1　課題 2

誤差を含むデータに対する，最小二乗法による関数近似の精度を調べるために，以下の実験を行え.

2　1. この資料の最後にあるプログラム例を参考に, 課題（1）のプログラムを改変して,csv ファイルから x と y のデータを読み込み, 近似式の傾き a と切片 b を出力するプログラムを作成せよ.

題意の通りのプログラムをソースコード 1 に示す.

ソースコード 1: approximate.c

```
1  #include <stdio.h>
2  #include <math.h>
3
4  #define N 10000
5  int main(int argc, char *argv[]){
6    int n = 0;
7    double x[N],y[N];
8    FILE*fp = stdin;
9
10   if (argc >= 2){
11       fp = fopen(argv[1],"r");
12   }
13   while(fscanf(fp,"%lf,%lf",&x[n],&y[n]) != EOF) {
14       n++;
15   }
16
17   fclose(fp);
18   double a,b,s1,s2,avex,avey,temp1=0,temp2=0,sumx=0,sumy=0;
19   for (int i = 0; i < n; i++) {
20      temp1 += x[i]* y[i];
21      temp2 += pow(x[i],2);
22      sumx += x[i];
23      sumy += y[i];
24   }
25
26   s1 = (1/(double)(n))*temp1;
27   s2 = (1/(double)(n))*temp2;
28   avex =(1/(double)(n))* sumx;
29   avey =(1/(double)(n))* sumy;
30   a = (s1-(avex*avey))/(s2-pow(avex,2));
31   b = ((avey*s2)-(avex*s1))/(s2-pow(avex,2));
32   printf("傾きa = %g,切片b= %g\n",a,b);
33   return 0;
34 }
```

ファイルがコンパイルできることを確かめる. 実行結果は 3 で示す.

3 2.generate-data.c（添付ファイル）をコンパイル・実行して,100対（つまり n=100）の x, y のデータを生成せよ. 生成したデータをファイル（例えば,data.csv）に保存せよ.

プログラムをソースコード 2 に示す.

ソースコード 2: generate-data.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <time.h>
5  #include <unistd.h>
6
7  /*********************************************
8   generate-data.c
9
10  y = ax + b + e (e は N(0,1)) から x,y を生成するプログラム
11
12  *********************************************/
13
14 #define randDouble ((double)rand()+1.0)/((double)RAND_MAX+2.0) // 0以上 1未満の
       実数値の乱数を生成
15
16 double rand_normal(double mu, double sigma)
17 {
18   double e = sqrt(-2.0*log(randDouble)) * sin(2.0*M_PI*randDouble);
19   return mu + sigma*e;
20 }
21
22 int main(int argc, char *argv[])
23 {
24   int i, n = 100;
25   double a = 0.7, b = 1.2, e;
26   double x, y;
27
28   if(argc >= 2)
29     n = atoi(argv[1]);
30
31   srand((unsigned int)time(NULL));
32   rand();
33
34   for(i=0; i<n; i++){
35     x = randDouble * 10.0;
36     e = rand_normal(0,1);
37     y = a * x + b + e;
38     printf("%.3lf,%.3lf\n",x,y);
39   }
40   sleep(1);
41 }
```

csv にこの出力結果をそのままコピーペーストすればよいが後に何度も使うのと, 数が多いため自動でファイルを生成するように変更する.

ソースコード 3: generate-data.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <time.h>
5  #include <unistd.h>
6  #include <string.h>
7
8  /*******************************************
9   generate-data.c
10
11  y = ax + b + e (e は N(0,1)) から x,y を生成するプログラム
12
13  *******************************************/
14
15 #define randDouble ((double)rand()+1.0)/((double)RAND_MAX+2.0) // 0以上 1未満の
       実数値の乱数を生成
16
17 double rand_normal(double mu, double sigma)
18 {
19   double e = sqrt(-2.0*log(randDouble)) * sin(2.0*M_PI*randDouble);
20   return mu + sigma*e;
21 }
22
23 int main(int argc, char *argv[])
24 {
25   FILE *file;
26   // ファイルを書き込みモードでオープン
27   file = fopen(argv[1], "w");
28
29   if (file == NULL){
30         printf("ファイルをオープンできませんでした。\n");
31         return 1;
32   }
33
34   int i, n = 100;
35   double a = 0.7, b = 1.2, e;
36   double x, y;
37
38   if(argc >= 2)
39     n = atoi(argv[2]);
40
41   srand((unsigned int)time(NULL));
42   rand();
43   for(i=0; i<n; i++){
```

```
44      x = randDouble * 10.0;
45      e = rand_normal(0,1);
46      y = a * x + b + e;
47      fprintf(file, "%.3lf,%.3lf\n",x,y);
48    }
49    sleep(1);
50    printf("ファイル %s を生成しました。\n", argv[1]);
51  }
```

追記分は string.h パッケージとファイルの作成, 書き込みである. 実行結果のスクリーンショットを以下に示す.

```
genku@gen-mba kadai2-2 % gcc generate-data.c -lm -o generate-data
genku@gen-mba kadai2-2 % ./generate-data data.csv 100
ファイル data.csv を生成しました。
```

data.csv には 100 行の x,y の数値が書き込まれていることを確認できた.

4 3. 上記 1 のプログラムを用いて,2 で保存したファイルからデータを読み込み, 近似式の傾きと切片を求めよ.

実行結果をいかに示す.

```
genku@gen-mba kadai2-2 % gcc approximate.c -lm -o approximate
genku@gen-mba kadai2-2 % ./approximate data.csv
傾き a = 0.729127, 切片 b= 1.25576
```

結果が妥当かどうかを表計算ソフトを用いて確認してみる.CSV をそのままインポートして散布図を作成し, 近似曲線を表示すると以下のようになる. また表から算出した傾きと切片はそれぞれ a=0.729127179,b=1.25575569 となった.

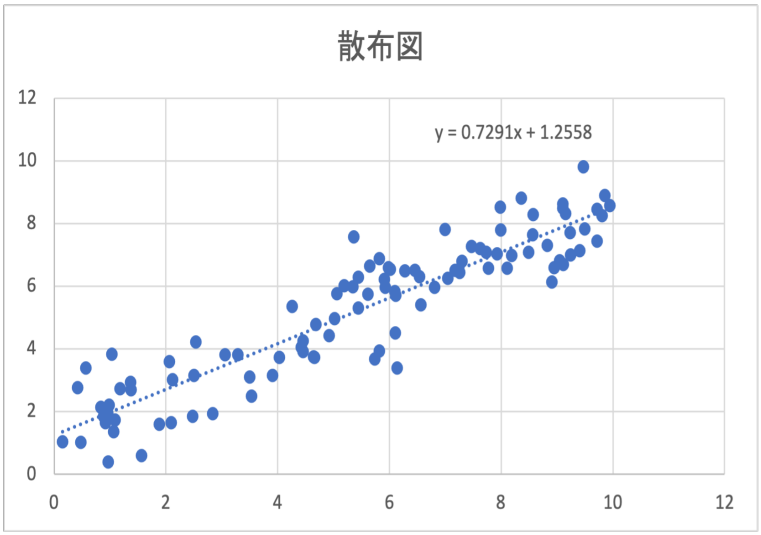

図 1: data.csv の散布図

これらの結果よりプログラム approximate.c は想定通りの動作をしているといえる.

5 4. 前ページの 2〜3 を 100 回繰り返して得られた 100 個の傾きの値に対して, ヒストグラムを作成せよ.

generate-data.c と approximate.c をそれぞれコピーして main から 100 回呼び出せるように変更する. 実際のプログラムをソースコード 4 に示す. 行数もそこまで多くないのでそのまま一つのファイルに書き下す.

ソースコード 4: main.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <time.h>
5  #include <unistd.h>
6  #include <string.h>
7
8  #define N 10000
9  double approximate(const char* filename) {
10     int n = 0;
11     double x[N],y[N];
12     FILE*fp = stdin;
13
14     fp = fopen(filename,"r");
15     while(fscanf(fp,"%lf,%lf",&x[n],&y[n]) != EOF) {
16         n++;
17     }
18
19     fclose(fp);
20     double a,b,s1,s2,avex,avey,temp1=0,temp2=0,sumx=0,sumy=0;
21     for (int i = 0; i < n; i++) {
22        temp1 += x[i]* y[i];
23        temp2 += pow(x[i],2);
24        sumx += x[i];
25        sumy += y[i];
26     }
27
28     s1 = (1/(double)(n))*temp1;
29     s2 = (1/(double)(n))*temp2;
30     avex =(1/(double)(n))* sumx;
31     avey =(1/(double)(n))* sumy;
32     a = (s1-(avex*avey))/(s2-pow(avex,2));
33     b = ((avey*s2)-(avex*s1))/(s2-pow(avex,2));
34     printf("傾きa = %g,切片b= %g\n",a,b);
35     remove(filename);
36     return a;
37 }
38
```

```
39  #define randDouble ((double)rand()+1.0)/((double)RAND_MAX+2.0) // 0以上 1未満
        の実数値の乱数を生成
40
41  double rand_normal(double mu, double sigma)
42  {
43    double e = sqrt(-2.0*log(randDouble)) * sin(2.0*M_PI*randDouble);
44    return mu + sigma*e;
45  }
46
47  int generate_data(const char* filename, int n)
48  {
49      FILE *file;
50      // ファイルを書き込みモードでオープン
51      file = fopen(filename, "w");
52
53      if (file == NULL){
54              printf("ファイルをオープンできませんでした。\n");
55              return 1;
56      }
57
58      int i= 100;
59      double a = 0.7, b = 1.2, e;
60      double x, y;
61
62      srand((unsigned int)time(NULL));
63      rand();
64      for(i=0; i<n; i++){
65          x = randDouble * 10.0;
66          e = rand_normal(0,1);
67          y = a * x + b + e;
68          fprintf(file, "%.3lf,%.3lf\n",x,y);
69      }
70      fclose(file);
71      sleep(1);
72      return 0;
73  }
74
75  int main(int argc, char *argv[])
76  {
77      FILE *file;
78      // ファイルを書き込みモードでオープン
79      int datasize = 100;
80      char slopefilename[100];
81      if(argc >= 2)
82      datasize = atoi(argv[1]);
83      sprintf(slopefilename,"slope%d.csv",datasize);
84      file = fopen(slopefilename, "w");
85
86      if (file == NULL){
87              printf("ファイルをオープンできませんでした。\n");
```

```
88                 return 1;
89         }
90         for(int i=0; i<100; i++)
91         {
92             char filename[100] = "data.csv";
93 
94             generate_data(filename,100);
95             double slope = approximate(filename);
96             fprintf(file, "%lf\n",slope);
97         }
98         fclose(file);
99         return 0;
100    }
```

この実行結果の slope100.csv を表計算ソフトで読み込みヒストグラムを作成する．結果は以下の通りとなった．

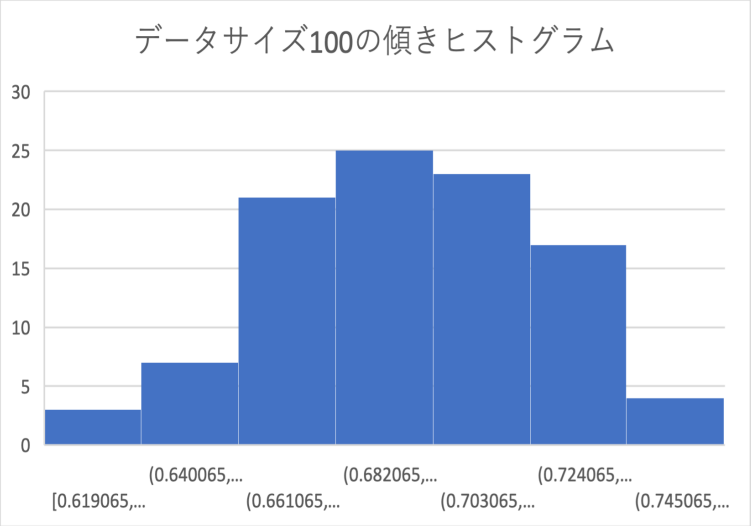

図 2: データサイズ 100 の傾きのヒストグラム

6 5. 前ページの 2 で生成されるデータサイズ（=データ対の個数 n）を変える（例えば,1000 や 10000 にする）ことで, ヒストグラムがどのようになるかを観察せよ．

ソースコード 4 に引数 1000 や 10000 を与えてデータを取り出す．4 と同様に表計算ソフトで読み込みヒストグラムを作成する．

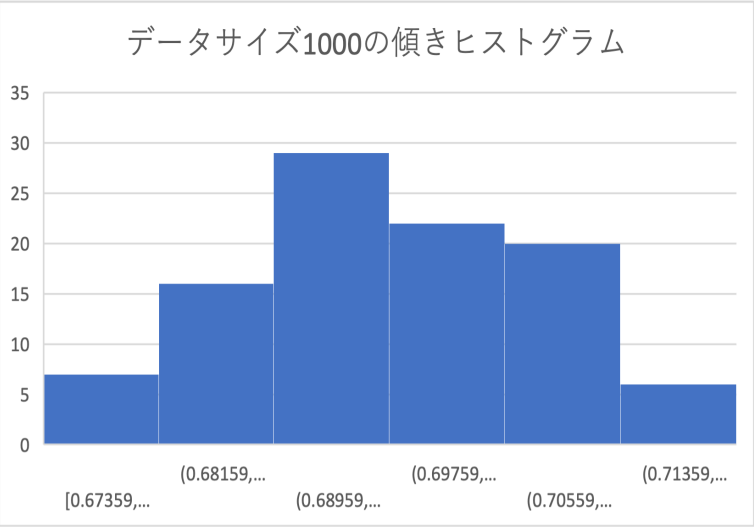

図 3: データサイズ 1000 の傾きのヒストグラム

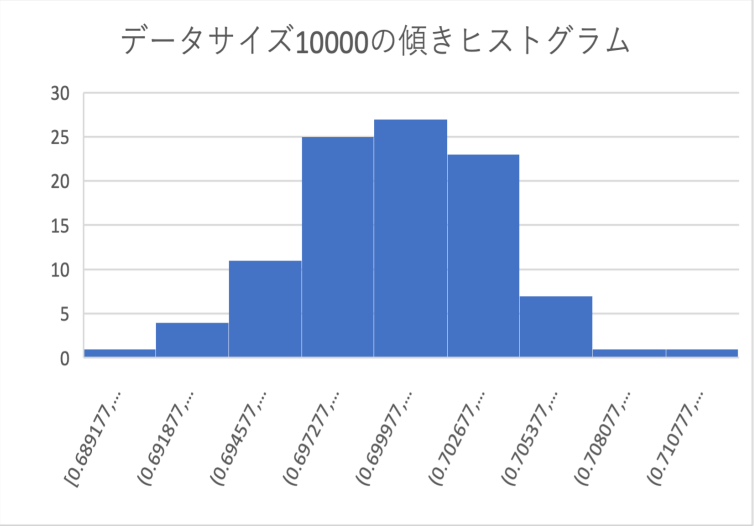

図 4: データサイズ 10000 の傾きのヒストグラム

7　6. 上記の 4〜5 を通じて, 実験によりわかったことや, その原因を考察して, レポートにまとめよ.

傾きのヒストグラムをそれぞれ比較してみるとデータサイズが大きくなるに連れて, 傾きは 0.7 から遠い値が少なくなっている. たとえばデータサイズが 100 のとき一番小さい値が 0.619, データサイズが 1000 のとき一番小さい値が 0.673, データサイズが 10000 のとき一番小さい値が 0.689 とどんどん 0.7 に近くなって言っているのがわかる.

このことより, データサイズの数を増やせば増やすほど,0.7 に近くなっていくと想定できる. よってデータ数が多いほど最小二乗法の近似式の精度が高くなることがわかる.

実際に自身で数個から数十個程度のデータをとりだして x 軸,y 軸のグラフに点をうち, すべての点に近い線を引こうとすると数個の点と数十個の点だと大きくずれが生じる. ある程度相関のある測定した 2 つの数, たとえば身長と体重というある程度個人差があるものでもデータの数が多くなれば近似式はより正確になる. 現実にありえる程度に極端に背が高く体重が軽い人が 100 人のデータの中に混じっていればかなりの誤差が生じるが 10000 人の中に一人混じっていると極端な数値の人より一般的な数値の人の数のほうが圧倒的に多くなるので近似式は正確になるだろう. このように少ないデータに現実にありえる大きく外れた値が混じってしまうと誤差が大きくなってしまうが, データ数が多い場合は大きく外れた値は相対的に少なくなるので誤差も少なくなる. このことがデータサイズの数によって, ヒストグラムが変化する原因だといえる.