Curve Fitting

M.Saketh Sai Ram - EE22B022

Import necessary libraries such as `numpy, matplotlib.pyplot, and curve_fit` from `scipy.optimize` for each dataset. The function `collect` is used in every Python file to extract the data points and would return X, and Y coordinates in the form of lists.

1 Dataset-1

Let n be the number of data points in the file `dataset1.txt`.

Since it has already been mentioned that these points form a straight line with noise. So, we can estimate the nearest values of slope and intercept using least squares curve fitting. To perform Least Square Regression, we need a matrix `M` of order $F^{n\times 2}$ and another matrix `v` of order $F^{2\times 1}$ which consists of variables we need to estimate.

1.1 Construction of Matrix M

The function `matrixM` in the code `dataset1.py` generates the matrix M using the function `column_stack`. The entries in the first column are filled with the X-coordinates and the second row with ones. The code that forms the matrix M is given below:

```
def matrixM(x):
     return np.column_stack([x,np.ones(len(x))])
```

$$\mathrm{M} = \begin{pmatrix} x_0 & 1 \\ x1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_{n-1} & 1 \end{pmatrix}$$

1.2 Matrix Equation

The matrix equation will take the form $Mv = Y$, where Y is a list of all Y-coordinates from the given data set. Below is the code to estimate the slope and intercept of a line:

```
def l_sq_c_fit(M,y):
    (p1, p2), _, _, _ = np.linalg.lstsq(M, y, rcond=None)
    return p1,p2
```

The function `l_sq _c _fit` returns the estimated values of Slope and Intercept.

1.3 Estimated Slope and Intercept

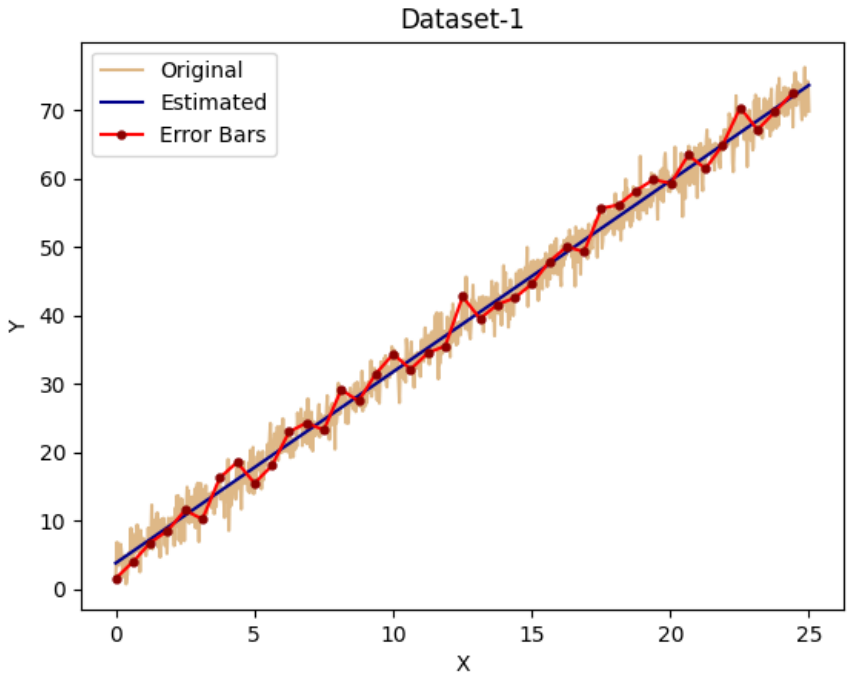
The slope of the Straight Line is 2.791124245414918.
The intercept of the Straight line is 3.8488001014307436.
The Approximate equation of the straight line would be $Y = 2.79X + 3.85$.

1.4 Plotting the Curves

The plot below consists of three curves:

- Original Curve
- Estimated Curve using Least Squares
- Estimated Curve with Error Bars

The Error bars are plotted for one point in every 25 data points.



2 Dataset-2

This is a signal consisting of three sine waves that have been added up and corrupted with noise. Let the output of the signal be $Y(t)$, will take the form $Asin(p_1t)+Bsin(p_2t)+Csin(p_3t)$. Assuming the frequencies are in the ratio 1:3:5, therefore, $p_1 = \frac{p_2}{3} = \frac{p_3}{5}$; A,B,C are the respective amplitudes of the individual signals. All the functions mentioned in the below content are from the Python script `dataset2.py`.

2.1 Estimation of Periodicity of the Signal

For a sinusoidal signal the periodicity can be estimated as:

- Find all possible times at which the output of the signal would become zero. To do this, draw a horizontal line L parallel to the time axis and find the intersection points of this straight with Y(t).
- The Difference between the two adjacent times will give half of the Time period, so all the differences in the times where Y(t) = 0 were noted.
- With the given data set, it is observed that there are five different times at which the line L intersects Y(t). Let these five times be t_0, t_1, t_2, t_3, t_4. From these values the periodicity can be estimated as Period(T) = $\frac{((t_1-t_0)+(t_2-t_1)+(t_3-t_2)+(t_4-t_3))}{2}$

2.1.1 Estimation of t_0, t_1, t_2, t_3, t_4

The function `interp_xfy(x,y)` would return the values of t_0, t_1, t_2, t_3, t_4. This function makes use of two more functions `inc_interp(x,xp,fp)` and `dec_interp(x,xp,fp)` to interpret the time at which Y(t) will become zero from the given data set. It is observed that the data points lie in the range of -3 and 3. Assume the reference time to be -3, let the signal's output change from negative to positive for the first time after -3 after p_0 data points, and again it changes from positive to negative after p_1 data points from the beginning. Similarly, it also finds the points p_2,p_3 and p_4. **Note:** `x,xp,fp = [0],Y(t)[`$p_i : p_{i+1}$`],t[`$p_i : p_{i+1}$`]`.

By estimation, the values of p_0,p_1,p_2,p_3 and p_4 are found to be 84, 292, 500, 711 and 917 respectively, these are evaluated relative to the reference point.

- Extract the data points in the range of 0 to $p_1 + 1$ and by making use of `inc_interp(x,xp,fp)` function, estimate the value of t_0.
- Similarly, extract the data points in the range of p_1+1 to p_2+1 and by making use of `dec_interp(x,xp,fp)` function, estimate the value of t_1.
- It performs the same algorithm until the end of the data set and returns all the values of t_0, t_1, t_2, t_3, t_4.

`inc_interp(x,xp,fp)` interpolates the required value for an increasing curve, `dec_interp(x,xp,fp)` interpolates the required value for a decreasing curve.

2.1.2 Estimated Periodicity

Periodicity = 2.5153939029247683

2.2 Least Squares Regression

Since we found out the periodicity of the signal, the next goal is to estimate the amplitudes of individual signals.

Let the frequencies of these individual signals be $\frac{2\pi}{T}, \frac{6\pi}{T} and \frac{10\pi}{T}$.

2.2.1 Construction of matrix M

Matrix M can be constructed using `np.column_stack()`. It contains three columns, the first column is filled with $\sin(\frac{2\pi t_i}{T})$, the second column with $\sin(\frac{6\pi t_i}{T})$ and the third column with $\sin(\frac{10\pi t_i}{T})$, where $(t_i, Y(t_i))$ represents i^{th} point in the data set for $i \in [0, n = No. of data points)$.

$$\mathrm{M} = \begin{pmatrix} \sin(\frac{2\pi t_0}{T}) & \sin(\frac{6\pi t_0}{T}) & \sin(\frac{10\pi t_0}{T}) \\ \sin(\frac{2\pi t_1}{T}) & \sin(\frac{6\pi t_1}{T}) & \sin(\frac{10\pi t_1}{T}) \\ \vdots & \vdots & \vdots \\ \sin(\frac{2\pi t_{n-1}}{T}) & \sin(\frac{6\pi t_{n-1}}{T}) & \sin(\frac{10\pi t_{n-1}}{T}) \end{pmatrix}$$

2.2.2 Estimation of Amplitudes

The function `amp(x,y,p)` would return the amplitudes of the individual signals. Let the amplitudes of the individual signals corresponding to the frequencies $\sin(\frac{2\pi t_i}{T})$, $\sin(\frac{6\pi t_i}{T})$, $\sin(\frac{10\pi t_i}{T})$ are a_1, a_2, a_3.

Basically, the function `np.linalg.lstsq()` collects the data points $Y(t_i)$ and the matrix M and then it solves for a_1, a_2, a_3 based on the equation $Ma = Y(t)$.

$$\begin{pmatrix} \sin(\frac{2\pi t_0}{T}) & \sin(\frac{6\pi t_0}{T}) & \sin(\frac{10\pi t_0}{T}) \\ \sin(\frac{2\pi t_1}{T}) & \sin(\frac{6\pi t_1}{T}) & \sin(\frac{10\pi t_1}{T}) \\ \vdots & \vdots & \vdots \\ \sin(\frac{2\pi t_{n-1}}{T}) & \sin(\frac{6\pi t_{n-1}}{T}) & \sin(\frac{10\pi t_{n-1}}{T}) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} Y(t_0) \\ Y(t_1) \\ \vdots \\ Y(t_{n-1}) \end{pmatrix}$$

2.2.3 Estimated Amplitudes

The amplitudes of the signals corresponding to the frequencies $\frac{2\pi}{T}$, $\frac{6\pi}{T} and \frac{10\pi}{T}$ are 6.011, 2.002 and 0.977 respectively.

2.3 curve_fit Estimation

Here, `curve_fit` is used to estimate the periodicity and amplitudes of individual signals.

```
def sinfunc(x,p1,a1,a2,a3):
    return a1*np.sin(p1*x)+a2*np.sin(3*p1*x)+a3*np.sin(5*p1*x)
```

The above function is used in `curve_fit` to estimate the required values. Since `curve_fit` couldn't able to estimate the required values for all range of data points, therefore, by trial and error method, the starting and end points were found to be 471 and 650. $p1$ is the frequency of the signal with amplitude a1.

2.3.1 Estimated Periodicity and Amplitudes

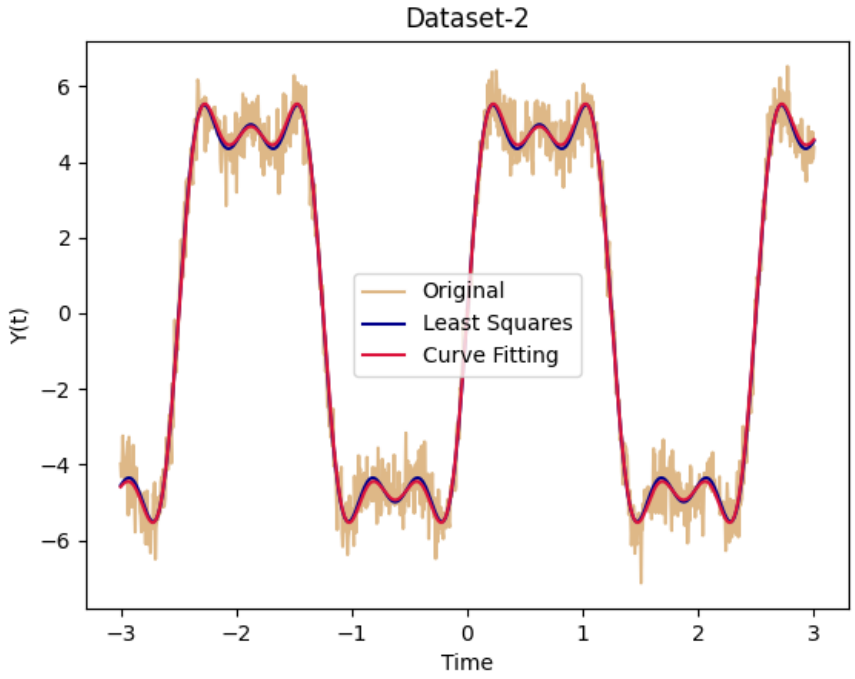
Periodicity($\frac{2\pi}{p1}$) = 2.4965819360108696
Amplitudes = 6.06, 2.024, 0.899 respectively.

2.3.2 Conclusion

There is no major difference between the values estimated using least squares and curve_fit. Since I did the trial and error method to get the correct approximation of the curve, there won't be any major changes to the periodicity and amplitudes.

2.4 Plotting the Curves

The plot below contains three curves, Original, Estimated curves using least squares, and curve_fit.



3 Dataset-3

The `dataset3.txt` corresponds to a simulation of the intensity of Black Body radiation emitted by an object at some temperature.

3.1 First Part

Here, the values of Planck's constant(h), Boltzmann's constant(k_b), and speed of the light(c) were assumed to be known. The goal here is to estimate the temperature at which the data is simulated. This can be done by using the function `curve_fit` with appropriate boundaries. f denotes the frequency of radiation and B denotes the corresponding intensity.

3.1.1 Rewriting the Intensity Equation

$$\frac{hf}{k_B T} = \ln\left(1 + \frac{2hf^3}{Bc^2}\right)$$

The value inside the logarithm can be computed by using the given data set and these values were added to a new list, say B1.

3.1.2 Estimation of Temperature

The function `esT(f,B)` would return the Temperature at which the data is simulated.

Arguments to be passed into 'estT' function:

- Modified list of frequency(fmod)
- Modified list of Intensity(Bmod)

`fmod` and `Bmod` are lists of points with appropriate starting and ending points, these starting and ending points can be found by using the trial and error method. It is found that 1655 and 2944 are the rough estimates of starting and ending points.

```
def estT(f,B):
    B1 = []
    for i in range(len(B)):
        b = (2*h*(f[i]**3))/(c*c*B[i])
        B1.append(np.log(b+1))
    def func(t,T):
        return (h*t)/(kb*T)
    T,pcov = curve_fit(func,f,B1)
    return T[0]
```

3.1.3 Estimated Temperature

The temperature at which the data is simulated is 4970.775990058394K.

3.2 Second Part

Here, the values of the speed of light and temperature(estimated in the first part) are assumed to be known. The goal here is to estimate Planck's constant and Boltzmann's constant using the function `curve_fit` with appropriate boundaries.

3.2.1 Estimation of h and k_b

The function `esthk(f,B)` would return the optimized values of h and k_b.

Similar to the first part, the function takes the modified frequencies and Intensity lists. But the starting and ending points will be different. These boundaries could be determined by using the trial and error method. It is observed that 2201 and 2900 are the rough estimates of starting and ending points. Here, an initial estimate of the h and k_b values is given as one of the arguments in `curve_fit`.

```
def esthk(f,B):
    def func(f,h,k):
        num = (2*h*(f**3))/(c**2)
        den = np.exp((h*f)/(k*T))-1
        return num/den
    h,k = curve_fit(func,f,B,p0=[6.62607015e-34, 1.380649e-23])
    return h,k
```

3.2.2 Estimated Values

Planck's constant: $6.621112329994696 \times 10^{-34}$
Boltzmann's constant: $1.3868321853629788 \times 10^{-23}$

3.2.3 Comments

The quality of the results will depend on the accuracy of the data and the precision of the estimation method. Convergence may require multiple iterations and fine-tuning of initial estimates. The estimated values in the second part are quite close to the actual values, these values were deviated by 0.005 approximately from the actual values.

3.2.4 To improve the estimates

There are a few chances where we could get the estimates more accurately:

1. The initial guess is reasonable and as close as possible to the actual values.
2. Ensure the data is accurate and collected under controlled conditions. High-quality data is crucial for accurate parameter estimation.
3. Reducing the noise.

3.3 Plots for Part1 and Part2