GROUP 7

Curve Fitting y = ax^p Python Program

Power Function Fit Using the Least Square Method

This Python program implements least square method to fit curve of type y = ax^p.

y = ax^p

log y = log ax^p

log y = log a + log x^p

log y = log a + plogx

Let Y = log y,

X = log x,

C = log a

Y = pX + C

We first read n data points from user and then we implement curve fitting for y = ax^p using least square approach in Python programming language as follow:

In [1]:
```
# -*- coding: utf-8 -*-
"""
Created on Tue Nov  2 21:02:00 2021

@author: OPEYEMI IBRAHIM
"""
# Curve Fitting y = ax^p Python Program

# This Python program implements least square fit method to fit curve of type y = ax

# y = ax^p
# log y = log ax^p
# log y = log a + log x^p
# log y = log a + plogx

# Let Y = log y,
#    X = log x,
#    C = log a

# Y = pX + C

# We first read n data points from user and then we implement curve fitting for y =

# Using the Least Square Fit (LSF) method

# Fitting y = ax^p to given n data points
import numpy as np

# Reading value of n
```

```
n = int(input("Please, enter number of data points, n = "))

# Creating numpy array x & y to store n data points;
x = np.zeros(n)
y = np.zeros(n)

# Reading data points for x and y;
print("Enter data points for x and y: ")
for i in range(n):
    x[i] = float(input("x["+str(i)+"]= "))
    y[i] = float(input("y["+str(i)+"]= "))

# Finding required sum for least square methods
sumX, sumX2, sumY, sumXY = 0, 0, 0, 0

for i in range(n):
    sumX = sumX + np.log(x[i])
    sumY = sumY + np.log(y[i])
    sumX2 = sumX2 + np.log(x[i])*np.log(x[i])
    sumXY = sumXY + np.log(x[i])*np.log(y[i])

# Finding coefficients a and p

# Let D = denominator
D = (n*sumX2)-(sumX*sumX)

p = (n*sumXY-sumX*sumY)/D
C = (sumY - p*sumX)/n

# Obtaining a from C
a = np.exp(C)

print ('\nOUTPUT RESULT WHEN PROMPTING THE USER TO ENTER THE VALUES:\n')

# Displaying coefficients a, p & equation
print("\nCoefficients are: ")
print("\na = %.3f" %(a))
print("\np = %.3f" %(p))

print("\nHence, y=%.3f(x)^%.3f" %(a, p))

print("\n'y = ax^p' is the best Power Fit method")
```

```
Please, enter number of data points, n = 6
Enter data points for x and y:
x[0]= 1
y[0]= 1200
x[1]= 2
y[1]= 900
x[2]= 3
y[2]= 600
x[3]= 4
y[3]= 200
x[4]= 5
y[4]= 110
x[5]= 6
y[5]= 50

OUTPUT RESULT WHEN PROMPTING THE USER TO ENTER THE VALUES:


Coefficients are:

a = 2033.966

p = -1.749
```

```
Hence, y=2033.966(x)^-1.749

'y = ax^p' is the best Power Fit method
```

Method 2: By reading our values rigidly into our codes:

In [2]:

```
# Using the Least Square Fit (LSF) method

# Fitting y = ax^p to given n data points
import numpy as np

# Reading value of n, i.e. number of data points
n = 6

# Creating numpy array x & y to store n data points;
x = np.zeros(n)
y = np.zeros(n)

# Reading data points for x;
x = [1, 2, 3, 4, 5, 6]

# Reading data points for y;
y = [0 for i in range(n)];
y[0] = 1200;
y[1] = 900;
y[2] = 600;
y[3] = 200;
y[4] = 110;
y[5] = 50;

# Finding required sum for least square methods
sumX, sumX2, sumY, sumXY = 0, 0, 0, 0

for i in range(n):
    sumX = sumX + np.log(x[i])
    sumY = sumY + np.log(y[i])
    sumX2 = sumX2 + np.log(x[i])*np.log(x[i])
    sumXY = sumXY + np.log(x[i])*np.log(y[i])

# Finding coefficients a and p

# Let D = denominator
D = (n*sumX2)-(sumX*sumX)

p = (n*sumXY-sumX*sumY)/D
C = (sumY - p*sumX)/n

# Obtaining a from C
a = np.exp(C)

print ('\nOUTPUT RESULT WHEN READING OUR VALUES RIGIDLY INTO OUR CODES:\n')

# Displaying coefficients a, p & equation
print("\nCoefficients are: ")
print("\na = %.3f" %(a))
print("\np = %.3f" %(p))

print("\nHence, y=%.3fx^%.3f" %(a, p))

print("\n'y = ax^p' is the best Power Fit method")
```

```
OUTPUT RESULT WHEN READING OUR VALUES RIGIDLY INTO OUR CODES:


Coefficients are:
```

a = 2033.966

p = -1.749

Hence, y=2033.966x^-1.749

'y = ax^p' is the best Power Fit method

In [ ]: