数値計算の理論と実際

第 6 回 関数の離散補間

1　最小 2 乗法 (1 次関数の場合)

x, y について次のようなデータが与えられたとする。

```
x  1.0  2.0  3.0  4.0  5.0
y  2.0  2.5  2.9  3.5  4.4
```

グラフにすると図 1 の赤丸のようになる。ほぼ直線上に存在しており、$y = ax + b$ のように y は x の 1 次関数として近似されると考えられる。

データ点は完全には直線上に存在しない。全体としてずれが最小になるように a, b を求めるのが**最小 2 乗法**である。つまり n 個のデータが存在する場合、

$$E = \sum_{k=1}^{n} (y_k - ax_k - b)^2 \tag{1}$$

を最小にすればよい。その条件は $\frac{\partial E}{\partial a} = 0, \frac{\partial E}{\partial b} = 0$ である。すると

$$\begin{aligned} \frac{\partial E}{\partial a} &= -2 \sum x_k (y_k - ax_k - b) = 0 \\ \frac{\partial E}{\partial b} &= -2 \sum (y_k - ax_k - b) = 0 \end{aligned} \tag{2}$$

であり、未知変数 a, b について整理すると

$$\begin{aligned} \left(\sum x_k^2\right) a + \left(\sum x_k\right) b &= \sum x_k y_k \\ \left(\sum x_k\right) a + \left(\sum 1\right) b &= \sum y_k \end{aligned} \tag{3}$$

という 2 元連立 1 次方程式を解くことに帰着する。計算すると

$$\begin{cases} 55a + 15b = 51.7 \\ 15a + 5b = 15.3 \end{cases} \tag{4}$$

のようになる。

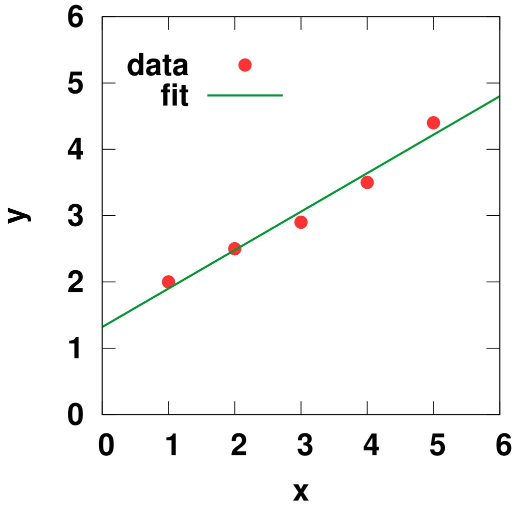

図 1:

1.1　練習問題

上記のデータをインプットし、a, b についての連立 1 次方程式を作成するプログラムを実装する。そしてこれまでの作成したガウスの消去法のプログラムと組み合わせて、最適な a, b を計算する。以下サンプルコードである。ガウスの消去法のプログラムを流用しているため、a, b, x などの変数名に注意する。サンプルコード上の $a[\quad][\quad]$ は式 (4) の係数行列、$b[\quad]$ は右辺、$x[\quad]$ は未知変数 a, b を表している。インプットファイルは 06-1.dat として moodle に置いてある。

フィッティング関数ができたらグラフ化して確かめよ。グラフ化は gnuplot を使うと簡単である。例えば gnuplot コマンドを打ち込み起動し、

```
p "06-1.dat" u 1:2
rep x+1
```

などとすると、データ点がまずプロットされ、y=x+1 が線で描画される。

```
#include<iostream>
#include<cstdio>
#include<cmath>

using namespace std;

const int n = 2;
double a[n][n], b[n], x[n];  // 連立1次方程式用の配列
const int ndata=5;  // データ数
double xk[ndata], yk[ndata];  //データ

void selectPivot( const int s){
  前回の実習を参照
}

void forward(){
```

```
  前回の実習を参照
}

void backward(){
  前回の実習を参照
}

int main(){

  for( int i=0; i<ndata; i++){
    cerr << "input xk[" << i << "]\tyk[" << i << "]" << endl;
    cin >> xk[i] >> yk[i];
    cerr << xk[i] << "\t" << yk[i] << endl;
  }

  a[0][0] = a[0][1] = b[0] = b[1] = x[0] = x[1] = 0.0;

  for( int i=0; i<ndata; i++){
    連立1次方程式の係数行列 a と b を作成する
  }

  cerr << endl << "Solve equations" << endl;
  cerr << a[0][0] << "\t" << a[0][1] << "\t" << b[0] << endl;
  cerr << a[1][0] << "\t" << a[1][1] << "\t" << b[1] << endl;

  forward();
  backward();

  cerr << "a=" << x[0] << endl;
  cerr << "b=" << x[1] << endl;

  E を計算し出力する

}
```

2 最小2乗法(高次関数)

最小2乗法は1次関数に限らず、m 次の多項式でも可能である。

$$y = a_0 + a_1 x + a_2 x^2 + \cdots + a_m x^m \tag{5}$$

とすると、

$$E = \sum_{k=1}^{n} (y_k - a_0 - a_1 x_k - a_2 x_k^2 - \cdots - a_m x_k^m)^2 \tag{6}$$

となる。1次関数の時と同様に、a_0 から a_m について $\frac{\partial E}{\partial a_k} = 0$ を満たすと E が最小になる。

式を整理すると

$$
\begin{bmatrix} S_0 & S_1 & \cdots & S_m \\ S_1 & S_2 & \cdots & S_{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ S_m & S_{m+1} & \cdots & S_{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} T_0 \\ T_1 \\ \vdots \\ T_m \end{bmatrix} \tag{7}
$$

という連立 $m+1$ 元 1 次方程式を解くことに帰着する。ただし、

$$
S_j = \sum_{k=1}^{n} x_k^j, \qquad T_j = \sum_{k=1}^{n} y_k x_k^j \tag{8}
$$

である。

2.1 練習問題

任意の次数 m を与えたときに m 次の多項式でデータ点をフィットするプログラムを作成せよ。そして以下のデータを最小二乗法でフィットし、$m = 1, 2, 3$ の場合について誤差を比較せよ (図 2)。

```
x     y
0.10   2.4824
0.20   1.9975
0.30   1.6662
0.40   1.3775
0.50   1.0933
0.60   0.7304
0.70   0.4344
0.80   0.2981
0.90  -0.0017
1.00  -0.0026
```

以下サンプルコードである。インプットファイルは 06-2.dat として moodle に置いてある。先ほどのデータとは異なり、1 行目にフィッティングの次数、2 行目にデータ数、3 行目以降にデータが記述されている。1 行目を適当に変更して比較せよ。1 次の場合は $-2.8116x+2.5539$、2 次の場合は $1.5817x^2-4.5515x+2.9019$、3 次の場合は $0.9894x^3-0.0508x^2-3.7986x+2.8170$ のようになる。

```
#include<iostream>
#include<cstdio>
#include<cmath>

using namespace std;

const int NMAX = 10;
```

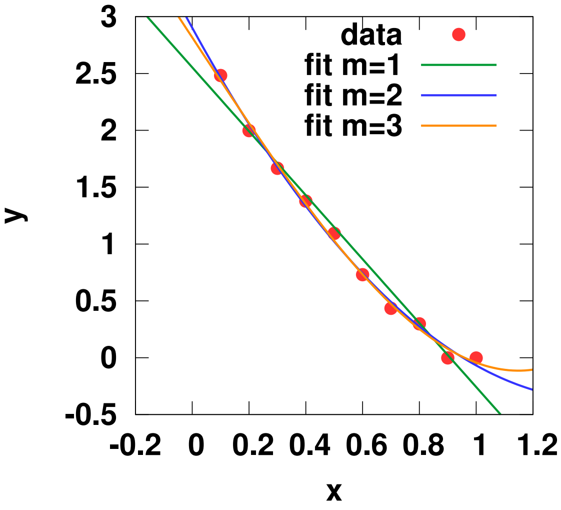

図 2:

```
int n = 0;  // 次数
double a[NMAX][NMAX], b[NMAX], x[NMAX]; // 連立1次方程式用の配列

void selectPivot( const int s){
  前回の実習を参照
}

void forward(){
  前回の実習を参照
}

void backward(){
  前回の実習を参照
}

int main(){

  int ndata= 0;
  cin >> n >> ndata;
  double xk[ndata], yk[ndata];

  for( int i=0; i<ndata; i++){
    cerr << "input xk[" << i << "]\tyk[" << i << "]" << endl;
    cin >> xk[i] >> yk[i];
    cerr << xk[i] << "\t" << yk[i] << endl;
  }

  n ++;
  for( int i=0; i<n; i++){
    for( int j=0; j<n; j++){
      a[i][j] = 0.0;
    }
    x[i] = 0.0;
    b[i] = 0.0;
```

```
  }

  for( int i=0; i<n; i++){
    連立1次方程式の係数行列 a と b を作成する
    ここはiを含め3重ループになる
  }

  cerr << endl << "Solve equations" << endl;
  for( int i=0; i<n; i++){
    for( int j=0; j<n; j++){
      cerr << a[i][j] << "\t";
    }
    cerr << b[i] << endl;
  }

  forward();
  backward();

  for( int i=0; i<n; i++){
    cerr << "a" << i << "=" << x[i] << endl;
  }

  E を計算し出力する

}
```