Assignment 3 : Curve Fit

Dataset 1

Construction of Matrix :

- First I have parsed the given file into x_data and y_data and stored them numpy arrays.
- I have converted the data values into float type.
- Then with the help of numpy library I created the matrix M:
  - I used `column stack` ; It basically takes columns of same length and appends into one single matrix of size *no. of columns x no. of rows in a column*
  - The first column has x_data obtained from the dataset.
  - And the second column has 1's.
  - The using least square function under linear algebra, I solve the matrix M. The equations that we obtained from the matrix are of the form **p1 * x_data + p2 = y_data** (correspondingly for each data value). We obtain an estimate for p1, p2 on solving.

Plots :

- I have plotted error bars for every 25 data points.(Figure1)
- I have defined a straight line equation as a function(`stline`) and using the estimated values obtained, I found values for y_estimated.
- Subtracting y_estimated and y_data gives noise.
- With x_data and y_estimated, using `stline`, estimated line has been plotted.
- Using legend() under matplotlib labels have been given to the graph.(Figure2)

Dataset 2

Construction of Matrix :

- Similar to the dataset 1, I have parsed the file into x_data and y_data, converted them into float and stored them as numpy arrays.
- The matrix M:
  - It was given that, the curve is superposition of 3 sine waves.
  - Using hit and trial method, I used different combinations for the frequencies like (x,2x,3x), (x,3x,4x),... so on. The best fit was achieved at the combinations (x,3x,5x). I used these frequencies for the rest of the code.
  - Assuming that they have different amplitudes, and may have been superimposed along with some constant, I chose 4 different variables i.e. Amplitudes - `p1,p2,p3`, Constant - `p4`.
  - The first column has `sin(x)` terms, second column has `sin(3x)` terms, third column has `sin(5x)` terms and the fourth column has 1's.

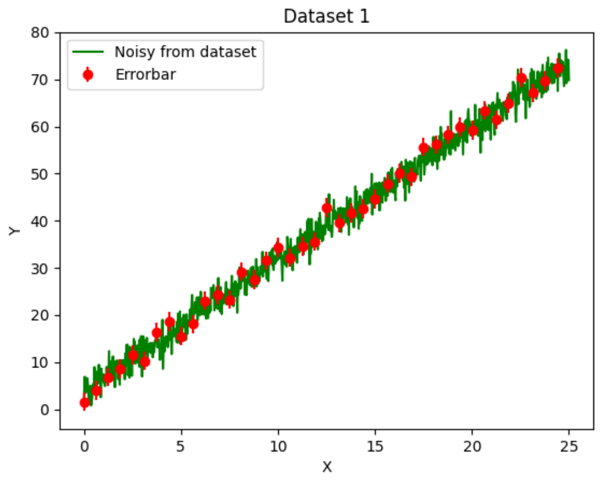

Figure 1: Plot of the original noisy data with error bars

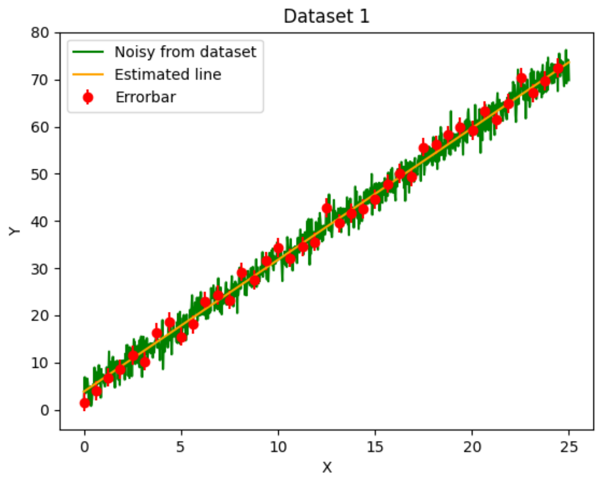

Figure 2: Plot of the estimated line with legend

  - The least square equation looks like **p1 * sin((2 * pi * x)/2.5) + p2 * sin(3(2 * pi * x)/2.5) + p3 * sin(5(2 * pi * x)/2.5) + p4**
  - On solving we obtain estimated values of p1, p2, p3, p4.
- I have defined `sin_superpos` function and calculated y_estimated using the estimated amplitudes and constant.
- With x_data, y_estimated, estimated curve has been plotted. #### Estimation of Periodicity :
- There is a function `np.correlate ( signal, signal, mode = 'full')` under numpy library.
- It calculates the autocorrelation of signal by comparing it to a delayed version of itself at different time lags.
- When this comparison is done, we are effectively looking for the repeating patterns in the signal, it is close to 1 if strong similarity is found, else close to 0, if no similarity is found.
- We check for the time lag where, autocorrelation is close to 1, this time lag(here it is x_data) is nothing but the periodicity of the curve.
- I tried doing this, periodicity was 3. It didn't give me satisfactory results, so I chose to estimate periodicity from the graph. And used periodicity - 2.5 for the rest of the code.

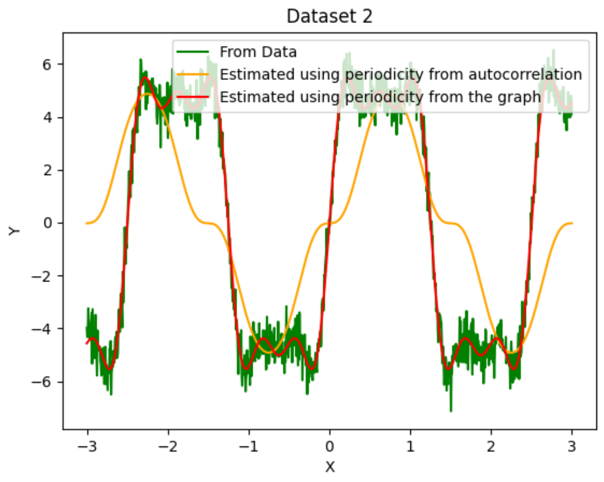

Figure 3: Plot of the curve using least square method

Using Curve Fit :

- from scipy.optimize, I have imported curve_fit.
- I passed the sin_superpos function, x_data and y_data as arguments, and estimated the values of sp1, sp2, sp3, sp4.
- With these values, I found y_curvefit and plotted it.

- Both curvefit and least square method plots overlapped in the graph. Here curvefit was as accurate as leastsquare method.
- It need not be same in other cases, the curvefit works properly, only if proper initial guesses are given, otherwise can give arbitrary results.

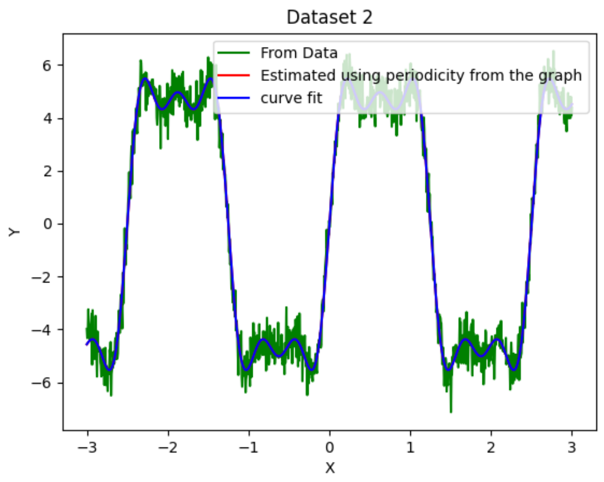

Figure 4: Plot of the curve using curve fit method

Dataset 3

First solution

- I looked up for values of h, c, kB. And defined function `planck_formula`.
- I tried without giving any initial guess for the curve fit, the estimated temperature turned out to be 1. And when it tried using this temperature there was as an overflow(may be because of high/low value in the exponential). The graph was a straight line parallel to x-axis.
- I gave several values and tested like 100,120,140,150,.. as initial guesses. It started working from 140. It returned the estimated value of temperature.
- With this as temperature curvefit is plotted.

Second solution

- This time I defined the function `planck_formula` with 5 arguments that are to be found - f, h, c, kB, T.
- Reference of temperature is taken from before solution. I tried giving exact values of h, c, kB, T. Though the graph fits, the values were higher than the actual values.
- I tried giving close values. I tried adjusting one parameter while keeping all others same, until the reslts were satisfactory. I tried for best approximate

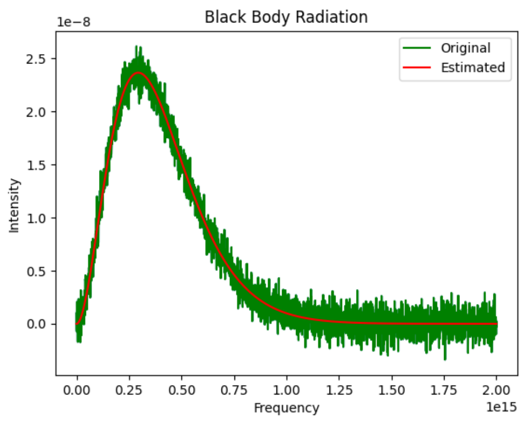

Figure 5: Plot of the curve using curve fit method - estimating temperature values.

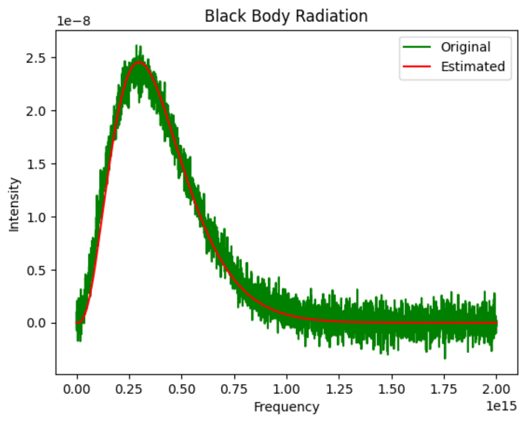

Figure 6: Plot of the curve using curve fit method - estimating all parameters