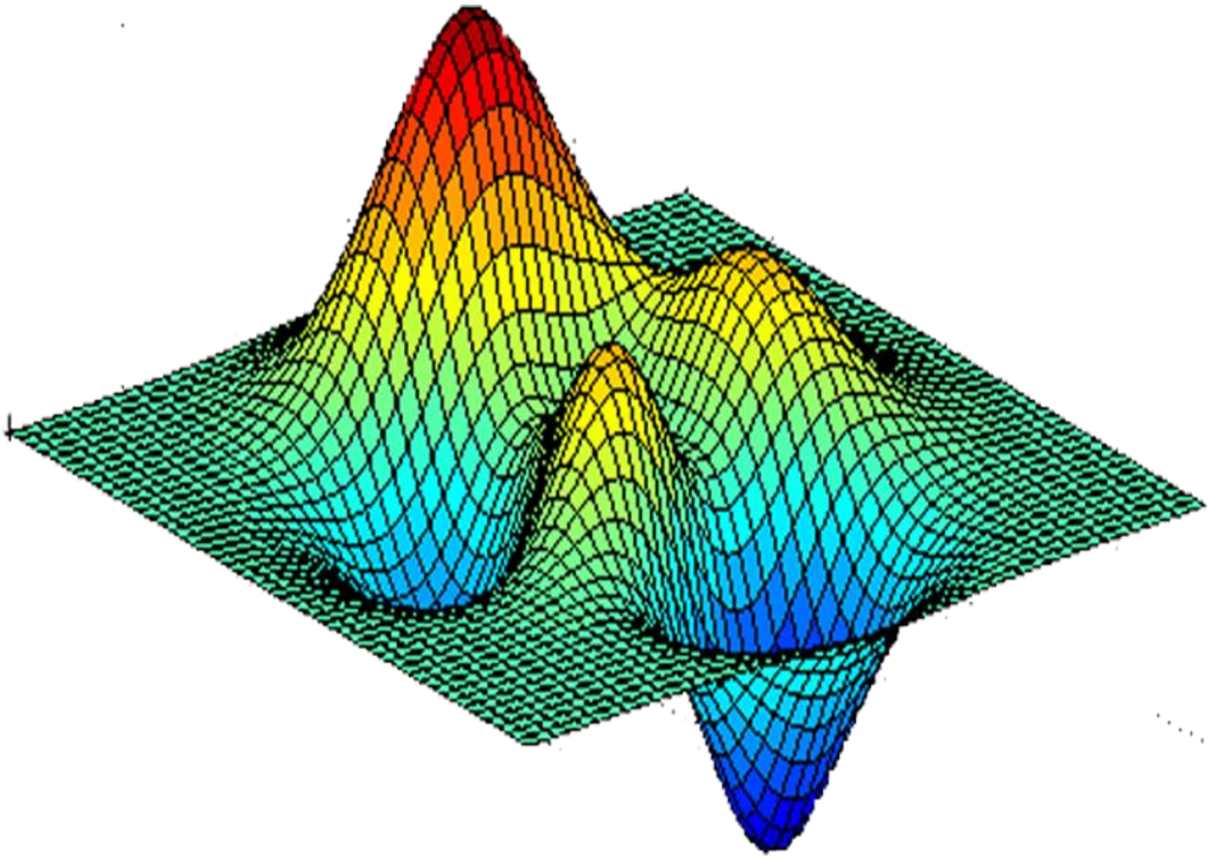
Numerical Computing

Least Squares Approximation and Gauss Jacobi Method



QASID AHMED ALEEM

COURSE TEACHER: DR. SHAHID QURESHI
BSC(HONOURS)
H-2 (MORNING)
INSTITUTE OF SPACE AND PLANETARY ASTROPHYSICS
UNIVERSITY OF KARACHI

Tasks:

Task 1:Least Squares Approximation

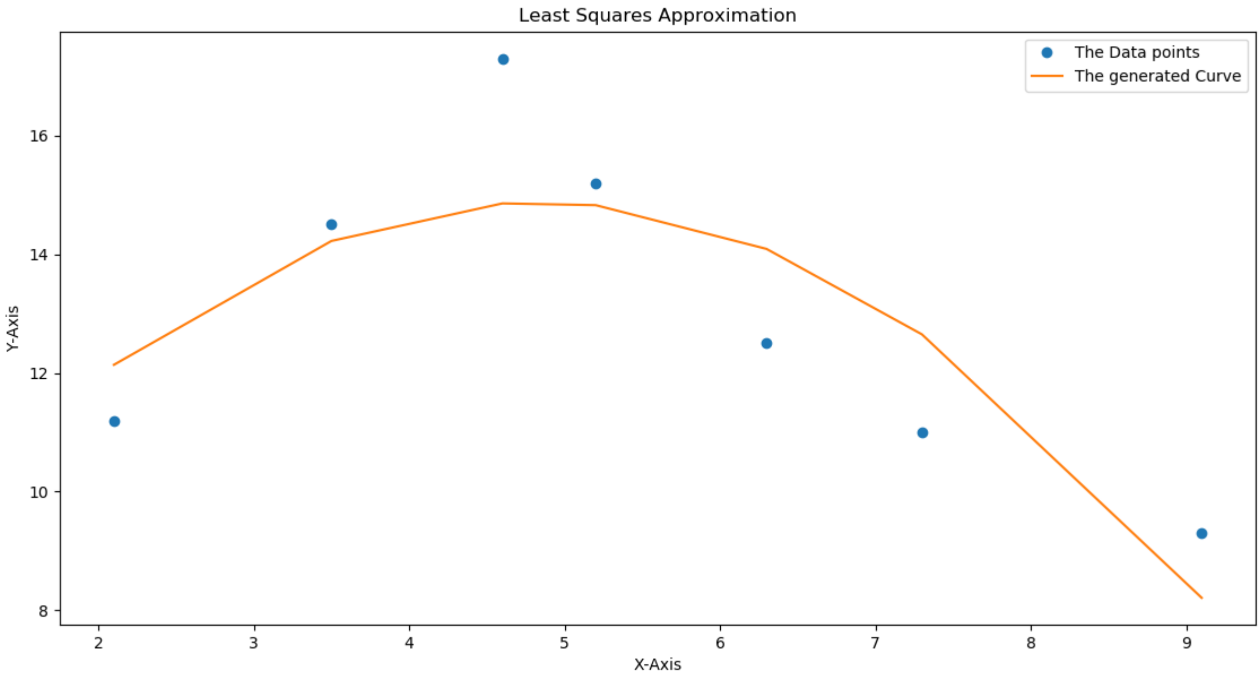
Code:

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

x=[2.1,3.5,4.6,5.2,6.3,7.3,9.1]
y=[11.2,14.5,17.3,15.2,12.5,11.0,9.3]

x2=[]
x3=[]
x4=[]
xiyi=[]
xiyi2=[]
print("{:^20}{:^15s}{:^15s}{:^15s}{:^15s}{:^15s}{:^15s}{:^15s}".\
format("i","x","y","xi^2","xi^3","xi^4","xiyi","xi^2yi"))
for i in range(0,len(x)):
    xt=x[i]**2
    x2.append(xt)
    xt=x[i]**3
    x3.append(xt)
    xt=x[i]**4
    x4.append(xt)
    xt=x[i]*y[i]
    xiyi.append(xt)
for i in range(0,len(x)):
    xt=x2[i]*y[i]
    xiyi2.append(xt)

for i in range(0,len(x)):
    print("{:^20d}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}\
{:^15.5f}".format(i,x[i],y[i],x2[i],x3[i],x4[i],xiyi[i],xiyi2[i]))
print("{:<20s}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}{:^15.5f}\
{:^15.5f}".format("Total",sum(x),sum(y),sum(x2),sum(x3),sum(x4)\
,sum(xiyi),sum(xiyi2)))
k1=[sum(x4),sum(x3),sum(x2),sum(xiyi2)]
k2=[sum(x3),sum(x2),sum(x),sum(xiyi)]
k3=[sum(x2),sum(x),len(x),sum(y)]

print("the matrix of the series of equations is thus")
for i in k1:
    print("{:10.5f}".format(i),end=" ")
print("\n")

for i in k2:
    print("{:10.5f}".format(i),end=" ")
print("\n")
for i in k3:
    print("{:10.5f}".format(i),end=" ")
print("\n")
def gauss_j(x,y,z):
    nr1=[1,1,1,1]
    nr2=[1,1,1,1]
    nr3=[1,1,1,1]
    nr4=[1,1,1,1]
    r1=[1,1,1,1]
    r2=[1,1,1,1]
    r3=[1,1,1,1]
    for i in range(0,len(x)):

        hit=x[0]
        nr2y=y[i]*hit-(x[i]*y[i-i])
```

```
        nr2[i]=nr2y

        nr3y=z[i]*hit-(x[i]*z[i-i])
        nr3[i]=nr3y
    y=nr2



    for i in range(0,len(x)):
        hit=y[1]
        nr1y=x[i]*hit-(x[i-i+1]*y[i])
        nr1[i]=nr1y



        nr4y=nr3[i]*hit-(nr2[i]*nr3[1])
        z[i]=nr4y

    for i in range(0,len(x)):
        hit=z[2]
        nr1y=nr1[i]*hit-(nr1[2]*z[i])
        r1[i]=nr1y

        nr2y=nr2[i]*hit-(nr2[2]*z[i])
        r2[i]=nr2y

    for i in range(0,len(x)):
        nr1y=r1[i]/r1[0]
        nr1[i]=nr1y
        nr2y=r2[i]/r2[1]
        nr2[i]=nr2y
        nr3y=z[i]/z[2]
        nr3[i]=nr3y




    return nr1,nr2,nr3 #returns 3 lists of the reduced echeleon matrix

a=gauss_j(k1,k2,k3)[0][3]
b=gauss_j(k1,k2,k3)[1][3]
c=gauss_j(k1,k2,k3)[2][3]
print("Applying Gauss Jordan Elimination....")
for i in gauss_j(k1,k2,k3)[0]:
    print("{:10.5f}".format(i),end=" ")
print("\n")
for i in gauss_j(k1,k2,k3)[1]:
    print("{:10.5f}".format(i),end=" ")
print("\n")
for i in gauss_j(k1,k2,k3)[2]:
    print("{:10.5f}".format(i),end=" ")
print("\n")

print("thus a={} b={} c={}".format(a,b,c))
def parabola(xi,x,y,z):
    para1=np.arange(0.0,len(xi),1)
    for i in range(0,len(xi)):
        para1[i]=float((x*(xi[i]**2)) +( y*xi[i]) + z)
    return para1


plt.plot(x,y,'o',label="The Data points")
plt.plot(x,parabola(x,a,b,c),'-', label="The generated Curve")
plt.legend(loc="upper right")
plt.title("Least Squares Approximation")
plt.xlabel("X-Axis")
```

```python
        plt.ylabel("Y-Axis")
        plt.grid
        plt.show()
```

Result:




```
        i                   x           y       xi^2       xi^3       xi^4        xiyi      xi^2yi
        0             2.10000    11.20000    4.41000    9.26100   19.44810    23.52000    49.39200
        1             3.50000    14.50000   12.25000   42.87500  150.06250    50.75000   177.62500
        2             4.60000    17.30000   21.16000   97.33600  447.74560    79.58000   366.06800
        3             5.20000    15.20000   27.04000  140.60800  731.16160    79.04000   411.00800
        4             6.30000    12.50000   39.69000  250.04700 1575.29610    78.75000   496.12500
        5             7.30000    11.00000   53.29000  389.01700 2839.82410    80.30000   586.19000
        6             9.10000     9.30000   82.81000  753.57100 6857.49610    84.63000   770.13300
Total                38.10000    91.00000  240.65000 1682.71500 12621.03410  476.57000  2856.54100
the matrix of the series of equations is thus
12621.03410 1682.71500  240.65000 2856.54100

1682.71500  240.65000   38.10000  476.57000

 240.65000   38.10000    7.00000   91.00000

Applying Gauss Jordan Elimination....
   1.00000    0.00000    0.00000   -0.36628

   0.00000    1.00000    0.00000    3.54150

   0.00000    0.00000    1.00000    6.31615

thus a=-0.36627540822340904 b=3.5414987617458245 c=6.316153452349639
```

Excel Work:

i	x	y	x2	x3	x4	xiyi	x2yi
1	2.1	11.2	4.41	9.261	19.4481	23.52	49.392
2	3.5	14.5	12.25	42.875	150.0625	50.75	177.625
3	4.6	17.3	21.16	97.336	447.7456	79.58	366.068
4	5.2	15.2	27.04	140.608	731.1616	79.04	411.008
5	6.3	12.5	39.69	250.047	1575.296	78.75	496.125
6	7.3	11.0	53.29	389.017	2839.824	80.3	586.19
7	9.1	9.3	82.81	753.571	6857.496	84.63	770.133
7	**38.1**	**91**	**240.65**	**1682.715**	**12621.03**	**476.57**	**2856.541**

Gauss Jordan Elimination:

a	b	c	=
12621.0341	1682.715	240.65	2856.541
1682.715	240.65	38.1	476.57
240.65	38.1	7	91

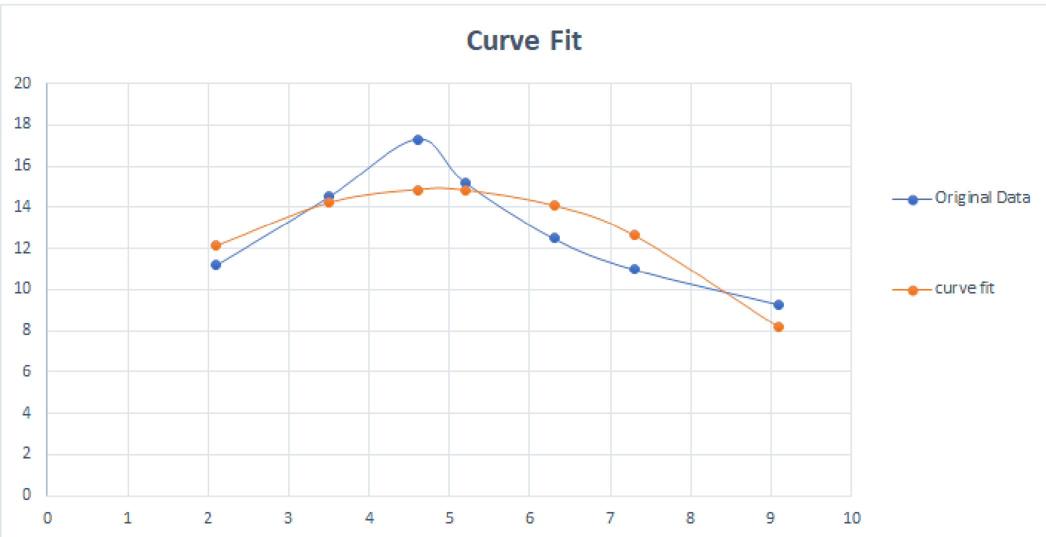
12621.0341	1682.715	240.65	2856.541
0	205722.0849	75916.0345	1208061.832
0	75916.03446	30434.8162	461087.5115

2596425449	0	-78238030	-1445170196
0	205722.0849	75916.0345	1208061.832
0	0	497869555	3144620511

1.29268E+18	0	0	-4.73477E+17
0	1.02423E+14	0	3.6273E+14
0	0	497869555	3144620511

1	0	0	-0.366275408
0	1	0	3.541498762
0	0	1	6.316153452

ax2+bx+c
12.1380263
14.2245254
14.8566601
14.82786
14.0901247
12.6502779
8.21252563



Task 2:Gauss Jacobi Method

Code:

```
print("the given Equations...")
print(" 6x-2y+z=11")
print("-2x+7y+2z=5")
print(" x+2y-5z=-1\n")

x1=0
x2=0
x3=0

print("First Method")
print("{:^15s}{:^15s}{:^15s}".format("x1","x2","x3"))
for i in range(0,10):
    print("{: 15.12f}{: 15.12f}{: 15.12f}".format(x1,x2,x3))
    a=(11+(2*x2)-x3)/6
    b=(5+(2*x1)-(2*x3))/7
    c=(1+x1+(2*x2))/5

    x1=a
    x2=b
    x3=c

print("x1={} x2={} x3={} after {} iterations".format(x1,x2,x3,i))

x1=0
x2=0
x3=0

print("\n\n\nSecond method")
print("{:^15s}{:^15s}{:^15s}".format("x1","x2","x3"))
for i in range(0,10):
    print("{: 15.12f}{: 15.12f}{: 15.12f}".format(x1,x2,x3))
    x1=(11+(2*x2)-x3)/6
    x2=(5+(2*x1)-(2*x3))/7
    x3=(1+x1+(2*x2))/5



print("\nx1={} x2={} x3={} after {} iterations".format(x1,x2,x3,i))
```

Result:

```
the given Equations...
 6x-2y+z=11
-2x+7y+2z=5
 x+2y-5z=-1

First Method
      x1             x2             x3
 0.000000000000 0.000000000000 0.000000000000
 1.833333333333 0.714285714286 0.200000000000
 2.038095238095 1.180952380952 0.852380952381
 2.084920634921 1.053061224490 1.080000000000
 2.004353741497 1.001405895692 1.038208616780
 1.994100529101 0.990327178490 1.001433106576
 1.996536875067 0.997904977864 0.994950977216
 2.000143163085 1.000453113672 0.998469366159
 2.000406143531 1.000478227693 1.000209878086
 2.000124429550 1.000056075841 1.000272519783
x1=1.9999732719832346 x2=0.9999576885047652 x3=1.0000473162465864 after 9 iterations



Second method
      x1             x2             x3
 0.000000000000 0.000000000000 0.000000000000
 1.833333333333 1.238095238095 1.061904761905
 2.069047619048 1.002040816327 1.014625850340
 1.998242630385 0.995319080013 0.997776158082
 1.998810333657 1.000295478736 0.999880258226
 2.000118449874 1.000068054757 1.000050911878
 2.000014199606 0.999989510780 0.999998644233
 1.999996729554 0.999999452949 0.999999127090
 1.999999963135 1.000000238870 1.000000088175
 2.000000064927 0.999999993358 1.000000010329

x1=1.9999999960645238 x2=0.9999999959245361 x3=0.9999999975827192 after 9 iterations
Press any key to continue . . .
```

Excel Work:

First Method:

x1	x2	x3
0	0	0
1.833333	0.714286	0.2
2.038095	1.180952	0.646667
2.119206	1.111837	0.866286
2.059565	1.072263	0.970356
2.029028	1.025488	1.000055
2.008487	1.008278	1.005828
2.001788	1.00076	1.004028
1.999582	0.99936	1.001969
1.999458	0.999318	1.000704
1.999655	0.999644	1.000173
1.999852	0.999852	1

Second Method:

x1	x2	x3
0	0	0
1.83333	1.238095	1.061905
2.06905	1.002041	1.014626
1.99824	0.995319	0.997776
1.99881	1.000295	0.99988
2.00012	1.000068	1.000051
2.00001	0.99999	0.999999
2.00000	0.999999	0.999999
2.00000	1	1
2.00000	1	1