Assignment 3

Sahithi Pallakonda-EE22B080

Dataset 1

The file dataset1.py contains the python code which uses the method of least squares curve fit from numpy and estimates the values of slope and the intercept of the straight line. The file **dataset1.txt** consists of the points that lie on the straight line

This code performs the following tasks:

- It begins by reading data from the file **dataset1.txt** and extracts pairs of x and y coordinates. These coordinates are stored in two separate lists : **X**[ ] and **Y**[ ].
- Next, the code constructs a matrix **matX** of order len(X) $\times$ 2, using the **np.column_stack()** function. The first column of this matrix consists of the x coordinates, and the second column consists of ones.
- After constructing the matrix **matX**, the code proceeds to calculate the slope (**m**) and intercept (**c**) for linear regression model using **np.linalg.lstsq()** function, which takes the matrix **matX** and the y coordinates as arguments.
- Finally, using the estimated values for the slope and intercept (**m** and **c**), the code generates another list of y coordinates. These new y coordinates are computed to match the points on a linear regression line based on the provided dataset. These calculated y coordinates are typically used for plotting the linear regression line on a graph.

Estimated values :

- Slope of the line is 2.791
- Intercept of the line is 3.849

Plot:

The figure below consists of 3 curves: Original, Estimated and the curve with error bars.

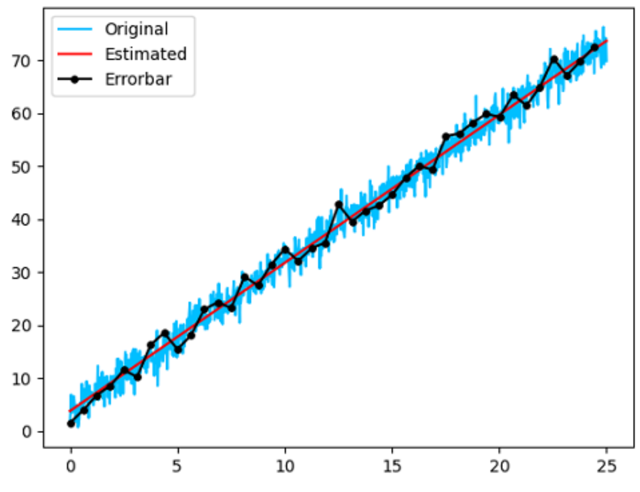


Figure 1: Dataset 1

Dataset 2

The file dataset2.py contains the python code which uses the method of least squares curve fit from numpy and estimates the periodicity of the given curve and the amplitudes of the 3 sine waves involved. The file **dataset2.txt** consists of the points that lie on the superpositioned curve of 3 sine waves.

This code performs the following tasks:

- It begins by reading data from the file **dataset2.txt** and extracts pairs of x and y coordinates. These coordinates are stored in two separate lists : **X**[ ] and **Y**[ ].
- Next, the code constructs an array **disx** which is obtained by extracting 150 x coordinates from the list **X**[ ] at equal intervals. Similarly, it constructs another array **disy**, which consists of the y coordinates corresponding to the x coordinates in the array **disx**.
- Then the code calculates the x coordinates of two points whose y coordinates are around zero, one which is leftmost to the right of y axis and the other rightmost to the left of y axis, the difference between these two x coordinates is stored into the variable (**per**).

Least Squares

- After constructing the matrix **M**, the code proceeds to calculate the Amplitudes for linear regression model using **np.linalg.lstsq()** function, which takes the matrix **M** and the **Y** as arguments.
- Now, the code constructs a list **ls** which contains the sum of the sine waves with different amplitudes for each of the element in list **X**.

Curve_Fit

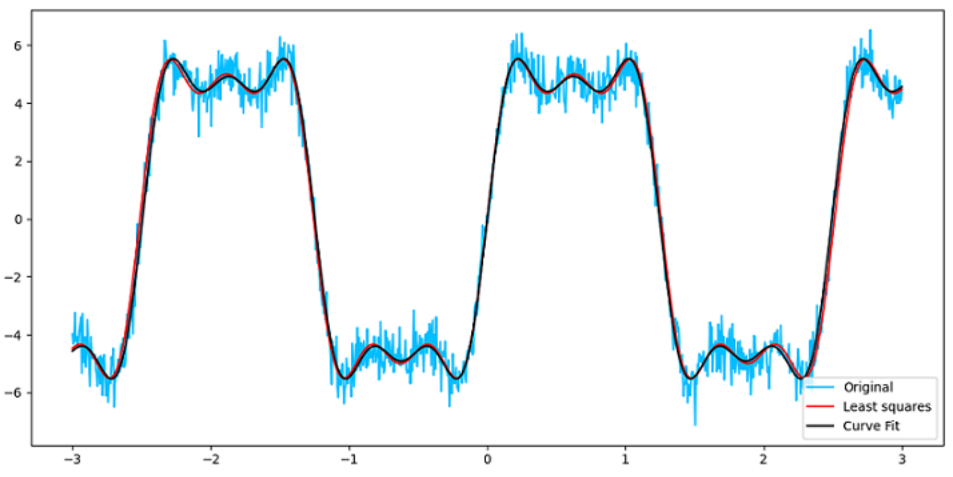
- The function **sup__of__signals** defined in the code takes x coordinate, 3 amplitudes and one of the frequency of one sine wave(so the other two will be 3 times and 5 times of this frequency) as parameters and returns the corresponding y coordinates on the sum of sine waves curve.
- By using trial and error, I determined the range of indices(start index(s) = 370, end index(e) = 649) of x coordinates which almost matches with the original curve.
- Then the code estimates the amplitudes and periodicities of the sine curves using **curve__fit**.
- Then the code constructs a list **cf** consisting of the y coordinates corresponding to x coordinates in the list **X** using the amplitudes and periodicity determined above.

Estimated values :

- Periodicity is 2.513
- Amplitudes using least squares are 6.008, 1.995, 0.99
- Periodicity using curve fit is 2.519
- Amplitudes using curve fit are 6.025, 2.03, 0.923

Plot:

The figure below consists of 3 curves: Original, Least squares and the Curve fit.



Dataset 3

The file dataset3_1.py contains the python code which uses the method of curve_fit to estimate the temperature at which the given data is simulated. The file dataset3_2.py also uses the method of curve_fit to estimate the constants such as planck's constant(h) and boltzmann's constant(k). The file **dataset2.txt** consists of frequencies(**X**) and their corresponding intensities(**Y**). The temperature is denoted by **T**.

Part 1

Let's modify the Intensity equation to

$\frac{1}{T} = \frac{k\log(v)}{hX}$ for $v = 1 + \frac{2hX^3}{c^2Y}$

Since we know the values of **v** for each X and Y, so add these values of **v** into another list **val**[ ] to use in curve_fit.

With the help of modified equation, construct a function which has to be passed into curve fit for estimation of h and k.

```
val = []
for i in range(len(X)):
    v = (2*h*(X[i]**3))/((c**2)*abs(Y[i]))+1
    val.append((k*np.log(v))/(h*X[i]))

def temperature(x,t):
    return 1/t
```

Then we need to find the good approximations for the starting points so that it converges. For the data points in the range of 2000(s) to 2800(e), the curve_fit approximation almost matches with the original curve.

```
temp,_ = curve_fit(temperature,X[s:e],val[s:e])
```

Estimated values

- Temperature(T) is 4994.764 in Kelvin.

Part 2

The goal here is to estimate the values of **h** and **k** by making use of **c** and temperature(estimated in Part 1).

Before proceeding into the algorithm, first we need to give an initial guess of the parameters **h** and **k**, so that the curve_fit can give the good approximations for a certain range of data points.

Similar to the first part, we need to find the good approximations for the starting points, these starting points are not same as in the first part, here the function might converge at some other data points. Therefore, by trial and error, I determined the range of indices (s = 2219, e = 2994) for which the the curve_fit approximation almost matches with the original curve.

```
def h_and_k_est(x,h,k):
    u = (2*h*(x**3))
    v = (np.exp((h*x)/(k*temp))-1)*(speed**2)
    return u/v
var,_ = curve_fit(h_and_k_est,X[s:e],Y[s:e],p0=initial_guess)
```

var is a list of estimated values.

Estimated values

- h = 6.6259×10^{-34}
- k = 1.3808×10^{-23}

Conclusions and improvements

- The estimated values in the second part are bit closer to the actual values, **h** is just deviated by 0.0001 and **k** is deviated by 0.0002.

- Reducing the noise in the dataset might reduce the deviations further.
- If the initial guess is reasonable and as close as possible.

Plot

The figure below consists of three curves: Original, Part 1, Part 2.

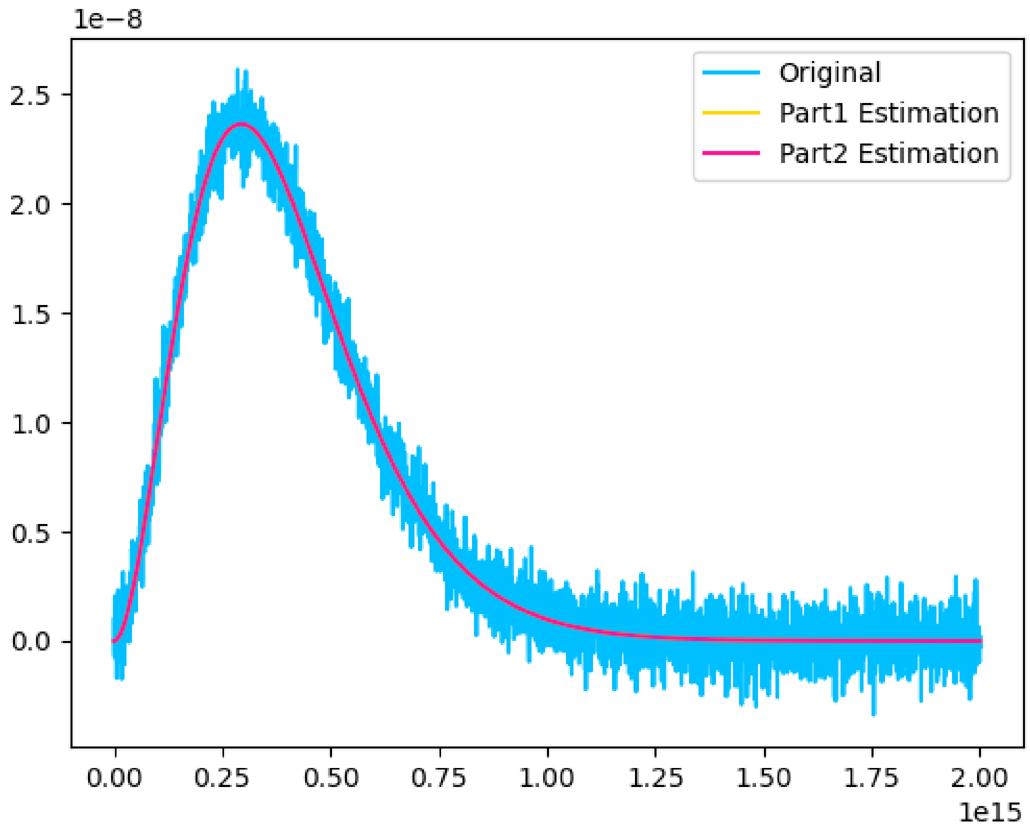


Figure 2: Dataset 3