Lab 2 Report

Math 391 - Introduction to Modern Physics Lab
Professor Wayne Lau
University of Michigan

Jinyan Miao

Fall 2022

Contents

Lab 2 - Blackbody Radiation

Introduction

In the early 20th century, British physicist Lord Rayleigh derived the Rayleigh-Jeans law through classical arguments and empirical facts:

$$\frac{dI}{df} = \frac{2\pi f^2}{c^2} kT \qquad \text{(Rayleigh-Jeans law)}$$

The Rayleigh-Jeans law, which approximates spectral radiance of electromagnetic radiation as a function of wavelength and temperature, agrees with experimental results for large wavelengths, but it diverges for short wavelengths, which leads to the so-called UV catastrophe. The resolution to the UV catastrophe was given by Max Planck of Planck's law for blackbody radiation, which assumes the quantization of photon energy, and that gives the correct result for blackbody radiation at both long and short wavelengths:

$$\frac{dI}{df} = \frac{2\pi h f^3}{c^2} \frac{1}{e^{hf/(kT)} - 1} \tag{2.1}$$

Planck's result also suggests that the intensity radiated by the blackbody is proportional to T^4 with a constant of proportionality σ_s, which agrees with the Stefan–Boltzmann law that was experimentally verified in the late 19th century:

$$I = \sigma_s T^4 \tag{2.2}$$

In Lab 2 of Physics 391, we verify the T^4 dependence of the radiation intensity, and we demonstrate the spectral intensity temperature dependence. We do so by measuring the temperature and the radiated intensity of an incandescent lamp when the lamp is powered by a source, then we fit our data with (2.1) and (2.2) to conclude our results. This lab is designed to help us to develop a better understanding of Planck's law and Stefan-Boltzmann law for blackbody radiation.

Experimental setup

In this lab, we explore the properties of blackbody radiation by using a 12-volt incandescent lamp. The intensity radiated by the lamp can be approximated by measuring the power that it consumes:

$$P = IV \tag{2.3}$$

where I is the current through the lamp and V is the voltage across the lamp. The temperature of the lamp is varied by changing the voltage across the lamp, and calculated from the effective resistance of the lamp.

The Lab consists of two parts. In the first part, we investigate the spectral intensity temperature dependence. A silicon photodiode is used for detecting the spectral intensity of the lamp at three different wavelengths, 550 nm (green), 750 nm (red), and 950 nm (dark). Each wavelength is selected by a bandpass filter. The lamp is held in place and wired in a box, and a bandpass filter is placed between the lamp and the silicon photodiode. When the lamp is powered, the light emitted from the lamp first goes through the bandpass filter, then the intensity of the filtered light is recorded by measuring the current through the photodiode, and at the same time, the current and voltage across the lamp is measured. We repeat this process for all three bandpass filters. The temperature of the lamp is calculated through the following approximation proposed by Howard Jones and Irving Langmuir at the General Electric Corporation in 1927:

$$T = \frac{a + br + cr^2}{1 + dr} \qquad \text{where } r = \frac{R - R_{\text{cord}}}{R_{\text{room temperature}}} \tag{T}$$

where $a = -4.129538 \cdot 10^{-1}$, $b = 4.360552 \cdot 10^{-3}$, $c = 7.399998 \cdot 10^{-7}$, and $d = 6.195380 \cdot 10^{-5}$. $R = V/I$ is the resistance of the lamp measured when different voltages V is applied across the lamp, R_{cord} is the resistance of the power cord which we measured before performing the experiment, and $R_{\text{room temperature}}$ is the resistance of the lamp at room temperature. We measure $R_{\text{room temperature}}$ before and after the experiment to ensure the consistency of our data. We then fit our data (the ratio of temperature T/T_{max} of the lamp, and the corresponding ratio of current i/i_{max} through the photodiode) with the following equation to determine the filter wavelengths λ of the bandpass filter:

$$\log_{10}\left(\frac{i}{i_{max}}\right) = \log_{10}\left(e^{\frac{hc}{\lambda k T_{max}}} - 1\right) - \log_{10}\left(e^{\frac{hc}{\lambda k T}} - 1\right) \tag{2.4}$$

Note that (1.4) can be derived by scaling both sides of (2.1) by the maximum intensity and then taking the logarithm.

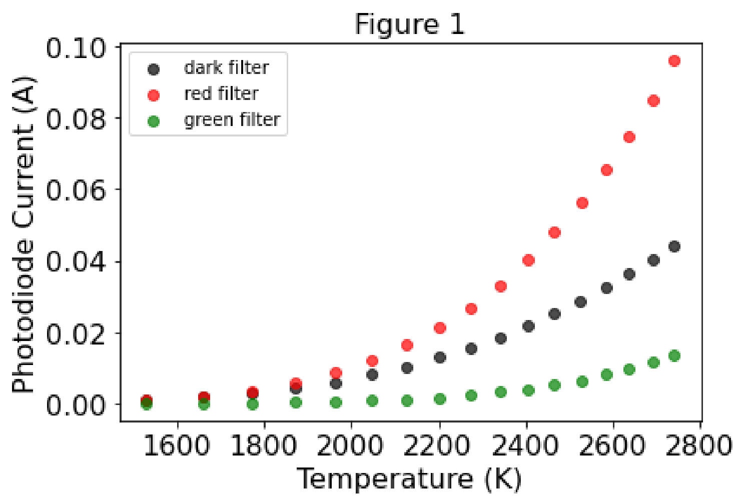
The second part of the lab is measuring the radiated power of the lamp at different temperatures. In this part, we simply repeat the measure in the first part except without the bandpass filters and the photodiode. That is, we measure the current and the voltage across the lamp, and calculate the temperature and power of the lamp. Then we fit our data (the power of the lamp P, and the temperature T) to the following equation to determine the power dependence T^4 in the Stefan-Boltzmann law:

$$\log(P) = m \log(T) + b \tag{2.5}$$

where m should give us the theoretical value of 4, and b is a constant. Here we note that (2.5) can be obtained by taking the log of both sides of (2.2).

Visualizing the data

For the measurement of spectral intensity temperature dependence, we obtain the following data for the three bandpass filters, red, green, and dark:



By observing Fig 1, we see that most of the energy radiated by the lamp has shorter wavelengths, as indicated by the plot that the red data points are greater than the black and green data points at a given temperature. One can also plot the log of the ratio i/i_{max} of the photodiode current, over the temperature inverse $1/T$ of the lamp, as shown in the following, for the three bandpass filters:

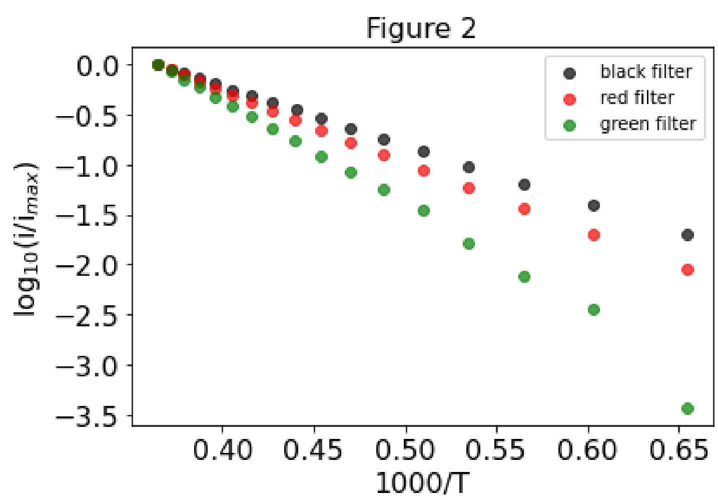
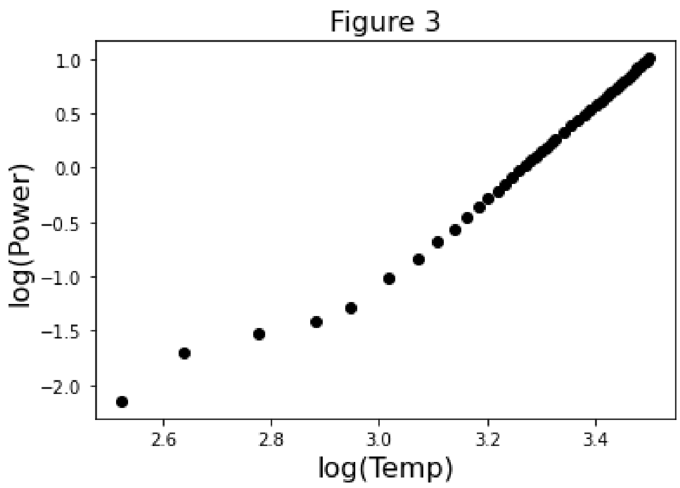


Fig. 2 indicates a linear relationship between $\log(i/i_{max})$ and $1000/T$, and this relationship is predicted by (2.4) because the right-hand side of (2.4) can be approximated as a polynomial of $1/T$ with some coefficient proportional to $-hc/(\lambda k)$. That is, we write:

$$\log\left(\frac{i}{i_{max}}\right) \propto \frac{1}{T} \qquad \text{with proportional coefficient } -\frac{hc}{\lambda k} \tag{2.6}$$

For the measurement of radiated power, we obtain the following data:



where we also see an approximately linear relationship between $\log(P)$ and $\log(T)$. This relationship is predicted by (2.5), with the slope of the linear relationship given by m, with the theoretical value of $m = 4$.

Analyzing the data

First, we fit our measurement of spectral intensity temperature dependence with (2.4) to find the wavelengths λ of the bandpass filter:

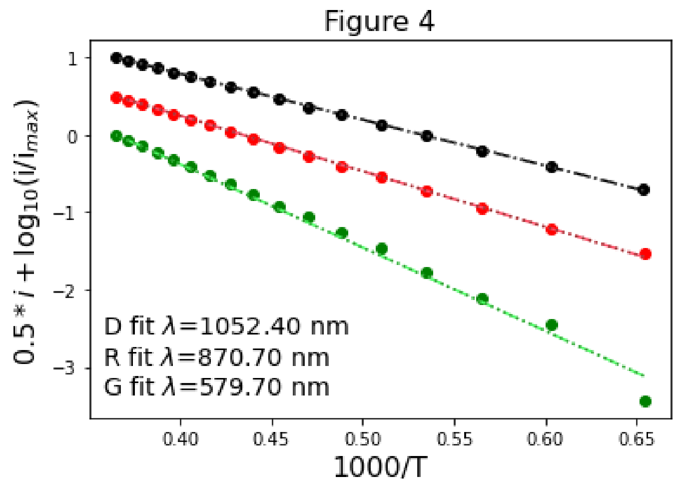


Fig. 4 plots the data in the same way as we did in Fig. 2, except the data for the red and the dark bandpass filters are translated vertically upwards by $0.5 \cdot i$, where $i = 1$ for the red filter and $i = 2$ for the black filter ($i = 0$ for the green filter). The data are fitted before the vertical translation, and they are fitted in two different methods. The first method is minimizing the unweighted Δ quantity:

$$\Delta = \sum_i (y_i - f(x_i))^2$$

where f is the function of the model to be fitted, and (x_i, y_i) is the data point. The fitted model (2.4) under this method is plotted using dash lines (--). Note that the dash lines almost overlap with the dotted lines $(\cdot\ \cdot)$, which represents the model fitted using the second method. For the second method, we fit the data using the scipy library function *scipy.optimize.curve_fit*, which fits the data using the method of non-linear squares fitting. As mentioned previously, the fitted model under the second method is plotted using the dotted lines. The error of the second method is calculated using variances. Here we note that the fitted wavelengths λ indicated in Fig. 4 are those generated by the first method. The results of this process can be summarized by the following tables:

First method: minimizing Δ	wavelength	Δ
Green filter	579.70 nm	0.1507
Red filter	870.70 nm	0.0029
Dark filter	1052.40 nm	0.0007

Second method: minimizing variance	wavelength	Variance
Green filter	579.66 nm	$9.732 \cdot 10^{-17}$
Red filter	870.72 nm	$9.826 \cdot 10^{-18}$
Dark filter	1052.39 nm	$5.203 \cdot 10^{-18}$

We see that the difference between the two methods is tiny, and hence the models of the two methods almost overlap. From the slope of the models, we see that the radiated intensity for an object with a lower temperature is smaller, and by comparing the slope of the three different bandpass filters, we have verified the relationship between the radiation intensity, wavelength, and temperature of the object as characterized by (2.6), that is the slope of the lines increases as the wavelength of the bandpass filter increases. The last thing to notice here is that the wavelengths predicted by the models are about 20% longer than that of the actual bandpass filters (550 nm for green, 750 nm for red, and 950 nm for dark). As mentioned in the lab manual, this is most likely due to an overestimate of the filament temperature by the Jones and Langmuir model given by equation (T).

For the second part of the experiment, we fit our spectral intensity data with equation (2.5).

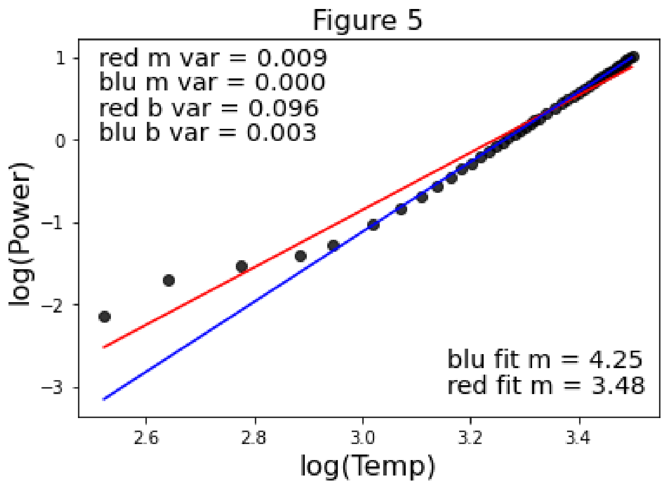
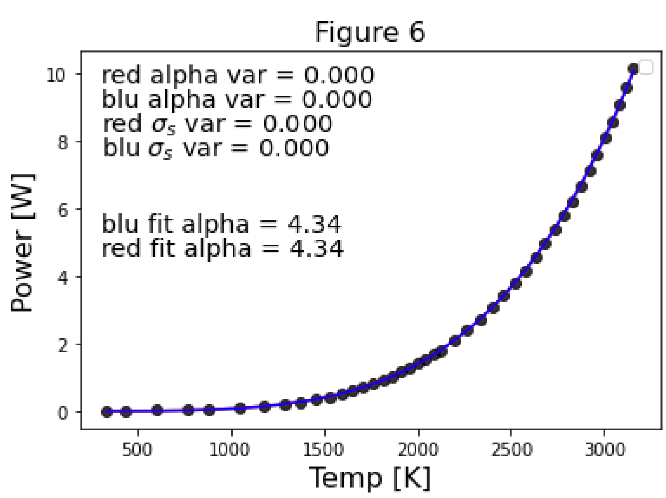


Fig. 5 plots the data in the same way as we did in Fig. 3. In addition we also plot two best-fit models for the data set. The first model is the red line, which is fitted by using all points in the data set. The slope of the red line is 3.48, and the variance of parameter m is 0.009. This indicates that the power α in the relationship $I = \sigma_s T^\alpha$ is around 3.48, but this is clearly an underestimate of m as we see in Fig 5. that there are some outlier data points at low temperature, which is mostly because, at low temperature, the blackbody radiation is negligible and most of the input power is dissipated by thermal conduction to the lamp filament supports and the rest of the world. By removing the first 4 data points on the left of Fig 5., we fit the rest of the data points using the blue line, and that gives us approximately zero variance for parameter m, with $m = 4.25$. Notice that the slope of the curve 4.25 is greater than the theoretical value of 4, yet it is close to 4, the difference is significant (differ by around 6%). Hence we are unable to conclude that the power dependence α in the Stefan-Boltzmann law is 4. The cause of this result is yet to be investigated, but one possible factor is the underestimation of the temperature of the lamp by the Jones and Langmuir model given by equation (T), and the quality of the lamp might have been downgraded since its use in the first part of this lab. While we should, if we get the chance in the future, repeat the spectral intensity experiment with different lamps to get more statistical convincing results.



On the other hand, one can also fit the original data with the power law $I = \sigma_s T^\alpha$ to find the parameter α. We again use the scipy library function *scipy.optimize.curve_fit* to obtain the model, minimizing the variance via fitting the data with parameters α and σ_s. The red curve in Fig. 5, which is mostly overlapped by the blue curve, is fitted using all data points in our data set. The blue curve in Fig. 5 is fitted by removing the first 4 data points at the low-temperature end. The variances for all parameters in these fittings are negligible, but we observe that the power dependence α is still much larger than 4.

Consistency of data

Efforts have been made to ensure the consistency of the data in this experiment. Before performing the first part of this experiment (measuring spectral intensity temperature dependence), we recorded the resistance of the lamp at room temperature, and we repeat this measurement 10 minutes after the first part of the experiment. The 10 minutes period is designed to give enough time for the lamp to cool down to room temperature. We also repeat the measurement 10 minutes after the second part of the experiment (measuring the radiated power). The results are given by the following:

	$R_{\text{room temperature}}$	Condition of lamp
Before the first part	$5.129\,\Omega$	clear
After the first part	$5.135\,\Omega$	clear
After the second part	$5.134\,\Omega$	not darken

From the data recorded, we see that the condition of the lamp has not significantly changed even after the second part of the experiment. This indicates that the result we obtained should have not been affected by the quality of the lamp that we used in this experiment.

In addition to that, in the first part of the experiment, we start with the spectral measurements of the dark bandpass filter, and when the measurement of all three bandpass filters has been done, we repeat the spectral measurements for the dark bandpass filter. Comparing the two data sets we obtain the followings:

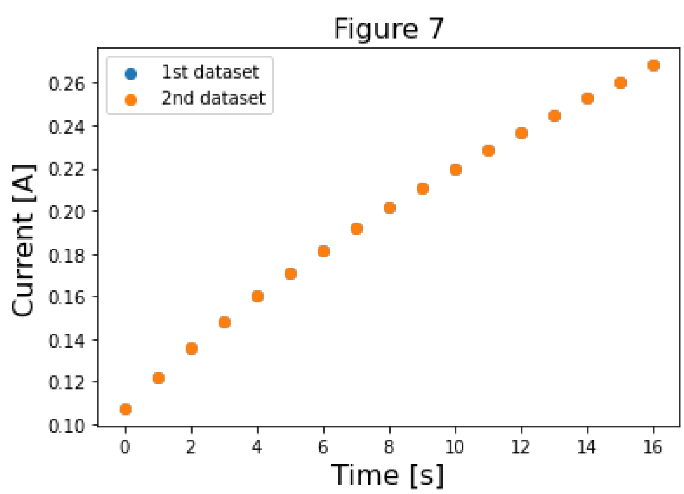


Fig. 7 Comparing the current through the lamp recorded in the two data sets

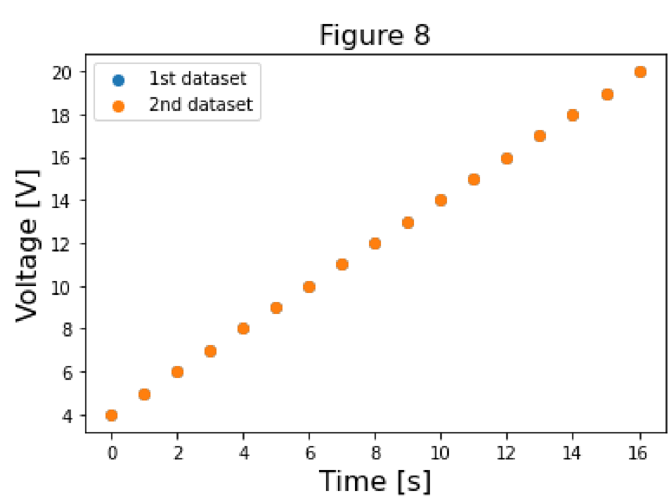


Fig. 8 Comparing the voltage across the lamp recorded in the two data sets

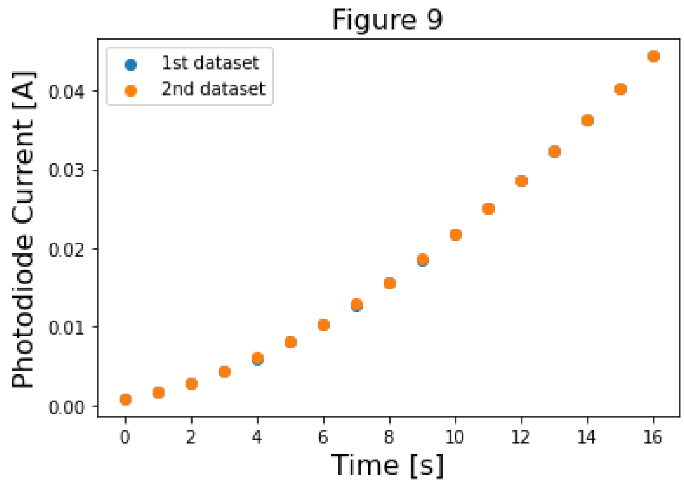


Fig. 9 Comparing the current through the photodiode recorded in the two data sets

From Fig. 7, 8, and 9, we see that the orange data points (representing the points recorded in the second data set) almost overlap with the blue data points (representing the points recorded in the first data set), which again suggests that the results in the first part of this lab should not be affected by the quality of the lamp.

Summary

In Lab 2 of Physics 391, we perform experiments intending to verify (I) the temperature power dependence in the Stefan-Boltzmann law, and (II) the relationship between the radiated intensity, the temperature of the blackbody, and wavelengths as revealed by Planck's law of radiation. In this text, we analyze the experiment results and conclude that (II) is well justified by our data, but we fail to verify (I) as the difference between our data and the theoretical power dependence of $\alpha = 4$ is statistically significant. We seek to perform the measurement of the radiated power of lamps again in the future to get more statistical convincing results. Nevertheless, the underlying physics in this experiment - energy of photons is quantized and hence (2.1) and (2.2) holds for blackbody radiation - is already well confirmed by many other experiments, but it is still worth spending time to experience the difficulty in generalizing those results.

Experiment Data

Black filter 1				
Index	**Time**	**Voltage**	**Current**	**PhotoCurrent**
0	0.0	3.999483	0.107399	0.000886
1	1.0	4.999443	0.121979	0.001709
2	2.0	5.999354	0.135549	0.002818
3	3.0	6.999373	0.148162	0.004316
4	4.0	7.9992	0.159983	0.006006
5	5.0	8.998361	0.171142	0.008062
6	6.0	9.998784	0.181742	0.010271
7	7.0	10.99892	0.191845	0.012834
8	8.0	11.99869	0.201536	0.015637
9	9.0	12.99837	0.210822	0.018553
10	10.0	13.99771	0.219802	0.02177
11	11.0	14.99761	0.228483	0.025139
12	12.0	15.99668	0.236637	0.028589
13	13.0	16.99652	0.24483	0.032264
14	14.0	17.99601	0.252799	0.036262
15	15.0	18.99576	0.260594	0.040228
16	16.0	19.99561	0.268219	0.04437
Green filter 1				
Index	**Time**	**Voltage**	**Current**	**PhotoCurrent**
0	0.0	3.99934	0.107498	5e-06
1	1.0	4.999364	0.122028	4.9e-05
2	2.0	5.999322	0.135591	0.000103
3	3.0	6.999341	0.148188	0.000225
4	4.0	7.999185	0.160002	0.00048
5	5.0	8.998329	0.17115	0.000764
6	6.0	9.9988	0.181738	0.001163
7	7.0	10.99903	0.191849	0.001648
8	8.0	11.99877	0.201521	0.002353
9	9.0	12.9984	0.210822	0.003131
10	10.0	13.99775	0.219794	0.004069
11	11.0	14.99771	0.228475	0.005271
12	12.0	15.99677	0.236641	0.006493
13	13.0	16.99661	0.244822	0.008048
14	14.0	17.99615	0.252792	0.009676
15	15.0	18.99589	0.260583	0.01148
16	16.0	19.9958	0.268204	0.013586
Red filter 1				
Index	**Time**	**Voltage**	**Current**	**PhotoCurrent**
0	0.0	3.999372	0.107501	0.000881
1	1.0	4.999395	0.122036	0.001893
2	2.0	5.999259	0.135587	0.003447
3	3.0	6.999357	0.148184	0.005675
4	4.0	7.999153	0.159983	0.008503
5	5.0	8.998298	0.171138	0.012075
6	6.0	9.998721	0.181734	0.016301
7	7.0	10.99889	0.191834	0.021266
8	8.0	11.99874	0.20151	0.026882
9	9.0	12.99835	0.210807	0.033221
10	10.0	13.99768	0.219779	0.040218
11	11.0	14.99766	0.228467	0.047923
12	12.0	15.99669	0.23663	0.056286
13	13.0	16.99647	0.244807	0.065306
14	14.0	17.99602	0.252784	0.074883
15	15.0	18.99586	0.260576	0.085151
16	16.0	19.99559	0.268197	0.096019

Black filter 2				
Index	**Time**	**Voltage**	**Current**	**PhotoCurrent**
0	0.0	3.999324	0.107475	0.000872
1	1.0	4.999332	0.12201	0.001738
2	2.0	5.99929	0.135572	0.002857
3	3.0	6.999341	0.148177	0.004311
4	4.0	7.999169	0.159991	0.006028
5	5.0	8.998282	0.171131	0.008094
6	6.0	9.998721	0.18173	0.010351
7	7.0	10.999	0.191864	0.012858
8	8.0	11.99874	0.201555	0.015615
9	9.0	12.99841	0.210849	0.018604
10	10.0	13.99775	0.219809	0.021704
11	11.0	14.99769	0.228498	0.025041
12	12.0	15.99677	0.23666	0.02863
13	13.0	16.9966	0.244841	0.032293
14	14.0	17.99612	0.252803	0.036203
15	15.0	18.99589	0.260598	0.04024
16	16.0	19.99575	0.268223	0.044339

Radiated Power			
Index	**Time**	**Voltage**	**Current**
0	0.0	0.199951	0.044339
1	1.0	0.399698	0.044339
2	2.0	0.599825	0.044339
3	3.0	0.799588	0.044339
4	4.0	0.999398	0.044339
5	5.0	1.49954	0.044339
6	6.0	1.999381	0.044339
7	7.0	2.498984	0.044339
8	8.0	2.999061	0.044339
9	9.0	3.498821	0.044339
10	10.0	3.999135	0.044339
11	11.0	4.499004	0.044339
12	12.0	4.999174	0.044339
13	13.0	5.499359	0.044339
14	14.0	5.999053	0.044339
15	15.0	6.499443	0.044339
16	16.0	6.999135	0.044339
17	17.0	7.498559	0.044339
18	18.0	7.998931	0.044339
19	19.0	8.498242	0.044339
20	20.0	8.998013	0.044339
21	21.0	9.498621	0.044339
22	22.0	9.998405	0.044339
23	23.0	10.99859	0.044339
24	24.0	11.99838	0.044339
25	25.0	12.99797	0.044339
26	26.0	13.9972	0.044339
27	27.0	14.99725	0.044339
28	28.0	15.9963	0.044339
29	29.0	16.99614	0.044339
30	30.0	17.99563	0.044339
31	31.0	18.99534	0.044339
32	32.0	19.9951	0.044339
33	33.0	20.99535	0.044339
34	34.0	21.99568	0.044339
35	35.0	22.99535	0.044339
36	36.0	23.99518	0.044339
37	37.0	24.99539	0.044339
38	38.0	25.99585	0.044339
39	39.0	26.99679	0.044339
40	40.0	27.99719	0.044339
41	41.0	28.99803	0.044339
42	42.0	29.99835	0.044339

Code

The code for computing statistics of the data sets is attached.

```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
import scipy.optimize as opt
import os

# Read in the three spectral intensity datasets from your data directory
    here.  Use pd.read_csv
speIntB1_df = pd.read_csv('data/Spectral_Intensity_Data_black1.txt',
                          skiprows=0, delimiter='\t').drop(17)
speIntB2_df = pd.read_csv('data/Spectral_Intensity_Data_black2.txt',
                          skiprows=0, delimiter='\t').drop(17)
speIntR1_df = pd.read_csv('data/Spectral_Intensity_Data_red1.txt',
                          skiprows=0, delimiter='\t').drop(17)
speIntG1_df = pd.read_csv('data/Spectral_Intensity_Data_green1.txt',
                          skiprows=0, delimiter='\t').drop(17)
radPower_df = pd.read_csv('data/Radiated_Power_Data.txt', skiprows=0,
                          delimiter='\t').drop(43)

# Define and calculate resistance constants, filter resistances, and
    resitance ratios

# resistance values here
r_cords = 0.028
r_filament_and_cords = 5.129
r_roomT = r_filament_and_cords - r_cords
r_filament_and_cords_rP = 5.135
r_roomTr = r_filament_and_cords_rP - r_cords

# Filter resistance and ratios
green_resistance = [V/I for V,I in zip(speIntG1_df['Voltage'], speIntG1_df['
    Current'])]
red_resistance = [V/I for V,I in zip(speIntR1_df['Voltage'], speIntR1_df['
    Current'])]
dark_resistance = [V/I for V,I in zip(speIntB2_df['Voltage'], speIntB2_df['
    Current'])]
radP_resistance = [V/I for V,I in zip(radPower_df['Voltage'], radPower_df['
    Current'])]


green_resistance_ratio = [r/r_roomT for r in green_resistance]
red_resistance_ratio = [r/r_roomT for r in red_resistance]
dark_resistance_ratio = [r/r_roomT for r in dark_resistance]
radP_resistance_ratio = [r/r_roomTr for r in radP_resistance]

def calculate_temperature(r, a=-4.129538e-1, b=4.360552e-3,
                          c=7.399998e-7, d=6.195380e-5) :
  '''Using fit parameters for the resistivity ratio of tungsten as a
    function of temperature,
  we calculate the temperature as a function of the resistance ratio.
  Input:
  r (float): resistance ratio with unity at 300K

  Output:
  temperature (float) : Temperature to convert values for lamp voltage and
    current to resistance
  '''
  return (d*r-b+np.sqrt((d*r-b)**2 + 4*(r-a)*c))/(2*c)


# Calculate temperatures for each filter

green_temperature = np.array([calculate_temperature(r) for r in
    green_resistance_ratio])
red_temperature = np.array([calculate_temperature(r) for r in
    red_resistance_ratio])
dark_temperature = np.array([calculate_temperature(r) for r in
    dark_resistance_ratio])
radP_temperature = np.array([calculate_temperature(r) for r in
    radP_resistance_ratio])

```

```
# Plot log10I_photodiode vs. 1/T for all three filters
bT = [1000/t for t in dark_temperature]
bI = [np.log10(I/np.max(speIntB2_df['PhotoCurrent'].values))
      for I in speIntB2_df['PhotoCurrent']]

rT = [1000/t for t in red_temperature]
rI = [np.log10(I/np.max(speIntR1_df['PhotoCurrent'].values))
      for I in speIntR1_df['PhotoCurrent']]

gT = [1000/t for t in green_temperature]
gI = [np.log10(I/np.max(speIntG1_df['PhotoCurrent'].values))
      for I in speIntG1_df['PhotoCurrent']]

plt.scatter(bT,bI, c='black', label="black filter",alpha=0.7)
plt.scatter(rT,rI, c='red', label="red filter",alpha=0.7)
plt.scatter(gT,gI, c='green', label="green filter",alpha=0.7)
plt.ylabel("log$_{10}$(i/i$_{max}$)",fontsize=16)
plt.xlabel("1000/T",fontsize=16)
plt.title('Figure 2',fontsize=16)
plt.legend()
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()

## comparing two datasets
plt.scatter(speIntB1_df['Time'], speIntB1_df['Current'], label='1st dataset'
    )
plt.scatter(speIntB2_df['Time'], speIntB2_df['Current'], label='2nd dataset'
    )
plt.ylabel("Current [A]",fontsize=16)
plt.xlabel("Time [s]",fontsize=16)
plt.title('Figure 7',fontsize=16)
plt.legend()
plt.show()

plt.scatter(speIntB1_df['Time'], speIntB1_df['Voltage'], label='1st dataset'
    )
plt.scatter(speIntB2_df['Time'], speIntB2_df['Voltage'], label='2nd dataset'
    )
plt.ylabel("Voltage [V]",fontsize=16)
plt.xlabel("Time [s]",fontsize=16)
plt.title('Figure 8',fontsize=16)
plt.legend()
plt.show()

plt.scatter(speIntB1_df['Time'], speIntB1_df['PhotoCurrent'], label='1st
    dataset')
plt.scatter(speIntB2_df['Time'], speIntB2_df['PhotoCurrent'], label='2nd
    dataset')
plt.ylabel("Photodiode Current [A]",fontsize=16)
plt.xlabel("Time [s]",fontsize=16)
plt.title('Figure 9',fontsize=16)
plt.legend()
plt.show()
plt.show()


# Plot I_photodiode vs. T for all three filters
plt.scatter(dark_temperature, speIntB2_df['PhotoCurrent'].values,
            c='black', label="dark filter",alpha=0.7)
plt.scatter(red_temperature, speIntR1_df['PhotoCurrent'].values,
            c='red', label="red filter",alpha=0.7)
plt.scatter(green_temperature, speIntG1_df['PhotoCurrent'].values,
            c='green', label="green filter",alpha=0.7)
plt.ylabel("Photodiode Current (A)",fontsize=16)
plt.xlabel("Temperature (K)",fontsize=16)
plt.title('Figure 1',fontsize=16)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.legend()
plt.show()

def planck_model_to_fit(temperature, wavelength_param) :
  '''Model to fit the intensity vs. temperature model to
  Inputs:
```

```
  temperature (array-like): temperature measured, calculated from resistivity ratio data
  wavelength_param (float): wavelength of filter, can be fit

  Outputs:
  log_norm_current (array-like): log of the current normalized by the max
  '''

  h = 6.626070040e-34
  c = 2.99792458e8
  k = 1.38064852e-23

  term1 = np.log10(np.exp(h*c/wavelength_param/k/np.max(temperature)) - 1)
  term2 = np.log10(np.exp(h*c/wavelength_param/k/temperature) - 1)

  log_norm_current = term1 - term2

  return log_norm_current

def plot_iteration_of_least_squares(temperature, log_current_norm_data,
                                    log_current_norm_model) :
  ''''''

  plt.plot(1000/temperature,log_current_norm_data)
  plt.plot(1000/temperature,log_current_norm_model, ls=':')
  plt.xlabel("1000/T",fontsize=16)
  plt.ylabel("log$_{10}$(i/i$_{max}$)",fontsize=16)
  plt.show()

def planck_chisq_to_minimize(wavelength_param, temperature, current, plot=True) :
  '''Chi squared to minimize when fitting the current vs.
     temperature data to the Planck spectrum
  Inputs:
  wavelength_param (float): wavelength of filter
                            divided by the speed of light, can be fit
  temperature (array-like): temperature measured,
                            calculated from resistivity ratio data
  current (array-like): current measured, calculated from resistivity ratio data
  plot (boolean, optional): option to plot the model and data with each iteration

  Outputs:
  least_sq (float): value of the least square value
  '''

  log_current_norm_data = np.log10(current/np.max(current))
  log_current_norm_model = planck_model_to_fit(temperature, wavelength_param)

  chi_squared_of_model = np.sum((log_current_norm_data-log_current_norm_model)**2)

  least_sq=chi_squared_of_model

  if plot :
    plot_iteration_of_least_squares(temperature, log_current_norm_data,
                                    log_current_norm_model)
    print("Least squares this iteration:%f.2 nm"%least_sq)

  return least_sq

# Fit points to Eqn 10
wavelengths = [500e-9 +i*10e-11 for i in range(0,10000)]
photo_Is = [speIntG1_df['PhotoCurrent'].values,
            speIntR1_df['PhotoCurrent'].values,
            speIntB1_df['PhotoCurrent'].values]
temps = [green_temperature,
         red_temperature,
         dark_temperature]
fit_lams = []
opt_lams = []
covars = []
chis = []
```

```python

# iterate all three colors to find best fit or optimized lambda
for i in [0,1,2]:
  # extract parameters for this color
  chi = []
  temp, photo_I = temps[i], photo_Is[i]
  # find best fit lambda
  for lam in wavelengths:
    chi.append(planck_chisq_to_minimize(lam, temp, photo_I, False))
  min_chi_index = chi.index(np.min(chi))
  chis.append(np.min(chi))
  fit_lams.append(wavelengths[min_chi_index])
  # find optimized lambda
  log_current_norm_data = np.log10(photo_I/np.max(photo_I))
  opt_lam, covariance = opt.curve_fit(planck_model_to_fit,
                                      temp,
                                      log_current_norm_data,
                                      p0 = [550e-9])
  opt_lams.append(opt_lam)
  covars.append(covariance)

# Overplotted best-fit model with data points here
colors = [['green', (0.3,1,0.4), (0.2,0.6,0.2)],
          ['red', (1,0.3,0.4), (0.6,0.2,0.2)],
          ['black', (0,0,0), (0.3,0.3,0.3)]]
print_cor = ['G', 'R', 'D']
for i in [0,1,2]:
  temp, photo_I = temps[i], photo_Is[i]
  log_I_norm_data = np.log10(photo_I/np.max(photo_I))+i*0.5
  fit_lam, opt_lam, covar = fit_lams[i], opt_lams[i], covars[i]
  log_current_norm_fit = planck_model_to_fit(temp, fit_lam)+i*0.5
  log_current_norm_opt = planck_model_to_fit(temp, opt_lam)+i*0.5
  plt.plot(1000/temp,log_I_norm_data, ls='none', marker='o', c=colors[i][0])
  plt.plot(1000/temp,log_current_norm_fit, ls='-.', c=colors[i][1])
  plt.plot(1000/temp,log_current_norm_opt, ls=':', c=colors[i][2])
  lam_print = fit_lam*10**9
  plt.annotate(print_cor[i]+' fit $\lambda$=%.2f nm'%lam_print, (0.15,0.2+i
    *0.06),
               xycoords='figure fraction',
               fontsize='x-large')

plt.xlabel("1000/T",fontsize=16)
plt.ylabel("$0.5*i + $log$_{10}$(i/i$_{max}$)",fontsize=16)
plt.title('Figure 4', fontsize=16)
plt.show()

print(covars)
print(opt_lams)
print(fit_lams)


# Read in the radiated power data from your data directory here.  Use pd.
    read_csv
radPower_df['Temp'] = radP_temperature
radPower_df['R/R0'] = radP_resistance_ratio
radPower_df['R'] = radP_resistance
radPower_df['Power'] = [I*V for I,V in zip(radPower_df['Current'],
                                           radPower_df['Voltage'])]


# Plot log P vs. log T
plt.plot(np.log10(radPower_df['Temp'].values) ,
         np.log10(radPower_df['Power'].values),
         ls='none', marker='o', c=colors[i][0])
plt.xlabel("log(Temp)",fontsize=16)
plt.ylabel("log(Power)",fontsize=16)
plt.title('Figure 3',fontsize=16)
plt.show()


def linearFit(x, m, b):
  """
  input:
    x: x-data to be fitted
    m: slope of the linear fit
```

```
    b: y-interception of the linear fit
  return:
    y: y=mx+b
  """
  return m*x+b


## linear fit the power law
opt_mb, covariance = opt.curve_fit(linearFit,
                                   np.log10(radPower_df['Temp'].values),
                                   np.log10(radPower_df['Power'].values),
                                   p0 = [4,0])
## linear fit but taking out the first few outliers
opt_mb_o, covariance_o = opt.curve_fit(linearFit,
                                       np.log10(radPower_df['Temp'].values
    [4:]),
                                       np.log10(radPower_df['Power'].values
    [4:]),
                                       p0 = [4,0])

opt_m, opt_b = opt_mb[0], opt_mb[1]
opt_m_o, opt_b_o = opt_mb_o[0], opt_mb_o[1]

print(opt_m, opt_b)
print(opt_m_o, opt_b_o)

## plot the linear fit of the power law
plt.plot(np.log10(radPower_df['Temp'].values),
         np.log10(radPower_df['Power'].values),
         ls='none', marker='o', c='black', alpha=0.8)
plt.plot(np.log10(radPower_df['Temp'].values) ,
         linearFit(np.log10(radPower_df['Temp'].values),
                   opt_m, opt_b),
         ls='-', c='r')
plt.plot(np.log10(radPower_df['Temp'].values) ,
         linearFit(np.log10(radPower_df['Temp'].values),
                   opt_m_o, opt_b_o),
         ls='-', c='b')
plt.xlabel("log(Temp)",fontsize=16)
plt.ylabel("log(Power)",fontsize=16)
plt.annotate('red fit m = %.2f '%opt_m, (0.65,0.2),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu fit m = %.2f '%opt_m_o, (0.65,0.25),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu m var = %.3f '%covariance_o[0][0], (0.15,0.81),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu b var = %.3f '%covariance_o[1][1], (0.15,0.71),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('red m var = %.3f '%covariance[0][0], (0.15,0.86),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('red b var = %.3f '%covariance[1][1], (0.15,0.76),
             xycoords='figure fraction',
             fontsize='x-large')
plt.title('Figure 5', fontsize=16)
plt.show()





def stefBoltzFit(x, sig, alp):
  """
  input:
    x: x-data to be fitted
    sig: stefan-boltzmann constant
    alp: power of the temperature
  return:
    y: sigma*x**(alpha)
  """
  return sig*x**(alp)
```

```

# power fit the power law
opt_sa, covariance_sa = opt.curve_fit(stefBoltzFit,
                                      radPower_df['Temp'].values,
                                      radPower_df['Power'].values,
                                      p0 = [0,4])
# power fit but taking out the first few outliers
opt_sa_o, covariance_sa_o = opt.curve_fit(stefBoltzFit,
                                          radPower_df['Temp'].values[4:],
                                          radPower_df['Power'].values[4:],
                                          p0 = [0,4])



opt_s, opt_a = opt_sa[0], opt_sa[1]
opt_s_o, opt_a_o = opt_sa_o[0], opt_sa_o[1]

print(opt_s, opt_a)
print(opt_s_o, opt_a_o)

## plot the power fit
plt.plot(radPower_df['Temp'].values,
         radPower_df['Power'].values,
         ls='none', marker='o', c='black', alpha=0.8)
plt.plot(radPower_df['Temp'].values,
         stefBoltzFit(radPower_df['Temp'].values,
                      opt_s, opt_a),
         ls='-', c='r')
plt.plot(radPower_df['Temp'].values,
         stefBoltzFit(radPower_df['Temp'].values,
                      opt_s_o, opt_a_o),
         ls='-', c='b')
plt.xlabel("Temp [K]",fontsize=16)
plt.ylabel("Power [W]",fontsize=16)
plt.annotate('red fit alpha = %.2f '%opt_a, (0.15,0.5),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu fit alpha = %.2f '%opt_a_o, (0.15,0.55),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu alpha var = %.3f '%covariance_sa_o[0][0], (0.15,0.80),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('blu $\sigma_s$ var = %.3f '%covariance_sa_o[1][1], (0.15,0.70)
    ,
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('red alpha var = %.3f '%covariance_sa[0][0], (0.15,0.85),
             xycoords='figure fraction',
             fontsize='x-large')
plt.annotate('red $\sigma_s$ var = %.3f '%covariance_sa[1][1], (0.15,0.75),
             xycoords='figure fraction',
             fontsize='x-large')
plt.title('Figure 6', fontsize=16)
plt.legend()
plt.show()
```