情報工学実験２
データ解析
(直線／曲線によるデータの当てはめ)

035743A：比嘉雅樹

実験日： 2004/11/22,29
提出日： 2004/12/06

1　実験の目的

与えられたデータに直線あるいは曲線を当てはめるアルゴリズムを C++ 言語を用いてプログラミングする事を目的とする。

2　報告事項

2.1　線形回帰当てはめのプログラム (lin-reg.cpp)

```
// Inputs
//   x        独立変数
//   y        従属変数
//   sigma    y における推定誤差
// Outputs
//   a_fit    a(1):切片, a(2):傾き
//   sig_a    パラメータ a() における推定誤差
//   yy       推定値
//   chisqr   カイ二乗関数

#include "NumMeth.h"

void lin_reg(Matrix x, Matrix y, Matrix sigma,
    Matrix &a_fit, Matrix &sig_a, Matrix &yy, double &chisqr) {

  //* Evaluate various sigma sums
   int i, nData = x.nRow();
   double sigmaTerm;
   double s = 0.0, sx = 0.0, sy = 0.0, sxy = 0.0, sxx = 0.0;
   for( i=1; i<=nData; i++ ) {
    //式 (5.10) の計算を行う
     sigmaTerm = sigma(i) * sigma(i);
     s += 1 / sigmaTerm;
     sx += x(i) / sigmaTerm;
     sy += y(i) / sigmaTerm;
     sxy += (x(i)*y(i)) / sigmaTerm;
     sxx += x(i)*x(i) / sigmaTerm;
  }
```

```
  // 切片 a_fit(1) と 傾き a_fit(2) の計算
   a_fit(1) = ((sy*sxx)-(sx*sxy)) / ((s*sxx)-(sx*sx));
   a_fit(2) = ((s*sxy)-(sy*sx)) / ((s*sxx)-(sx*sx));

  //* 切片と傾きの誤差範囲を計算
   sig_a(1) = sqrt( sxx/((s*sxx)-(sx*sx)) );
   sig_a(2) = sqrt( s/((s*sxx)-(sx*sx)) );

  //* 推定値とカイ二乗関数の計算
   chisqr = 0.0;
   for ( i=1; i<=nData; i++ ) {
     sigmaTerm = sigma(i) * sigma(i);
     chisqr += 1 / sigmaTerm  * (( a_fit(1) + a_fit(2)*x(i) - y(i) )
              * ( a_fit(1) + a_fit(2)*x(i) - y(i) ));
     yy(i) = a_fit(1) + a_fit(2) * x(i);
   }
}
```

2.1.1 実行結果

```
[MasakiPC:~/works/jikken2/rep5] higamasa% demo
Curve fit data is created using the quadratic
  y(x) = c(1) + c(2)*x + c(3)*x^2
Enter the coefficients:
c(1) = 3
c(2) = 2
c(3) = 1
Enter estimated error bar: 100
Fit parameters:
 a(1) = -469.07 +/- 28.7139
 a(2) = 53.3162 +/- 0.979992
Chi square = 253.124; N-M = 48
```

図１に、実行結果をプロットした図を示す。

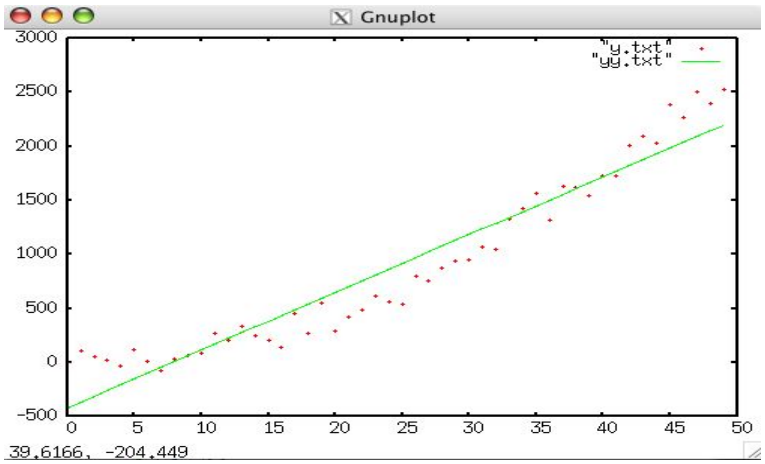

図 1: データと回帰直線

2.2 線形回帰、最小二乗法について調べて簡潔にまとめよ。

- 線形回帰
  ２変数間の変動傾向をまとめる為に用いる一つの統計処理を回帰分析といい、パラメータが線形のものを線形回帰、そうでないものを非線形回帰という。

- 最小二乗法
  予測値に基づく y=ax+b の直線上の値と、実際の値との差の２乗和が最小となるような回帰直線を求める方法で、図２のようなデータとがあるとすると、$d1^2 + d2^2 + d3^2 + d4^2$ が最小になるような直線を求める方法である。

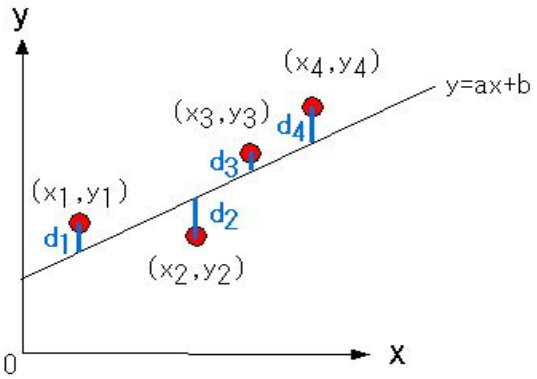

図 2: 最小二乗法の例

2.3 カイ２乗関数について調べて簡潔にまとめよ。

カイ２乗関数とは、カイ２乗値 X^2 が求められる分布関数で、X^2 は、

$$X^2 = (\text{(観測度数} - \text{期待度数)}^2 \div \text{期待度数)} \text{ の総和}$$

で求められる。

2.4 線形回帰当てはめの、多項式当てはめのプログラムを作成せよ。

- プログラム (lsf-demo2.cpp)

```
// lsfdemo - 最小二乗当てはめのプログラム
#include "NumMeth.h"

void lin_reg( Matrix x, Matrix y, Matrix sigma,
    Matrix &a_fit, Matrix &sig_a, Matrix &yy, double &chisqr );
void poly_reg(Matrix x, Matrix y, Matrix sigma,
    Matrix &a_fit, Matrix &sig_a, Matrix &yy, double &chisqr, int M2);

double randn( long& iseed );

int main() {

  int M, M2,i, j;
  //M は近似したい関数の次数
  cout << "Input M (It is Degree), M > 1 : "; cin >> M;
  if ( M < 2 ) {
    cout << "Error" << endl;
    exit(1);
  }
  Matrix c(M+1);
  //M2 は回帰直線、回帰曲線の次数
  cout << "Number of fit parameter M2(>1)："; cin >> M2;
  if ( M2 < 2 ) {
    cout << "Error" << endl;
    exit(1);
  }
  cout << "Curve fit data is created using the quadratic" << endl;
  cout << "y(x) = c(1) + c(2)*x + c(3)*x^2 +....." << endl;
```

```
cout << "Enter the coefficients:" << endl;
for(i = 1; i <= M + 1; i++){
  cout << "c(" <<  i << ") = "; cin >> c(i);
}
double alpha;
cout << "Enter estimated error bar: "; cin >> alpha;
int N = 50;        // Number of data points
long seed = 1234;     // Seed for random number generator
Matrix x(N), y(N), sigma(N);
for( i=1; i<=N; i++ ) {
  x(i) = i;                  // x = [1, 2, ..., N]
  for( j = 1; j <= M +1; j++){
    y(i) += c(j) * pow(x(i), j - 1);
  }
  y(i) += alpha * randn(seed);
  sigma(i) = alpha;          // Constant error bar
}

//* Fit the data to a straight line
Matrix a_fit(M2), sig_a(M2), yy(N);      double chisqr;
if ( M2 == 2 ) {
  lin_reg( x, y, sigma, a_fit, sig_a, yy, chisqr);
}
else {
  poly_reg( x, y, sigma, a_fit, sig_a, yy, chisqr, M2);
}
//* Print out the fit parameters, including their error bars.
cout << "Fit parameters:" << endl;
for( i=1; i<=M2; i++ ) {
    cout << " a(" << i << ") = " << a_fit(i) << " +/- " << sig_a(i) << endl;
}
cout << "Chi square = " << chisqr << "; N-M = " << N-M << endl;

//* Print out the plotting variables: x, y, sigma, yy
ofstream xOut("x.txt"), yOut("y.txt"),
    sigmaOut("sigma.txt"), yyOut("yy.txt");
for( i=1; i<=N; i++ ) {
  xOut << x(i) << endl;
  yOut << y(i) << endl;
```

```
    sigmaOut << sigma(i) << endl;
    yyOut << yy(i) << endl;
  }
}
```

- プログラム (poly-reg.cpp)

```
// Inputs
//   x 独立変数
//   y 従属変数
//   M 多項式の次数
// Outputs
//   a_fit   a(1):切片, a(2):傾き
//   sig_a パラメータ a() における推定誤差
//   yy 推定値
//   chisqr カイ二乗関数

#include "NumMeth.h"

void poly_reg(Matrix x, Matrix y, Matrix sigma,
              Matrix &a_fit, Matrix &sig_a, Matrix &yy, double &chisqr, int M2) {

  double inv(Matrix A, Matrix& Ainv); //逆行列の為の関数

  // Evaluate various sigma sums
  int i, j, k, nData = x.nRow();
  double sigmaTerm;
  double sum;
  // A：計画行列。   AT：A の転置行列。   ATA：AT*A。   C：ATA の逆行
列。   CAT：C*AT
  Matrix A(nData, M2), AT(M2, nData), ATA(M2, M2), C(M2, M2), CAT(M2, nData);
  Matrix b(nData);
  for( i=1; i<=nData; i++ ) {
    b(i) = y(i) / sigma(i);
    for( j = 1; j <= M2; j++){
      A( i, j) = pow(x(i), j-1) / sigma(i);
      AT( j, i) = pow(x(i), j-1) / sigma(i);
    }
  }
```

```
  for( j = 1; j <= M2; j++){
    for( i = 1; i <= M2; i++){
      sum = 0.0;
      for(k = 1; k <= nData; k++){
        sum += AT(j, k) * A(k, i);
      }
      ATA(j, i) = sum;
    }
  }

  inv(ATA, C); //ATA の逆行列 C を生成
  for(j = 1; j <= M2; j++){
    for( i = 1; i <= nData; i++){
      sum = 0.0;
      for( k = 1; k <= M2; k++){
        sum += C(j, k) * AT(k, i);
      }
      CAT(j, i) = sum;
    }
  }

  for( j = 1; j <= M2; j++){
    sum = 0.0;
    for(i = 1; i <= nData; i++){
      sum += CAT(j, i) * b(i);
    }
    a_fit(j) = sum;
    sig_a(j) = sqrt( C(j, j) );
  }

  chisqr = 0.0;
  for(i = 1; i <= nData; i++){
    for( j = 1; j <= M2; j++){
      yy(i) += a_fit(j) * pow(x(i), j-1);
    }
    sigmaTerm = sigma(i) * sigma(i);
    chisqr += ( pow( yy(i) - y(i), 2) ) / sigmaTerm;
  }
}
```

2.5 octave もしくは gnuplot を用いて結果をプロットし、それぞれの当てはめプログラムに関して考察せよ。なお３種類のデータを発生させ、それぞれグラフ化する事。

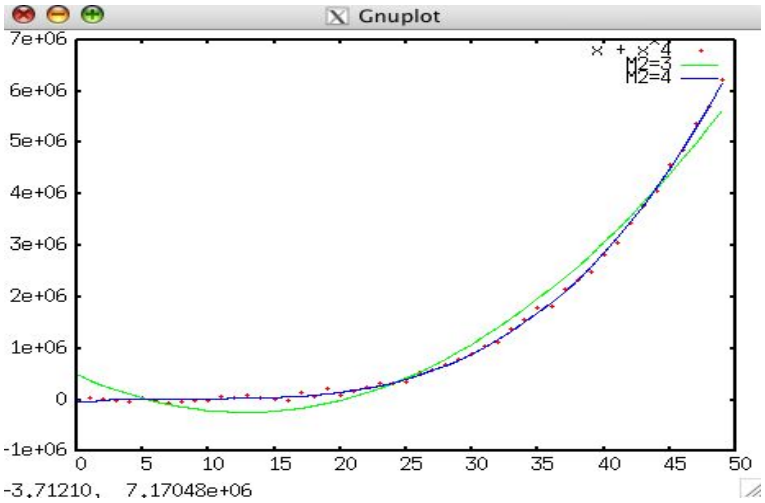

図 3: $f(x) = x + x^4$ の回帰曲線（3 次、4 次）

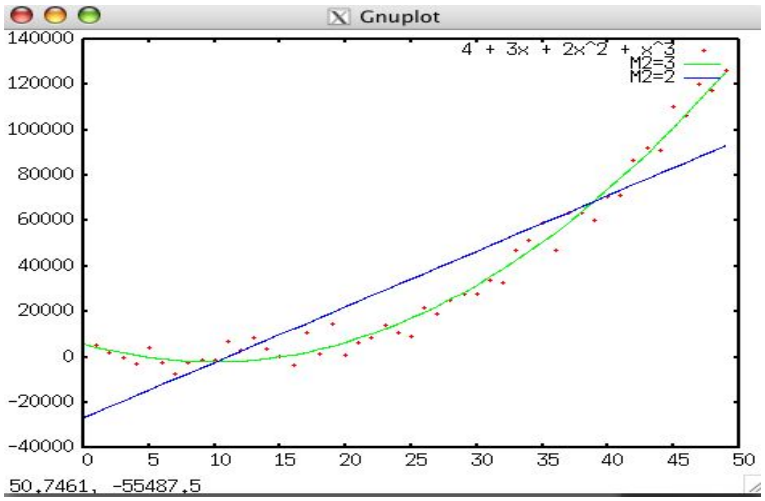

図 4: $f(x) = 4 + 3x + 2x^2 + x^3$ の回帰曲線（3 次）と回帰直線（誤差 5000）

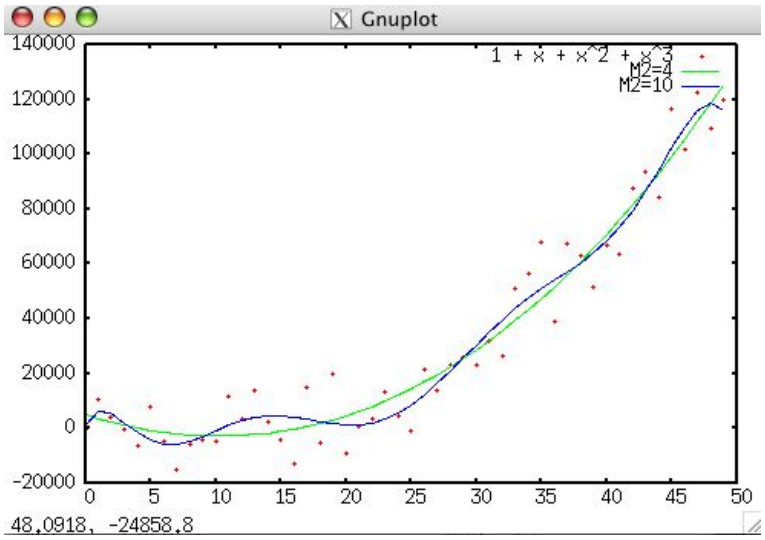

図 5: $f(x) = 1 + x + x^2 + x^3$ の回帰曲線（10 次）と回帰直線（誤差 10000）

2.6　それぞれのプログラムの説明、考察

- 線形回帰当てはめのプログラム
  線形回帰当てはめプログラム lin-reg.cpp では、講義の際に配布された資料をもとに、式（5.4)、(5.5)、(5.10)〜(5.13) の計算を行っている。
- 多項式当てはめプログラム
  多項式当てはめプログラム poly-reg.cpp では、資料の式を参考に、(5.21)〜(5.30) の計算を行っている。また、変数 M では表示するデータの次数を受け取り、M2 では近似する式の次数を受け取っている。

プロットした結果を見てみると、近似する式の次数が大きいほどよりデータを近似していることが分かる。しかし、データが 1 次式で表せる時などは、近似する式の次数は少ない方がよい。よって、データをより近似する為の式の次数は M の値によって考える方がよい。

参考文献

[1] 確率・統計　著：柴田文明

[2] http://szksrv.isc.chubu.ac.jp/lms/lms1.html　　”最小二乗法について”

[3] http://www.rerf.or.jp/nihongo/glossary/regression.htm　”回帰分析”

[4] http://kogolab.jp/elearn/hamburger/chap3/sec2.html　”カイ 2 乗値とカイ 2 乗分布”