アルゴリズム第 2 期末レポート

61908697 佐々木良輔

問 3 最小二乗法の拡張

最小二乗法について

最小二乗法は測定で得られた離散的なデータを関数で近似するために用いられる．ここでは特に近似を行う関数形が m 次多項式の場合を考える．すなわち関数は

$$g(x) = \sum_{k=0}^{m} a_k x^k = a_0 + a_1 x + \cdots + a_m x^m \tag{1}$$

という形で表される．ここで n 組の離散データの組が (x_i, y_i) で与えられるとき最小二乗法ではこれらのデータと $g(x)$ の残差の二乗和 E が最小となるように a_k を定める．すなわち目的関数は以下で与えられる．ただし $a_0, \ldots, a_m$ を a_k と略記している．

$$\text{minimize} : E(a_k) = \sum_{i=0}^{n-1} (y_i - g(x_i))^2 \tag{2}$$

ここで a_k について $E(a_k)$ が唯一極小値を持つことを仮定すると $E(a_k)$ が最小となる条件は

$$\begin{aligned} \frac{\partial E}{\partial a_k} &= \frac{\partial}{\partial a_k} \sum_{i=0}^{n-1} (y_i - (a_0 + \cdots + a_k x_i^k + \cdots + a_m x_i^m))^2 \\ &= 2 \sum_{i=0}^{n-1} (y_i - g(x_i))(-x_i^k) = 0 \end{aligned} \tag{3}$$

すなわち

$$\sum_{i=0}^{n-1} (a_0 x_i^k + a_1 x_i^{k+1} + \cdots + a_m x_i^{k+m}) = \sum_{i=0}^{n-1} y_i x_i^k \tag{4}$$

となる．$k = 0, \ldots, m$ までを並べて行列とすると

$$\begin{pmatrix} n & \overline{x} & \cdots & \overline{x^m} \\ \overline{x} & \overline{x^2} & \cdots & \overline{x^{m+1}} \\ \vdots & \vdots & \vdots & \vdots \\ \overline{x^m} & \overline{x^{m+1}} & \cdots & \overline{x^{2m}} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{pmatrix} = \begin{pmatrix} \overline{y} \\ \overline{xy} \\ \vdots \\ \overline{x^m y} \end{pmatrix} \tag{5}$$

ここで $\overline{x^k y}$ などは $\sum_{i=0}^{n-1} x_i^k y_i$ を意味する. 以上から (5) 式の連立方程式を解くことで $E(a_k)$ を最小化する a_k が得られる．また m 次多項式での最小二乗法は係数行列の生成に $O(m^2 \times n)$ の総和計算, 連立方程式を解くのには $O(m^2)$ の計算量が必要と見積もられる.

実装について

以上の最小二乗法を数値計算するプログラム (lsm_polynomial.c) を C 言語で実装した. ソースコードは補遺 A に示した．入力データは 1 列目に時刻, 2 列目に値を持ったテキストファイルである．また入力ファイルの 1 行 1 列にはデータ数を, 1 行 2 列には多項式の次数を与える．連立方程式のソルバーとしてガウスの消去法を用いた.

動作確認

まず動作確認として, 第 10 回授業で配布された xy9.txt と同一のデータを直線回帰し, その結果を同じく第 10 回授業で配布された lsm.c の出力結果と比較した．それぞれのプログラムで得られた回帰直線は以下の通りである．この結果から 2 つの lsm_polynomial.c は lsm.c と全く同一の出力をしており, 正常に動作していると考えられる.

表 1　動作確認の結果

プログラム	回帰直線
lsm.c	$y = 12.068x - 5.011$
lsm_polynomial.c	$y = 12.068x - 5.011$

性能評価

幾つかの次数の多項式から生成されたデータを用いてプログラムの性能を評価した．ここではデータの生成プログラムとして generate_data.c を作成した．これは任意の多項式関数に対して正規分布に従うノイズを乗せたものをファイルに保存するプログラムである．ソースコードは補遺 A に示した．正規分布に従う乱数は [1] を参考にした．ここではデータ数 $n = 101$ である．また性能は計算に要した CPU 時間の 5 回平均を用いて評価した．表 2 に最小二乗法による fitting の結果を示す. それぞれの fitting 結果のプロットを図から図に示した．これらの結果から作成したプログラムは正常に fitting を行っていると言える.

表 2　fitting 結果

次数	元の関数	fitting 結果
1	$10 + 5x$	$9.8 + 5.0x$
2	$10 + 3x + x^2$	$9.9 + 3.0x + 1.0x^2$
3	$10 + 5x + 0.2x^2 + 1.2x^3$	$9.8 + 5.1x + 0.2x^2 + 1.2x^3$
4	$10 + 5x + 0.4x^2 + 0.5x^3 + 2x^4$	$9.7 + 5.1x + 0.4x^2 + 0.5x^3 + 2.0x^4$
5	$10 + 5x + 0.1x^2 + 0.6x^3 + x^4 + 1.3x^5$	$9.7 + 5.1x + 0.2x^2 + 0.6x^3 + 1.0x^4 + 1.3x^5$

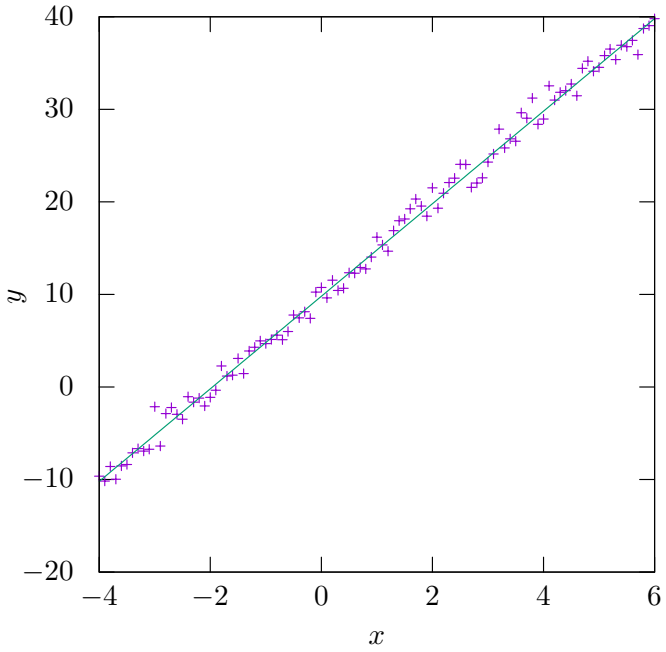

図 1　1 次の回帰曲線

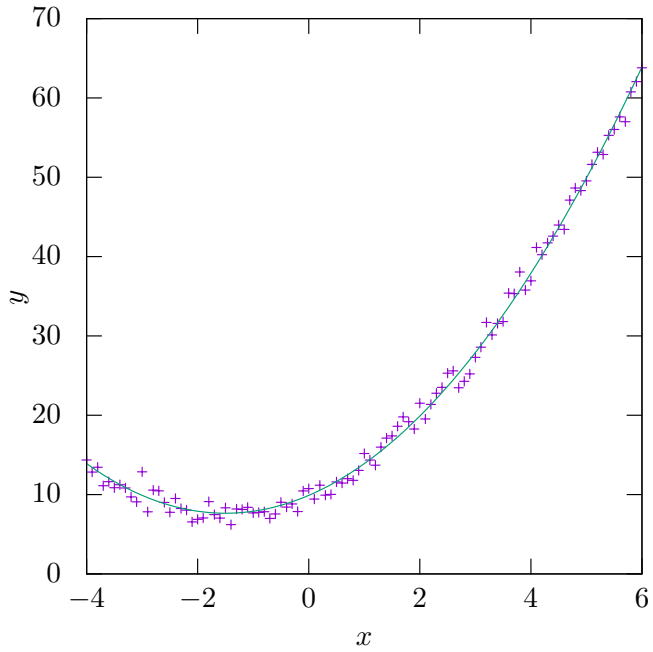

図 2　2 次の回帰曲線

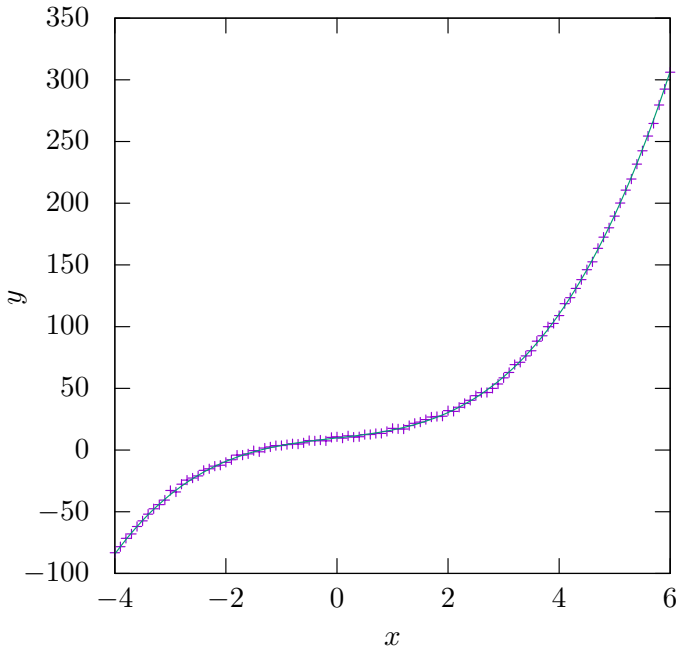

図 3　3 次の回帰曲線

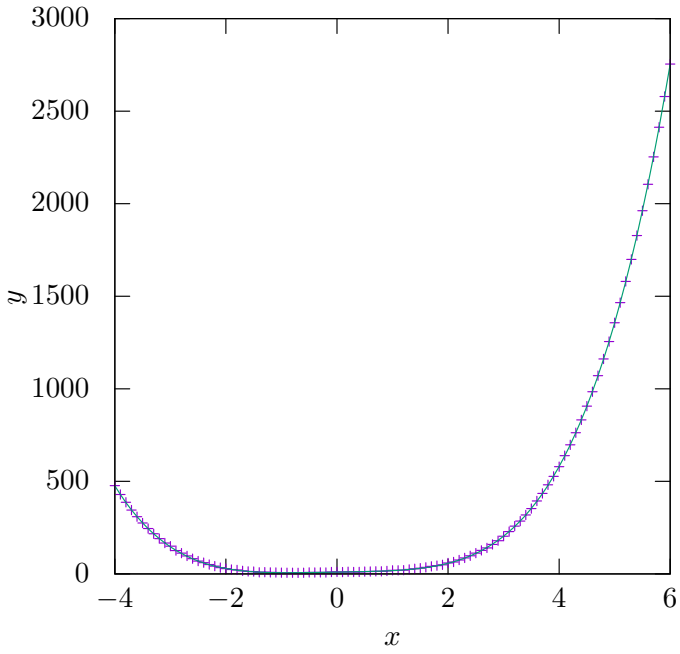

図 4　4 次の回帰曲線

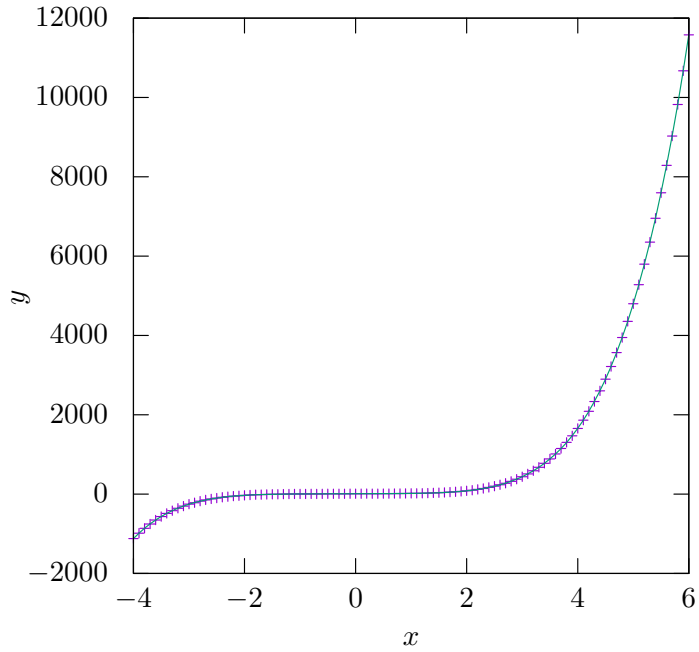

図 5　5 次の回帰曲線

図 6, 図 7 に次数と係数行列の初期化時間, 計算時間の関係を示す. 回帰曲線は gnuplot の非線形最小二乗法機能を用いて作成し, それぞれの関数は表 3 のようになった. 以上の結果から初期化時間, 計算時間ともに大凡次数の二乗に比例していることが確かめられた. また最小二乗法の計算においては係数行列の初期化が連立方程式の計算に比べて非常に多くの時間を要することがわかった. これは連立方程式の計算時間は次数のみに依存するのに対し, 係数行列の初期化時間は元データ数に線形に比例することに依ると考えられる. 係数行列の初期化を高速化する手段として, 係数行列の対称性から不要な総和計算を省略することが考えられる. これにより上手く実装を行えば $O(m^2 \times n)$ だった計算量は $O(m \times n)$ にできると考えられる.

表 3　回帰曲線

初期化時間	$y = 21.4x^{1.73} + 62.6$
計算時間	$y = 8.85 \times 10^{-2}x^{2.11} + 2.74$

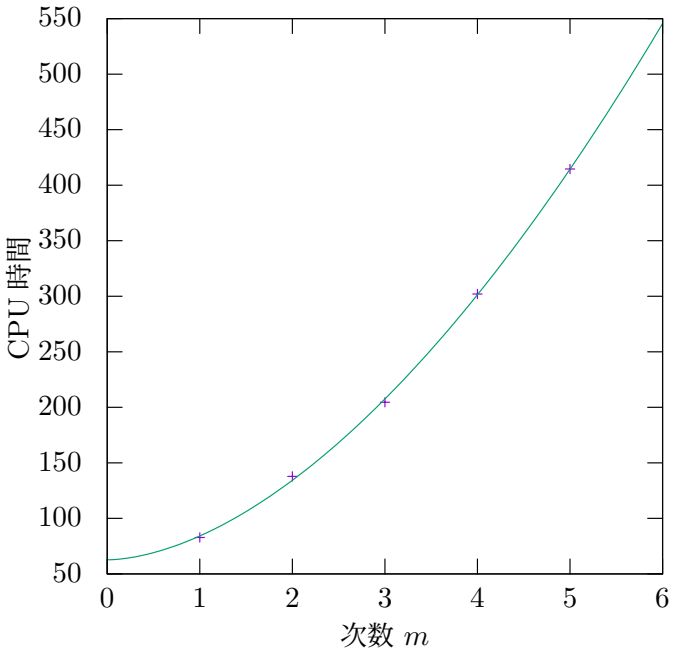

図 6　次数と初期化時間の関係

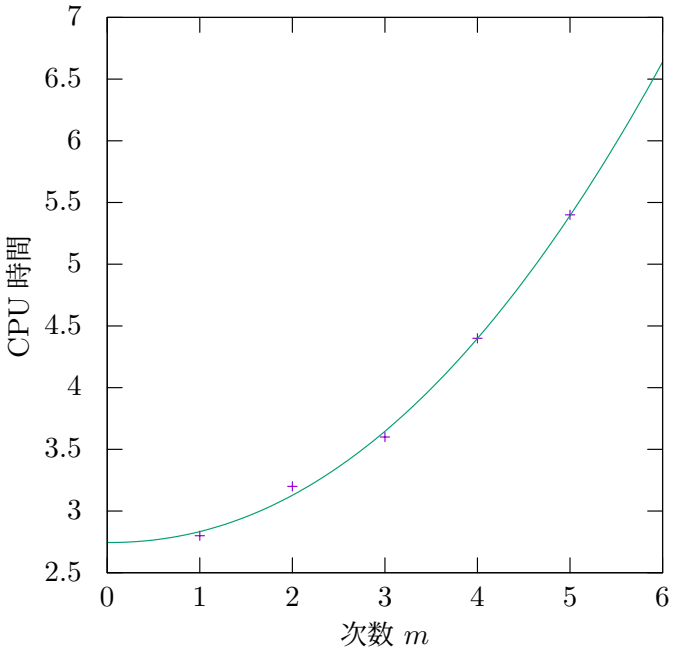

図 7　次数と計算時間の関係

参考文献

[1] 近江崇宏. C 言語による乱数生成. https://omitakahiro.github.io/random/random_variables_generation.html#Prepare_rand. (Accessed on 07/30/2021).

補遺 A　ソースコード

Listing 1　lsm_polynomial.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
#define BUF_SIZE 4096
double *array_x, *array_y, **factor_matrix, *x;
int DegreeOfPolynomial, NumOfData;

void Clear() {
    free(array_x);
    free(array_y);
    for (int i = 0; i < (DegreeOfPolynomial + 1); i++)
        free(factor_matrix[i]);
    free(factor_matrix);
    free(x);
}
void init_factor_matrix(int n)
{
    int i, j, k;
    double temp;
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            factor_matrix[i][j] = 0.0;
            for (k = 0; k < NumOfData; k++)
                factor_matrix[i][j] += pow(array_x[k], i+j);
        }
        factor_matrix[i][n] = 0.0;
        for (k = 0; k < NumOfData; k++)
            factor_matrix[i][n] += pow(array_x[k], i)*array_y[k];
    }
}
```

```
32 void disp_factor_matrix(int n, double** matrix)
33 {
34     int i, j;
35     for (i = 0; i < n; i++) {
36         for (j = 0; j < n + 1; j++) printf("%f ", matrix[i][j]);
37         printf("\n");
38     }
39 }
40
41 void disp_vector_x(int n, double* x)
42 {
43     for (int i = 0; i < n; i++) printf("a%d: %f\n", i, x[i]);
44 }
45
46 void gauss_elimination_method(int n, double** matrix, double* x_)
47 {
48     int i, j, k;
49     double pivot, temp;
50     for (i = 0; i < n - 1; i++) {
51         pivot = matrix[i][i];
52         for (j = i + 1; j < n; j++) {
53             temp = matrix[j][i] / pivot;
54             for (k = i + 1; k < n + 1; k++) matrix[j][k] -= matrix[i][k] *
                   temp;
55         }
56     }
57     for (i = n - 1; i >= 0; i--) {
58         temp = 0.0;
59         for (k = i + 1; k < n; k++) temp += matrix[i][k] * x_[k];
60         x_[i] = (matrix[i][n] - temp) / matrix[i][i];
61     }
62 }
63
64 int main(int argc, char *argv[])
65 {
66     FILE *fp;
67     int i;
68     char buf[BUF_SIZE];
69     char command;
70     if (argc != 2) {
71         printf("Usage: %s <filename>\n", argv[0]);
72         exit(EXIT_FAILURE);
```

```
73      }
74      if ((fp = fopen(argv[1], "r")) == NULL) {
75          printf("Cannot open file (%s) \n", argv[1]);
76          exit(EXIT_FAILURE);
77      }
78      /* Read the file to get the number of (x,y) pairs (N)
79      and the degree of approximate polynomial (m) */
80      fgets(buf, BUF_SIZE, fp);
81      sscanf(buf, "%d %d", &NumOfData, &DegreeOfPolynomial);
82      /* Memory allocation for the (x,y) pairs */
83      if ((array_x = malloc(sizeof(double) * NumOfData)) == NULL) {
84          printf("Cannot allocate memory \n");
85          exit(EXIT_FAILURE);
86      }
87      if ((array_y = malloc(sizeof(double) * NumOfData)) == NULL) {
88          printf("Cannot allocate memory \n");
89          exit(EXIT_FAILURE);
90      }
91      if ((factor_matrix = malloc(sizeof(double *) * (DegreeOfPolynomial + 1)))
             == NULL) {
92          printf("Cannot allocate memory \n");
93          exit(EXIT_FAILURE);
94      }
95      for (i = 0; i < (DegreeOfPolynomial + 1); i++) {
96          if ((factor_matrix[i] = malloc(sizeof(double) * (DegreeOfPolynomial +
                2))) == NULL) {
97              printf("Cannot allocate memory \n");
98              exit(EXIT_FAILURE);
99          }
100     }
101     if ((x = malloc(sizeof(double) * (DegreeOfPolynomial + 1))) == NULL) {
102         printf("Cannot allocate memory \n");
103         exit(EXIT_FAILURE);
104     }
105     /* Read the (x,y) pairs */
106     i = 0;
107     while (fgets(buf, BUF_SIZE, fp) != NULL) {
108         sscanf(buf, "%lf %lf", &array_x[i], &array_y[i]);
109         i++;
110     }
111     fclose(fp);
112
```

```
113     clock_t start_initial, end_initial;
114     clock_t start_calc, end_calc;
115     start_initial = clock();
116     init_factor_matrix(DegreeOfPolynomial + 1);
117     end_initial = clock();
118     start_calc = clock();
119     gauss_elimination_method(DegreeOfPolynomial + 1, factor_matrix, x);
120     end_calc = clock();
121     disp_vector_x(DegreeOfPolynomial + 1, x);
122     printf("initialize:%ld\ncalculation:%ld",
123     (end_initial - start_initial),
124     (end_calc - start_calc));
125 
126     Clear();
127     exit(EXIT_SUCCESS);
128 }
```

Listing 2 generate_data.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #define START -4.0
5  #define END 6.0
6  #define INTERVAL 0.1
7  #define N 5
8  
9  double Uniform(void){
10   return ((double)rand()+1.0)/((double)RAND_MAX+2.0);
11 }
12 
13 double rand_normal(double mu, double sigma){
14     double z=sqrt(-2.0*log(Uniform())) * sin( 2.0*M_PI*Uniform() );
15     return mu + sigma*z;
16 }
17 
18 double func(double x) {
19     return rand_normal(0, 1) + 10 + 5*x + pow(x, 2)*0.1 + pow(x, 3)*0.6 +
          pow(x, 4)*1 + pow(x, 5)*1.3;
20 }
21 
22 int main(int argc, char *argv[])
23 {
```

```
24      double x;
25      FILE *fp;
26
27      if (argc != 2) {
28          printf("Usage: %s <filename>\n", argv[0]);
29          exit(EXIT_FAILURE);
30      }
31      if((fp = fopen(argv[1], "w")) == NULL) {
32          printf("Cannot open file (%s) \n", argv[1]);
33          exit(EXIT_FAILURE);
34      }
35      fprintf(fp, "%d\t%d\n", (int)((END - START) / INTERVAL) + 1, N);
36      for (x=START; x < END; x += INTERVAL) {
37          fprintf(fp, "%f\t%f\n", x, func(x));
38      }
39      fclose(fp);
40  }
```