------------
--------------------
1.00 2.00 1.00 5.00 | 20.50
0.00 -15.00 -5.00 -39.00 | -149.50
0.00 0.00 -1.67 -17.00 | -51.83
0.00 0.00 0.00 -55.80 | -167.40
--------------------
--------------------
1.00 2.00 1.00 5.00 | 20.50
0.00 -15.00 -5.00 -39.00 | -149.50
0.00 0.00 -1.67 -17.00 | -51.83
0.00 0.00 0.00 -55.80 | -167.40
--------------------
Answer
x[1] = 1.000000
x[2] = 2.000000
x[3] = 0.500000
x[4] = 3.000000
```

3.3 手計算

式 (1)〜式 (3) に手計算の結果を示す. 式 (3) より, 手計算と実行結果が一致していることがわかる. これより作成したプログラムが正しく動作していると考える.

$$\left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 8 & 1 & 3 & 1 & 14.5 \\ 1 & 7 & 1 & 1 & 18.5 \\ 1 & 1 & 6 & 1 & 9.0 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & -15 & -5 & -39 & -149.5 \\ 0 & 5 & 0 & -4 & -2 \\ 0 & -1 & 5 & -4 & -11.5 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & 0 & -5 & -51 & -155.5 \\ 0 & 0 & 25 & -24 & -59.5 \\ 0 & 1 & -5 & 4 & 11.5 \end{array}\right) \tag{1}$$

$$\rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & 0 & -5 & -51 & -155.5 \\ 0 & 0 & 0 & 279 & 837 \\ 0 & 1 & -5 & 4 & 11.5 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & 0 & -5 & 0 & -2.5 \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 1 & -5 & 4 & 11.5 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 1 & -5 & 4 & 11.5 \end{array}\right) \tag{2}$$

$$\rightarrow \left(\begin{array}{cccc|c} 1 & 2 & 1 & 5 & 20.5 \\ 0 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 1 & 0 & 0 & 2 \end{array}\right) \rightarrow \left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 3 \\ 0 & 1 & 0 & 0 & 2 \end{array}\right) \tag{3}$$

4 課題 12

本章では課題 12 のプログラムおよび実行結果について述べる.

4.1 プログラム

リスト 3 に課題 12 のコードを示す. ピボットの選択の有無は 89 行目の変数 isPivoing で管理している. 変数 isPivoing=1 のときピボット選択あり, 変数 isPivoing=0 のときピボット選択なしで動作する.

リスト 3: 課題 12 のコード

```
#include<stdio.h>
#include<math.h>
#define N 3

double x[N]={0,0,0};

double A[N][N]={{2,4,2},
                {1,2,3},
                {4,6,2}};
double b[N]={6,9,8};

void disp(){
    int i,j;
    printf("--------------------\n");
    for(i=0;i<N;i++){
        for(j=0;j<N;j++){
            printf("%.2lf ",A[i][j]);
        }
        printf("| %.2lf\n",b[i]);
    }
    printf("--------------------\n");
}

int pivoting(int m){
    int i;
    int maxvalue=0;
    int maxcolumn=m;
    for(i=m+1;i<N;i++){
        if(maxvalue<fabs(A[m][i])){
            maxvalue=fabs(A[m][i]);
            maxcolumn=i;
        }
    }
    return maxcolumn;
}

void changeindex(int p,int q){
    int tmp,i;
    for(i=0;i<N;i++){
        tmp = A[p][i];
        A[p][i] = A[q][i];
        A[q][i]=tmp;
    }
    tmp = b[p];
    b[p]=b[q];
    b[q]=tmp;
}

void forwardElimination(int isPivoting){
    int i,j,k;
    double m;
    int pivot;

   for(k=0;k<N-1;k++){
        for(i=k+1;i<N;i++){
            if(isPivoting){
                pivot = pivoting(i);
                changeindex(i,pivot);
            }
            m=A[i][k]/A[k][k];
            A[i][k]=0;
            for(j=k+1;j<N;j++){
                A[i][j] = A[i][j]-A[k][j]*m;
            }
            b[i]=b[i] -b[k]*m;
        }
    }
}
```

```
void dispAns(){
    int i;
    for(i=0;i<N;i++){
        printf("x[%d] = %lf\n",i+1,x[i]);
    }
}

void backwordSubstitution(void){
    int k,j;
    x[N-1]=b[N-1]/A[N-1][N-1];
    for(k=N-2;k>=0;k--){
        x[k]=b[k];
        for(j=k+1;j<=N;j++){
            x[k]-=A[k][j]*x[j];
        }
        x[k]/=A[k][k];
    }
}

int main(void){
    int isPivoting=1;
    // 入力行列の表示
    printf("\nisPivoting = %d\n",isPivoting);
    printf("Init Array A and Vector b\n");
    disp();
    // 前進消去
    forwardElimination(isPivoting);
    // 前進消去の結果を表示
    printf("\nResult of forward elimination\n");
    disp();
    // 後退代入
    backwordSubstitution();
    // 結果を表示
    printf("\nAnswer\n");
    dispAns();
    printf("\n");
    return 0;
}
```

4.2 実行結果

本節では課題 12 のコードの実行結果を示す.

4.3 ピボット選択なしの実行結果

ピボット選択なしのときの実行結果をリスト 4 に示す. リスト 4 からピボット選択なしのとき (x_1, x_2, x_3) はすべて負の無限大になっていることが読み取れる.

リスト 4: ピボット選択なしのときの実行結果

```

isPivoting = 0
Init Array A and Vector b
--------------------
2.00 4.00 2.00 | 6.00
1.00 2.00 3.00 | 9.00
4.00 6.00 2.00 | 8.00
--------------------

Result of forward elimination
--------------------
2.00 4.00 2.00 | 6.00
0.00 0.00 2.00 | 6.00
```

```
0.00 0.00 inf | inf
--------------------

Answer
x[1] = -nan
x[2] = -nan
x[3] = -nan
```

4.4 ピボット選択ありの実行結果

ピボット選択ありのときの実行結果をリスト 5 に示す. リスト 5 からピボット選択ありのとき $(x_1, x_2, x_3) = (2, -1, 3)$ になってることが読み取れる.

リスト 5: ピボット選択ありのときの実行結果

```

isPivoting = 1
Init Array A and Vector b
--------------------
2.00 4.00 2.00 | 6.00
1.00 2.00 3.00 | 9.00
4.00 6.00 2.00 | 8.00
--------------------

Result of forward elimination
--------------------
2.00 4.00 2.00 | 6.00
0.00 -2.00 -2.00 | -4.00
0.00 0.00 2.00 | 6.00
--------------------

Answer
x[1] = 2.000000
x[2] = -1.000000
x[3] = 3.000000
```

4.5 解が正しいことの確認

リスト 5 の解が正しいことを確認する. 式 (4) にリスト 5 の結果を代入して計算した結果を示す. 式 (4) から, 課題で与えられた式の右辺と, 計算結果が一致していることがわかる. これより正しい解になっていることがわかる. また, ピボット選択なしのときには正しく動作しないが, ピボット選択ありでは正しく動作する例があることが確認できた.

$$\begin{cases} 2x_1 + 4x_2 + 2x_3 = 2 \cdot 2 + 4 \cdot (-1) + 2 \cdot 3 = 6 \\ 1x_1 + 2x_2 + 3x_3 = 1 \cdot 2 + 2 \cdot (-1) + 3 \cdot 3 = 9 \\ 4x_1 + 6x_2 + 2x_3 = 4 \cdot 2 + 6 \cdot (-1) + 2 \cdot 3 = 8 \end{cases} \tag{4}$$

5 課題 13

本章では課題 13 のプログラムおよび実行結果について述べる.

5.1 プログラム

本節では課題 13 および課題 13b のプログラムについて述べる.

5.1.1 課題 13 のプログラム

リスト 6 に課題 13 のコードを示す. リスト 6 のコードでは回帰方程式の計算結果に加えて, 説明変数および従属変数の平均および分散, 共分散, 相関係数, 決定係数を計算して表示している.

リスト 6: 課題 13 のコード

```
#include<stdio.h>
#include<math.h>
#define N 4

double X[N] = {1,2,3,4};
double Y[N] = {0,1,2,4};


void RMSE(void){
    double meanx=0;
    double meany=0;
    double cov=0;
    double varx=0;
    double vary=0;
    double slope,intercept;
    double r;
    int i;
    // calculate mean
    for(i=0;i<N;i++){
        meanx+=X[i];
        meany+=Y[i];
    }
    meanx/=N;
    meany/=N;
    // calculate variance
    for(i=0;i<N;i++){
        varx+=(X[i]-meanx)*(X[i]-meanx);
        vary+=(Y[i]-meany)*(Y[i]-meany);
    }
    varx/=N;
    vary/=N;
    // calculate covariance
        for(i=0;i<N;i++){
        cov+=(X[i]-meanx)*(Y[i]-meany);
    }
    cov/=N;
    slope = cov/varx;
    intercept = meany-slope*meanx;
    r=cov/(sqrt(varx)*sqrt(vary));
    printf("説明変数の平均 %lf\n",meanx);
    printf("説明変数の分散 %lf\n",varx);
    printf("従属変数の平均 %lf\n",meany);
    printf("従属変数の分散 %lf\n",vary);
    printf("\n共分散 %lf\n",cov);
    printf("相関係数 %lf\n",r);
    printf("\n回帰方程式 y= %lf + %lf x\n",intercept,slope);
    printf("決定係数 %lf\n",r*r);
}

int main(void){
    RMSE();
    return 0;
}
```

5.1.2 課題 13b のプログラム

課題 13b は二次多項式の近似をガウスの消去法を用いて行うプログラムを作成する課題である. 本プログラムでは二次多項式の近似よりも更に一般に,n 次多項式または,n 変量の近似を行えるプログラムを作成し

た. リスト 7 に課題 13b のコードを示す. 多項式近似と重回帰分析の切り替えは 6 行目から 23 行目で行うことができる.

リスト 7: 課題 13b のコード

```
#include<stdio.h>
#include<math.h>
#define N 7
#define M 2

#define polynomial //多項式近似
//#define multiple // 重回帰分析

#ifdef polynomial
    // 課題13b
    double X[N] = {0.0,0.1,0.2,0.3,0.4,0.5,0.6};
    double Y[N] = {0.0,0.034,0.138,0.282,0.479,0.724,1.120};
    double transX[M][N];
#else // multiple
    // 課題13bと等価な問題
    //double transX[M][N]={{0.0,0.1,0.2,0.3,0.4,0.5,0.6},
    {0.0,0.01,0.04,0.09,0.16,0.25,0.36}};
    //double Y[N] = {0.0,0.034,0.138,0.282,0.479,0.724,1.120};

    // マンションの価格データ
    //double transX[M][N]={{51,38,57,51,53,77,63,69,72,73},{16,4,16,11,4,22,5,5,2,1}};
    //double Y[N]={3.0,3.2,3.3,3.3,3.9,4.4,4.5,4.5,5.4,5.4,6.0};
#endif


double x[M+1];
double A[M+1][M+1];
double b[M+1];

// データを標準出力
void dispData(void){
    int i,j;
    printf("       データ       \n");
    printf("--------------------\n");
    #ifdef polynomial
        for(i=0;i<N;i++){
            printf("%lf ",X[i]);
            printf("| %lf \n",Y[i]);
        }
    #else
    for(i=0;i<N;i++){
        for(j=0;j<M;j++){
            printf("%lf ",transX[j][i]);
        }
        printf("| %lf \n",Y[i]);
    }
    #endif
    printf("--------------------\n");
    printf("\n");
}
// A,bを標準出力
void disp(){
    int i,j;
    printf("      A           b \n");
    printf("--------------------\n");
    for(i=0;i<M+1;i++){
        for(j=0;j<M+1;j++){
            printf("%.2lf ",A[i][j]);
        }
        printf("| %.2lf\n",b[i]);
    }
    printf("--------------------\n\n");
}
// 結果を標準出力
```

```
65 void dispx(){
66     int i;
67     printf("y = %lf ",x[0]);
68     for(i=1;i<M+1;i++){
69         #ifdef polynomial
70             printf("%+lf x^%d ",x[i],i);
71         #else // multiple
72             printf("%+lf x%d ",x[i],i);
73         #endif
74     }
75     printf("\n");
76 }
77 // ピボット選択
78 int pivoting(int m){
79     int i;
80     int maxvalue=0;
81     int maxcolumn=m;
82     for(i=m+1;i<M+1;i++){
83         if(maxvalue<fabs(A[m][i])){
84             maxvalue=fabs(A[m][i]);
85             maxcolumn=i;
86         }
87     }
88     return maxcolumn;
89 }
90
91 void changeindex(int p,int q){
92     int tmp,i;
93     for(i=0;i<M+1;i++){
94         tmp = A[p][i];
95         A[p][i] = A[q][i];
96         A[q][i]=tmp;
97     }
98     tmp = b[p];
99     b[p]=b[q];
100    b[q]=tmp;
101 }
102
103 // Gauss Elimination
104 // solve Ax=b
105 // 前進消去
106 void forwardElimination(int isPivoting){
107     int i,j,k;
108     double m;
109     int pivot;
110
111    for(k=0;k<M;k++){
112         for(i=k+1;i<M+1;i++){
113             if(isPivoting){
114                 pivot = pivoting(i);
115                 changeindex(i,pivot);
116             }
117             m=A[i][k]/A[k][k];
118             A[i][k]=0;
119             for(j=k+1;j<M+1;j++){
120                 A[i][j] = A[i][j]-A[k][j]*m;
121             }
122             b[i]=b[i] -b[k]*m;
123         }
124     }
125 }
126 // 後退代入
127 void backwordSubstitution(void){
128     int k,j;
129     x[M]=b[M]/A[M][M];
130     for(k=M-1;k>=0;k--){
131         x[k]=b[k];
132         for(j=k+1;j<=M+1;j++){
133             x[k]-=A[k][j]*x[j];
134         }
```

```
135         x[k]/=A[k][k];
136     }
137 }
138 
139 // 分析関数
140 void RMSE(int isPivoting){
141     int i,j,k;
142     double tmp,p,q;
143     // transfit X shape
144     #ifdef polynomial
145         for(i=0;i<M;i++){
146             for(j=0;j<N;j++){
147                 tmp=1;
148                 for(k=0;k<i+1;k++){
149                     tmp*=X[j];
150                 }
151                 transX[i][j]=tmp;
152             }
153         }
154     #endif
155 
156     dispData();
157 
158     // cal A,b,x
159     for(k=0;k<M+1;k++){
160         for(j=0;j<M+1;j++){
161             A[k][j]=0;
162             for(i=0;i<N;i++){
163                 if(j==0){
164                     p=1;
165                 }else{
166                     p=transX[j-1][i];
167                 }
168 
169                 if(k==0){
170                     q=1;
171                 }else{
172                     q=transX[k-1][i];
173                 }
174                 A[k][j]+=p*q;
175             }
176         }
177         b[k]=0;
178         for(i=0;i<N;i++){
179             if(k==0){
180                 q=1;
181             }else{
182                 q=transX[k-1][i];
183             }
184             b[k]+=Y[i]*q;
185         }
186         x[k]=0;
187     }
188 
189     disp();
190     // solve by Gauss Elimination
191     forwardElimination(isPivoting);
192     backwordSubstitution();
193     dispx();
194 }
195 
196 int main(void){
197     int isPivoting=0;
198     RMSE(isPivoting);
199     return 0;
200 }
```

5.2 実行結果

本節では, 課題 13 のプログラムおよび課題 13b のプログラムの実行結果について述べる.

5.2.1 課題 13 のプログラムの実行結果

リスト 8 に課題 13 のコードの実行結果を示す. リスト 8 から回帰方程式は $-1.5 + 1.3x$ であることがわかる読み取れる. これは「わかりやすい数値計算入門」[1] の例題の答えと一致しているから, 作成したプログラムが正しいことが確認できた.

リスト 8: 課題 13 のプログラムの実行結果

```
説明変数の平均 2.500000
説明変数の分散 1.250000
従属変数の平均 1.750000
従属変数の分散 2.187500

共分散 1.625000
相関係数 0.982708

回帰方程式 y= -1.500000 + 1.300000 x
決定係数 0.965714
```

5.2.2 課題 13b のプログラムの実行結果 (多項式近似)

リスト 9 に課題 13b で与えられた二次近似の実行結果を示す. リスト 9 から近似結果は $y = 0.008333 - 0.057500x + 3.120238x^2$ で課題ページの値と一致していることが確認できる.

リスト 9: 課題 13b のプログラムの実行結果 (多項式近似)

```
        データ
--------------------
0.000000 | 0.000000
0.100000 | 0.034000
0.200000 | 0.138000
0.300000 | 0.282000
0.400000 | 0.479000
0.500000 | 0.724000
0.600000 | 1.120000
--------------------

       A            b
--------------------
7.00 2.10 0.91 | 2.78
2.10 0.91 0.44 | 1.34
0.91 0.44 0.23 | 0.69
--------------------

y = 0.008333 -0.057500 x^1 +3.120238 x^2
```

5.2.3 課題 13b のプログラムの実行結果 (重回帰分析)

リスト 10 に重回帰分析の実行結果を示す. 理論値は $y = 1.020130 + 0.066805x_1 - 0.080830x_2$ である. リスト 10 から, 理論値と実行結果の値が一致していることが確認できる.

リスト 10: 課題 13b のプログラムの実行結果 (重回帰分析)

```
        データ
--------------------
51.000000 16.000000 | 3.000000
38.000000 4.000000 | 3.200000
57.000000 16.000000 | 3.300000
51.000000 11.000000 | 3.900000
53.000000 4.000000 | 4.400000
77.000000 22.000000 | 4.500000
63.000000 5.000000 | 4.500000
69.000000 5.000000 | 5.400000
72.000000 2.000000 | 5.400000
73.000000 1.000000 | 6.000000
--------------------

       A            b
--------------------
10.00 604.00 86.00 | 43.60
604.00 37876.00 5224.00 | 2724.20
86.00 5224.00 1204.00 | 339.40
--------------------

y = 1.020130 +0.066805 x1 -0.080830 x2
```

6 課題 14

本章では課題 14 のプログラム, 実行結果および考察について述べる.

6.1 プログラム

リスト 11 に課題 14 のコードを示す. リスト 11 のコードはコンパイル時に「gcc -o kadai14.exe kadai14.c -DN=1000」という風にランダムウォークのステップ数を指定することができる.「-DN=1000」のとき,1000 ステップのランダムウォークを TRIAL 回実行する. 実行結果では, 結果を課題 13b のプログラムに入力して, 一次近似を行っている.

リスト 11: 課題 14 のコード

```
#include<stdio.h>
#include <stdlib.h>
#include <time.h>

//#define ALLSTEP
#define TRIAL 30
//#define N 1000

int randomWalk(int n,double p,int L){
    int S=0;
    double E,V,rnd;
    int i;
    #ifdef ALLSTEP
        //printf("%d,%d\n",0,S);
    #endif
    for(i=0;i<n;i++){
        rnd = 1 + (rand()%100);
        rnd/=100;
        if(rnd<=p){
            S+=L;
        }else{
            S-=L;
        }
        #ifdef ALLSTEP
```

```
25              //printf("%d,%d\n",i+1,S);
26          #endif
27      }
28      return S;
29  }
30
31  int main(void){
32      int i,j,r;
33      double p=0.5;
34      int L=1;
35      double S1,S2,V,Vmean;
36      srand((unsigned) time(NULL));
37      Vmean=0;
38      for(i=0;i<TRIAL;i++){
39          S1=0;
40          S2=0;
41          #ifdef ALLSTEP
42          printf("\n%d回目の試行\nr(%d) = ",i+1,N);
43          #endif
44          for(j=0;j<TRIAL;j++){
45              r=randomWalk(N,p,L);
46              #ifdef ALLSTEP
47              printf("%d, ",r);
48              #endif
49              S1+=r;
50              S2+=r*r;
51          }
52          #ifdef ALLSTEP
53          printf("\n");
54          #endif
55
56          V=(S2/TRIAL) - (S1/TRIAL)*(S1/TRIAL);
57          Vmean+=V;
58          #ifdef ALLSTEP
59          printf("V = %lf\n",V);
60          printf("----------\n");
61          #endif
62      }
63      Vmean/=(TRIAL);
64      //printf("Vmean = %lf\n",Vmean);
65      printf("%d,%lf\n",N,Vmean);
66      return 0;
67  }
```

6.2 実行結果と考察

リスト 11 のプログラムを次の条件で実行し, 特徴を考察する. ステップ数 N は 1000 から 5000 の範囲で 100 おきに実行する.

1. p=0.5,TRIAL=5
2. p=0.5,TRIAL=30
3. p=0.7,TRIAL=30

まず,p=0.5,TRIAL=5 の条件で実行する. 図 1 に p=0.5,TRIAL=5 の条件での実行結果を示す. 図 1 から, ステップ数 N と分散 V には, 正の相関があることが読み取れる.

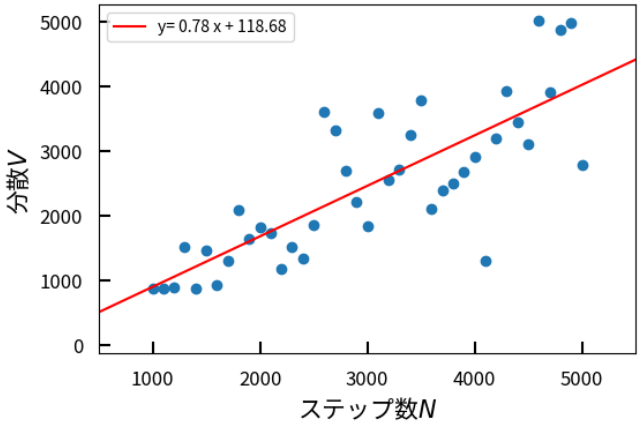

図 1: p=0.5,TRIAL=5 のときの実行結果

次に TRIAL=30 にして実行する. 図 2 に p=0.5,TRIAL=30 の条件での実行結果を示す. 図 2 から, ステップ数 N と分散 V に強い正の相関があることが読み取れる. さらに, 回帰直線がおおよそ $V = N$ であることが読み取れる.

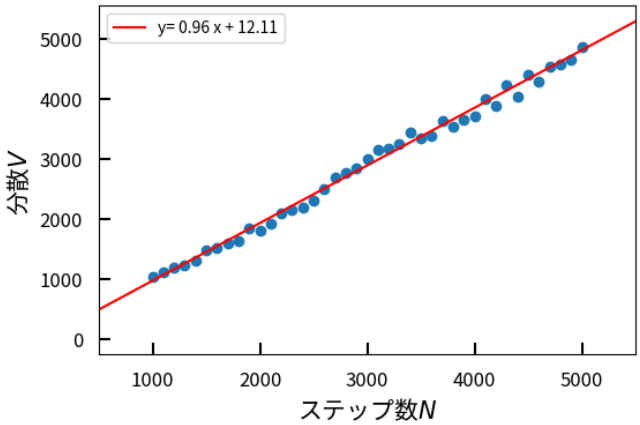

図 2: p=0.5,TRIAL=30 のときの実行結果

最後に p=0.7,TRIAL=30 の条件で実行する. 図 3 に p=0.7,TRIAL=30 の条件での実行結果を示す. 図 3 から, 単回帰係数が 0.83 と,p=0.5 と比較して傾きが緩やかになっていることが読み取れる.

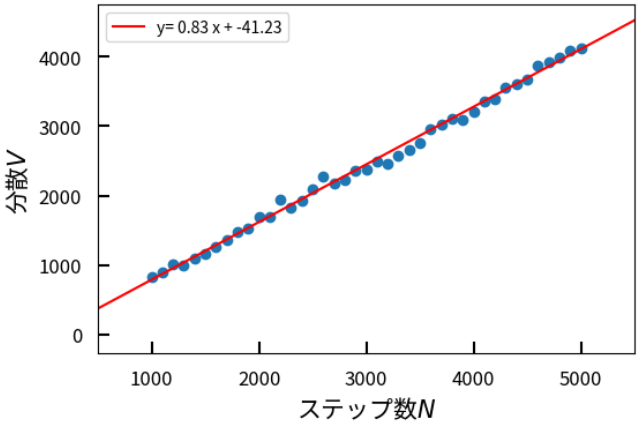

図 3: p=0.7,TRIAL=30 のときの実行結果

分散と N,p の関係について考察する. $S_1, S_2, \ldots S_n$,i.i.d(独立同分布) とし,$P(S_i = 1) = p, P(S_i = -1) = 1 - p$ とする ($L = 1$ に限定して考える). $n \geq 1$ に対して確率変数 X_n を式 (5) のように定義する. 式 (5) のように定義すると X_n は時刻 n における位置を表している. また, この運動がランダムウォークである.

$$X_n = S_1 + S_2 + \cdots + S_n \tag{5}$$

$E[X_n]$ および $V[X_n]$ を計算するために, 時刻 n に位置が k である確率 $P(X_n = k)$ を求める必要がある. 確率 $P(X_n = k)$ を計算するために W_n,L_n という 2 つの変数を定義する. $S_i = 1$ なる i の個数 W_n,$S_i = -1$ なる i の個数 L_n とする. このとき,W_n は二項分布 $Bin(n, p)$ に従う. W_n,L_n を用いると $X_n = W_n - L_n$,$n = W_n + L_n$ と表せる. これより $X_n + n = 2W_n$ と表せる. したがって,$X_n = k$ は $W_n = \frac{k+n}{2}$ と表せる. $k + n$ が奇数のとき,$\frac{k+n}{2}$ は整数にならない. このため,${}_nC_{\frac{k+n}{2}}$ は計算できないから,$P(X_n = k) = 0$ となる.$k + n$ が偶数のときは $P(X_n = k)$ は式 (6) になる.

$$P(X_n = k) = P\left(W_n = \frac{n+k}{2}\right) = {}_nC_{\frac{n+k}{2}} p^{\frac{n+k}{2}} (1-p)^{\frac{n-k}{2}} \tag{6}$$

式 (6) は二項分布だから期待値と分散は簡単に計算できる. X_n の期待値および分散は式 (7) および式 (8) になる.

$$E[X_n] = E[2W_n - n] = 2np - n = n(2p - 1) \tag{7}$$

$$V[X_n] = V[2W_n - n] = 4V[W_n] = 4np(1 - p) \tag{8}$$

$p = 0.5$ のとき, 期待値と分散を計算してみる. 式 (7) および式 (8) に $p = 0.5$ を代入すると式 (9) および式 (10) のようになる. 式 (10) は, 数値計算で求めた分散 $V[\Delta x^2(N)] = N$ になることを示している. 図 2 で示した回帰係数がおおよそ 1 になっていることから,$p = 0.5$ のときの分散 $V[\Delta x^2(N)] = N$ という理論と一致していることがわかる.

$$E[X_n] = n(2p - 1) = 0 \tag{9}$$

$$V[X_n] = 4np(1 - p) = n \tag{10}$$

$p = 0.7$ のとき, 期待値と分散を計算してみる. 式 (7) および式 (8) に $p = 0.7$ を代入すると式 (11) および式 (12) のようになる. 式 (12) は, 数値計算で求めた分散 $V[\Delta x^2(N)] = 0.84N$ になることを示している. 図

3 で示した回帰係数がおおよそ 0.83 になっていることから,$p = 0.7$ のときの分散 $V[\Delta x^2(N)] = 0.84N$ という理論と一致していることがわかる.

$$E[X_n] = n(2p-1) = 0.4n \tag{11}$$

$$V[X_n] = 4np(1-p) = 0.84n \tag{12}$$

7 課題 14-2

本章では, 課題 14-2 のプログラムおよび実行結果と考察について述べる.

7.1 プログラム

リスト 12 に課題 14-2 のコードを示す. リスト 12 のコードは N ステップの 2 次元ランダムウォークを 1 回行うコードであり, 複数回呼び出すためにマクロを組んでいる.

リスト 12: 課題 14-2 のコード

```
#include<stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

// 試行回数n
// 移動距離 L
void randomWalk2(int n,int L){
    double X=0;
    double Y=0;
    double rnd;
    int i;
    printf("%d,%lf,%lf\n",0,0.0,0.0);
    for(i=0;i<n;i++){
        // make random[0,1]
        rnd = (double)rand()/RAND_MAX;
        // make random[0,2pi]
        rnd*=2*M_PI;
        X+=L*cos(rnd);
        Y+=L*sin(rnd);
        printf("%d,%lf,%lf\n",i+1,X,Y);
    }
}

int main(void){
    srand((unsigned) time(NULL));
    randomWalk2(5000,1);
    return 0;
}
```

7.2 実行結果と考察

リスト 12 のコードをステップ数が 100,1000,5000 のときについて, それぞれ 200 回実行した. 図 4〜図 6 に実行結果を示す. 図 4 はステップ数が 100, 図 5 は 1000, 図 6 は 5000 のときのグラフである. 図 4〜図 6 から, ステップ数を大きくすると移動距離の半径が多くなる傾向が読み取れる.

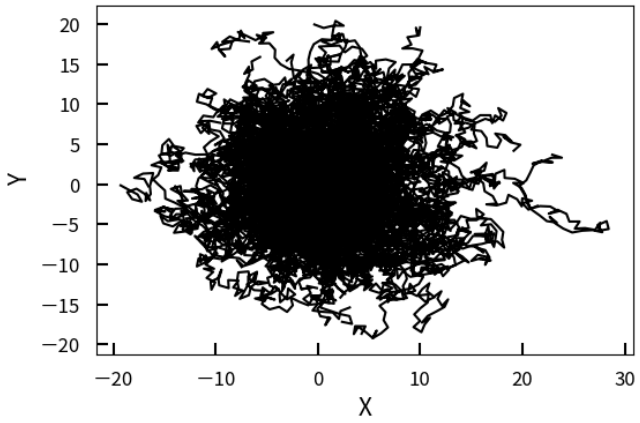

図 4: ステップ数が 100 のときの実行結果

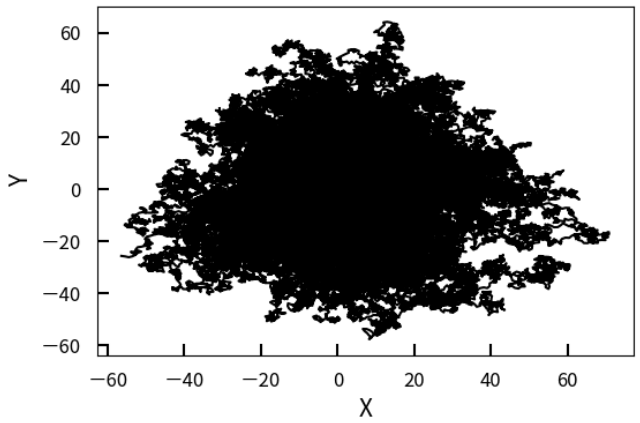

図 5: ステップ数が 1000 のときの実行結果

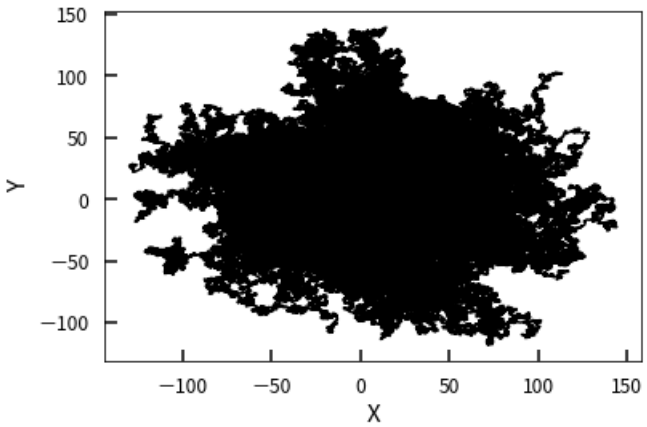

図 6: ステップ数が 5000 のときの実行結果

ステップ数を大きくしたときの半径について考察する. 表 2 に N ステップ経過したときの粒子の原点 $(0, 0)$ からの距離の平均および標準偏差を示す. 表 2 から,N ステップ数後の原点からの距離はおおよそ平均 N, 標

準偏差 N であることが読み取れる. このことから,2 次元ランダムウォークの場合も分散は N になると考えられる.

表 2: N ステップ後の距離の平均と標準偏差

ステップ数	平均	標準偏差
100	103.62	101.18
1000	964.67	855.02
5000	4653.74	4475.67

参考文献

[1] 栗原正仁,”わかりやすい数値計算入門”, ムイスリ出版,2018

[2] 久保田達也,”現代数理統計学の基礎”, 共立出版社,2020