

【最小二乗法 Least Squares Method】

データ y_n (データ数 N), 近似関数 $f(x) = \sum_{k=0}^m c_k x^k$ とする.

$$\mathbf{A}_{i,j} = \sum_{n=0}^{N-1} x_n^{i+j}, \quad \mathbf{B}_i = \sum_{n=0}^{N-1} x_n^i y_n \text{ とするとき,}$$

$$\begin{bmatrix} \mathbf{A}_{i,j} \end{bmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{bmatrix} \mathbf{B}_i \end{bmatrix}$$

■ 係数ベクトル $\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{bmatrix} \mathbf{A}_{i,j} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{B}_i \end{bmatrix}$

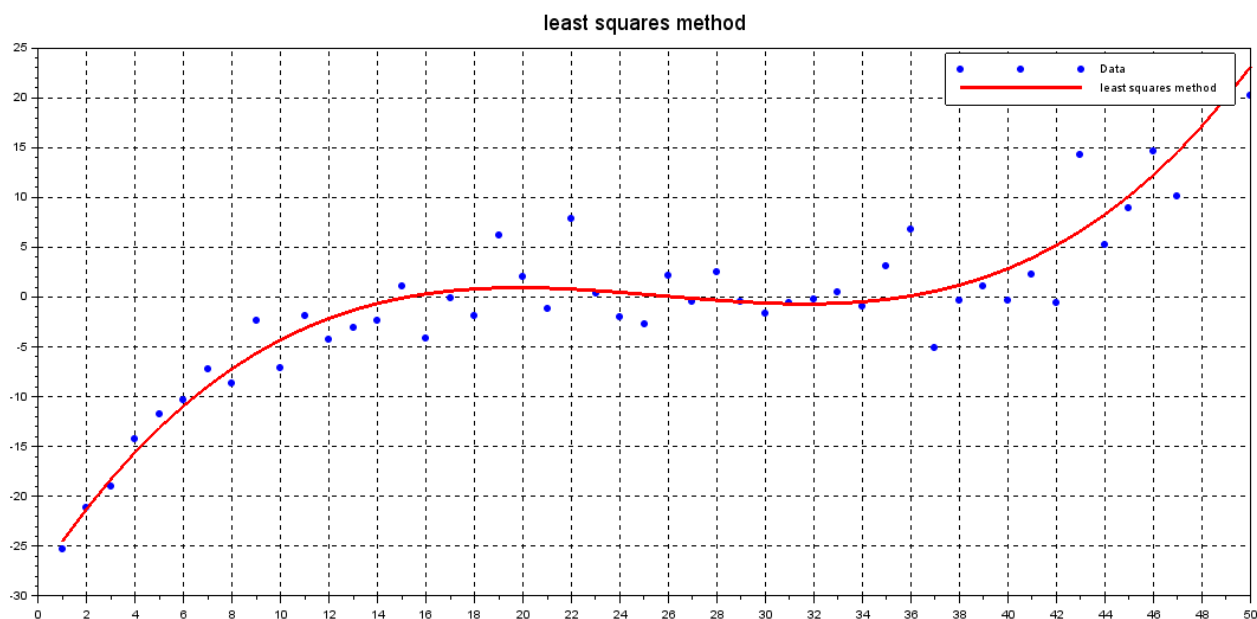


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//  最小二乗法
//  Least Squares Method
//
//                               M.Tsutsui
////////////////////////////////////

clear;

funcprot(0)
function [res]=lsm_m(order,d_size,x_a); //引数 order:次数,d_size:データ数,x_a:基底

[size_l,size_r]=size(x_a);

A=[];

```

```

B=[];
for i=1:1:order+1;
    for j=1:1:order+1;
        hat=(i-1)+(j-1);
        loop_a=[];
        for n=1:1:size_r;
            loop_a=[loop_a,x(n)^hat];
        end
        A(i,j)=sum(loop_a);//Aij
    end
end

for i_b=1:1:order+1;
    hat_b=i_b-1;
    loop_b=[];
    for k=1:1:size_r;
        loop_b=[loop_b,(x(k)^hat_b)*y(k)];
    end
    B(i_b)=sum(loop_b);//Bi
end

coe=inv(A)*B;//係数 C

[c_size_l,c_size_r]=size(coe);

res=[];
for h=1:1:order+1;
    res=res+coe(h)*x_a.(h-1);//近似関数
end
endfunction

//----main-----//
order=3;//次数
d_size=50;//データ数
x_a=linspace(0,6,d_size);//x軸
x=x_a;//基底
y=(x_a-3).^3-x_a+3+9*rand(x_a)-8*rand(x_a);//ターゲット

plot(y,'. ');
plot(lsm_m(order,d_size,x_a),'r');
a=gce();
c=a.children;
c.thickness = 3;
xgrid();
hl=legend(['Data';'least_squares_method']);
title('least_squares_method','fontsize',4);

```

Source Code 2: Python

```

#-----
# module Name:least squares method
# Author:m.tsutsui
#-----

#_Library_Import-----
from numpy import*
import math, numpy as np
import matplotlib.pyplot as plt
#_Library_Import-----

def lsm_m(order,d_size,x_a):
    size_r=size(x_a,0)

```

```

A=np.zeros([order+1,order+1])
B=np.zeros([order+1,1])

for i in np.arange(0,order+1,1):
    for j in np.arange(0,order+1,1):
        loop_a=[]
        for n in np.arange(0,size_r,1):
            loop_a.append(x[n]**(i+j))
        A[i][j]=np.sum(loop_a) #Aij

for i_b in np.arange(0,order+1,1):
    loop_b=[]
    for k in np.arange(0,size_r,1):
        loop_b.append(x[k]**(i_b)*y[k])
    B[i_b]=np.sum(loop_b) #Bi

coe=array(matrix(A).I*B)

res=0
for h in np.arange(0,order+1,1):
    res+=coe[h]*(np.power(x_a,h))

return res

if __name__ == '__main__':
    order=3
    d_size=50
    x_a=linspace(0,6,d_size)
    x=x_a
    y=((x_a-3)**3-x_a+3+9*np.random.rand(1,d_size)-8*np.random.rand(1,d_size)).T

#_plot_command-----
plt.figure(facecolor='w')#Backgroundcolor_white
plt.plot(y,'o')
plt.plot(lsm_m(order,d_size,x_a).T,'r',linewidth=3)
plt.grid()
plt.legend(('Data','LSM'))
plt.title('least_squares_method')
plt.show()
#_plot_command-----

```