ID - 202218061

NAME - JATAN SAHU

ASSIGMENT - 04

+ Code + Text

1. Write a program for Linear Spline, Quadratic Spline and Cubic Spline.

X	Y= f(x)
3.0	2.5
4.5	1.0
7.0	2.5
9.0	0.5

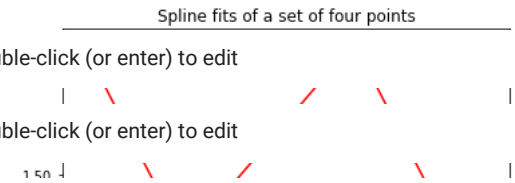
```
from sympy import symbols, Eq
import matplotlib.pyplot as plt
import numpy as np

```

Linear Spline

```
X=[3,4.5,7,9]
Y=[2.5,1,2.5,0.5]
```

```
def Spline():
  # X = list(map(float , input("Enter X values for Spline seprated by space :").split(" ")))
  # Y = list(map(float , input("Enter Y values for Spline seprated by space :").split(" ")))
  # while len(X)!=len(Y):
  #     X = list(map(float , input("Enter X values for Spline seprated by space :").split(" ")))
  #     Y = list(map(float , input("Enter Y values for Spline seprated by space :").split(" ")))
  X=[3,4.5,7,9]
  Y=[2.5,1,2.5,0.5]
  #Finding slope
  M =[]
  for i in range(len(X)-1):
    M.append( (Y[i+1] - Y[i])/(X[i+1] - X[i]) )

  x = symbols('x')
  lines =[]
  for i in range(len(M)):
    eq1 = (Y[i] + M[i]*(x - X[i]) )
    print(f"Equation{i+1} = ",eq1 , f"if {X[i]} <= X <= {X[i+1]}")
    print()

  # #Plotting
  plt.plot(X,Y , color = 'r',marker = 'o', ms = 5, mec = 'b', mfc = 'r')
  plt.scatter(X,Y)
  plt.title("Spline fits of a set of four points")


Spline()


```

```
Equation1 =  5.5 - 1.0*x if 3 <= X <= 4.5

Equation2 =  0.6*x - 1.7 if 4.5 <= X <= 7

Equation3 =  9.5 - 1.0*x if 7 <= X <= 9
```



Double-click (or enter) to edit

Double-click (or enter) to edit

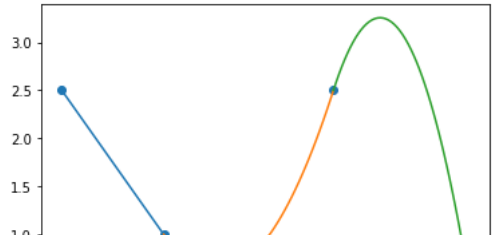
Quadratic Spline

```
1 x=[3,4.5,7,9]
2 y=[2.5,1,2.5,0.5]
```

```
1 m = []
2 for i in range(len(x)-1):
3   slope = (y[i+1]-y[i])/(x[i+1]-x[i])
4   m.append(slope)
```

```
1 # Quadratic Spline
2 A = np.zeros([3*3,3*3])
3 b = np.zeros([3*3])
4 e = 0
5 # 1st Rule : Functional Value of adjacent polynmial must be equal at internal knots
6 print(A)
7 for i in range(1,3):
8   print(f'Equation {e+1} : {x[i]**2}*a{i} + {x[i]}*b{i} + c{i} = {y[i]} ')
9   A[e][(i-1)*3:(i-1)*3+3] =  x[i]**2,x[i],1
10  b[e] = y[i]
11  print(f'Equation {e+2} : {x[i]**2}*a{i+1} + {x[i]}*b{i+1} + c{i+1} = {y[i]} ')
12  A[e+1][(i)*3:(i)*3+3] = x[i]**2,x[i],1
13  b[e+1] = y[i]
14  e += 2
15
16 # 2nd Rule : First and Last function must pass through th end points
17 print(f"Equation {e} : {x[0]**2}*a1 + {x[0]}*b1 + c1 = {y[0]}")
18 A[e][0:3] = x[0]**2,x[0],1
19 b[e] = y[0]
20 e+=1
21 print(f"Equation {e} : {x[3]**2}*a1 + {x[3]}*b1 + c1 = {y[3]}")
22 A[e][-3:] = x[3]**2,x[3],1
23 b[e] = y[3]
24 e+=1
25
26 # 3rd Rule : The first derivatives at interior knots must be equal
27 for i in range(1,3):
28   print(f'Equation {e} : {2*x[i]}*a{i} + b{i} = {2*x[i]}*a{i+1} + b{i+1}')
29   A[e][(i-1)*3:(i)*3+3] =  2*x[i],1,0,-2*x[i],-1,0
30   b[e] = 0
31   e+=1
32
33 # 4th Rule : Second Derivative is zero at first point
34 print(f"Equation {e} : a1 = 0")
35 A[e][0] = 1
36
37 print()
38 print(A)
39 print()
40 print(b)
41
42 sol = np.dot(np.linalg.inv(A),b)
43 sol = sol.reshape(3,3)
44 print(sol)
45
46 plt.scatter(x,y)
47 for i in range(len(m)):
48   p = np.linspace(x[i],x[i+1])
49   plt.plot(p,sol[i][0]*p**2 + sol[i][1]*p + sol[i][2])
50 plt.show()
```

```
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0.]]
Equation 1 : 20.25*a1 + 4.5*b1 + c1 = 1
Equation 2 : 20.25*a2 + 4.5*b2 + c2 = 1
Equation 3 : 49*a2 + 7*b2 + c2 = 2.5
Equation 4 : 49*a3 + 7*b3 + c3 = 2.5
Equation 4 : 9*a1 + 3*b1 + c1 = 2.5
Equation 5 : 81*a1 + 9*b1 + c1 = 0.5
Equation 6 : 9.0*a1 + b1 = 9.0*a2 + b2
Equation 7 : 14*a2 + b2 = 14*a3 + b3
Equation 8 : a1 = 0

[[ 20.25   4.5    1.     0.     0.     0.     0.     0.     0.  ]
 [  0.     0.     0.    20.25   4.5    1.     0.     0.     0.  ]
 [  0.     0.     0.    49.     7.     1.     0.     0.     0.  ]
 [  0.     0.     0.     0.     0.     0.    49.     7.     1.  ]
 [  9.     3.     1.     0.     0.     0.     0.     0.     0.  ]
 [  0.     0.     0.     0.     0.     0.    81.     9.     1.  ]
 [  9.     1.     0.    -9.    -1.     0.     0.     0.     0.  ]
 [  0.     0.     0.    14.     1.     0.   -14.    -1.     0.  ]
 [  1.     0.     0.     0.     0.     0.     0.     0.     0.  ]]

[1.  1.  2.5 2.5 2.5 0.5 0.  0.  0. ]
[[-6.49686066e-16 -1.00000000e+00  5.50000000e+00]
 [ 6.40000000e-01 -6.76000000e+00  1.84600000e+01]
 [-1.60000000e+00  2.46000000e+01 -9.13000000e+01]]
```



CUBIC SPLINE

```
#For cubic
# Quadratic Spline
A = np.zeros([12,12])
B = np.zeros([12])
c=1
n=3
# 1st : The functional Value must be equal at interior knots
# (2n - 2) conditions

for i in range(2,4): #for 3n i =2,3
  print(f"Equation{c} : {(X[i-1])**3}a{i-1} + {(X[i-1])**2}b{i-1} + {X[i-1]}c{i-1} + d{i-1} = {Y[i-1]}")
  A[c-1][(i-2)*4 : (i-2)*4 +4] = (X[i-1])**3 , (X[i-1])**2 , X[i-1] , 1
  B[c-1]=Y[i-1]

  print(f"Equation{c+1} : {(X[i-1])**3}a{i} + {(X[i-1])**2}b{i} + {X[i-1]}c{i} + d{i-1} = {Y[i-1]}")
  A[c][(i-1)*4 : (i-1)*4 +4] = (X[i-1])**3 , (X[i-1])**2 , X[i-1] , 1
  B[c]=Y[i-1]
  c+=2

# 2nd : First and Last function must pass through th end points
print(f"Equation{c} : {(X[0])**3}a{1} + {(X[0])**2}b{1} + {X[0]}c{1} + d{1} = {Y[0]}")
A[c-1][(0)*4 : (0)*4 +4] = (X[0])**3  ,(X[0])**2 ,X[0] ,1
B[c-1] = Y[0]
c+=1
print(f"Equation{c} : {(X[3])**3}a{n} + {(X[n])**2}b{n} + {X[n]}c{n} + d{n} = {Y[n]}")
A[c-1][(2)*4 : (2)*4 +4] = (X[n])**3  ,(X[n])**2 ,X[n] ,1
B[c-1] = Y[n]
c+=1

# 3rd : First derivative at the interior knot, must be equal (n-1) condition
for i in range(2,4): #for 3n i =2,3  eqaution 7 and 8
  print(f"Equation{c} : {3*(X[i-1])}a{i-1} + {(2*X[i-1])}b{i-1} + c{i-1} = {3*(X[i-1])}a{i} + {(2*X[i-1])}b{i} + c{i}")
  A[c-1][(i-2)*4 : (i-2)*4 +4] = 3*((X[i-1])**2) , (X[i-1])*2 , 1 , 0
  A[c -1][(i-1)*4 : (i-1)*4 +4] = -(((X[i-1])**2)*3) , -((X[i-1])*2)  , -1, 0
  B[c-1]= 0
  c+=1

```

```
# 4th : Second  derivative at the interior knots, must be equal
for i in range(2,4): # equation 9 and 10
  print(f"Equation{c} : {6*(X[i-1])}a{i-1} + {2}b{i-1}  = {6*(X[i-1])}a{i} + {2}b{i} ")
  A[c-1][(i-2)*4 : (i-2)*4 +4] = 6*((X[i-1])) , 2 , 0 , 0
  A[c -1][(i-1)*4 : (i-1)*4 +4] = -((X[i-1])*6) , -2  , 0, 0
  B[c-1]=0
  c+=1


# 5th : The second derivatives at the end knots are zero (2 condition)
print(f"Equation{c} : {(X[0])*6}a{1} + {2}b{1} = {0}")
A[10][(0)*4 : (0)*4 +4] = (X[0])*6  ,2 ,0 ,0
B[10]=0
c+=1
print(f"Equation{c} : {(X[n])*6}a{n} + {2}b{n} = {0}")
A[11][(2)*4 : (2)*4 +4] = (X[n])*6  ,2 ,0 ,0
B[11]=0


print(A)
#X=A^-1 .B
sol = np.dot(np.linalg.inv(A),B)
sol = sol.reshape(3,4)
print("size :", sol.shape)
print(sol)

plt.scatter(X,Y)
for i in range(len(m)):
  p = np.linspace(X[i],x[i+1])
  plt.plot(p,sol[i][0]*p**3 + sol[i][1]*p**2 + sol[i][2]*p +sol[i][3])
plt.show()

```

```
    Equation1 : 91.125a1 + 20.25b1 + 4.5c1 + d1 = 1
1
    Equation4 : 343a3 + 49b3 + 7c3 + d2 = 2.5
1
    Equation7 : 13.5a1 + 9.0b1 + c1 = 13.5a2 + 9.0b2 + c2
    Equation8 : 21a2 + 14b2 + c2 = 21a3 + 14b3 + c3
    Equation9 : 27.0a1 + 2b1  = 27.0a2 + 2b2
    Equation10 : 42a2 + 2b2  = 42a3 + 2b3
    Equation11 : 18a1 + 2b1 = 0
    Equation12 : 54a3 + 2b3 = 0
    [[  91.125   20.25     4.5      1.       0.       0.       0.       0.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.      91.125   20.25     4.5      1.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.     343.      49.       7.       1.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.       0.       0.       0.       0.
       343.      49.       7.       1.   ]
     [  27.       9.       3.       1.       0.       0.       0.       0.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.       0.       0.       0.       0.
       729.      81.       9.       1.   ]
     [  60.75     9.       1.       0.     -60.75    -9.      -1.       0.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.     147.      14.       1.       0.
      -147.     -14.      -1.       0.   ]
     [  27.       2.       0.       0.     -27.      -2.       0.       0.
         0.       0.       0.       0.   ]
     [   0.       0.       0.       0.      42.       2.       0.       0.
        42.       2.       0.       0.   ]
```