Assignment 3

V.Manaswini-EE22B065

Import the libraries: numpy, matplotlib.pyplot, scipy.constants and scipy.optimize.

Dataset 1

The `dataset1.py` contains the python script that is used to estimate the slope and intercept of a straight line. The file `dataset1.txt` contains the points that lie on the straight line corrupted with noise. To perform this estimation, python script makes use of least squares curve fit from `numpy`.

Description of the code

- Initially, it extracts the x and y co-ordinates from the file `dataset1.txt` and append them into two different lists `xcord[]` and `ycord[]` respectively.
- **Construction of matrix M**: It is a matrix of order $len(xcord) \times 2$, in which the first column is filled with x coordinates and the second column is filled with ones. This matrix can be formed by using the function `np.column_stack()`.
- **Solving for Slope and Intercept**: Slope(m) and intercept(c) can be estimated by using the function `np.linlag.lstsq()`, it takes the matrix M and y coordinates as its arguments.
- Using the estimated values, construct another list of y co-ordinates for plotting.

Estimated Values

Slope of the line: 2.791124245414918

Intercept of the line: 3.848800101430742

Plotting

Three curves have been plotted in the given figure below: Original, Estimated, and the plot with error bars.

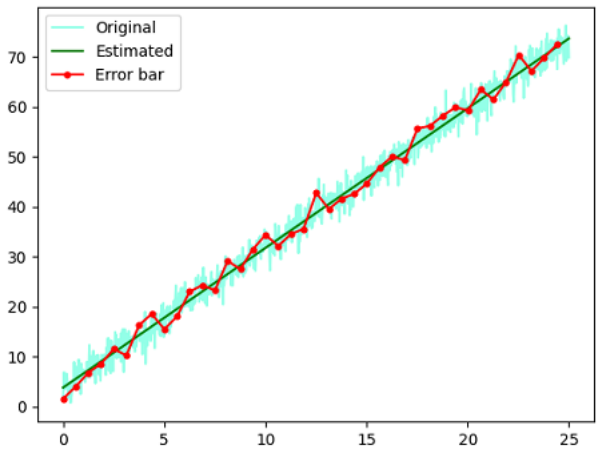

Figure 1: Dataset 1

Dataset 2

The file `dataset2.py` contains the python script that is used to estimate the amplitudes of three different sine waves that have been added together to form a signal. Let us consider the output of the signal is represented by `y(x)` and the input is represented by `x`. These `x` and `y(x)` will be extracted from the file `dataset2.txt`.

Description of the code

1. **Estimation of Fundamental Period**:

- The given data is simulated in between the points (-3,-3.98) and (3,4.37). It is observed that there are two crests and two troughs while plotting the curve with the given set of data points.
- Since the curve started with the negative `y`, it definitely crosses the x axis at some point. First, we should be able to find the point which is just after -3 where the value of `y` would become zero. Let's say the x coordinate of this point to be x[l[0]].
- Since the curve ends with positive y, the objective is now to estimate the point before 3 where the value of `y` will become zero. Let's say the x coordinate of this point to be x[l[1]].
- As I mentioned there are two crests and two troughs from the observation, so one crest will be on left side of y axis, another crest will be on right side of y axis. Similarly for the troughs.
- Periodicity is defined as the distance between two consecutive crests or troughs, therefore, the periodicty(T) would become $\frac{x[l[1]]-x[l[0]]}{2}$.
- Estimated value of periodicity is 2.4984984984984986.

2. **Estimation of amplitudes**:

- Assuming the frequencies of the individual sine waves are in the ratio 1:3:5, hence these frequencies will become $f_1 = \frac{2\pi}{T}, f_2 = \frac{6\pi}{T}, f_3 = \frac{10\pi}{T}$ and also their corresponding amplitues to be a_1, a_2, a_3.
- Estimation of amplitudes can be done with the help of least squares curve fitting,.
- **Construction of matrix M**: It is a matrix of the order $len(x) \times 3$, in which the first column is filled $\sin(f_1 x_i)$, second column with $\sin(f_2 x_i)$, third column with $\sin(f_3 x_i)$ for $i \in [0, len(x))$.
- **Solving for the Amplitudes**: Amplitudes corresponding to the frequencies f_1, f_2, f_3 could be generated using the function `np.linlag.lstsq()`, it takes `M` and `y(x)` as its arguments.
- Create another list of `expy` for plotting `y(x)` by making use of the values estimated by least squares.

Estimated Values using Least Squares

Amplitudes for the frequencies f_1, f_2, f_3 are 6.011206381448963, 2.0016993031105454, 0.978247434262425 respectively.

- **Estimation of amplitudes using `curve_fit`**: Since `curve_fit` couldn't able to estimate the required values for the whole set of data points, so we will try to figure out the starting and ending points where `curve_fit` gives good approximation. These starting and ending points were observed to be 390 and 658 respectively, these estimations were done by using trial and error method.

```
def signal(x,pc,ac1,ac2,ac3):
    value = ac1*np.sin(pc*x)+ac2*np.sin(pc*3*x)+ac3*np.sin(pc*5*x)
    return value
```

The function above is used to estimate the frequency and amplitudes.

Estimated Values using Curve Fit

Fundamental Period($\frac{2\pi}{pc}$): 2.5122451513521957

Amplitudes: 6.011139227172248, 2.030908968107392, 0.9480655349320031

Plotting

Three curves have been plotted in the given figure below: Original, Using Least squares, Using Curve fit.

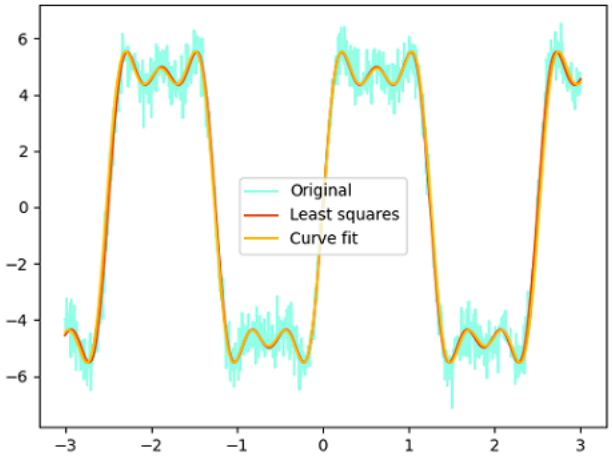

Figure 2: Dataset 2

Dataset 3

Part 1

The file `dataset3_1.py` contains the python script that is used to estimate the temperature at which `dataset3.txt` was simulated. Let us consider the frequency of radiation is represented by `x` and the intensity of radiation is represented by `y`. `x` and `y` will be extracted from the file `dataset3.txt`.

Description of the code:

The values of Planck's constant(h), Boltzmann constant(k), Speed of the light(c) are assumed to be known.

Equation for Temperature: $T = \frac{hx}{k_B \log(s+1)}$ where $s = \frac{2hx^3}{yc^2}$

Estimation of Temperature:

Using the equation for `s`, a new list(v1) is created to append all the values of `s` for the points in `x`. Now, the function defined below is used in `curve_fit` to estimate the temperature.

```
def temp(x,t):
    return ((h*x)/(k*t))
T,_ = curve_fit(temp,x[start:end],v1[start:end])
```

- Since the curve fit couldn't able to give the good approximation for the whole set of data points, so we'll try to figure out the starting and ending points of `x` and `y`. This can be done by trial and error method, and it is observed to be 1900 and 2600 respectively. Data points in the range of 1900 to 2600 represent the portion of the curve nearer to maximum intensity.

Estimated Temperature: T = 4998.813308041924K

Part 2

In this part, we'll try to estimate the values of Planck's constant and Boltzmann's constant using the same function `curve_fit`.

Description of the code:

Here, the values of speed of light and temperature at which data is simulated are assumed to be known.

The function below is used in `curve_fit` to estimate the required valus.

```
def handk(x,h,k):
    n = (2*h*(x**3))
    d = (np.exp((h*x)/(k*T))-1)*(c**2)
    return n/d
```

Similar to the first part, the `curve_fit` takes certain range of values where it can produce good approximation. It is observed that 2200 and 2700 are the good estimates for starting and ending points. This estimation is purely trial and error.

- In this part, we give an initial guess of h and k as one of the arguments to `curve_fit`. Giving an initial guess is important, since the curve fit cannot estimate the required values for a non linear curve.

```
est,_ = curve_fit(handk,x[start:end],y[start:end],p0 = [constants.h, constants.k])
```

The first element of `est` would give Planck's constant and the second element gives the value of Boltzmann's constant.

Estimated Values:

Planck's constant(h) = $6.625088971556951 \times 10^{-34}$

Boltzmann's constant(k) = $1.3796726208859357 \times 10^{-23}$

Plotting

The plot for the first and second part are shown in the figure below.

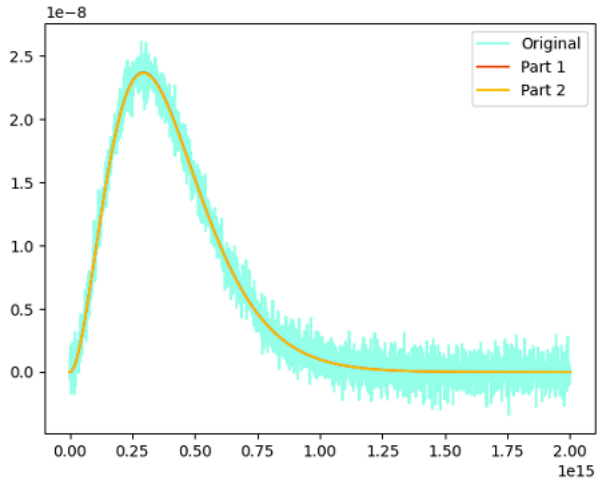

Figure 3: Dataset 2