#Straight Line: Dataset_1

Approach:

Here, we import the data from the .txt file by using the open file python command and readlines to extract the data. Thereby, we make a list of all the x and y data points by using the append function. We use plt.plot to plot the graph and plt.savefig() to save it.

We use the stline function defined and return the best fit straight line by using the M matrix from the Least Squares Fit Method method and obtain fit_y, and thereby we plot the graph.

```
plt.plot(x, y,label="noisy values")
plt.plot(x, fit_y,label="straight line")
plt.legend(["straight line"])
plt.legend()
```

These lines of code are used to plot the legend in the given graph.

```
plt.plot(x, fit_y)
plt.errorbar(x[::25], y[::25], y_error, fmt='ro',label="errorbar")
```

The error sidebars are also plotted using the function given, and the parameters are given an indexing of 25, so that the points are sampled in every 25 points.

How the M matrix is constructed?

```
M = np.column_stack([x, np.ones(len(x))])
```

Here, the stack of the x values inputted with the coefficient of the constant term involved is inputted in the matrix.

```
(m, c), _, _, _ = np.linalg.lstsq(M, y, rcond=None)
```

Here, to solve the equations, the parameters to be computed, i.e. m, and c are on the LHS of the linalg.lstsq command.

np.column_stack is used to stack two columns horizontally to form the M matrix. These two columns represent:

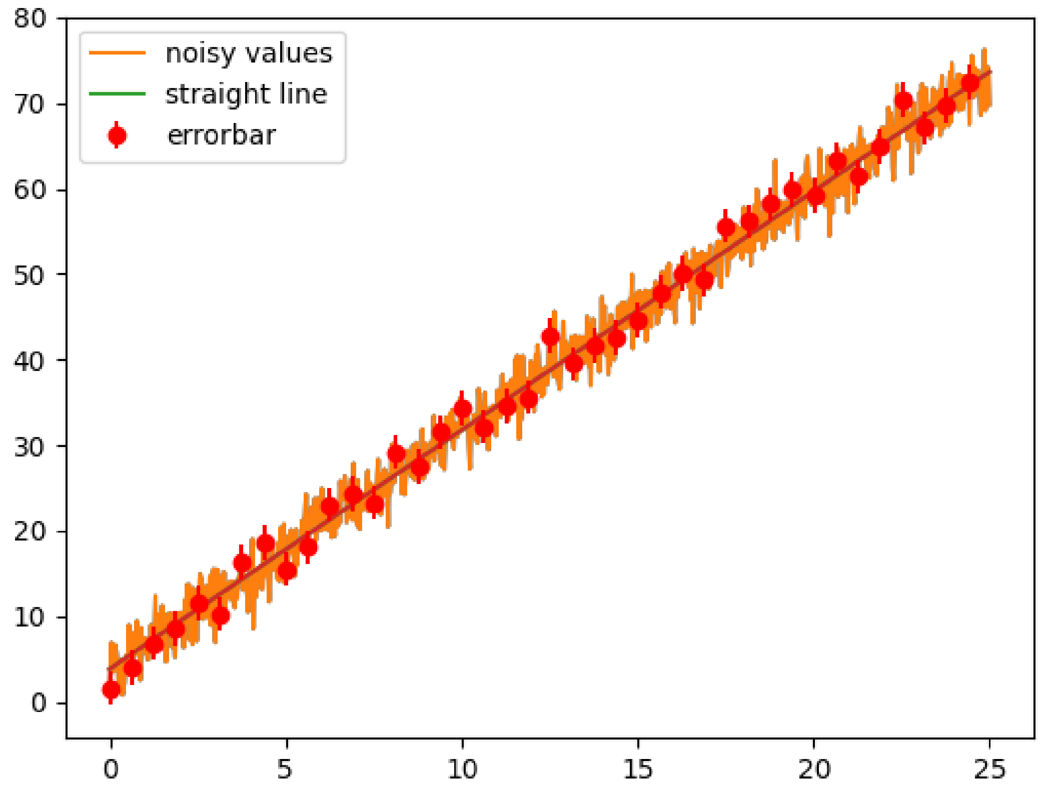
x: This is the input variable x itself.
np.ones(len(x)): This creates a column of ones with the same length as x. This column represents the constant term in a linear equation (the y-intercept).

Later in the code, this M matrix is used as the independent variable in a linear regression problem to estimate the coefficients m and c that best fit the data y in a linear equation of the form y = mx + c.

Result:

The estimated equation is 2.791124245414918 x + 3.848800101430742

This is the result obtained.



This is the plot attached herewith of the straight line that best fits the noisy data.

# Fourier Series: Dataset_2

Approach:

Here, we import the data from the .txt file by using the open file python command and readlines to extract the data. Thereby, we make a list of all the x and y data points by using the append function.

We use plt.plot to plot the graph, and plt.savefig() to save it.

A function called func with parameters to compute as f1, and p1, p2, and p3 (wherein p1, p2, and p3 are the coefficients of the three sine waves) is taken and returns the summation of the three sinusoid waves constituting the graph given.

Using the least_squares method:

We print the value f1 the frequencies also to figure out which frequency fits in the best by using curve_fit with redefining the variables whose values need to be found out using '(f1, p1, p2, p3)'. We give initial guesses of (2, 1, 1, 1) to the values of (f1, p1, p2, p3). We thereby print the required p1, p2, p3 and f1, obtained using the curve_fit algorithm.

Using the best curve_fit method:

We define a function to solve for the values of p1, p2, and p3 using the best curve_fit method, and thereby use the M matrix to solve for it.
The M-matrix makes use of the inputted values of y2, which basically is the value of the function at the given values of x2.

These three sine functions are stacked horizontally using np.column_stack. So, M will have three columns, and each column will represent one of these sine functions evaluated at the values of x2.

Each row represents a different data point or observation, and the columns represent the values of the three sine functions at those data points.

Later in the code, this M matrix is used as the independent variable in a linear regression problem to estimate the coefficients p that best fit the data y2 in the equation y2 = p[0] * sin(2.5*x2) + p[1] * sin(7.5*x2) + p[2] * sin(12.5*x2).

<u>How did you estimate the periodicity of the sine waves?</u>

<u>How was the M matrix constructed?</u>

To estimate the periodicity of the waves, we are given that the frequency of the three waves constituting the curve are in the ratio of 1:3:5. We look at the graph obtained and see that the maximum time period of the wave is 3 seconds, the frequency being the inverse, and take the corresponding frequencies of the waves in that ratio.

We use curve_fit to obtain the ratio of 2.5, 7.5, and 12.5 by giving initial estimate values, and similarly the coefficients.

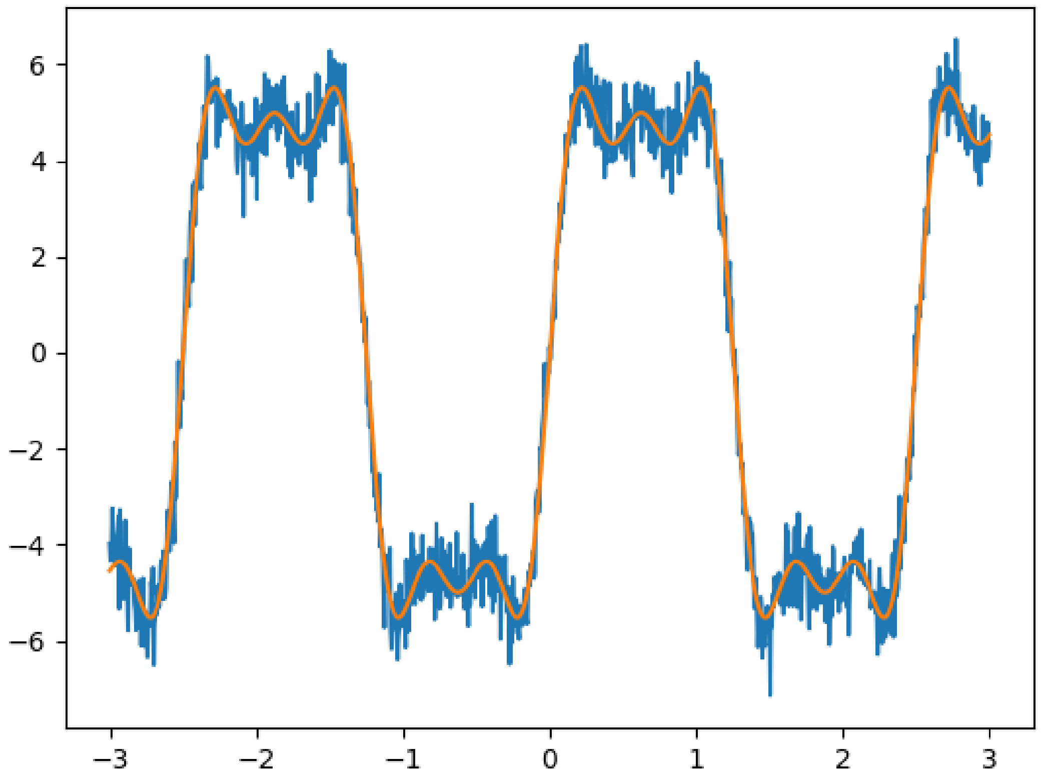
```
(f1, p1, p2, p3), pcov = curve_fit(func, x2, y2, p0 = [2, 1, 1, 1])

# Building a model to obtain the coefficient values

M = np.column_stack([np.sin(2.5*x2), np.sin(7.5*x2), np.sin(12.5*x2)])

# Use the lstsq function to solve for the array p

p, _, _, _ = np.linalg.lstsq(M, y2, rcond=None)
```

The M matrix was constructed using the stacking of [np.sin(2.5*x2), np.sin(7.5*x2), np.sin(12.5*x2)] as we need to perform curve_fit to obtain the best possible values of p, namely, p[0], p[1], p[2].

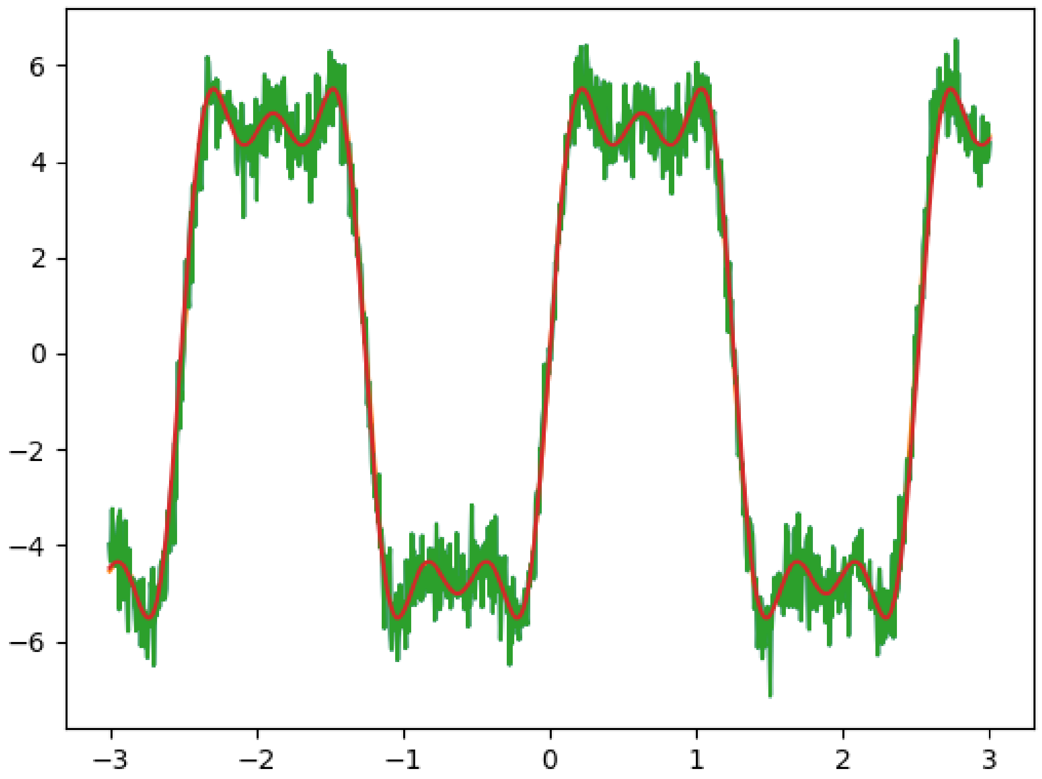
<u>Result:</u>

Estimated function: 6.011120039922203 * np.sin(2.5127345570127146*x2) + 2.001458525077708 * np.sin(3*2.5127345570127146*x2) + 0.9809072132533909 * np.sin(5*2.5127345570127146*x2)

The estimated parameters are: 6.008241657836032 sin(1/3*x2) + 1.9949067627692931 sin(1*x2) + 0.9899459271190323 sin(5/3*x2)



This is the plot attached herewith of the curve that best fits the noisy data using least_squares method.



This is the plot attached herewith of the curve that best fits the noisy data using best curve_fit method.

These are the results obtained.

```
# Plank Experiment: Dataset_3_1
```

Approach:

We import the libraries necessary and then open teh file and read through the lines to assort the x and y values into arrays and plot them to obtain a Gaussian-of kind distribution plot of temperature versus the function B.

We define a function called planklaw which takes the expression of the plank law and returns that value. We define the constant values of kb, h and c here first.
Then we use the curve_fit algorithm to identify the variables whose
values need to be estimated, namely t here.

We define initial guesstimates for the same while mentioning these to be the parameters that are to be guessed from the values given.
We thereby plot the estimated y plot versus x and see that the results are
fairly accurate. We also obtain the estimated values of the parameters.

Result/ Documentation of Trial and Error:

By giving initial guesses of 3000 or 300 both, the Temperature yielded a value of 4986 in each of the cases, which is pretty accurate an estimate. Here, since all of the constant values have also been defined and assigned the values, it is more likely to get the correct value of the Temperature.

```
# Plank Experiment: Dataset_3_2
```

Approach:

We define a function called planklaw which takes the expression of the plank law and returns that value.

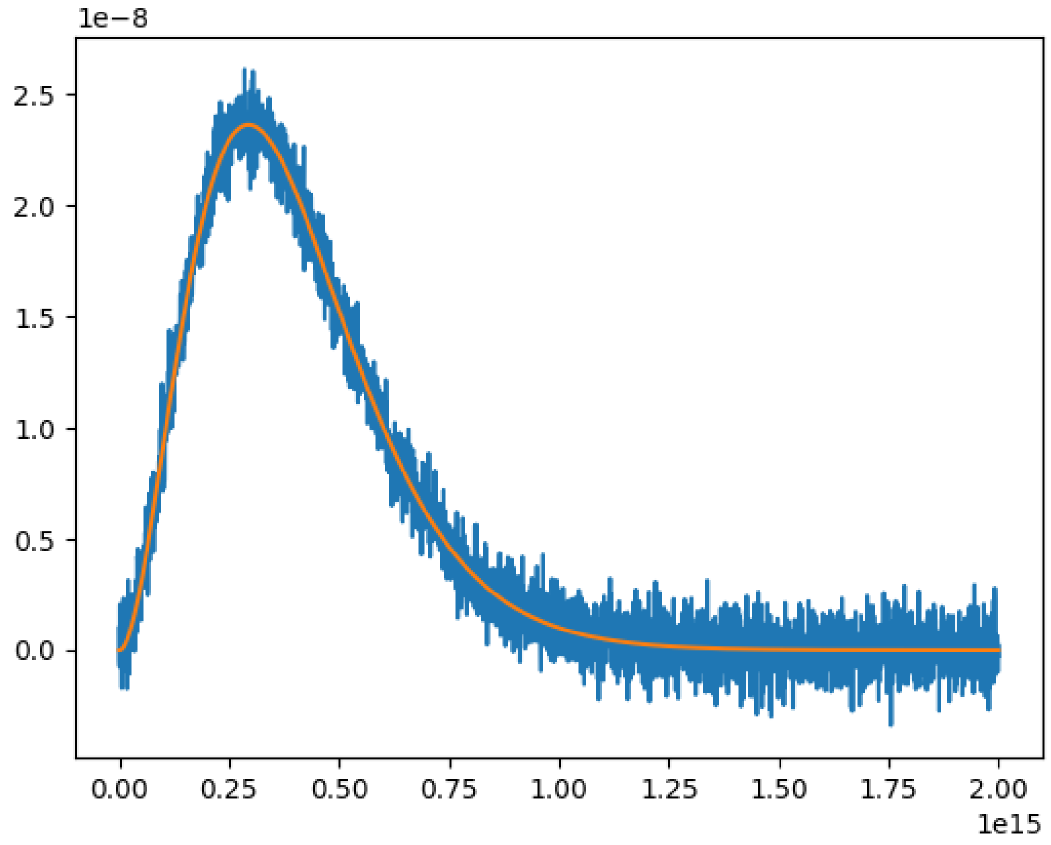
Then we use the curve_fit algorithm to identify the variables whose values need to be estimated, namely h, kb, t and c here.

We define initial guesstimates for the same while mentioning these to be the parameters that are to be guessed from the values given.
We thereby plot the estimated y plot versus x and see that the results are fairly accurate. We also obtain the estimated values of the parameters.

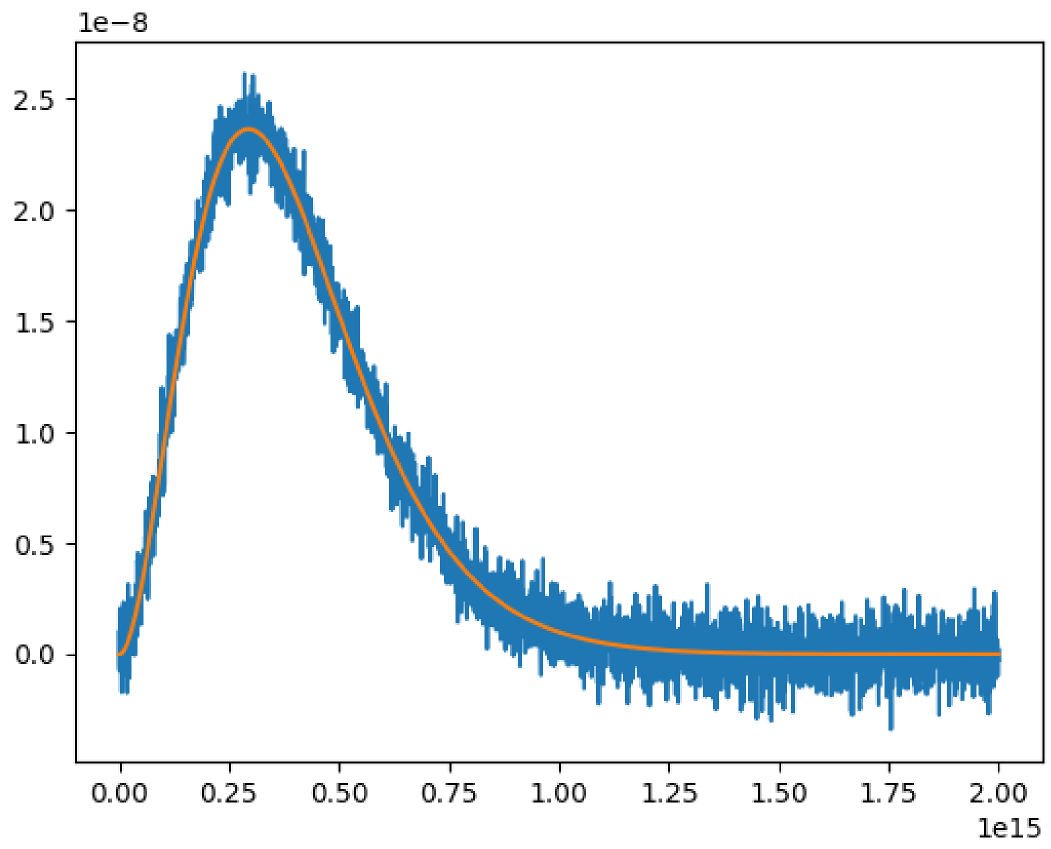
(h, t, kb, c), _ = curve_fit(plancklaw, x3, y3, p0 = [1e-33, 2000, 1e-23, 1e8]:
The initial guesses of these values give the most accurate results. The other values gave the Temperature to be twice of the actual value, otherwise the Plank’s constant and Boltzmann constant values were pretty much accurate except that they varied by an order of ten in some of the guesstimates given initially.

Result/ How close are they to the actual values, and can you think of ways to improve your estimates:

Estimated T = 3498.5646124238597, Planck's constant = 5.871005274543504e-34, Boltzmann constant = 1.7490161715025108e-23, Speed of light = 282712294.2797647



This is the plot attached herewith of the curve that best fits the noisy data using best curve_fit method wherein we had to find the value of Temperature.



This is the plot attached herewith of the curve that best fits the noisy data using best curve_fit method wherein we had to find the value of h, kb, c and t.

These are the results obtained.

The results are pretty accurate given that the initial estimates of the values are pretty close to the actual values to be obtained. Better the initial guess, better is the measure of the values.