Assignment 3 - Data Estimation

Abhinav I S EE23B002

September 16, 2024

---

1 Introduction

The data given to us in the experiments, corresponds to the spectral radiance per unit wavelength. We are expected to estimate the various parameters by curve fitting this data.

The equation for spectral radiance per unit wavelength is given by

$$B_\lambda(\lambda, T) = \frac{2hc^2}{\lambda^5} \frac{1}{e^{\frac{h\nu}{k_B T}} - 1}$$

1.1 Plotting the datasets

The data sets can be plotted by executing the function `plotDataSet`, with the path to the dataset
As showing in figures 1 - 4, d1 seems to have lesser data points, while d2, and d4 seems to have too much nosie. Meanwhile, d3 looks like it has more points, as well as a reasonable amount of noise.

2 Curve Fitting

I will try to use `scipy.optimize.curve_fit` to fit the curve with different datasets, to obtain curve fits for this I need to write a function that returns spectral radiance, after taking in the unknown parameters.

- This is the `plancksLaw` function in the Jupyter Notebook

2.1 All Parameters unknown

When we try calling `scipy.optimize.curve_fit` without passing any initial conditions, it returns the following error

```
    RuntimeWarning: overflow encountered in exp
  B = (2*h*np.power(c,2))/(np.power(Ld,5)*(np.exp(h*c/(Ld*k*T))-1))
```

This is because the value in the `np.exp` function becomes too large to handle, this can be fixed by giving better initial conditions to the `curve_fit` function.
We need to give better initial conditions to `sp.optimize.curve_fit`, This can be done by calling `curve_Fit` with a different `initial_conditions` each time.
The function `plotFit` can be called with an initial conditions array, the dataset, and a function to curve fit, and then plot and then calls `scipy.optimize.curve_fit` and then plots the fitted curve with the original dataset, it also prints out the fitting parameters

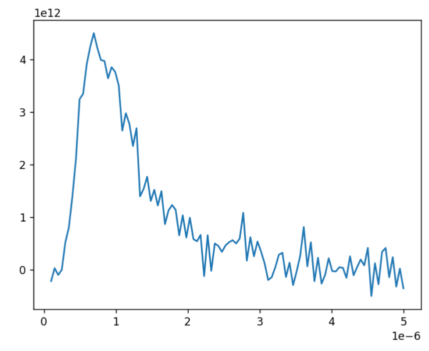

Figure 1: d1.txt

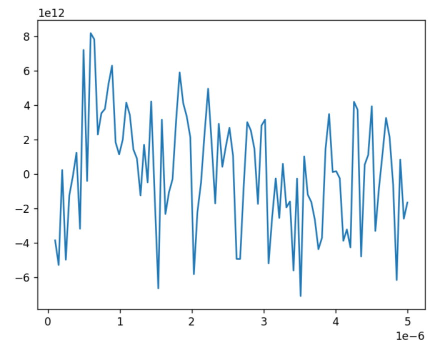

Figure 2: d2.txt

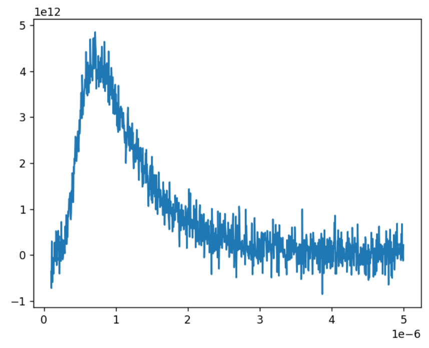

Figure 3: d3.txt

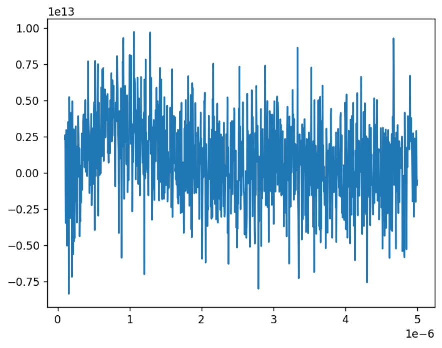

Figure 4: d4.txt

We can try fitting for different datasets, with different initial conditions by then simply calling, the `plotFit` function, as demonstrated in the Jupyter notebook

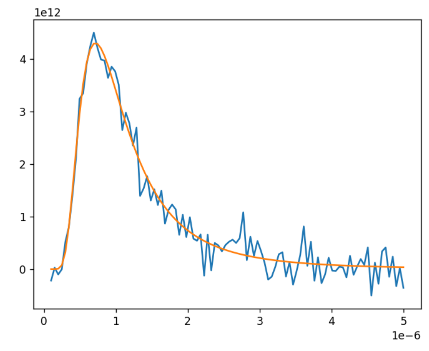

Figure 5: d1.txt

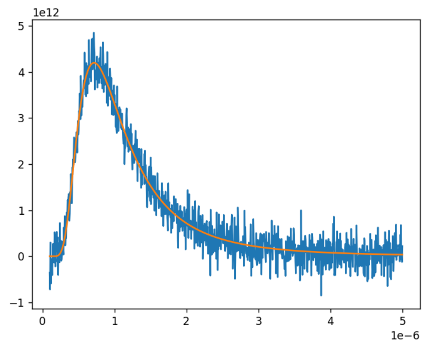

Figure 6: d3.txt

For example, after trying the initial conditions

$$T = 1000, h = 10^{-30}, c = 10^{6}, k = 10^{-20}$$

I got the fitted parameters

$$T = 1.56 * 10^3, h = 4.99 * 10^{-30}, c = -3.4 * 10^6, k = -3.4 * 10^{-21}$$

Even though the fitted curves look good, the values are very bad, as expected.
The same results can be reproduced in the Jupyter Notebook by simply calling the `plotFit` function multiple times

2.2 Partial Application

Now, we fix parameters and try to fix such that there is only one unknown at a time, in order to obtain better results.

2.2.1 T is the only unknown

Leaving temperature as the only variable, because in reality, we know all other constants, except T, so we create a new function `newPlanksLaw` which is a wrapper around `plancksLaw` itself, and on curve fitting with d3.txt, we get T = 4010.745 K, this can be done, by calling, the `plotFit` function with the updated `newPlancksLaw` function with the initial conditions, as in the Jupyter Notebook
Repeating the same exercise with d1 results in T = 4030.094, no meaningful T from d2 and T = 3914.61 with d4

2.2.2 C is the only unknown

Now, I will fix every other parameter, except c(speed of light) in order to get a better fit. I will move on with the T obtained from d3, because d3 dataset looks like it gives the best fit.
T = 4010.74559518
On calling `plotFit` with the updated `newPlancksLaw`, we get accurate values for c, with good curve fits. I have tried fitting c for each of the given datasets in the Jupyter Notebook.

2.2.3 h is the only unknown

Now, I will fix every parameter, except h (Plancks constant). I will move on with T obtained from d3, and c obtained from d3, to find out value for h, by following in the same procedure as above.
We get accurate results for h, while using d3, and the values get progressively worse for d1, f2, and d4, which is expected.
I have fitted for h, for each of the datasets given in the Jupyter Notebook.

2.2.4 k is the only variable

Now, I will fix every parameter, except k (Boltzmann's constant).

I will use all the values obtained from fitting d3, again, as they are the most accurate I got k = 1.379603e-23 for d3, and the values are slightly different for each of the different datasets, datsets. I have fitted for k, for each of the datasets given in the Jupyter Notebook

3 Results

It is clear that, even though not fixing values for any of the parameters, can give a decently good fit, the values are all over the place. Using the dataset with the most number of points, and less noise, also tends to give more accurate results, and keeping one value unknown at a time produces the best results.