Assignment 3 - Curve Fitting Report

Dataset 1

EE22B175

Overview

The python file `dataset1.py` takes in a dataset that represents linearly related data, which might have noise, and fits the data to a linear model by solving it using Least Squares approach. It also plots the modelled linear equation along side the data with error bars.

Least Squares Calculation

Least squares fitting is a statistical method for finding the best fit of a model to data. The goal of least squares fitting is to minimize the sum of the squared residuals, where the residuals are the differences between the actual and predicted y values.

In this example, we are using least squares fitting to find the best fit of a linear model to the data.

$$y = mx + c$$

Constructing the M Matrix

Since the data corresponds to a linear relation between the given values, from the above equation, the M matrix has to have 2 columns, one for the values of the independent variable and the other for values of the constant, in this case `x` and `c` respectively.

The first column corresponding to the independent variable is just the values of the independent variable itself. The second column, since it corresponds to a constant, is just a column of 1s. Hence, the M Matrix becomes, if the values of the independent variable go from x_1 to x_n,

$$\mathbf{M} \equiv \begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}$$

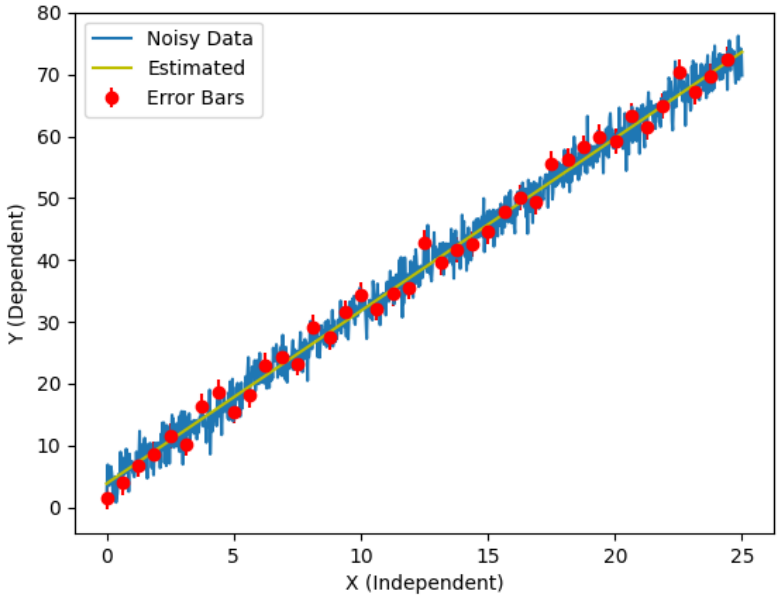
This M matrix multiplied by the `p` matrix results in the `g` matrix which is a column matrix of the observations, in other words the values of the independent variable. Solving for the `p` matrix gives the coefficients of the independent variable and the constant.

Results and Plots

Fitting the given data in `dataset1.txt` to a linear model using Least Squares resulted in the following estimated equation, assuming `x` is the independent variable and `y` is the dependent variable.

$$y = 2.791124245414918x + 3.8488001014307436$$

Error bars are plotted on the given noisy data for 1 in every 25 points and the estimated line is overlaid on it. The length of the error bars is the standard deviation of the calculated residuals (i.e., the difference between the actual and predicted y values). The estimated line is in Yellow, the noisy data is in Blue, and the error bars are in Red. This also has the legend of the plot



Dataset 2

Overview

The python file `dataset2.py` takes in a dataset that represents data of superposition of 3 Sine Waves of frequencies `f, 3f, 5f`, which might have noise, finds the time period of the given data and fits the data to a sinusoidal model by solving it using Least Squares approach. It also plots the modelled sinusoidal equation along side the data. The same modelling is also done using `scipy.optimize.curve_fit()` which is also subsequently plotted.

Finding the Period of the Data

The `period` variable is initialized to 0. This will be the period of the signal. The spacing between the data points is calculated. The `difference` variable is initialized to 1. This will be used to track the smallest difference between two Y values that are more than 100 data points apart. Iterating over each pair of Y values, if the difference between them is less than the `difference` variable and if the two Y values are more than 100 data points apart, the difference between the current Y value and the next Y value. The `period` variable is also updated to the number of data points between the current Y value and the next Y value. the `period` variable is multiplied by the `spacing` variable to get the Time Period of the signal in the original units.

The 100 units apart restriction was imposed to avoid plateaus giving the time period to be very small, and also so that the random chance of two close values to be equal because added noise is also taken care of.

Least Squares Calculation

Least squares fitting is a statistical method for finding the best fit of a model to data. The goal of least squares fitting is to minimize the sum of the squared residuals, where the residuals are the differences between the actual and predicted y values.

In this example, we are using least squares fitting to find the best fit of a sinusoidal model to the data.

$$y = a\sin(fx) + b\sin(3fx) + c\sin(5fx)$$

Constructing the M Matrix

Since the data corresponds to a sinusoidal relation between the given values, from the above equation, the M matrix has to have 3 columns, one each for the values of the $\sin(fx)$, $\sin(3fx)$, and $\sin(5fx)$.

The first column corresponding to the values of $\sin(fx)$, the second $\sin(3fx)$, and the third $\sin(5fx)$. Hence, the M Matrix becomes, if the values of the independent variable go from x_1 to x_n,

$$\mathbf{M} \equiv \begin{pmatrix} \sin(fx_1) & \sin(3fx_1) & \sin(5fx_1) \\ \sin(fx_2) & \sin(3fx_2) & \sin(5fx_2) \\ \vdots & \vdots & \vdots \\ \sin(fx_n) & \sin(3fx_n) & \sin(5fx_n) \end{pmatrix}$$

This M matrix multiplied by the `p` matrix results in the `g` matrix which is a column matrix of the observations, in other words the values of the dependent variable. Solving for the `p` matrix gives the coefficients of $\sin(fx)$, $\sin(3fx)$, and $\sin(5fx)$.

Results and Plots

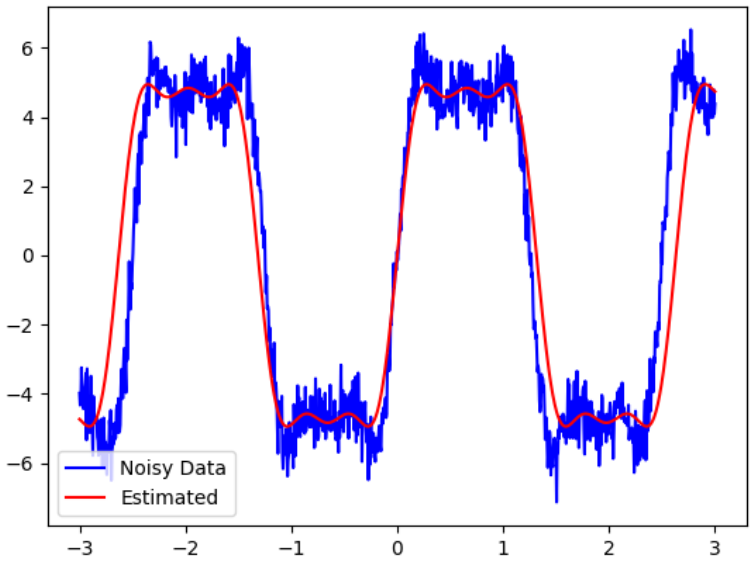
Fitting the given data in `dataset2.txt` to a sinusoidal model made up of 3 Sine waves of frequencies f, 3f and 5f using Least Squares resulted in the following estimated equation, assuming `x` is the independent variable and `y` is the dependent variable.

$$y = 5.85283087692017\sin(fx) + 1.5729264941184493\sin(3fx) + 0.557176177370758\sin(5fx)$$

Where,

$$f = 2.388471127044260$$

The estimated line is in Red, the noisy data is in Blue. This also has the legend of the plot



Using `curve_fit()`

The `scipy.optimize` library contains the `curve_fit` function that can perform a non-linear curve fitting on given data. Unlike the least squares method, a parametrized function must be given that can be used to estimate the parameters. The function `equation()` had to be modified to be parameterized and hence `equationUnknown()` was created using `numpy.sin()` instead of `math.sin()` which is not vectorizable. No starting guesses were given to the function.

The function outputs a tuple containing the values of the coefficients of the three Sine terms and it also outputs a 2-D covariances array.

Results and Plots

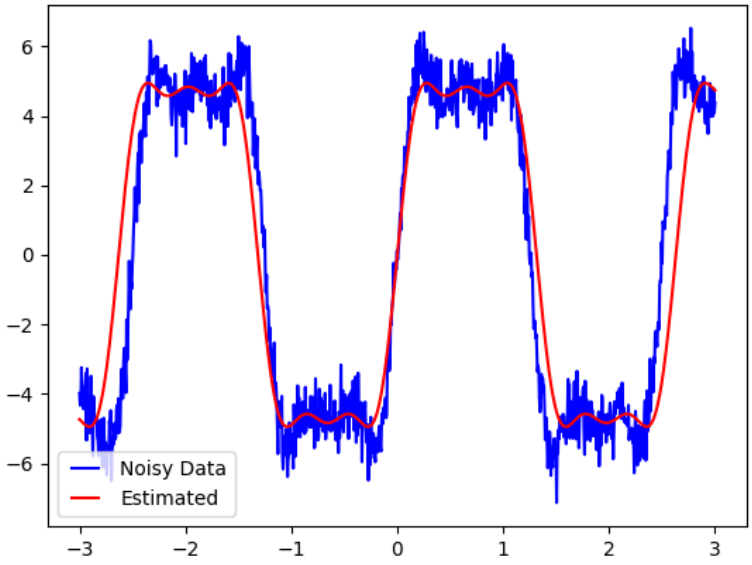
Fitting the given data in `dataset2.txt` to a sinusoidal model made up of 3 Sine waves of frequencies f, 3f and 5f using Curve Fit resulted in the following estimated equation, assuming `x` is the independent variable and `y` is the dependent variable.

$$y = 5.8528308759125816 \sin(fx) + 1.5729264947599473 \sin(3fx) + 0.5571761771317888 \sin(5fx)$$

Where,

$$f = 2.388471127044260$$

The estimated line is in Red, the noisy data is in Blue. This also has the legend of the plot



Comparing Both

Each of the methods produced estimates within 10^{-8} of the other, and it can be concluded that the difference between the two methods of estimation in this case is practically non-existent.

Dataset 3 First Solution

Overview

This code, `dataset3_1.py` solves a least squares problem to fit a blackbody radiation formula to data. This code uses `sci_py.optimize.curve_fit` to solve the problem and estimate the Temperature at which the Black Body Radiances were calculated. The blackbody radiation formula is a physical model that describes the spectrum of electromagnetic radiation emitted by a black body. The formula for the radiance is given by

$$B(f,T) = \frac{2hf^3}{c^2} \frac{1}{e^{\frac{hf}{k_B T}} - 1}$$

where h is Planck's constant, k_B is Boltzmann's constant, and c is the speed of light in vacuum.
Here, the values of the constants h, c, k_b are assumed to be known.

Since the relation between T and B is not linear, Least Squares method cannot be used, and `curve_fit()` has to be used.

Using `curve_fit()`

The `scipy.optimize` library contains the `curve_fit` function that can perform a non-linear curve fitting on given data. Unlike the least squares method, a parametrized function must be given that can be used to estimate the parameters. The function `planck()` has to be parameterized and was created using `numpy.exp` instead of `pow(e, exponent)` which is not vectorizable.

Starting Guesses and Constants

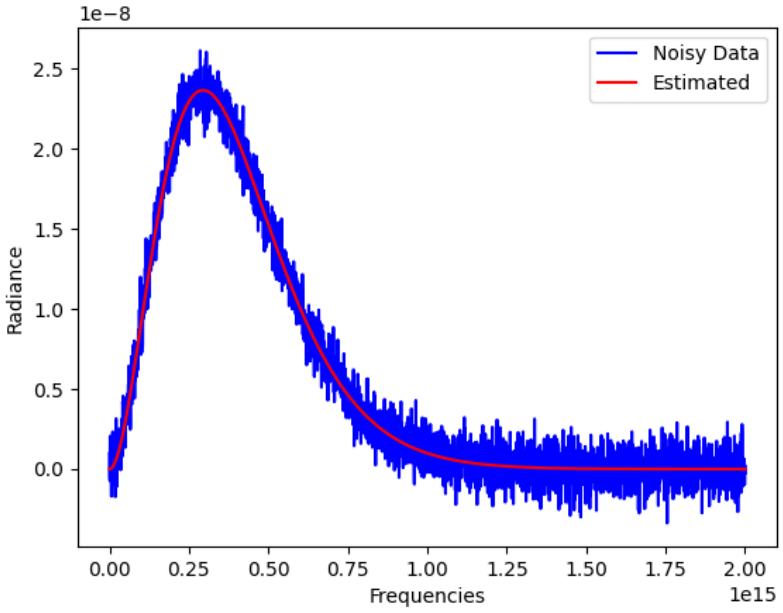
The starting guesses for the curve fitting algorithm were taken by generating a list of 10 equally spaced values between 1000 and 10000. This range was chosen because it is reasonable to assume that the temperature of the black body is within this range. The estimated T was calculated using the first guess and if any other subsequent guess had a lower covariance, the estimate generated using it was considered. The `try - except` part takes care of starting values which may result in divergence by catching a `RuntimeError` and letting it pass through.

Results and Plots

Fitting the given data in `dataset3.txt` using Curve Fit resulted in the following estimated Temperature at which the Black Body Radiances were calculated.

$$T = 4997.30254684K$$

The estimated line is in Red, the noisy data is in Blue. This also has the legend of the plot



Dataset 3 Second Solution

Overview

This code, `dataset3_2.py` solves a least squares problem to fit a blackbody radiation formula to data. This code uses `sci_py.optimize.curve_fit` to solve the problem and estimate the Temperature at which the Black Body Radiances were calculated. The blackbody radiation formula is a physical model that describes the spectrum of electromagnetic radiation emitted by a black body. The formula for the radiance is given by

$$B(f,T) = \frac{2hf^3}{c^2}\frac{1}{e^{\frac{hf}{k_BT}}-1}$$

where h is Planck's constant, k_B is Boltzmann's constant, and c is the speed of light in vacuum.
Here, the values of the constants h, c, k_B are assumed to **not** be known unlike the first solution. Thus this code estimates the values of h, c, k_B, and T by fitting the data using `curve_fit()` .

Using `curve_fit()`

The `scipy.optimize` library contains the `curve_fit` function that can perform a non-linear curve fitting on given data. Unlike the least squares method, a parametrized function must be given that can be used to estimate the parameters. The function `planck()` has to be parameterized and was created using `numpy.exp` instead of `pow(e, exponent)` which is not vectorizable.

Starting Guesses and Constants

The starting guesses for the curve fitting algorithm were taken to be very close to the actual value for the constants as the estimates of numbers of such small scale, 10^-34 and 10^-23 in the case of h, and k_B, and of large scale like 10^8 for c can be changed drastically with slight variations in the starting values. Hence, the staring values taken are

$$h = 6.62 \times 10^{-34},\ k_B = 1.4 \times 10^{-23},\ c = 2.99 \times 10^8$$

The starting guess of the Temperature is taken to be $4930K$, and it has been chosen purely by trial and error.

Results and Plots

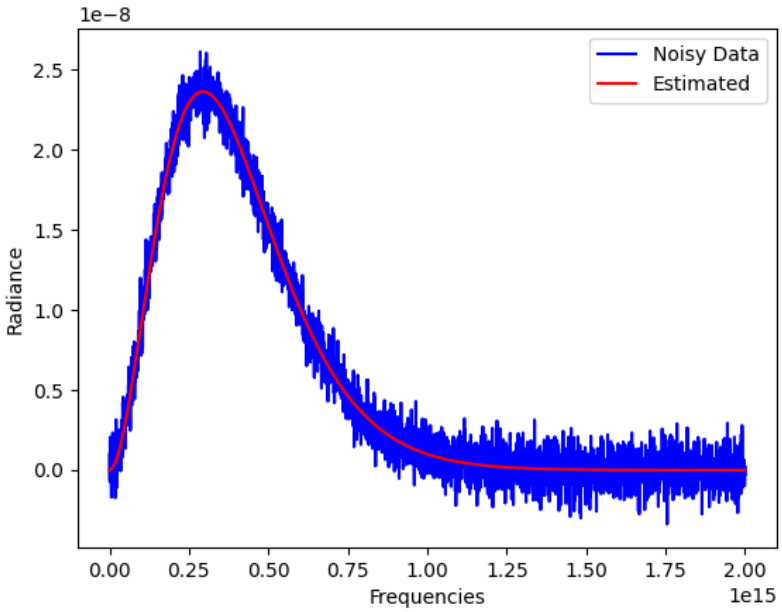
Fitting the given data in `dataset3.txt` using Curve Fit resulted in the following estimated Temperature at which the Black Body Radiances were calculated.

$$T = 4878.921583800454K$$

The constants have been estimated to be

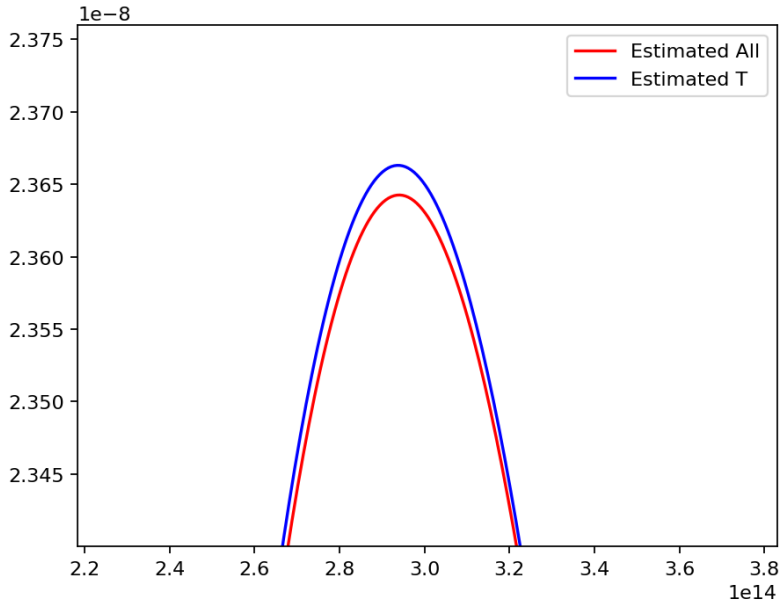
$$h = 6.4857102923871645 \times 10^{-34},\ c = 2.971441688900885 \times 10^8,\ kB = 1.3854948920261862 \times 10^{-23}$$

The estimated line is in Red, the noisy data is in Blue. This also has the legend of the plot



Comparing Estimates and Improvements

The constants estimated were close to the actual values of the constants, but not quite there, so was the estimated Temperature. The values estimated were nonetheless accurate to a usable degree. Below is a zoomed in plot of the radiance calculated both ways, based on solution 1 and 2. The Solution2 line is in Red, the Solution1 line is in Blue. This also has the legend of the plot



This plot illustrates that the radiances calculated are not far off.

The estimates can be improved in the following ways

- One way to improve the estimates would be to use more data points. The more data points that are used, the more accurate the estimates will be.
- The starting points can be fine-tuned even further to get better estimates
- Another way to improve the estimates would be to use a different curve fitting algorithm. There are many different curve fitting algorithms available, and some algorithms may be more accurate than others for certain types of data.
- The estimates could be improved by using a more sophisticated model. The blackbody radiation formula is a simple model, and there are more sophisticated models that can be used to describe the spectrum of electromagnetic radiation emitted by real objects.
- - **Use Bayesian methods.** Bayesian methods are a powerful way to approach curve fitting. Bayesian methods allow you to incorporate prior knowledge about the parameters of the model into the estimation process. This can lead to more accurate estimates, especially when the data is limited.