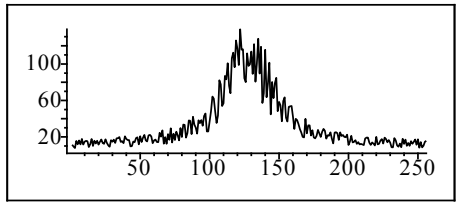
非線形最小2乗法

ー数値計算(10/11/26)ー

関西学院大学理工学部 西谷滋人

非線形最乗2乗法の原理

前回の授業では，データに近似的にフィットする最小二乗法を紹介した．今回は，フィット式が多項式のような線形関係にない関数の最小二乗法を紹介する．図のようなデータにフィットする場合を考えよう．



直観的な解説

このデータにあてはめるのはローレンツ関数，

$$F(x;\boldsymbol{a}) = a_1 + \frac{a_2}{a_3 + (x - a_4)^2}$$

である．この関数の特徴は，今まで見てきた関数と違いパラメータが線形関係になっていない．誤差関数は，いままでと同様に

$$\chi^2(\boldsymbol{a}) = \sum_i^N d_i^2 = \sum_i^N \left(F(x_i;\boldsymbol{a}) - y_i\right)^2$$

で，$\boldsymbol{a} = \{a_0, a_1, \dots\}$をパラメータとして変えた時に最小となる値を求める点もかわらない．しかし，線形の最小二乗法のように微分しても一元の方程式にならず，連立方程式を単に解くだけでは求まらない．

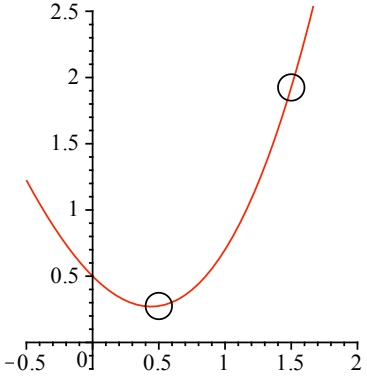
そこで図のような2次関数の最小値を求める場合を考える．最小値の点$\boldsymbol{a_0}$のまわりで，Taylor展開すると

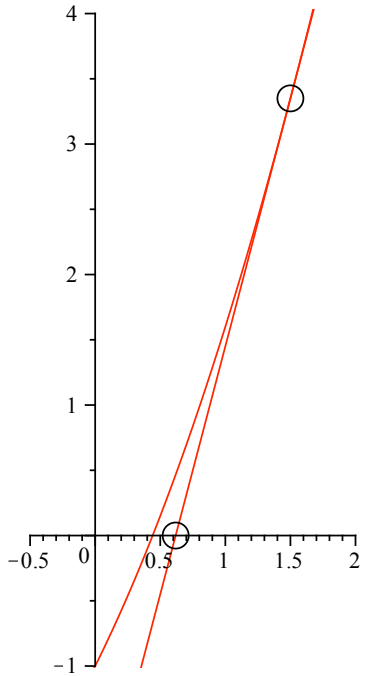
$$\chi^2(\boldsymbol{a}) = \chi^2(\boldsymbol{a_0}) - \boldsymbol{d}(\boldsymbol{a} - \boldsymbol{a_0}) + \frac{1}{2}\boldsymbol{D}(\boldsymbol{a} - \boldsymbol{a_0})^2$$

であるから，最小の点$\boldsymbol{a_0}$は

$$\boldsymbol{a_0} = \boldsymbol{a} + \boldsymbol{D}^{-1} \times (-\boldsymbol{d})$$

と予測される．図を参照して上の式を導け．またその意味を考察せよ．





現実には高次項の影響で計算通りにはいかず，単に最小値の近似値を求めるだけである．これは，$\chi^2(\boldsymbol{a})$の微分関数の解をNewton法で求める操作に対応する．つまり，この操作を何度も繰り返せばいずれ解がある精度で求まるはず．

具体的な手順

パラメータの初期値を$(a_0+\Delta a, b_0+\Delta b, c_0+\Delta c, d_0+\Delta d)$とする．このとき関数$f$を真値$(a_0, b_0, c_0, d_0)$のまわりでテイラー展開し，高次項を無視すると

$$\Delta f = f(a_0+\Delta a_1, b_0+\Delta b_1, c_0+\Delta c_1, d_0+\Delta d_1) - f(a_0, b_0, c_0, d_0)$$
$$= \left(\frac{\partial}{\partial a} f\right)_0 \Delta a_1 + \left(\frac{\partial}{\partial b} f\right)_0 \Delta b_1 + \left(\frac{\partial}{\partial c} f\right)_0 \Delta c_1 + \left(\frac{\partial}{\partial d} f\right)_0 \Delta d_1 \quad (1.2.1)$$

となる．

課題でつくったデータは$t=1$から$t=256$までの時刻に対応したデータ点$f_1, f_2, \cdots f_{256}$とする．各測定値とモデル関数から予想される値との差$\Delta f_1, \Delta f_2, \cdots, \Delta f_{256}$

$$\begin{bmatrix} \Delta f_1 \\ \Delta f_2 \\ \vdots \\ \Delta f_{256} \end{bmatrix} = J \begin{bmatrix} \Delta a_1 \\ \Delta b_1 \\ \Delta c_1 \\ \Delta d_1 \end{bmatrix} \quad (1.2.2)$$

となる．ここでJはヤコビ行列と呼ばれる行列で，4行256列

$$J = \begin{bmatrix} \left(\frac{\partial}{\partial a} f\right)_1 & \left(\frac{\partial}{\partial b} f\right)_1 & \left(\frac{\partial}{\partial c} f\right)_1 & \left(\frac{\partial}{\partial d} f\right)_1 \\ \vdots & \vdots & \vdots & \vdots \\ \left(\frac{\partial}{\partial a} f\right)_{256} & \left(\frac{\partial}{\partial b} f\right)_{256} & \left(\frac{\partial}{\partial c} f\right)_{256} & \left(\frac{\partial}{\partial d} f\right)_{256} \end{bmatrix} \quad (1.2.3)$$

である．このような矩形行列の逆行列は転置行列J^Tを用いて，

$$J^{-1} = (J^T J)^{-1} J^T \quad (1.2.4)$$

と表わされる．したがって，真値からのずれは

$$\begin{bmatrix} \Delta a_2 \\ \Delta b_2 \\ \Delta c_2 \\ \Delta d_2 \end{bmatrix} = (J^T J)^{-1} J^T \begin{bmatrix} \Delta f_1 \\ \Delta f_2 \\ \vdots \\ \Delta f_{256} \end{bmatrix} \quad (1.2.5)$$

理想的には$(\Delta a_2, \Delta b_2, \Delta c_2, \Delta d_2)$は$(\Delta a, \Delta b, \Delta c, \Delta d)$に一致するはずだが，測定誤差と高次項のために一致しない．初期値に比べ，より真値に近づくだけ．そこで，新たに得られたパラメータの組を新たな初期値に用いて，より良いパラメータに近付けていくという操作を繰り返す．新たに得られたパラメータと前のパラメータとの差がある誤差以下になったところで計算を打ち切り，フィッティングの終了となる．

Mapleによる解法の指針

線形代数計算のためにサブパッケージとしてLinearAlgebraを呼びだしておく．

```
> restart;
  with(plots):
  with(LinearAlgebra):
```

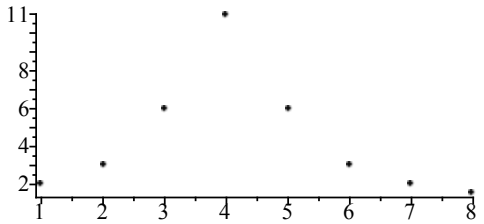
データを読み込む．

```
> ndata:=8:
  f1:=t->subs({a1=1,a2=10,a3=1,a4=4},a1+a2/(a3+(t-a4)^2) );
  T:=[seq(f1(i),i=1..ndata)]:
```

$$f1 := t \to subs\left(\{a1=1, a2=10, a3=1, a4=4\}, a1 + \frac{a2}{a3+(t-a4)^2}\right) \quad (2.3.1)$$

データの表示

```
> listplot(T,connect=false);
  l1:=listplot(T,connect=false):
```



ローレンツ型の関数を仮定し，関数として定義．

```
> f:=t->a1+a2/(a3+(t-a4)^2);
  nparam:=4:
```

$$f := t \to a1 + \frac{a2}{a3+(t-a4)^2} \quad (2.3.2)$$

ヤコビアンの中の微分を新たな関数として定義．

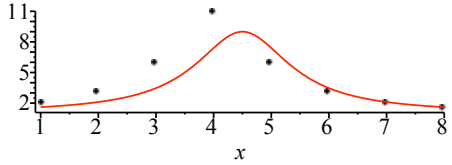
```
> for i from 1 to nparam do
  dfda||i:=unapply(diff(f(x),a||i),x);
  end do;
```

$$dfda1 := x \to 1$$
$$dfda2 := x \to \frac{1}{a3+(x-a4)^2}$$
$$dfda3 := x \to -\frac{a2}{(a3+(x-a4)^2)^2}$$
$$dfda4 := x \to -\frac{a2\,(-2x+2\,a4)}{(a3+(x-a4)^2)^2} \quad (2.3.3)$$

初期値を仮定して，データとともに関数を表示．

```
> g1:=Vector([1,8,1,4.5]):
  guess1:={}:
  for i from 1 to nparam do
   guess1:={op(guess1),a||i=g1[i]};
  end do:
  guess1;
  p1:=plot(subs(guess1,f(x)),x=1..ndata):
  display(l1,p1);
```

$$\{a1=1, a2=8, a3=1, a4=4.5\}$$




```
> interface(displayprecision=3);
  df:=Vector([seq(subs(guess1,T[i]-f(i)),i=1..ndata)]);
```

3

$$df := \begin{bmatrix} 0.396 \\ 0.897 \\ 2.538 \\ 3.600 \\ -1.400 \\ -0.462 \\ -0.103 \\ -0.016 \end{bmatrix}$$

```
> Jac:=Matrix(ndata,nparam):
  for i from 1 to ndata do
  for j from 1 to nparam do
    Jac[i,j]:=evalf(subs(guess1,dfda||j(i)));
  end do:
  end do:
```

```
> Jac;
```

$$\begin{bmatrix} 1.000 & 0.075 & -0.046 & -0.319 \\ 1.000 & 0.138 & -0.152 & -0.761 \\ 1.000 & 0.308 & -0.757 & -2.272 \\ 1.000 & 0.800 & -5.120 & -5.120 \\ 1.000 & 0.800 & -5.120 & 5.120 \\ 1.000 & 0.308 & -0.757 & 2.272 \\ 1.000 & 0.138 & -0.152 & 0.761 \\ 1.000 & 0.075 & -0.046 & 0.319 \end{bmatrix}$$

(2.3.4)

```
> tJac:=(MatrixInverse(Transpose(Jac).Jac)).Transpose(Jac);
```

$$tJac := \begin{bmatrix} 0.565 & 0.249 & -0.354 & 0.040 & 0.040 & -0.354 & 0.249 & 0.565 \\ -2.954 & -0.506 & 4.012 & -0.552 & -0.552 & 4.012 & -0.506 & -2.954 \\ -0.352 & -0.029 & 0.557 & -0.176 & -0.176 & 0.557 & -0.029 & -0.352 \\ -0.005 & -0.012 & -0.035 & -0.080 & 0.080 & 0.035 & 0.012 & 0.005 \end{bmatrix}$$

```
> g2:=tJac.df;
  g1:=g1+g2;
```

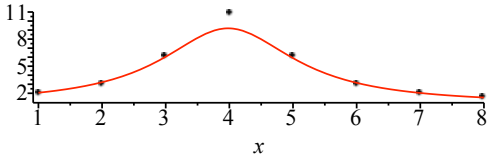
$$g2 := \begin{bmatrix} -0.235 \\ 5.592 \\ 0.613 \\ -0.520 \end{bmatrix}$$

$$g1 := \begin{bmatrix} 0.765 \\ 13.592 \\ 1.613 \\ 3.980 \end{bmatrix}$$

これをまたもとの近似値(guess)に入れ直して表示させると以下のようになる．カーブがデータに近づいているのが確認できよう．この操作をずれが十分小さくなるまで繰り返す．

```
> guess1:={}:
  for i from 1 to nparam do
   guess1:={op(guess1),a||i=g1[i]};
  end do:
  guess1;
  p1:=plot(subs(guess1,f(x)),x=1..ndata):
  display(l1,p1);
```

{$a1 = 0.765107054282076260$, $a2 = 13.5919251223407489$, $a3 = 1.61274649042098117$, $a4 = 3.98049166112393404$}



非線形最小二乗法に関するメモ

このGauss-Newton法と呼ばれる非線形最小二乗法は線形問題から拡張した方法として論理的に簡明であり，広く使われている．しかし，収束性は高くなく，むしろ発散しやすいので注意が必要．2次の項を無視するのでなく，うまく見積もる方法を用いたのがLevenberg-Marquardt 法である．明快な解説がNumerical Recipes in C(C 言語による数値計算のレシピ）WilliamH.Press 他著，技術評論社1993にある．

課題

1 一山ピークへのフィット

以下の256個のデータ

```
> ndata:=256;
  f1:=t->subs({a1=10,a2=40000,a3=380,a4=128},a1+a2/(a3+(t-
  a4)^2) );
```

```
T:=[seq(f1(i)*(0.6+0.8*evalf(rand()/10^12)),i=1..ndata)]:
> f:=t->a1+a2/(a3+(t-a4)^2);
```
で近似したときのパラメータa1,a2,a3,a4を求めよ．ただし，パラメータの初期値は，ある程度近い値にしないと収束しない．

Mapleによる解法の指針

線形代数計算のためにサブパッケージとしてLinearAlgebraを呼びだしておく．

```
> with(LinearAlgebra):
```

データを読み込む．

```
> datapoint:=[seq([i,T[i]],i=1..256)]:
```

ローレンツ型の関数を仮定し，関数として定義．

```
> f:=a+b/(c+(x-d)^2):
  f1:=unapply(f,x);
```

$$f1 := x \to a + \frac{b}{c + (x - d)^2} \qquad (3.1.1.1)$$

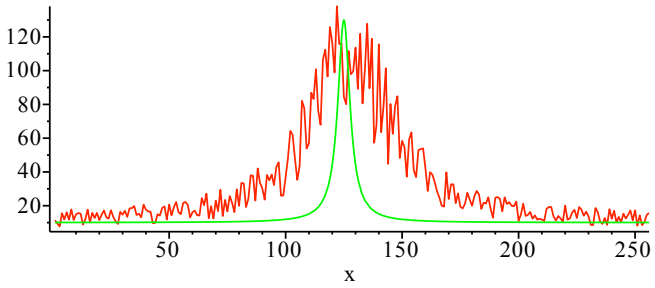
ヤコビアンの中の微分を新たな関数として定義．

```
> dfda:=unapply(diff(f,a),x):
  dfdb:=unapply(diff(f,b),x):
  dfdc:=unapply(diff(f,c),x):
  dfdd:=unapply(diff(f,d),x):
```

初期値を仮定して，データとともに関数を表示．

```
> guess1:={a=10,b=1200,c=10,d=125};
  plot([datapoint,subs(guess1,f1(x))],x=1..256);
```

$$guess1 := \{a = 10, b = 1200, d = 125, c = 10\}$$



解法のヒントと手順

1 (1.2.2)式のΔf_iを求めよ．T[i]-f1(i)を1..imaxまで求め，Vectorに入れる．
2 (1.2.3)式のヤコビ行列を求めよ．
3 (1.2.4)式にしたがってヤコビ行列の逆行列を求めよ．また，(1.2.5)式にしたがって新たなパラメータの組を求めよ．
4 求めたパラメータを用いたモデル関数と，データをプロットしてみよ．前回より近づいているのがわかるだろう．
5 上の操作を適当に繰り返し，パラメータを収束させよ．その値とプロットを示せ．

2 二山ピークのフィット

以下のように作成したデータ

```
> ndata:=256;
  f1:=t->subs({a=10,b=40000,c=380,d=128},a+b/(c+(t-d)^2) );
  f2:=t->subs({a=10,b=40000,c=380,e=90},a+b/(c+(t-e)^2) );
  T:=[seq((f1(i)+f2(i))*(0.6+0.2*evalf(rand()/10^12)),i=1..
  ndata)]:
> f:=t->a1+a2/(a3+(t-a4)^2)+a2/(a3+(t-a5)^2);
```

で近似したときのパラメータを求めよ．

解答例

```
> restart;
  with(plots):
  with(LinearAlgebra):
```

Warning, the name changecoords has been redefined

```
> f1:=t->subs({a=10,b=40000,c=380,d=128},a+b/(c+(t-d)^2) )
  ;
  f2:=t->subs({a=10,b=40000,c=380,e=90},a+b/(c+(t-e)^2) );
  T:=[seq((f1(i)+f2(i))*(0.6+0.2*evalf(rand()/10^12)),i=1.
  .256)]:
```

$$f1 := t \to subs\left(\{a = 10, b = 40000, d = 128, c = 380\}, a + \frac{b}{c + (t - d)^2}\right)$$

$$f2 := t \to subs\left(\{a = 10, b = 40000, c = 380, e = 90\}, a + \frac{b}{c + (t - e)^2}\right)$$

```
> writedata("/Users/bob/Desktop/data1",T);
> l1:=listplot(T):
> f:=t->a1+a2/(a3+(t-a4)^2)+a2/(a3+(t-a5)^2);
  nparam:=5;
```

$$f := t \to a1 + \frac{a2}{a3 + (t - a4)^2} + \frac{a2}{a3 + (t - a5)^2}$$

$$nparam := 5$$

```
> for i from 1 to nparam do
  dfda||i:=unapply(diff(f(x),a||i),x);
  end do;
```

$$dfda1 := x \to 1$$

$$dfda2 := x \to \frac{1}{a3 + (x - a4)^2} + \frac{1}{a3 + (x - a5)^2}$$

$$dfda3 := x \to -\frac{a2}{\left(a3 + (x - a4)^2\right)^2} - \frac{a2}{\left(a3 + (x - a5)^2\right)^2}$$

$$dfda4 := x \to -\frac{a2\,(-2x + 2\,a4)}{\left(a3 + (x - a4)^2\right)^2}$$

$$dfda5 := x \to -\frac{a2\,(-2x + 2\,a5)}{\left(a3 + (x - a5)^2\right)^2}$$

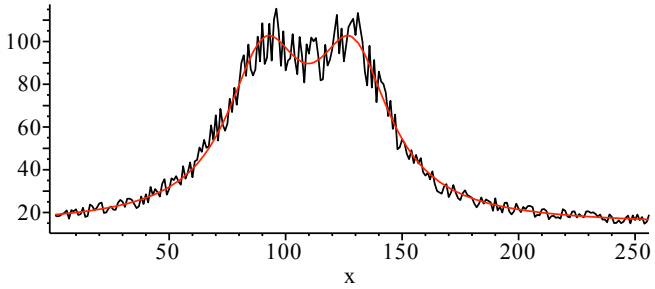
```
> g1:=Vector([10,1200,10,125,90]);
```

$$g1 := \begin{bmatrix} 10 \\ 1200 \\ 10 \\ 125 \\ 90 \end{bmatrix}$$

```
> guess1:={};
  for i from 1 to nparam do
  guess1:={op(guess1),a||i=g1[i]};
  end do:
```

$$guess1 := \{\}$$

```
> p1:=plot(subs(guess1,f(x)),x=1..256):
  display(l1,p1);
```




```
> df:=Vector([seq(subs(guess1,T[i]-f(i)),i=1..256)]):
> Jac:=Matrix(1..256,1..nparam,sparse):
> for i from 1 to 256 do
  for j from 1 to nparam do
    Jac[i,j]:=evalf(subs(guess1,dfda||j(i)));
  end do:
  end do:
> tJac:=(MatrixInverse(Transpose(Jac).Jac)).Transpose(Jac)
  :
> g2:=tJac.df;
  g1:=g1+g2;
```

$$g2 := \begin{bmatrix} -0.390553882992161205 \\ 1584.55290636967129 \\ 24.9577909601538366 \\ -0.0472041829705451138 \\ -0.00719532042503852940 \end{bmatrix}$$

$$g1 := \begin{bmatrix} 13.6348019182603064 \\ 29567.3667677707381 \\ 410.545681677467769 \\ 128.512734548828887 \\ 90.9223109918718678 \end{bmatrix}$$