数値計算 Class-10 演習

21T2166D 渡辺大樹

2023 年 7 月 27 日

1　演習内容

Class-10 では前回行った最小二乗法を拡張し、適当なデータファイルとモデルファイルを読み取り、モデルファイルから読み取った基本関数とデータファイルから作った補間関数と、もともとのデータファイルを Python でグラフにし可視化、またモデルの誤差を数値として計算する。

今回使ったソースコードをソースコード 1,2,3,4 に示す (長くなるため pdf 末尾に記載)。ソースコード 4 は for 文などを用いて高速にグラフを取得できるようにしている。

このソースコードを 6 つあるデータセットと 26 個あるモデルのパターンで動かし、Python で作成したグラフから補間結果について考察していく。

2　演習結果-考察

以下では実際に 6 つあるデータセットと 26 個あるモデルのパターンで最小二乗法を計算し、特に誤差の少ないものについてグラフを示し、考察していく。

2.1　example2.txt

初めにデータセット example2.txt で最小二乗法を行った結果を図 1 に示す。

図 1 ではデータセット example2.txt をモデル 23($= a_1x + a_2x^2 + a_3\frac{1}{x}$) で補間したグラフになる。この図を見るとデータ点に対してかなり正確な精度で補間出来ていることが分かる。

また error の値を見てもほかのモデルが 20-0.1 程度であるのに対し 0.007 とかなり小さく収まっていることがわかる。ただモデル 10($= a_1x + a_2\frac{1}{x}$) も誤差が 0.007 程度に収まっており、モデル 23 での x^2 の係数もかなり小さいことから、場合や用途によっては x^2 の項を無視したモデル 10 を用いてもよいかもしれない。

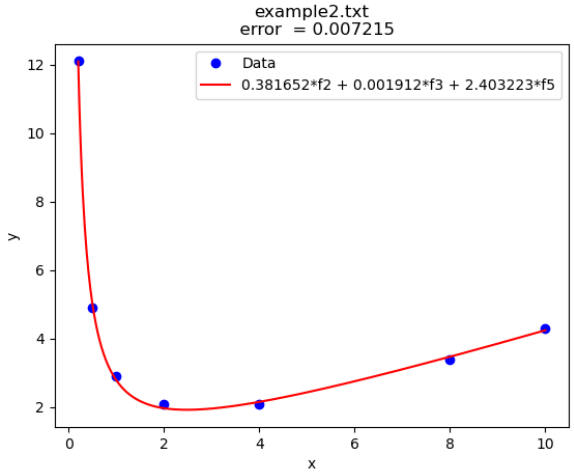

図 1　example2.txt でのデータ点と補間関数 (model=23)

2.2　nh_bb_age_length.txt

続いてデータセット nh_bb_age_length.txt で最小二乗法を行った結果を図 2 に示す。

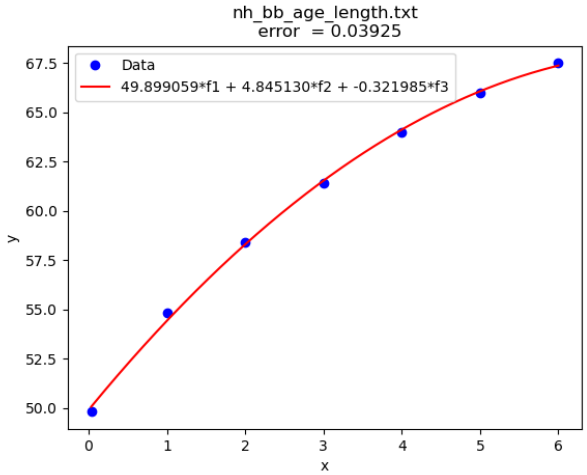

図 2　nh_bb_age_length.txt でのデータ点と補間関数 (model=18)

図 2 ではデータセット nh_bb_age_length.txt をモデル 18($= a_1 + a_2x + a_3x^2$) で補間したグラフになる。この図でもかなり正確な補間ができていることが分かる。

また error の値も小さくほかのモデルで行った際は 1000-0.1 ほどの誤差が出ており、このモデルでの無視出来るほど小さな係数も存在しないので、今回調べたモデルの中ではこのモデルでの補間が最適であると考える。

このデータセットは名前を見る限り乳児期から幼少期にかけての身長に関するデータであるた

め、この結果より二次関数で近似できることが分かる。

2.3 nh_bb_age_weigth.txt

続いてデータセット nh_bb_age_weigth.txt で最小二乗法を行った結果を図 3 に示す。

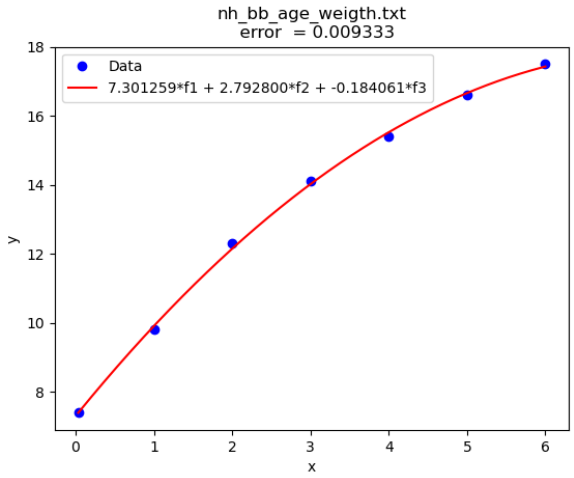

図 3　nh_bb_age_weigth.txt でのデータ点と補間関数 (model=18)

図 3 ではデータセット nh_bb_age_weigth.txt をモデル 18($= a_1 + a_2x + a_3x^2$) で補間したグラフになる。この図でもかなり正確な補間ができている。

この補間に関しても error が小さく、ほかのモデルでは 90-0.03 までの誤差が出ており、2.2 同様このモデルでの補間が最適であると考えられる。

このデータセットは 2.2 同様に乳児期から幼少期にかけての体重に関するデータであると思うので、実際にそういった統計が必要の際には二次関数での補間で都合がよくなると分かる。

2.4 nh_covid-italy.txt

続いてデータセット nh_covid-italy.txt で最小二乗法を行った結果を図 4 に示す。

図 4 はデータセット nh_covid-italy.txt をモデル 26($= a_1x^2 + a_2x^3 + a_3e^x$) で補間したグラフになる。引き続きこの図でも正確な補間ができていることが分かる。

この補間は 2.1-3 に比べると誤差の値が $3.1 \cdot 10^6$ と大きいが、このデータセットでのほかのモデルと比較するとほかのモデルは $10^9 - 10^7$ までかなり広い範囲に大きな数字で誤差が出ていることが分かるのでこれが最適なモデルであることが分かる。

このデータセットはこれも名前から予想するにイタリアでのコロナ感染者のデータになっていると思うが、よく言われている伝染病の感染者数は指数関数的に増加するという言葉通り、項に指数関数が含まれている (係数はかなり小さいが) のでかなり面白い分析になっていると思う。

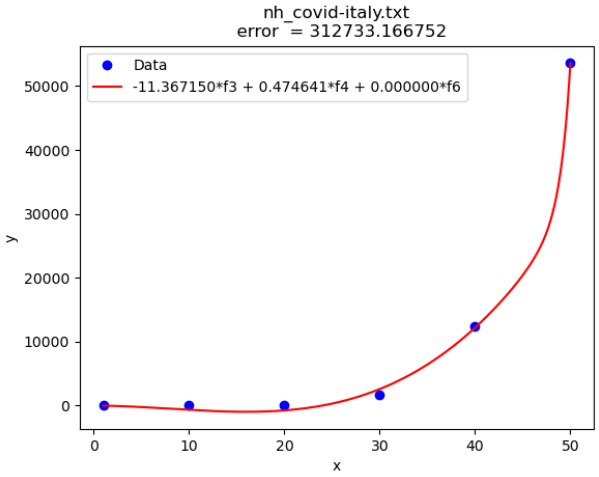

図 4　nh_covid-italy.txt でのデータ点と補間関数 (model=26)

2.5　nh_fish.txt

続いてはデータセット nh_fish.txt で最小二乗法を行った結果を図 5 に示す。

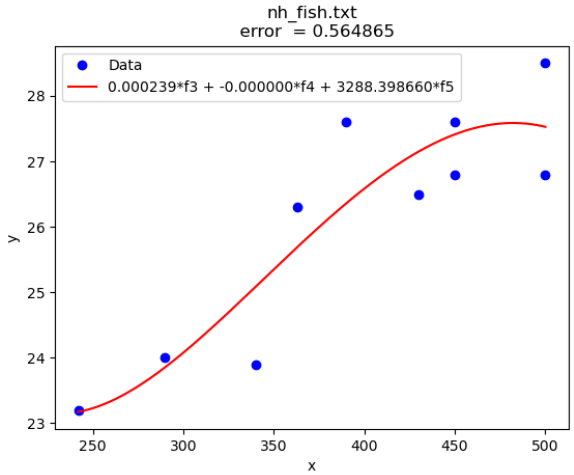

図 5　nh_fish.txt でのデータ点と補間関数 (model=25)

図 5 はデータセット nh_fish.txt をモデル 25($= a_1x^2 + a_2\frac{1}{x} + a_3e^x$) で補間したグラフになる。

このデータセットはデータ点自体がかなりバラバラであるため図を見ただけでは良い補間になっているかは分からない。error の値をほかのモデルと比較してみても 15-0.6 の付近、特に誤差が 0.6 付近になるモデルは 7 つあり、実際にこのデータがこのモデルの関係性を持つとは言い切れないような補間になっている。

2.6 nh_wine.txt

最後にデータセット nh_wine.txt で最小二乗法を行った結果を図 6 に示す。図 6 はデータセット

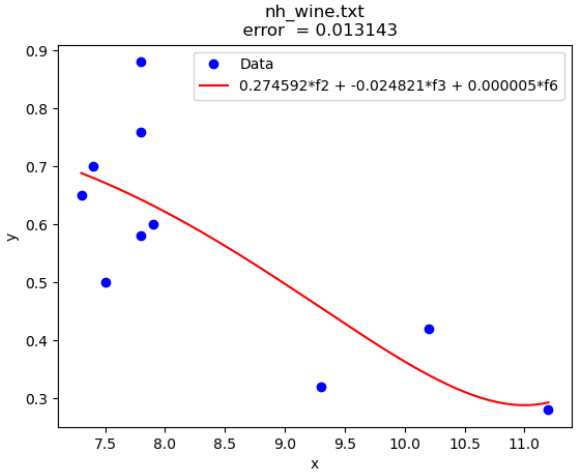

図 6　nh_wine.txt でのデータ点と補間関数 (model=24)

nh_wine.txt をモデル 24($= a_1x + a_2x^2 + a_3e^x$) で補間したグラフになる。

このデータセットもグラフを見るだけではデータ点により近いような補間になっているとは考えずらい。error の値もモデルの半分以上が 0.013 ほどであり一概にこのモデルが一番良いとは言えない結果であると考えられる。

ソースコード 1　minjijo_lusolve_extended.h

```
/* 基本関数fi の x での関数値 y を求める */
double ffv(int fi, double x)
{
    double y;
    switch (fi)
    {
    case 1:
        y = 1.0;
        break;
    case 2:
        y = x;
        break;
    case 3:
        y = pow(x, 2);
        break;
    case 4:
        y = pow(x, 3);
        break;
    case 5:
        y = 1.0 / x;
        break;
    case 6:
        y = exp(x);
        break;
    default:
        y = x;
        break;
    }
    return y;
}
/*** データの入力 ***/
int data_input(char *fname, double x[N], double y[N])
{
    int n = -1, i;
    FILE *fp;

    fp = fopen(fname, "r");
    if (fp == NULL)
    {
        printf("ファイル「%s」を開くことができません", fname);
        exit(0);
```

```
42      }
43      /* データの個数は何個ですか？(1< n < %d) n = ", N); */
44      fscanf(fp, "%d", &n);
45      if ((n <= 1) || (N <= n))
46          printf("データの個数は (1< n < %d) でなければなりません, n = %d", N, n);
47
48      /* printf("\n データ x の値は小から大の順に入力する. \n"); */
49      for (i = 1; i <= n; i++)
50      {
51          fscanf(fp, "%lf", &x[i]);
52          fscanf(fp, "%lf", &y[i]);
53      }
54
55      fclose(fp);
56      return n;
57  }
58  /*  モデルを読み取る  */
59  int read_model(FILE *fp_models, int f_id[N])
60  {
61      int k = 0;
62      char line[81], *token;
63      /* 基本関数fk(x)を１～k の番号を読み取る */
64      /* ファイルの 1 行から文字列としてモデルの仕様を読み取る  */
65      fgets(line, sizeof(line), fp_models);
66      /* 空白で区切られた文字列に分割 */
67      token = strtok(line, " ");
68      while (token != NULL)
69      {
70          k++;
71          /* 関数のID を文字列から整数に変換する */
72          f_id[k] = atoi(token);
73          /* 次の関数のID を取得します */
74          token = strtok(NULL, " ");
75      }
76      /* 基本関数の数を返す */
77      return k;
78  }
79  /* A[n][k],tA[k][n]とb[k]を設定する */
80  void set_A_tA_b(int n, int k, int f_id[N], double x[N], double y[N], double A
        [N][N], double tA[N][N], double b[N])
81  {
82      int i, j;
```

```
83      /* 関数を呼び出し、A と tA を設定する */
84      for (i = 1; i <= n; i++)
85      {
86          for (j = 1; j <= k; j++)
87          {
88              A[i][j] = ffv(f_id[j], x[i]);
89              tA[j][i] = A[i][j];
90          }
91          /* b を設定する */
92          b[i] = y[i];
93      }
94  }
95  /* tA[k][n]・A[n][k]を計算して配列tAA[k][k]に入れる */
96  void seki_tA_A(int n, int k, double tA[N][N], double A[N][N], double tAA[N][N
        ])
97  {
98      int i, j, jj;
99      double s;
100
101     for (i = 1; i <= k; i++)
102     {
103         for (j = 1; j <= k; j++)
104         {
105             s = 0.0;
106             for (jj = 1; jj <= n; jj++)
107             {
108                 s = s + tA[i][jj] * A[jj][j];
109             }
110             tAA[i][j] = s;
111         }
112     }
113 }
114 /* tA[k][n]・b[k]を計算してtAb[k]配列に入れる */
115 void seki_tA_b(int n, int k, double tA[N][N], double b[N], double tAb[N])
116 {
117     double s;
118     int i, j;
119     tAb[0] = 0;
120     for (i = 1; i <= k; i++)
121     {
122         s = 0.0;
123         for (j = 1; j <= n; j++)
```

```
124         {
125             s = s + tA[i][j] * b[j];
126         }
127         tAb[i] = s;
128     }
129 }
130 /* モデルの係数を求める  */
131 void compute_model(int n, int k, double x[N], double y[N], int f_id[N],
132                    double sol[N])
133 {
134     double A[N][N], tA[N][N], b[N], tAA[N][N], tAb[N];
135     int i;
136 
137     for (i = 0; i <= k; i++)
138         sol[i] = 0; /* 求めたa1,...ak の係数を扱う*/
139     /* A[n][k],tA[k][n]とb[k]を設定する */
140     set_A_tA_b(n, k, f_id, x, y, A, tA, b);
141     /* tA[k][n]・A[n][k]を計算して配列tAA[k][k]に入れる */
142     seki_tA_A(n, k, tA, A, tAA);
143     /* tA[k][n]・b[k]を計算してtAb[k]配列に入れる */
144     seki_tA_b(n, k, tA, b, tAb);
145     /* LU 分解法で解く */
146     lu_solve(k, tAA, tAb, sol);
147 }
148 /* モデルの誤差を求める  */
149 double model_error(int n, int k, double x[N], double y[N], double sol[N],
150                    int f_id[N])
151 {
152     int i, j;
153     double err = 0.0, yy;
154 
155     for (i = 1; i <= n; i++)
156     {
157         yy = 0.0;
158         for (j = 1; j <= k; j++)
159         {
160             yy += sol[j] * ffv(f_id[j], x[i]);
161         }
162         err += (yy - y[i]) * (yy - y[i]);
163     }
164     return err / n;
165 }
```

```
166 /*  モデルの誤差，モデルの基本関数と求めた係数を出力 */
167 void model_error_output(FILE *fp_out, int i, double err, int k, double sol[N],
        int f_id[N])
168 {
169     int j;
170     fprintf(fp_out, "%d\t%f", i, err);
171     for (j = 1; j <= k; j++)
172     {
173         fprintf(fp_out, "\t%f*f%d", sol[j], f_id[j]);
174     }
175     fprintf(fp_out, "\n");
176 }
177 /*  補間結果ファイル　: グラフを描くための準備（数表を出力）*/
178 void data_output(char *fname, int n, int k, double x[N], double y[N],
179                  double sol[N], int f_id[N])
180 {
181     double h, xx, yy;
182     int i, j;
183     FILE *fp;
184
185     fp = fopen(fname, "w");
186     if (fp == NULL)
187     {
188         printf("ファイル「%s」を開くことができません", fname);
189         exit(0);
190     }
191     h = (x[n] - x[1]) / 500.0;
192     xx = x[1];
193     for (i = 0; i <= 500; i++)
194     {
195         yy = 0.0;
196         for (j = 1; j <= k; j++)
197             yy += sol[j] * ffv(f_id[j], xx);
198         fprintf(fp, "%lf\t%lf\n", xx, yy);
199         xx = xx + h;
200     }
201     fclose(fp);
202 }
```

ソースコード 2　minjijo_lusolve_extended.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
```

```c
#include <string.h>
#include <math.h>
#include <float.h>
#define N 220
#include "my_library_v3.h"
#include "minjijo_lusolve_extended.h"
int main(void)
{
    int n; /* n 個データ点数, */
    double x[N], y[N]; /* データ */
    int k, f_id[N]; /* モデル:基本関数の数k、基本関数の id */
    double sol[N] = {0}; /*求めたのモデル係数a1,...ak*/
    double err; /* モデルの誤差*/
    int m; /*  調べるモデル数  */
    /* データファイル  */
    //char *data = "example2.txt";
    //char *data="nh_covid-italy.txt";
    // char *data="nh_bb_age_weigth.txt";
    // char *data="nh_bb_age_length.txt";
    // char *data="nh_fish.txt";
    char *data="nh_wine.txt";

    char *path =
        "C:/Program_code/NumMeth/Class10STU";

        /* ファイル名を扱う変数  */
        char fname_data[200],
         fname_models[200], fname_hokan[200],
         fname_out[200];
    /* ファイルポインタを扱う変数  */
    FILE *fp_models, *fp_out;

    /* データファイル名を設定する  */
    sprintf(fname_data, "%s/data/%s", path, data);
    /*  モデルの仕様ファイルを設定する */
    sprintf(fname_models, "%s/data/models.txt", path);
    /* 補間結果ファイルを設定する  */
    sprintf(fname_out, "%s/results/out_%s", path, data);
    /* モデルの仕様ファイルを開く*/
    fp_models = fopen(fname_models, "r");
    /* モデルの仕様ファイルを開く*/
    fp_out = fopen(fname_out, "w");
```

```
45
46      printf("このプログラムは最小 2乗法によって  \n");
47      printf("y = a1*f1(x) + a2*f2(x) +...+ ak*fk(x) \n");
48      printf("の形の曲線をあてはめるものです. \n\n");
49      /*** データの入力 ***/
50      n = data_input(fname_data, x, y);
51      printf("%s ファイルから %d 個データ点数を読み込んだ\n", data, n);
52      /* 調べるモデルの数を読み込む */
53      fscanf(fp_models, "%d", &m);
54      printf("models.txt ファイルから %d 個モデルを調べる\n", m);
55      /* models.txt ファイルで指定された各モデルを調べる  */
56      for (int i = 1; i <= m; i++)
57      {
58          /* モデルを読み込む  */
59          k = read_model(fp_models, f_id);
60          printf("m%d k %d ", i, k);
61          /* モデルの係数を求める  */
62          compute_model(n, k, x, y, f_id, sol);
63          /* モデルの誤差を求める  */
64          err = model_error(n, k, x, y, sol, f_id);
65          printf("err %lf\n", err);
66          /*  モデルの誤差，モデルの基本関数と求めた係数を出力 */
67          model_error_output(fp_out, i, err, k, sol, f_id);
68          /* 補間結果ファイルを設定する  */
69          sprintf(fname_hokan, "%s/results/m%d_est_%s", path, i, data);
70          /*  補間結果ファイル  : グラフを描くための準備（数表を出力）*/
71          data_output(fname_hokan, n, k, x, y, sol, f_id);
72      }
73      fclose(fp_models);
74      fclose(fp_out);
75      printf("モデル誤差と求めた基本関数の係数はファイル out_%s に保存されます\n",
            data);
76      printf("補間結果はファイル m#_est_%s に保存されます\n", data);
77      printf("処理の終了\n");
78  }
```

ソースコード 3　my_library_v3.h

```
1  /*L1-ノルム*/
2  double vector_norm1(double a[N+1]) {
3          int i = 0;
4          double norm = 0.0;
5          for (i = 1; i <= N; i++) {
```

```
6                 norm += fabs(a[i]);
7         }
8
9         return norm;
10 }
11
12 /*行列の入力*/
13 int input_matrix(double a[N][N]) {
14         int n, i, j;
15         char z, zz;
16         while(1) {
17                 printf("行列の次数の入力 (1 < n < %d) n = ", N-1);
18                 scanf("%d%c",&n, &zz);
19                 if ((n <= 1) || (N-1 <= n)) continue;
20                 printf("\n 行列 A の成分を入力します\n\n");
21                 for(i=1;i<=n;i++) {
22                         for(j=1; j<=n; j++) {
23                                 printf("a(%d, %d)=", i, j);
24                                 scanf("%lf%c",&a[i][j], &zz);
25                         }
26                         printf("\n");
27                 }
28                 printf("正しく入力しましたか？(y/n)");
29                 scanf("%c%c", &z, &zz);
30                 if (z == 'y') break;
31         }
32
33         return n;
34 }
35
36 /*行列の出力*/
37 void print_matrix(double a[N][N], int n) {
38         int i, j;
39         for(i=1; i<=n; i++) {
40                 for(j=1; j<=n; j++) {
41                         printf(" %10.6lf",a[i][j]);
42                 }
43                 printf("\n");
44         }
45 }
46
47
```

```
48
49
50
51
52
53 /*LU 分解*/
54 int ludecomp(int n, double a[N][N], double l[N][N], double u[N][N]) {
55         int i, j, k;
56         double p, q;
57
58     /*LU 分解とガウス消去*/
59         for(i=1; i<=n; i++) {
60                 p = a[i][i];
61                 if (fabs(p) < 1.0e-6) {
62                         printf("この行列はLU 分解出来ません. \n");
63                         return -1;
64                 }
65
66                 for(j=i; j<=n; j++) {
67                         l[j][i] = a[j][i];
68                         a[i][j] = a[i][j] / p;
69                 }
70
71                 for(k=i+1; k<=n; k++) {
72                         q = a[k][i];
73             for(j=1; j<=n+1; j++) {
74                                 a[k][j] = a[k][j] - a[i][j] * q;
75                         }
76                 }
77
78                 for(j=i; j<=n; j++) {
79                         u[i][j]= a[i][j];
80                 }
81         }
82
83         return 0;
84 }
85
86
87
88
89
```

```
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106 /*LU 分解による連立一次方程式の解*/
107 int lu_solve(int n, double a[N][N], double b[N], double x[N]) {
108         int i,j;
109         double l[N][N] = {0}, u[N][N] = {0};
110         double y[N] = {0};
111
112         /*LU 分解*/
113         int ret = ludecomp(n, a, l, u);
114
115         if (ret != 0) {
116                 return ret;
117         }
118
119         /*前進代入*/
120         for(i=1; i<= n; i++){
121                 double py = b[i];
122                 for(j=1; j < i; j++) {
123                         py -= l[i][j]*y[j];
124                 }
125                 y[i] = py/l[i][i];
126         }
127
128         /*後退代入*/
129         for (i = n; i >= 1; i--) {
130                 double px = y[i];
131                 for(int j = i+1; j <= n; j++) {
```

```
                        px -= u[i][j] * x[j];
                }
                x[i] = px/u[i][i];
        }

        return 0;
}
```

ソースコード 4　fig.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jun 22 14:03:55 2020

@author: hernan
"""
import pandas as pd
import matplotlib.pyplot as plt


#元のデータファイル名
fname_datas = ["example2.txt", "nh_covid-italy.txt", "nh_bb_age_weigth.txt",
    "nh_bb_age_length.txt", "nh_fish.txt", "nh_wine.txt"]

for fname_data in fname_datas:
    fm = "../results/out_"+fname_data #モデルの誤差，モデルの係数と基本関数
    # モデルをエラーごとに並べ替える
    column_names = [i for i in range(0, 5)]
    #df_m_sort = pd.read_csv(fm, sep="\t", header=None, names = column_names)
    #df_m_sort.sort_values(by=[1])

    #調べるモデル
    for model_number in range(1,27):

        fd = "../data/"+fname_data #データ
        fe = "../results/m"+str(model_number)+"_est_"+fname_data #補完の結果

        #出力されたモデルを取得する
        df_m = pd.read_csv(fm, sep="\t", header=None, skiprows = model_number
            -1 , nrows = 1)
        error = str(df_m.iloc[0,1])
        model = ' + '.join(list(df_m.iloc[0,2:]))
```

```
32
33
34          #元のデータを読み込む
35          df_d = pd.read_csv(fd, sep="\t", header=None, skiprows=1)
36
37          #補間されたデータを読み込む
38          df_e = pd.read_csv(fe, sep="\t", header=None)
39
40          # 元のデータと補間されたデータをプロットする
41          plt.figure()
42          plt.plot(df_d[0],df_d[1], 'ob')
43          plt.plot(df_e[0],df_e[1], '-r')
44          plt.title(fname_data + "\n error = " + str(error))
45          plt.xlabel('x')
46          plt.ylabel('y')
47          plt.legend(["Data", model])
48          plt.savefig("./"+ fname_data +"/plot_" + str(model_number) + "_" +
                fname_data + ".png")
49          plt.show()
```