# D8 最小二乗適合

さんしゃさんよー

36021341末冨光輝

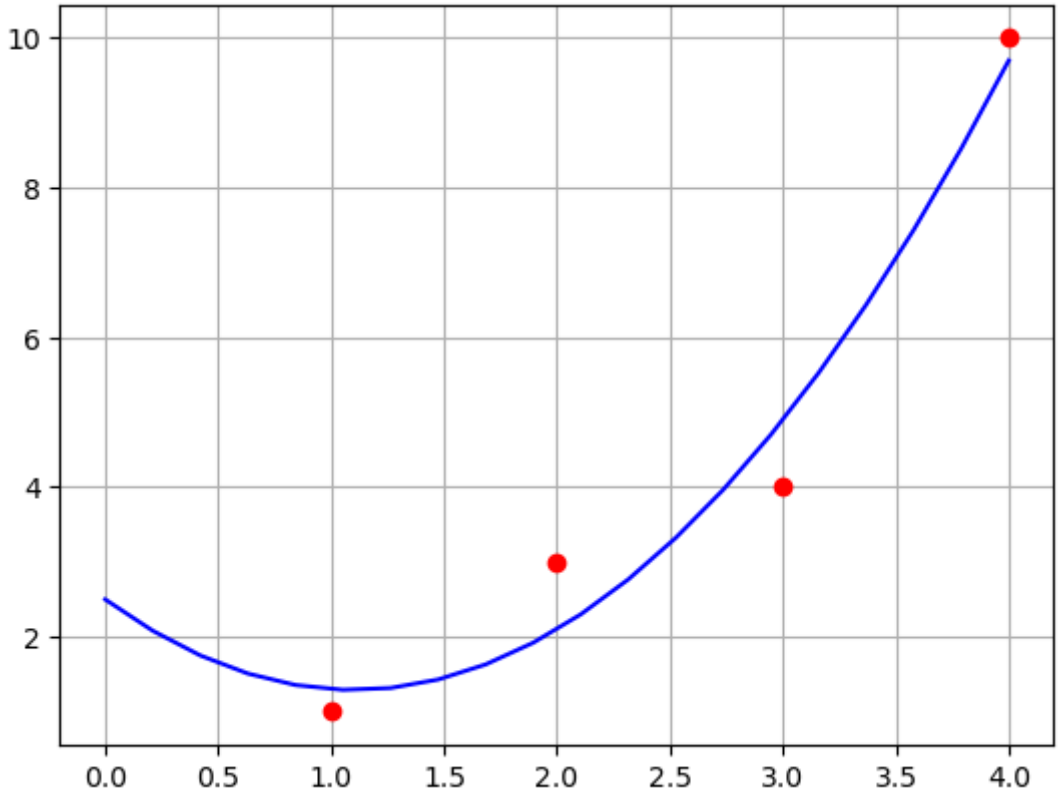
36021364古谷颯大

36021363道一世

[2]の場…

```
#最小二乗法で、データに曲線をフィット（近似）させたもの。
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
#関数
def f(x, a0, a1, a2):
    return a0 + a1*x + a2*x**2
#データ点を表したコード
xdata = np.array([1,2,3,4])
ydata = np.array([1,3,4,10])
plt.plot(xdata,ydata, 'o', color='r')

#データ点を最小二乗法で近似を表したコード
params, cov = curve_fit(f, xdata, ydata)
print(params)

#色と範囲
x =np.linspace(0,4,20)
y = f(x,params[0],params[1],params[2])
plt.plot(x,y, color='b')

plt.grid()
plt.show()
```

```
[ 2.5 -2.2  1. ]
```




[23]の場…

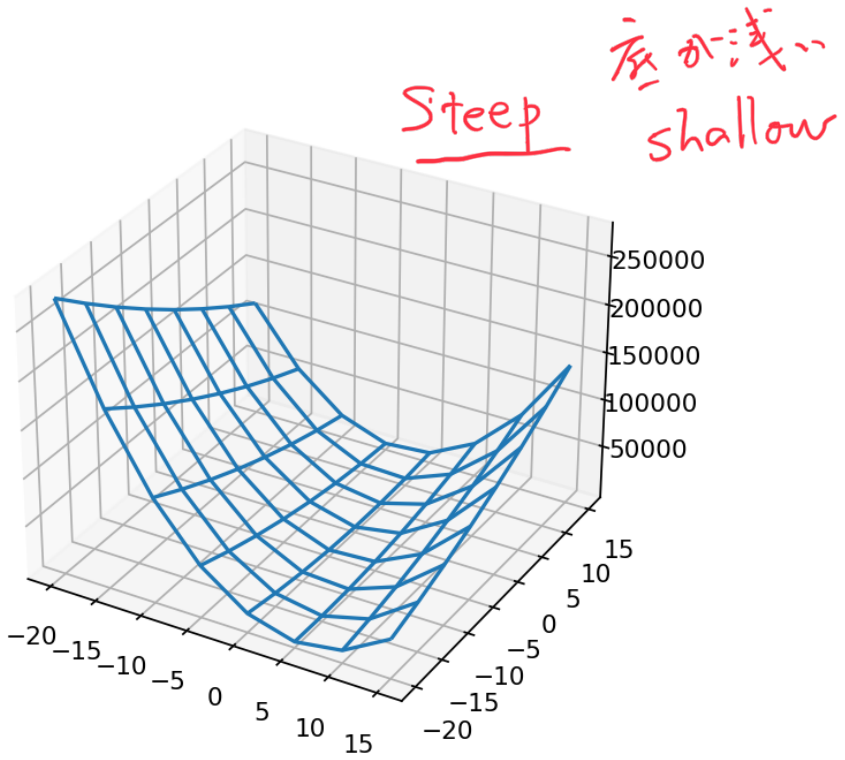
```
#χ2の極小値
%matplotlib notebook
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import numpy as np
from sympy import *

a0, a1, a2 = symbols('a0, a1, a2')
def func(x):
    return a0+a1*x+a2*x**2

def z_surf(xx,yy):
    sum = 0
    for i in range(0,4):
        tmp = xx[i] - yy[i]
        sum = sum + tmp*tmp
    return sum

x1 = np.array([1,2,3,4])
y1 = np.array([1,3,4,10])
eq = z_surf(func(x1),y1)
print(eq)
print(expand(eq))
```

```
(a0 + a1 + a2 - 1)**2 + (a0 + 2*a1 + 4*a2 - 3)**2 + (a0 + 3*a1 + 9*a2 - 4)**2 + (a0
+ 4*a1 + 16*a2 - 10)**2
4*a0**2 + 20*a0*a1 + 60*a0*a2 - 36*a0 + 30*a1**2 + 200*a1*a2 - 118*a1 + 354*a2**2 -
418*a2 + 126
```

[40]の場…

```
%matplotlib notebook
def f(a0, a1):
    return 4*a0**2 + 20*a0*a1 + 60*a0*a2 - 36*a0 + 30*a1**2 + 200*a1*a2 - 118*a1

a0 = np.arange(-20, 20, 5)
a1 = np.arange(-20, 20, 5)

A0, A1,  = np.meshgrid(a0, a1)
Z1 = f(A0, A1)
```

```
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.plot_wireframe(A0, A1, Z1)

plt.show()
```




[3]の場…

```
#正規方程式(Normal Equations)を用いた方法
import numpy as np
from pprint import pprint
import scipy.linalg as linalg

xdata=np.array([1,2,3,4])
ydata=np.array([1,3,4,10])

def ff(x,i):
    return x**i

Av = np.zeros([4,3])
for i in range(0,3):
    for j in range(0,4):
        Av[j][i]=ff(xdata[j],i)

pprint(Av)

Ai = linalg.inv(np.dot(np.transpose(Av),Av))
b = np.dot(np.transpose(Av),ydata)
np.dot(Ai,b)
```

```
array([[ 1.,  1.,  1.],
       [ 1.,  2.,  4.],
       [ 1.,  3.,  9.],
       [ 1.,  4., 16.]])
```

アウト[3]:

```
array([ 2.5, -2.2,  1. ])
```

[4]の場…

```
#特異値分解(Singular Value Decomposition)を用いた方法
import numpy as np
from pprint import pprint
import scipy.linalg as linalg

xdata=np.array([1,2,3,4])
ydata=np.array([1,3,4,10])

def f(x,a0,a1,a2):
    return a0+a1*x+a0*x**2
#def ff(x,i):
    #return x**i

Av = np.zeros([4,3])
for i in range(0,3):
    for j in range(0,4):
        Av[j][i]=ff(xdata[j],i)
m,n = Av.shape
pprint(Av)

U, s, Vs = linalg.svd(Av)
pprint(s)

S = linalg.diagsvd(s,m,n)
pprint(S)
iS = np.zeros([3,4])
for i in range(0,3):
    iS[i][i] = 1.0/s[i]
print(iS)
left = np.dot(np.transpose(Vs),iS)
right= np.dot(np.transpose(U),ydata)


np.dot(left,right)
#print(right)
```

アウト[4]:

```
array([[ 1.,  1.,  1.],
       [ 1.,  2.,  4.],
       [ 1.,  3.,  9.],
       [ 1.,  4., 16.]])
array([19.62136402,  1.71206987,  0.26625288])
array([[19.62136402,  0.        ,  0.        ],
       [ 0.        ,  1.71206987,  0.        ],
       [ 0.        ,  0.        ,  0.26625288],
       [ 0.        ,  0.        ,  0.        ]])
[[0.05096486 0.         0.         0.        ]
 [0.         0.58408831 0.         0.        ]
 [0.         0.         3.75582793 0.        ]]
array([ 2.5, -2.2,  1. ])
```

[45]の場…

```
#デザイン行列 Aで、Ax=bとみなした場合のxについて解いたもの
import numpy as np
from pprint import pprint
import scipy.linalg as linalg

xdata=np.array([1,2,3,4])
ydata=np.array([1,3,4,10])

def ff(x,i):
    return x**i

Av = np.zeros([4,3])
for i in range(0,3):
```

```
    for j in range(0,4):
        Av[j][i]=ff(xdata[j],i)
print(Av)

c, resid, rank, sigma = linalg.lstsq(Av, ydata)
print(c,resid,rank,sigma)
```

```
[[ 1.  1.  1.]
 [ 1.  2.  4.]
 [ 1.  3.  9.]
 [ 1.  4. 16.]]
[ 2.5 -2.2  1. ] 1.7999999999999958 3 [19.62136402  1.71206987  0.26625288]
```

[44]の場…

```
#SVD
Ai = linalg.inv(np.dot(np.transpose(Av),Av))
b = np.dot(np.transpose(Av), ydata)
np.dot(Ai,b)
```

アウト[44]:

```
array([ 2.5, -2.2,  1. ])
```

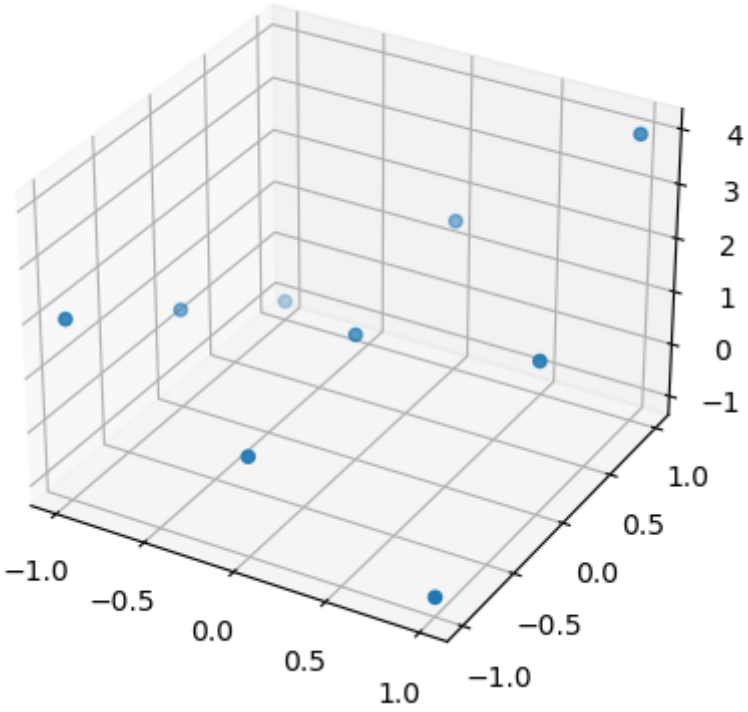
[21]の場…

```
#2次元の最小二乗フィットしたもの
import numpy as np
z = np.array([2, 0.5, -1, 0.5, 1, 1.5, -1, 1.5, 4])
x = [-1, -1, -1, 0, 0, 0, 1, 1, 1]
y = [-1, 0, 1, -1, 0, 1, -1, 0, 1]


for i in range(-2,3):
    for j in range(-2,3):
        x.append(i*0.0005)
        y.append(j*0.0005)

x = x[:9]  # xの要素数を9に変更
y = y[:9]  # yの要素数を9に変更


print(x)
print(y)
```

```
[-1, -1, -1, 0, 0, 0, 1, 1, 1]
[-1, 0, 1, -1, 0, 1, -1, 0, 1]
```

[6]の場…

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D



fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(np.array(x),np.array(y),z)

plt.show()
```




[7]の場…

```
from pprint import pprint
import scipy.linalg as linalg

n = z.size
n_j = 6
bb=np.zeros([n])
A=np.zeros([n,n_j])
for i in range(0,n):
    A[i,0]=1
    A[i,1]=x[i]
    A[i,2]=y[i]
    A[i,3]=x[i]*y[i]
    A[i,4]=x[i]**2
    A[i,5]=y[i]**2
    bb[i]=z[i]

c, resid, rank, sigma = linalg.lstsq(A, bb)
pprint(c)

Ai = linalg.inv(np.dot(np.transpose(A),A))
b = np.dot(np.transpose(A),bb)
np.dot(Ai,b)
```

```
array([ 1.00000000e+00,  5.00000000e-01,  5.00000000e-01,  2.00000000e+00,
        1.88737914e-15, -1.44328993e-15])
```

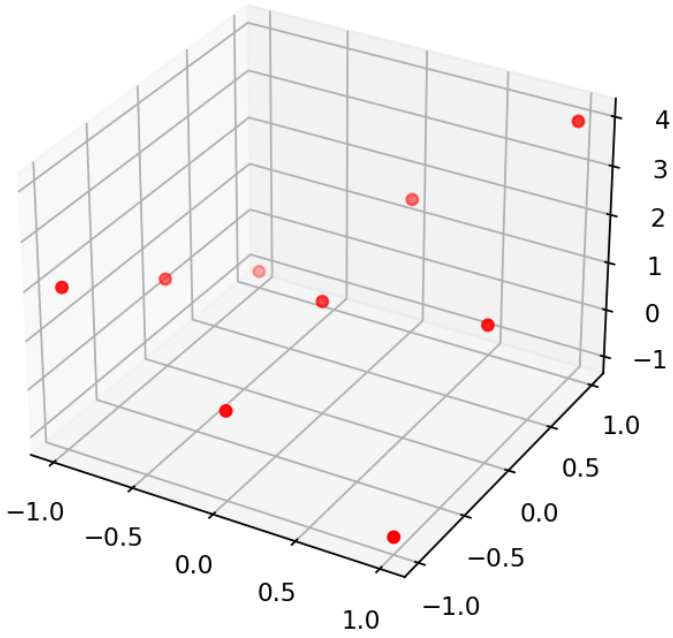
アウト[7]:

```
array([1. , 0.5, 0.5, 2. , 0. , 0. ])
```

[17]の場…

```
%matplotlib notebook
#for jupyter notebook
# %matplotlib inline # for vscode

#関数
def z_surf(xx,yy):
    val = c[0] + c[1]*xx + c[2]*yy
    val += c[3]*xx*yy + c[4]*xx**2
    val += c[5]*yy**2
    return val
#データ点
```

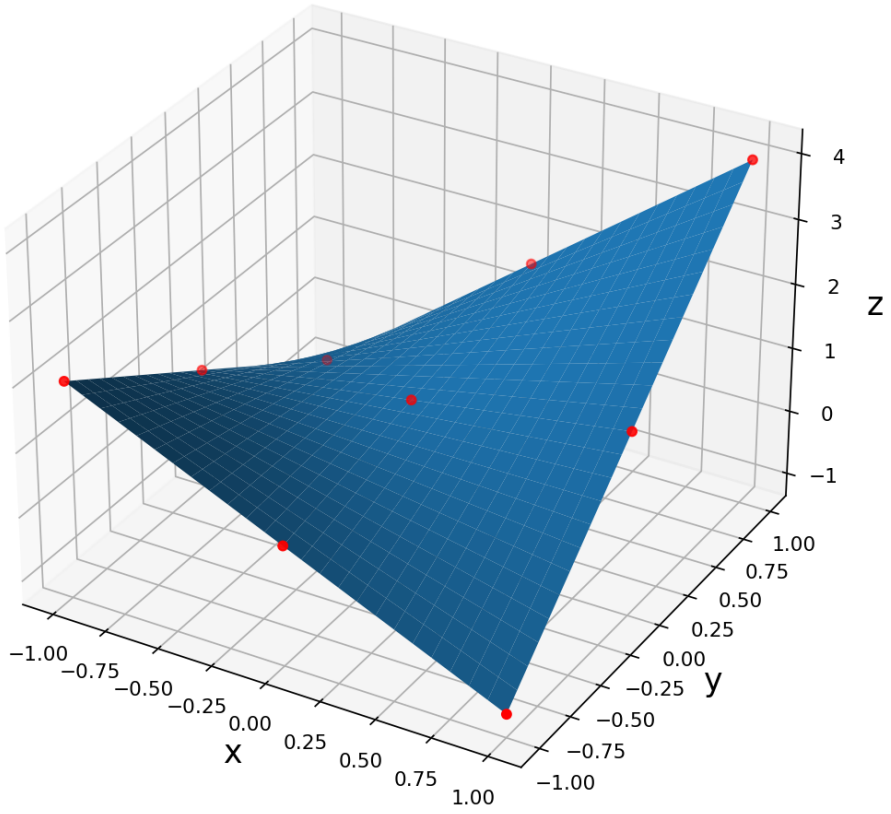
```
x1 = np.arange(-0.001, 0.00125, 0.00025)
y1 = np.arange(-0.001, 0.00125, 0.00025)
X, Y = np.meshgrid(x1, y1)
Z1 = z_surf(X,Y)

#表示
fig = plt.figure()
ax = fig.add_subplot(111, projection="3d")
ax.scatter(np.array(x), np.array(y), z, color='r')

#鞍点の表示
ax.plot_surface(X, Y, Z1)
#こっちだと鞍点が表示されなかった。

plt.show()
```




[18]の場…

```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt

#関数
def z_surf(xx,yy):
    val = c[0] + c[1]*xx + c[2]*yy
    val += c[3]*xx*yy + c[4]*xx**2
    val += c[5]*yy**2
    return val

# データ点を表したコード
x = np.array([-1, -1, -1, 0, 0, 0, 1, 1, 1])
y = np.array([-1, 0, 1, -1, 0, 1, -1, 0, 1])
z = np.array([2, 0.5, -1, 0.5, 1, 1.5, -1, 1.5, 4])


# グラフの範囲を表したコード
x_range = np.linspace(-1, 1, 25)
```

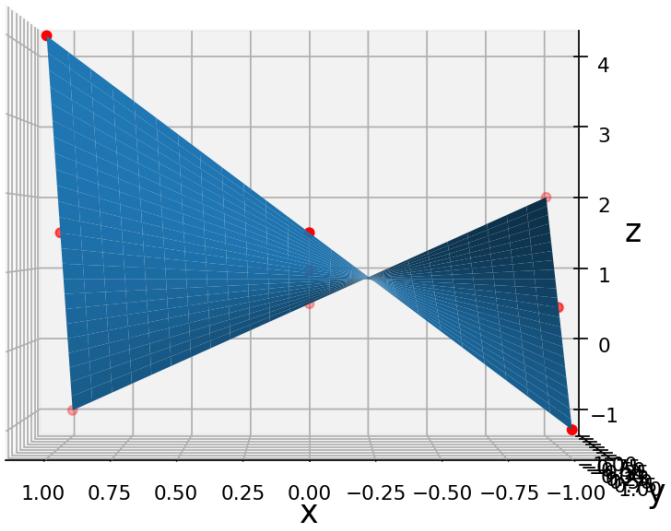
```
y_range = np.linspace(-1, 1, 25)
X, Y = np.meshgrid(x_range, y_range)
Z = z_surf(X,Y)


# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)
plt.show()
```




[19]の場…

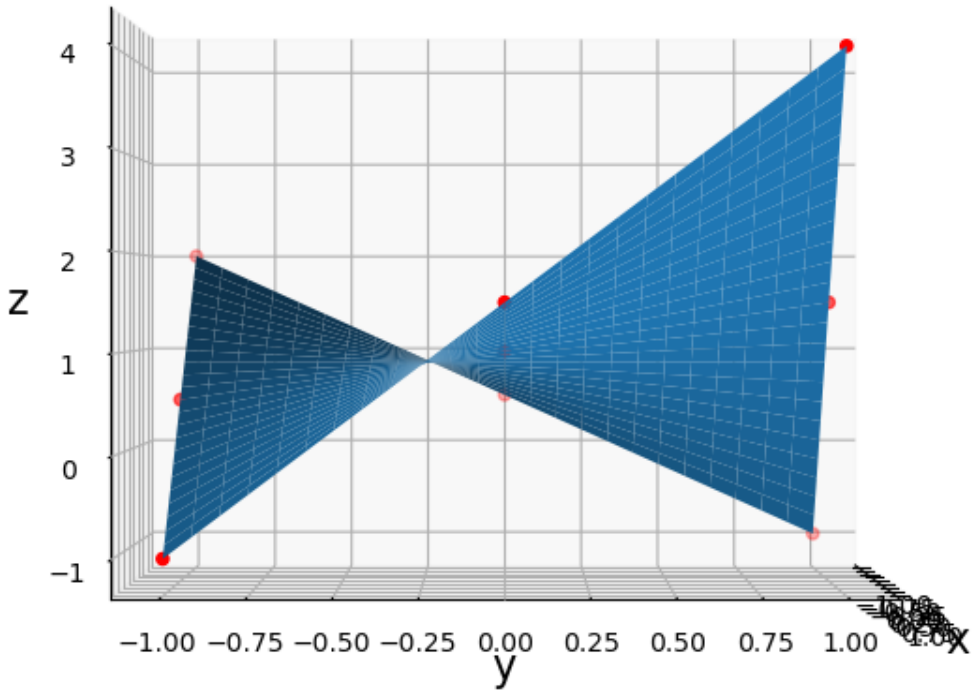
```
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt

def z_surf(xx,yy):
    val = c[0] + c[1]*xx + c[2]*yy
    val += c[3]*xx*yy + c[4]*xx**2
    val += c[5]*yy**2
    return val

# データ点を表したコード
x = np.array([-1, -1, -1, 0, 0, 0, 1, 1, 1])
```

```
y = np.array([-1, 0, 1, -1, 0, 1, -1, 0, 1])
z = np.array([2, 0.5, -1, 0.5, 1, 1.5, -1, 1.5, 4])


# グラフの範囲を表したコード
x_range = np.linspace(-1, 1, 25)
y_range = np.linspace(-1, 1, 25)
X, Y = np.meshgrid(x_range, y_range)
Z = z_surf(X,Y)

# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)

# 任意の仰角と方位角
#方位角方向に90度回転
#x z 軸で見たもの
ax.view_init(elev=0, azim=90)

plt.show()
```



[42]の場…

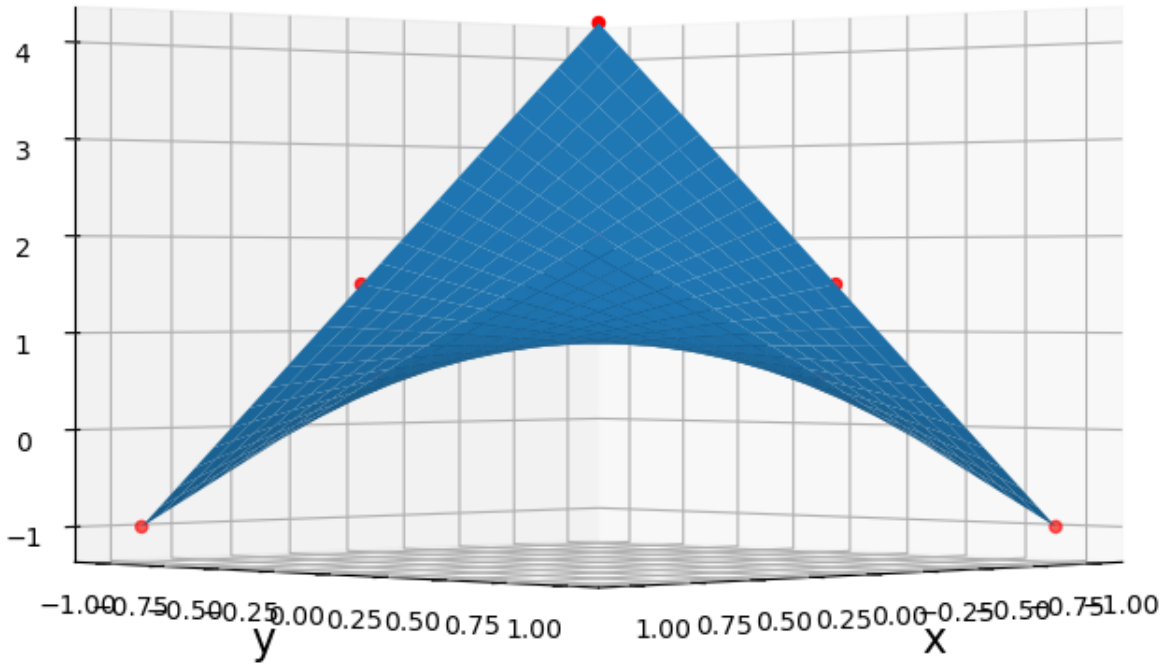
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)

# 任意の仰角と方位角
#ｙ ｚ軸でみたもの
ax.view_init(elev=0, azim=0)

plt.show()
```




[12]の場…

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

fig = plt.figure(figsize = (8, 8))
```

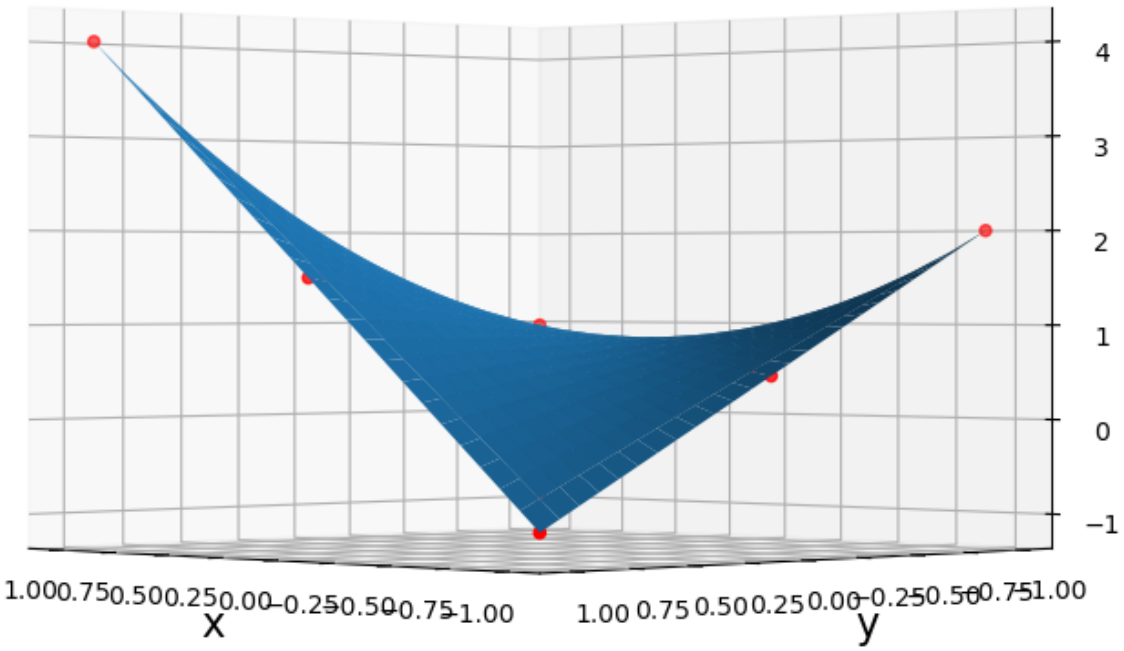
```
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)

# 任意の仰角と方位角
#方位角方向に45度回転
ax.view_init(elev=0, azim=45)

plt.show()
```




[13]の場…

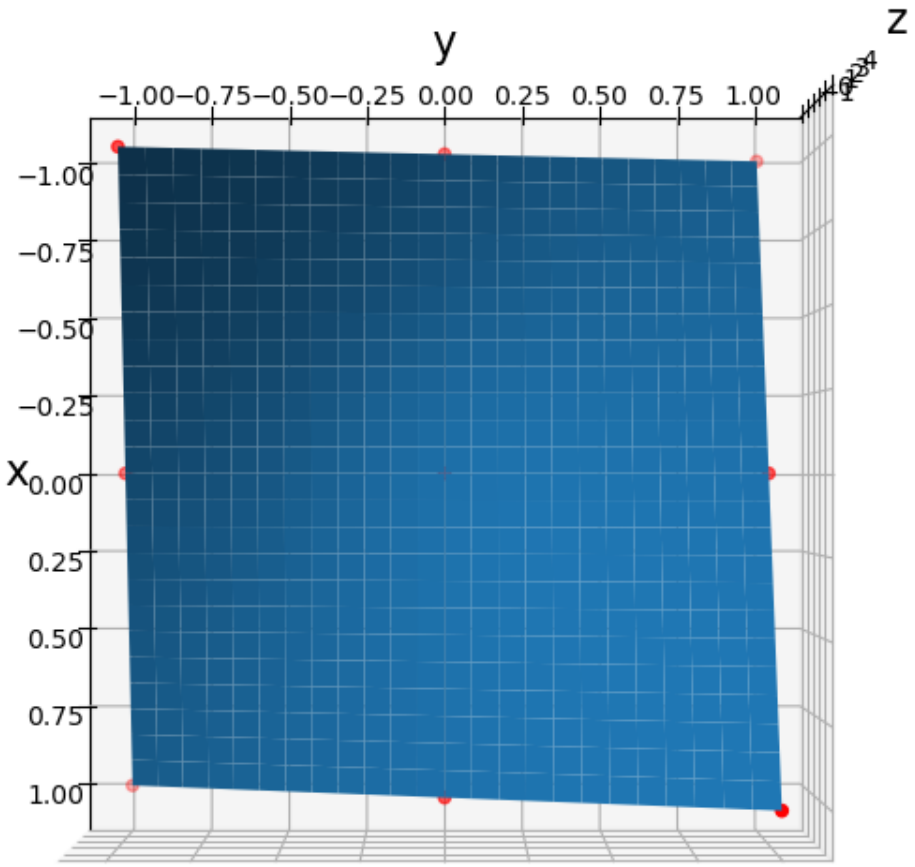
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)
```

```
# 任意の仰角と方位角
#方位角方向に135度回転
ax.view_init(elev=0, azim=135)

plt.show()
```



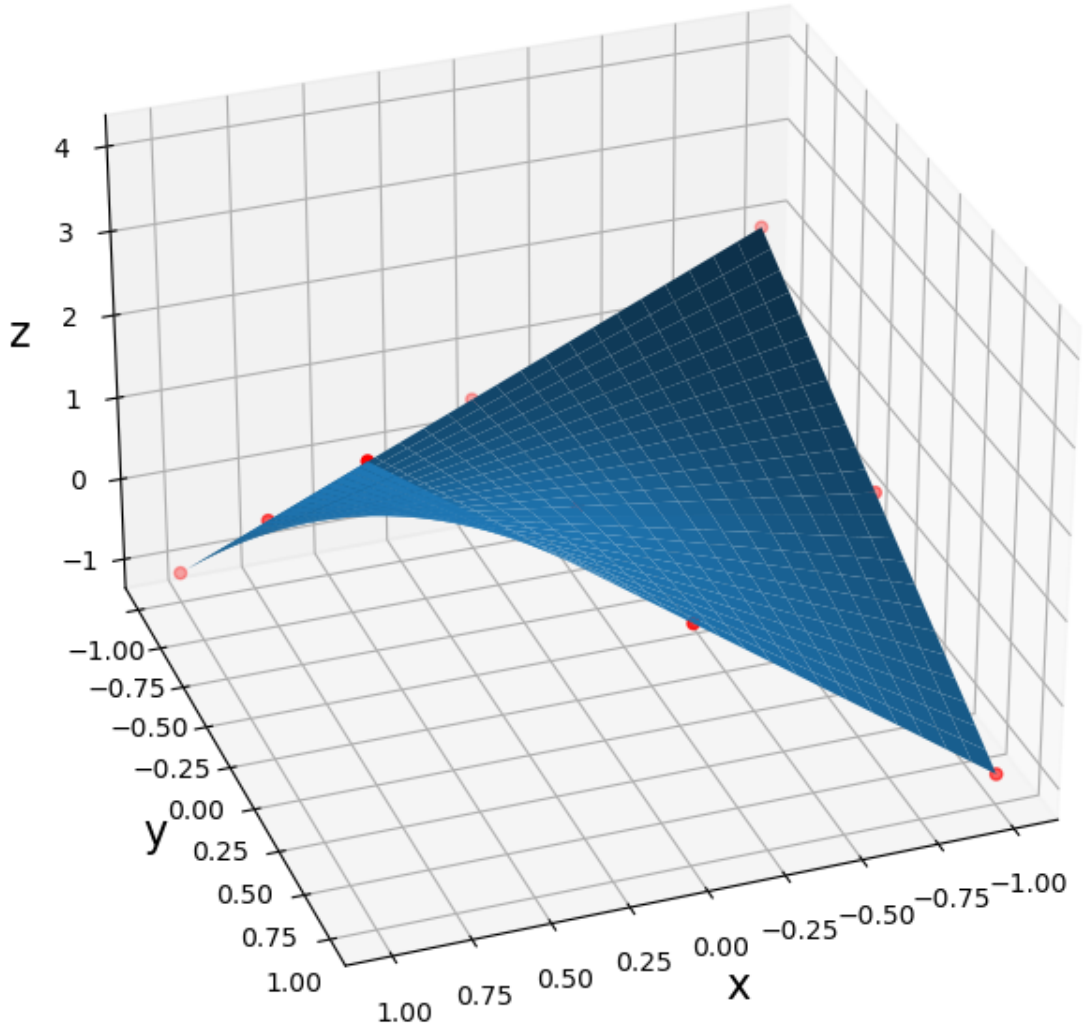

[14]の場…

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)

# 任意の仰角と方位角
#仰角方向に90度回転させたもの
ax.view_init(elev=90, azim=0)

plt.show()
```



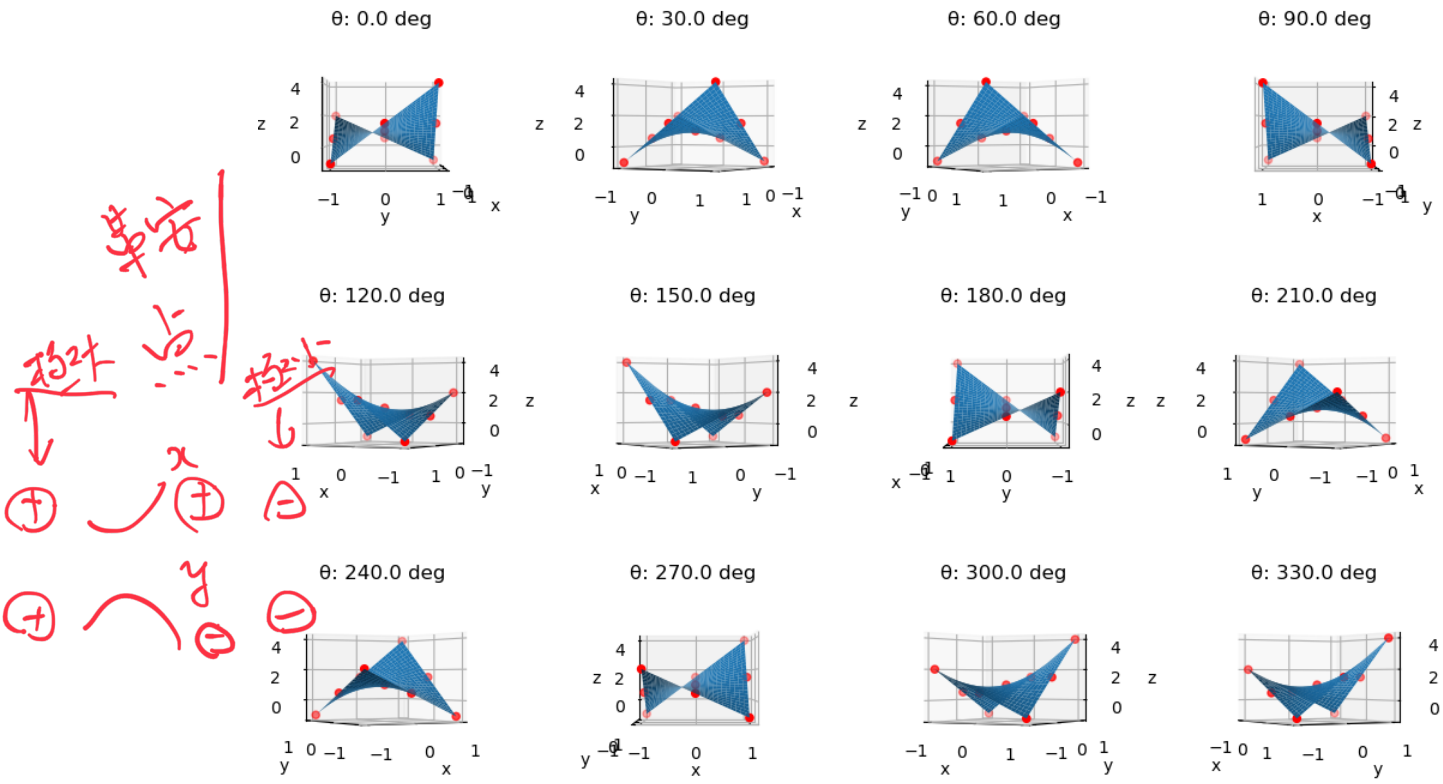

[15]の場…

```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# グラフの描写を表したコード
fig = plt.figure(figsize = (8, 8))
ax = fig.add_subplot(111, projection="3d")
ax.set_xlabel("x", size = 16)
ax.set_ylabel("y", size = 16)
ax.set_zlabel("z", size = 16)
ax.scatter(np.array(x),np.array(y),z, color='r')
ax.plot_surface(X, Y, Z)

# 任意の仰角と方位角
ax.view_init(elev=30, azim=70)

plt.show()
```




[16]の場…

```
#いろいろな角度から見たやつ（去年のやつから）
fig = plt.figure(figsize=(12,9))
for i in range(12):
    plt.subplots_adjust(wspace=0.7, hspace=0)
    ax = fig.add_subplot(3,4,i+1, projection="3d")
    ax.set_title("θ： {} deg".format(360/12*i))
    ax.scatter(x,y,z, color='r')
    ax.plot_surface(X,Y,Z)
    ax.set_xlabel("x")
    ax.set_ylabel("y")
    ax.set_zlabel("z")
    ax.view_init(0, 360/12*i)
```




# まとめ

最小二乗法でデータに曲線をフィット(近似)させたものの値と、正規方程式(Normal Equations)で、デザイン行列を用いた方法、特異値分解(Singular Value Decomposition)を用いた方法、ザイン行列 Aで、Ax=bとみなした場合のxについて解いた方法では、a0=2.5, a1=-2.2, a2=1と同じような近似結果が得られた。

また、χ2 の極小値をとり、データを関数に入れてsumをとった関数の値から、a0, a1, a2でパラメーターをとり、3次関数のグラフを作成した。

次に、x,y,z座標で,scipy.linalgのlinalg.lstsqで近似させ、グラフ化したものでは、9つのデータが表示され、近似された曲面の形状が可視化することができた。これにより、関数の値が低いところ（暗い部分）と、関数の値が高いところ（明るい部分）の勾配の変化も見やすくなった。