**CURVE FITTING**

**INTRODUCTION:** A curve can be estimated from a given set of coordinates. We essentially use matplotlib library to plot the same. But in real world situations data can be corrupted, fitting such data into curves is challenging

**APPROACH:**

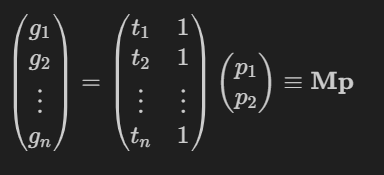
* **DATASET 1:**

Dataset contains noisy data of a straight line. Aim is to figure out slope and intercept of the line. We use LEAST SQUARE REGRESSION for the same.

Equation of a straight line: y = mx + c

Error in least square method is calculated as: , where f(x) is the function that we are predicting. For least square regression we use the lstsq function of the numpy library.

We build the M matrix for the same by using column stack, we column stack x values and 1 for line estimation.



The lstsq method returns the unknown parameters slope and intercept.

We plot the function obtained along with the error bar, that is the standard deviation of the data from estimated line.

**FUNCTIONING OF THE CODE:**

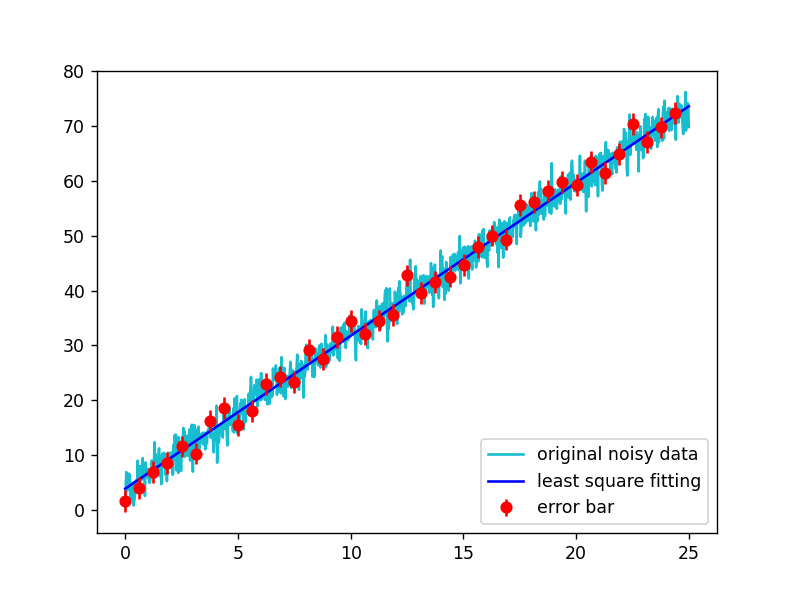
***def collect\_data(filename):* Function is used to retrieve numpy arrays of the dataset from the text file. This uses the numpy inbuilt function genfromtext. It splits the dataset into x and y coordinates and returns two separate numpy arrays.**

***def estimate\_line(x,y):* Funciton generates the line intercept and slope by using inbuilt numpy.linalg.lstsq function. Returns the resultant y coordinates.**

**The error array is calculated as difference between estimation and actual value. Standard deviation of the error array gives us the error for plotiing error bars.**

**The result is plotted along with the original dataset and error bar.**

**REQUIRED PLOT:**

****

* **DATASET 2:**

Dataset contains noisy data for sum of 3 Sine functions. We know two of the have time period T and 3T.

**WE USE CUSTOM LEAST SQUARE APPROXIMATION BY APPLYING LEAST SQUARES TWICE.**

Our first aim is to identify the time

Period of the dataset. If we plot the

Dataset we get a figure as shown.

This plot intersects y=0 in 5 points.

We can say that :

TIME PERIOD = 2\*DISTANCE BETWEEN

CONSECUTIVE POINTS

The user defined function Generate\_time\_period

does the same.

We find from the function that time period is : **2.495**

* LCM of the 3 time periods (T, 3T , UNKNOWN) = 2.495

Without loss of generality, We assume that this is the time period 3T and generate the plot for the known sine waves T and 3T. We calculate the constant coefficients of the sine functions using linalg.lstsq. M matrix was constructed by column stacking the two sin functions and 1s.

We get the curve as **2.0535 sin(7.55x) + 5.955 sin(2.52 x) - 0.025**

We can clearly verify that the plot almost coincides

Except for the 3rd bump (missing sine wave).

Hence we can argue that we have predicted the time

Periods of two sine waves correctly.

**Error array= y\_actual – y\_estimation**

Plotting error array we find that we get the missing

Sine Function (green colour)

The time period of this sine function can also

Be predicted using the user defined

**Generate\_time\_period** function.

We find the time period of unknown sine function is = 0.5116

Now we apply least square approximation again to find the unknown constant coefficients. M matrix was constructed by column stacking the 3 sine functions and 1s.

We get the curve as: 1.989 sin (7.55x) + 5.994 sin(2.518x) + 0.878 sin (12.27x)

We also use the in built curve\_fit function from scipy.optimize. We defined the required function with unknown parameters in generate\_fucntion function. We plot the estimation we got in both the methods and calculate the error.

We get the curve as: 1.99 sin (7.54x) + 6.015 sin(2.513x) + 0.977 sin (12.53x)

Error in custom lst sq = 0.582 Error in curve\_fit= 0.5

**FUNCITONING OF THE CODE:**

***def collect\_data(filename):* Function is used to retrieve numpy arrays of the dataset from the text file. This uses the numpy inbuilt function genfromtext. It splits the dataset into x and y coordinates and returns two separate numpy arrays.**

***def generate\_time\_period(read\_data):* Function takes in the coordinates as arguments and returns the time period by calculating number of times it crosses 0. We set up a flag=1 to calculate the x value when function crosses 0 from 0- to 0+. We setup the flag=-1 to calculate the x value when function crosses 0 from 0+ to 0-. The labels latest and first occurrence are defined to tackle the redundancies in the dataset (it is not increasing or decreasing monotonically as expected). Function gets the data points of all x values as mentioned above, takes the difference between consecutive points in data points and returns the mean difference. This is half of the required time period.**

**DIFFERNECE BETWEEN CUSTOM LST SQ AND CURVE\_FIT:**

* Clearly our estimation has greater error than the curve\_fit
* The function coefficients vary in the estimated unknown sine function
* The 3 bumps are not very accurate in some areas.
* We see that curve fit produces symmetric bumps

Whereas our prediction is slightly distorted.

* **DATASET 3\_1:**

The dataset contains the data of Planck’s measurement with a noise added to it.

We use the in built curve fit function. In this we use the standard values of h,c and KB.

We estimate the unknown parameter T using curve\_fit. We start giving the initial values for curve fit. When no value is given our output is bizarre. We get overflow error in run time errors until we increase the starting value up to 150. By hit and trial method we find that any initial value >=150 works perfectly fine.

estimated temperature= 5001.413374520575

error in estimation= 9.952300906356534e-10

**FUNCTIONING:**

***def collect\_data(filename):* Function is used to retrieve numpy arrays of the dataset from the text file. This uses the numpy inbuilt function genfromtext. It splits the dataset into x and y coordinates and returns two separate numpy arrays.**

***def generate\_funciton(xd3,T):* generated the required plack’s formula from the given input xd3 and unknown parameter T.**

**ESTIMATED CURVE:**

* **DATASET 3\_2:**

The dataset contains the data of Planck’s measurement with a noise added to it.

We use the in built curve fit function. In this we estimate the values of h,c and KB.

We estimate the unknown parameter T using curve\_fit. We start giving the intial values for curve fit. When no value is given our output is bizarre.

We give the starting values close to the standard values. After multiple hit and trial shots,

The given set of input parameters is considered almost ideal:

T=5000(we got this from first method) [4000,6000] c=3e8 [2e8,4e8]

kB = 1e-23 [ 0.8e-23, 2e-23] h=7.2e-34[6e-34, 8e-34]

estimated temperature= 5898.927714833197

error in curve fit= 9.952063309143707e-10

estimated values:

------------------------------------------------------------

h=6.677363523529996e-34 error%=0.7903928079999458

c=301502482.9580805 error%=0.5008276526934902

kB=1.1797855418578128e-23 error%=14.508294068274441

**FUNCTIONING:**

***def collect\_data(filename):* Function is used to retrieve numpy arrays of the dataset from the text file. This uses the numpy inbuilt function genfromtext. It splits the dataset into x and y coordinates and returns two separate numpy arrays.**

***def generate\_funciton(xd3,T):* generated the required plack’s formula from the given input xd3 and unknown parameter T,h,c,kB.**

***def error(given,actual):* generated the error in the calculation of the constant**

**ESTIMATED CURVE:**