Assignment 3 - CURVE FITTING by EE22B025

DATASET 1

Approach to the Problem:

- The first thing to do was to read the `dataset1.txt` file into the program. For this I created a readFile() function. This function reads in the data using open() and readlines() functions and then assigns the "first" column of data to the t list and the "second" column of data to the y list. It then converts them to numpy arrays and returns them.
- Then I created a CreateMatrix() function to create the M matrix for which the explanation is given seperately below.
- Then I created a seperate function getCoef() to estimate the slope and the y-intercept. Here I use the `np.linalg.lstsq()` function to find the estimated parameter values.
- Then finally there is a Main() function where all the function calls happen and I finally print out the required estimates.

Graphs Plotted:

- For plotting, I imported the matplotlib.pyplot library as `plt`.
- In the readFile() function, I plot the Original Noisy graph by using `plt.plot(t,y)`
- In the Main() function, I plot another graph, an error bar and then add the legend as well:
  - First, I plot the output with the estimated parameters using `plt.plot(t,y1)` , where y1 is output comupted with estimated slope and intercept.
  - Second, I plot the error bar every 1 in 25 data points using `plt.errorbar()`
  - Finally, I added the legend to the graph using `plt.legend([])`. Here the legend() function takes a list or an array as argument, where the list should contain the string value of the corresponding plot in the graph.
  - Then I use the `plt.savefig()` function to save the plot as a PNG image whenever the code is run.

Construction of Matrix M:

- The M matrix will have dimensions of N x 2, where N is the number of data points given.

- The 1st column of the Matrix will have all the x axis data values (here, t values).
- The 2nd column will only contain 1's because when the M matrix is multiplied with the parameter matrix, all the t values should get added to a constant c (y-intercept) value.
- The Matrix is constructed using column_stack from numpy. Colum_stack basically stacks two arrays side to side, as if each array was a single column.

The Estimated parameters are :

1) m = 2.791124245414918
2) c = 3.848800101430742

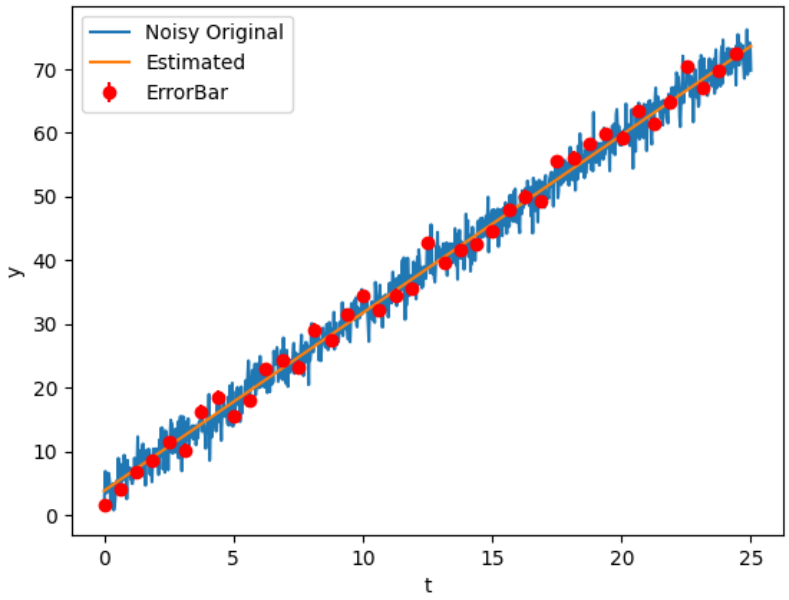

Figure 1: Plot of Noisy and estimated data with `Error Bars` and a `Legend`.

DATASET 2

Approach to the Problem :

- Again, the first thing to do here is to read in the file using open() and readfiles() functions and store the x and y values in seperate arrays.

- Since the given data is of the form of sum of 3 sinusoids, I create a function f1() which takes the most general form of the sum of sum of 3 sine waves with frequencies f , 3f and 5f. The value of f (variable `dum` in the code) = 0.8pi for which explanation is given seperately.
- The parameters are A , B , C which are the coefficients of the 3 individual sine waves and D, which is an extra constant.
- As given in the problem statement, first I try to estimate coefficients using Least Squares method.
- Similar to the Dataset 1, I created CreateMatrix() and GetCoeff() functions, which create the M matrix and get the parameter values respectively.
- The main_F() function has all the function calls in it. It first prints out the estimated parameter values which we got from Least Squares and then it uses the Curve_Fit function to again estimate the same A , B , C , D parameters and then it prints that out as well.

ESTIMATING PERIODICITY OF WAVES :

- The idea behind finding Time Period of the individual waves is that of finding the overall Time Period of the given data and then use LCM to find individual Time Periods.
- We can approximately find the Time Period of the overall function from the graph below. The graph was plotted using `plt.plot(x,y)`, where x is input data and y is output data.

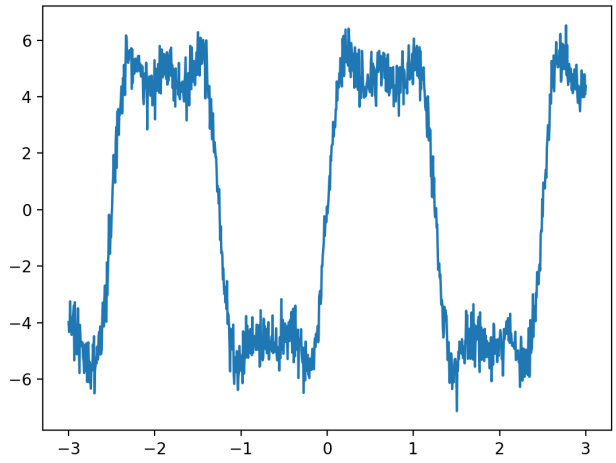

Figure 2: Plot of given data with Noise

- The time period upon inspection approximately comes out to be 2.5

- We know that if there are sum of sinusoids, then the overall Time period is the LCM of the individual Time periods and here it is given that the two of the waves have frequency f and 3f. So I `assumed` that the third wave has period of 5f.
- After computing LCM of the three, we get the value of f = 0.8pi and from that we can get 3f and 5f as well.

Construction of Matrix M :

- The M matrix is constructed in almost a similar way as in Dataset 1.
- Here instead of 2 columns, we have 4 columns because there are 4 parameters.
- The 1st column will have sin(fx) values, where x is the corresponding input data point.
- Similarly the 2nd and 3rd columns will have sin(3fx) and sin(5fx) values respectively.
- The 4th column will only have 1's because the last parameter is just a constant in the function.
- And finally the Matrix is constructed using column_stack from numpy.

Using Curve_Fit :

- As you can see from the answer, the coefficients/parameters A , B , C , D exactly match in both the cases, i.e there is very little difference in the values of the estimated parameters.
- This happens because the parameters are `linear` in the function and thus the curve_fit ends up having almost the same answer as least squares.

The estimated parameters using Least Squares are :

1) A = 6.011152934913331

2) B = 2.0015438200424898

3) C = 0.9802390885802197

4) D = -0.025875188673358248

The estimated parameters using Curve_Fit are :

1) A = 6.011152929571733
2) B = 2.0015438241102568
3) C = 0.9802390876415563
4) D = -0.025875183787915867

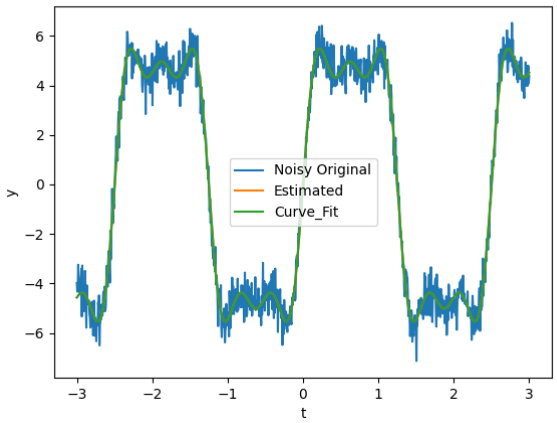

Figure 3: Final plot of DATASET 2 with `Estimated Parameters`.

DATASET 3 - PART 1

Approach to the Problem :

- Read the file in the same way as before, i.e , using open() and readlines() functions.
- Here the function is `NOT` linear, so we cannot estimate the curve using Least Squares method, and instead we need to use Curve_Fit() function offered by `scipy` module.
- In CASE 1, we are given the values of h (Plank's constant) , c(Speed of light) and Kb(Boltzmann's Constant). So the only parameter in the Intensity formula is the `TEMPERATURE`.

Tackling the "Convergence" problem :

- The problem with the given data is that without providing an `initial guess`, the curve_fit function will raise a warning saying "Covariance parameters could not be estimated" and then it doesn't return the proper parameter value.
- This problem happens because the algorithm used by curve_fit is not able to `converge` to a solution.
- To overcome this problem, we supply the curve_fit() function with an extra argument, p0 = initial_guess.
- This initial_guess is an array of the same length as the no. of parameters (1 in this case) which contains an approximate guess of the parameter value.
- I have used an initial guess of T = 6492.227 K. I got this value by taking

one data point from the given data and substituting it in the Black Body equation.
- The data point which I chose is (1.5e15 , 7.53e-10), which, upon substituting in the equation gives me the above T value.

The Estimated value of the Temperature is approximately `4997.34 K`.

DATASET 3 - PART 2

Approach to the Problem :

- The difference between this part and the previous part is that here there are 4 parameters in the Black Body equation, being, T (Temperature) , h (Planck's Constant) , c (Speed of light) , Kb (Boltzmann's Constant).

Finding Initial Guess :

- Again we face the same problem as the previous part, so we need to provide a p0 argument in the curve_fit function.
- This time the p0 argument will be an array/list of length 4, because there are 4 parameters instead of the only 1 in the previous case.
- Here the since we already know the approximate value of the 3 constants (h, c , Kb) , we can use those values for the initial guess.

Finding Temperature for initial guess :

- For temperature we need a little more accurate value this time around because there are multiple parameters in the given function.
- To get a good initial_guess value of Temperature, I tried the method of trial and error.
- I kept putting different values of T into the ini_guess list and kept checking which Temperature value gives the correct parameter estimates of the other 3 constants (h , c , Kb).
- I started at 6492.227K and went down and when I reached aroud 5000K, the estimated parameter values were closely matching the actual values of the constants.
- So I put the initial guess of Temperature as 4990 K.

The estimated values are:-

1) T = 5038.624291865093
2) h = 6.730090884125898e-34
3) c = 302690553.63320047
4) k = 1.3921310257507416e-23

`The estimates can be improved by changing the initial guess of T by using trial and error.`

FINAL GRAPH PLOTS FOR DATASET 3 :

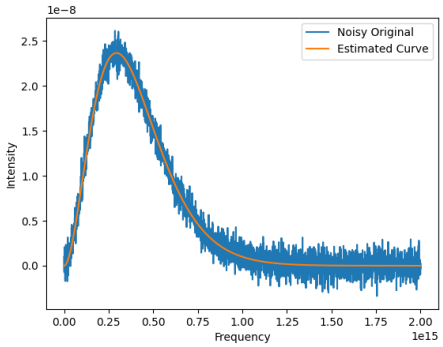

Figure 4: Final plot of the curve with `Estimated Parameters` for `part 1`.

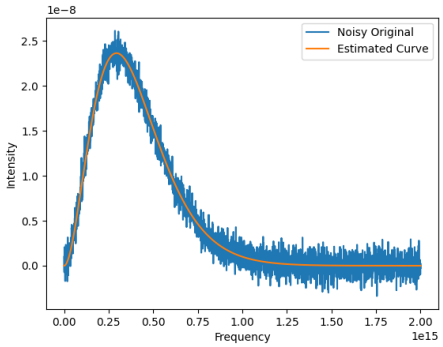

Figure 5: Final plot of the curve with `Estimated Parameters` for `part 2`.