3rd Year Computational Physics : Jessica Murphy

21/03/2023

Assignment 1

This Python notebook demonstrates how to use Rayleigh-Jean's Law to plot the spectral radiance for a radiative blackbody. The Rayleigh-Jeans Law for the spectrum of a balckbody is:

$$U(v,T) = \frac{8\pi v^2}{c^3}kT$$

where $k = 1.38x10^{-23}JK^{-1}$ and $c = 3.0x10^8 ms^{-1}$

The programme is then developed to show the relationship between Planck's function and the Stefann-Boltzmann law. A plot of Planck's function of a spectral blackbody vs. wavelength is made, using a range of temperatures between 2000K-10000K. From this the max wavelength is found for each temperature, and thus a calculation is made for Wien's displacement Law, which is as follows:

$$\lambda_{peak}T = Constant$$

In the last task, a numerical integration of the Planck function is carried out to verify that the area under each respective curbe is proportional to T^4. This is because the area under the graph is equal to the power(intensity) of the radiative blackbodies, and using a constant of proportionality, this gives us:

$$P = \sigma T^4$$

which is the Stefann-Boltzmann law.

The numerical integration techniques used are the trapezoidal rule and Simpson's rule, a comment is made at the end of the notebook discussing the accuracy of each, and which has a higher accuracy relative to the 'experimental' results.

Task 1

The objective of this task is to calculate and display the spectral radiance of a blackbody by applying the Rayleigh-Jeans law at a certain frequency and temperature. The answer was checked by a hand calculation. The formula used is:

$$U(v,T) = \frac{8\pi v^2}{c^3}kT$$

1. Import the python library NumPy as np, which is the natural log funtion
2. Set the valeus of the constants k and c, the Boltzmann constant and the speed of light
3. Input the initial values for frequency and temperature
4. Calculate the spectral radiance using the Rayleigh-Jean law formula
5. Print the calculated spectral radiance

In [1]:
```python
# Task 1
#The Rayleigh Jean Law for a spectrum of a blackbody
# U(v,T) = (8piv^2/c^3)KT

# import NumPy for the natural log function
import numpy as np

# set the values of the constant
k = 1.38e-23 # Boltzmann constant (J K^-1)
c = 3.0e8 # speed of light (ms^-1)

# Input the initial value for frequency and temperature
v = 1.5e12 # Hz
T = 2500 #Kelvin

# Rayleigh-Jeans law for the spectrum of a Blackbody
U = ((8 * np.pi * v**2)/c**3) * k * T

#print the rayleigh jean law for the spectrum of a blackbody formatting the analytical equation
print('The Rayleigh Jean Law for the spectrum of a blackbody is = {0:10} W /(m^2 Hz) '.format(U))
```

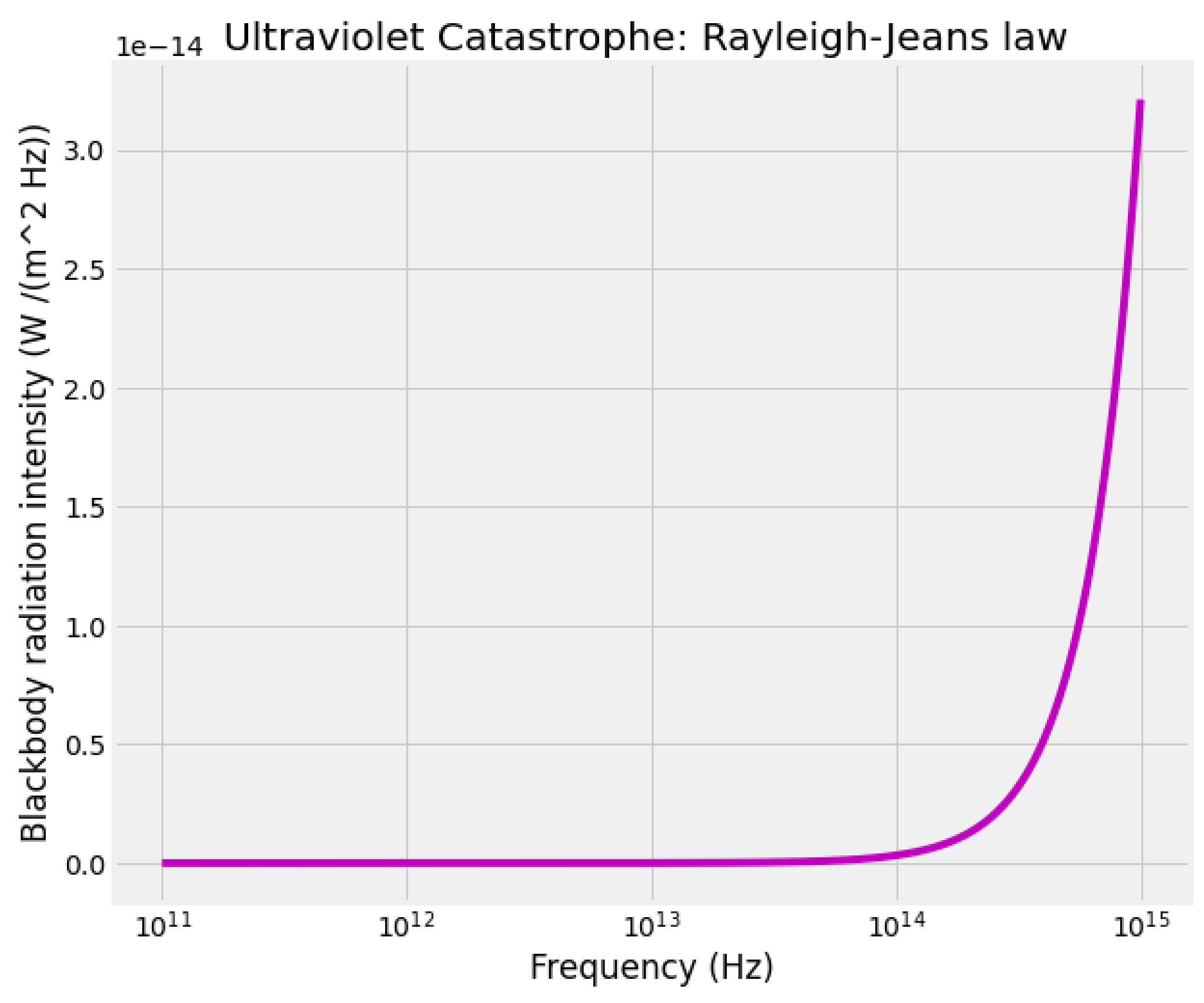
```
The Rayleigh Jean Law for the spectrum of a blackbody is = 7.225663103256526e-20 W /(m^2 Hz)
```

Task 2

Using the Rayleigh-Jeans law, this code creates a plot that illustrates 'Ultraviolet Catastrophe'. It demonstrates how, ata a given temperature, the spectral radiance varies with frequency, highlighting the problem where the law predicts unlimited radiation instensity at high frequencies, which is inconsostent with experimental observations, thus the 'catastrophe' - the spike in the graph. The equation used is the same as above.

1. Import python library matplotlib to plot the graph, and instruct the kernel to prepare for plot instructions
2. Input the initial value for temperature and create a linearly spaced array for frequency
3. Prepare the plot by adjusting the size of the figure and use a style sheet
4. Calculate the Rayleigh-Jeans law for this array of frequency values
5. Plot the spectral radiance as a function of frequency, and apply a logarithmic scale to the x-axis (frequency), label the axes, title the graph and display the plot

In [2]:
```python
# Task 2
# Plot of Rayleigh-Jeans law to demonstrate 'Ultraviolet Catastrophe'

# import matplotlib to plot graph
import matplotlib.pyplot as plt
# instruction to the kernel to prepare for plot instructions
%matplotlib inline

# Input the initial value for frequency and temperature
T = 2500 #Kelvin
f_101 = np.linspace(1e11, 1e15, 101) # plotting for lienarly spaced arrays (Hz)


# plotting the Rayleigh-Jeans law vs. the log scale of frequency
# increase the figure size
plt.figure(figsize=(10, 8))
plt.style.use('fivethirtyeight') # for background use style sheet from matplotlib


U_frequency = ((8 * np.pi * f_101**2)/c**3) * k * T # Rayleigh Jeans law
plt.xscale("log") # log applied to all x-values

plt.plot(f_101, U_frequency, 'm-')


plt.xlabel('Frequency (Hz)') # increase axes font
plt.ylabel('Blackbody radiation intensity (W /(m^2 Hz))')
plt.title('Ultraviolet Catastrophe: Rayleigh-Jeans law ')

plt.show()
```



Task 3

For the spectral radiance emitted by a blackbody, Planck came up with a new function which is as follows:

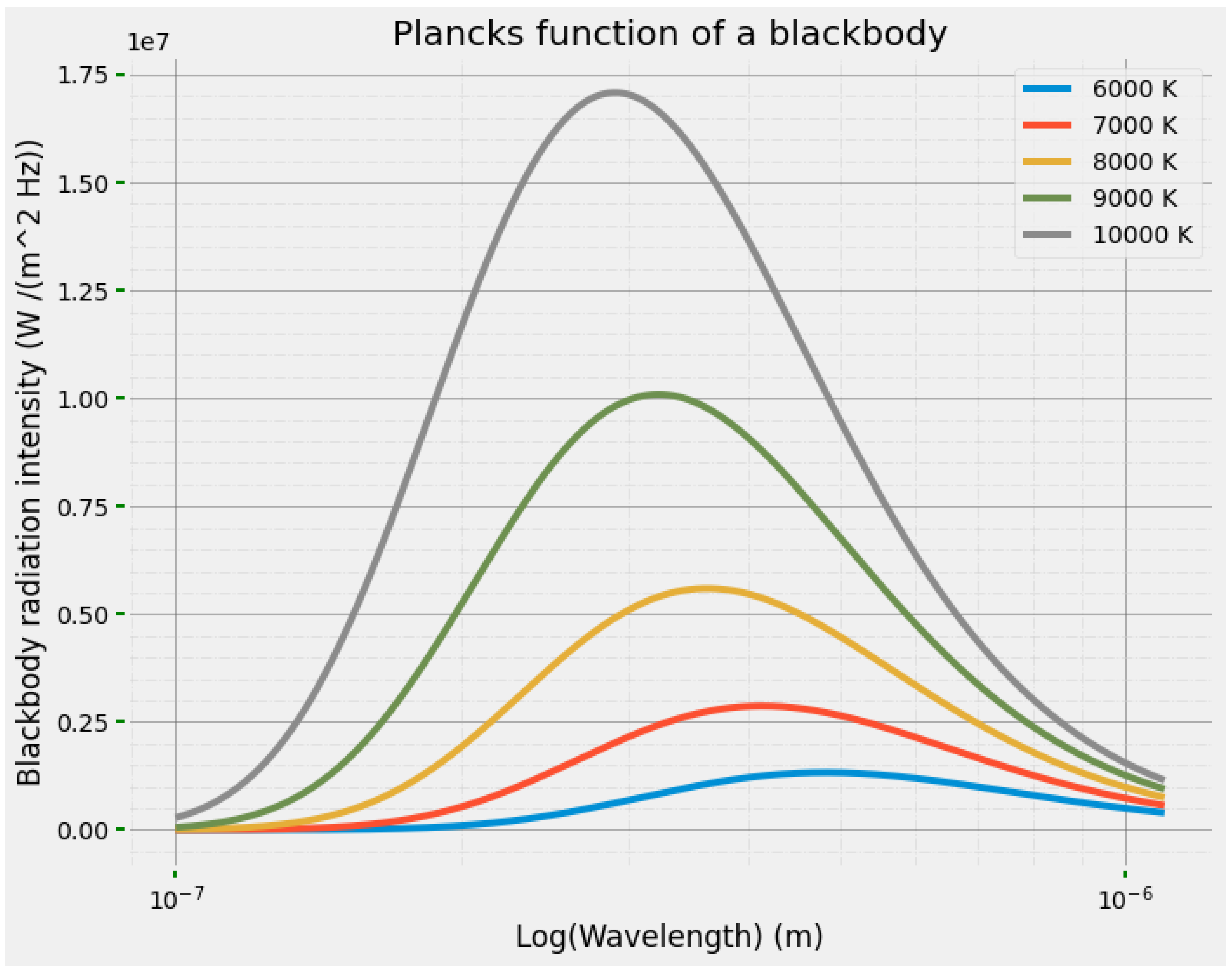
$$u(\lambda, T) = \frac{8\pi hc}{\lambda^5}\frac{1}{e^{\frac{hc}{\lambda kT}} - 1}$$

where h is Planck's constant, = 6.626x10-34 J s

This code plots a graph of this equation vs. wavelength, with plots of various temperatures. It provides a visual representation of the relationship between temperature and the emitted radiation spectrum according to a Planck's law. The graph has log scales on the x axis for better visualisation.

1. Set the value of Planck's constant
2. Define the temperatures to plot by assigning them to an array
3. Define the wavelength range
4. Define a function called plancks_function:
   - This function calculates the spectral radiance of a blackbody at different wavelengths for a given temperature using Planck's radiation law.
   - It calculates the exponent of the exponential term and uses it to compute the spectral radiance.
   - The function returns the calculated spectral radiance, representing the intensity of electromagnetic radiation emitted by a blackbody at different wavelengths for the given temperature.
5. Increase the figure size
6. Create a for loop to iterate through temperatures and plot planck's function for each
   - The label argument is used to label each curve with the corresponding temperature in Kelvin.
7. Apply a logarithmic scale to the x-axis (wavelength) for better visualization.
8. Set axes labels, title, and customize tick parameters, grid lines, and legend to improve the plot's appearance and readability.
9. Display the graph

In [3]:
```python
# Task 3
# Plotting wavelength version of Planck's function
# equation: u(λ,T) = (πhc/λ^5)* (1/(e^(hc)/λkT)− 1)

# set the value of the constants
h = 6.626e-34 # Planck's constant (J*s)

# array of temperatures to plot
temperatures = [6000, 7000, 8000, 9000, 10000] # in Kelvin

# wavelength range (in metres)
lambda_range = np.linspace(1e-7, 11e-7, 1001) # in metres - linearly spaced array


# define a function called 'plancks_function' that calculates the spectral radiance
# of a blackbody at different wavelengths for a given temperature using Planck's radiation law
def plancks_function(lambda_range, temperature):
    # calculating the exponent of the exponential term in Planck's law
    exponent = (h * c)/(lambda_range * k * temp)

    # calculate spectrum of the blackbody using the limited exponent to compute the spectral radiance
    U_wavelength = ((8 * np.pi * h * c)/lambda_range**5) * (1/( -1 + np.exp(exponent)))

    #the function returns the calculated spectral radiance,which represents the intensity of electromagnetic radiation
    # emitted by a blackbody at different wavelengths for the given temperature
    return U_wavelength


# increase the figure size
plt.figure(figsize=(10, 8))

# loop through temperatures and plot Planck's function for each
for temp in temperatures:

    # planck's law
    U_wavelength = plancks_function(lambda_range, temp)

    plt.xscale("log") # log applied to all x-values
    # plot with label showing the different temperatures
    plt.plot(lambda_range , U_wavelength, label=f'{temp} K') # label each curve


# adding title axes and title of graph
plt.xlabel('Log(Wavelength) (m)')
plt.ylabel('Blackbody radiation intensity (W /(m^2 Hz))')
plt.title('Plancks function of a blackbody')
plt.tick_params(axis='both', direction='out', length=6, width=2, labelcolor='k', colors='g', grid_color='gray', grid_alpha=0.5) # make axes ticks green and writing black
plt.grid(visible=True, color='dimgray',linestyle='-', linewidth=1) # adjust major grid lines to grey
plt.minorticks_on() # add minor ticks
plt.grid(visible=True, which='minor', color='lightgray', alpha=0.5, ls='-.', lw=1) # adjust minor grid lines to faint grey



plt.legend() # adding legend to see different temperatures
plt.show()
```



Task 4

The programme was then developed to find the Peak of the Planck function for each temperature. In a new cell, from this information Wien's displacemnet constant was calculated.

$$\lambda_{peak}T = Constant$$

1. Define a function called find_peak_wavelength:
   - This function calculates the peak wavelength for a given temperature using Planck's function.
   - It calculates Planck's function U_wavelength for the specified temperature.
   - It finds the index of the maximum value in U_wavelength, which corresponds to the peak wavelength.
   - The function returns the peak wavelength.
2. Create a loop to iterate through each temperature:
   - peak_wavelengths = []: Initializes an empty list called peak_wavelengths to store the peak wavelengths for each temperature.
   - The loop iterates through the list of temperatures and calls the find_peak_wavelength function for each temperature.
   - For each temperature, it calculates the peak wavelength and appends it to the peak_wavelengths list. As a result, peak_wavelengths stores the peak wavelengths corresponding to different temperatures.
3. Display the results

In [4]:
```
# define function called find_peak_wavelength to calculate the peak wavelength for a function
def find_peak_wavelength(temperature):
    U_wavelength = plancks_function(lambda_range, temperature) # planck's function
    peak_wavelength_i = np.argmax(U_wavelength) # finding the wavelength (index) at which planck's law reaches a maximum
    peak_wavelength = lambda_range[peak_wavelength_i]
    return peak_wavelength # returning peak wavelength

# for loop iterating through each temperature to find peak wavelength for each
peak_wavelengths = [] # initialising an empty list to store the max wavelength for each tenmperature
for temp in temperatures:
    peak_wavelength = find_peak_wavelength(temp) # calls the function and assigns the max wavlength to the variable peak_wavelength
    peak_wavelengths.append(peak_wavelength) # this adds the calculated peak-wavelength to the peak_wavelengths list, as it loops through different temperatures
    # each temperatures corresponding max wavelength is added to it

# Display the results by iterating through a loop of pairs of temperature and peak wavelength values
for temp, peak_wavelength in zip(temperatures, peak_wavelengths): # iterating over pairs of values from two lists; 'temperatures' and 'peak_wavelengths'
    print(f'Temperature: {temp} K, Peak Wavelength: {peak_wavelength :.2e} m') # printing to two significant figures
```

```
Temperature: 6000 K, Peak Wavelength: 4.84e-07 m
Temperature: 7000 K, Peak Wavelength: 4.14e-07 m
Temperature: 8000 K, Peak Wavelength: 3.63e-07 m
Temperature: 9000 K, Peak Wavelength: 3.22e-07 m
Temperature: 10000 K, Peak Wavelength: 2.90e-07 m
```

In [5]:
```
# task 4 continued
# evaluating the constant in Wien's displacemnet law for each temperature
for temp, peak_wavelength in zip(temperatures, peak_wavelengths): # iterating over pairs of values from two lists; 'temperatures' and 'peak_wavelengths'
    wien = temp * peak_wavelength # Wien's displacement law λmaxT=Constant
    print(f'Wiens displacement law = {wien * 1e3:.3f}x10-3 m*K') # printing for 4 significant figures to avoid rounding off error

print(f'Average of Wiens displacement constant: {((2.904+2.898+2.904+2.898+2.900)/5) * 1:.2f}x10-3 m*K') # averaging values calculated
# printing to 2 significant digits after the decimal point as the peak wavelength values had the same number of digits after the decimal pt.
```

```
Wiens displacement law = 2.904x10-3 m*K
Wiens displacement law = 2.898x10-3 m*K
Wiens displacement law = 2.904x10-3 m*K
Wiens displacement law = 2.898x10-3 m*K
Wiens displacement law = 2.900x10-3 m*K
Average of Wiens displacement constant: 2.90x10-3 m*K
```

Printing to 2 decimal places as that was how accurate the peak wavelength was.

Task 5

In this task, a numerical integration of the Planck function was carried out for a few temperatures to verify that the area under the curve is proportional to T^4 as in the Stefan-Boltzmann law

$$P = \sigma T^4$$

The numerical integration techniques used are the trapezoidal rule and Simpson's rule.

1. Import the integrate package from SciPy
2. Create a for loop to perform integration for each temperature, the factor of 1e8 is applied to the function values to ensure that the units match
   - Perform numerical integration using the trapezoidal rule
   - Perform numerical integration using Simpson's rule
3. Calculate T^4 and the power from the Stefan-Boltzmann law
4. Print the results

In [6]:
```
# task 5
# numerical integration of the Planck function for a few temperatures to verify that the area under the curve is proportional to T^4 as in the Stefan-Boltzmann law
# from the scipy python library import the integrate package to use numerical integration methods
from scipy import integrate


# Calculate the integral using both trapezoidal rule and Simpson's rule
# for loop iterating through each temperature while defining plancks radiation law to find the area under teh graph for each curve (the power)
for temp in temperatures:

    # plancks radiation law
    U_wavelength = plancks_function(lambda_range, temp)
    # Numerical integration using the trapezoidal rule
    integral_trapezoidal = integrate.trapezoid(U_wavelength*1e8, lambda_range) # multiply wavelength by factor of


    # Numerical integration using Simpson's rule
    integral_simpson = integrate.simpson(U_wavelength*1e8, lambda_range)


    # Calculate T^4 and the power, from the Stefan-Boltzmann law
    sigma = 5.670374419e-8 # stefann-boltzmann constant, units: W / (m2 x K4)
    temp4 = temp**4 #Calculating each temperature to the power of 4
    power =  sigma * temp4 # stefann-boltzmann law

    # Print the results
    print(f'Temperature: {temp} K') # temperature used in calculation
    print(f'Trapezoidal Rule Integral: {integral_trapezoidal :.8e}') # printing to 8 significant digits to clearlt compare values that are similar
    print(f'Simpson\'s Rule Integral: {integral_simpson :.8e}')
    print(f'Power: {power:.5e}')
    print()
```

```
Temperature: 6000 K
Trapezoidal Rule Integral: 7.64402912e+07
Simpson's Rule Integral: 7.64402989e+07
Power: 7.34881e+07

Temperature: 7000 K
Trapezoidal Rule Integral: 1.52570145e+08
Simpson's Rule Integral: 1.52570158e+08
Power: 1.36146e+08

Temperature: 8000 K
Trapezoidal Rule Integral: 2.72812688e+08
Simpson's Rule Integral: 2.72812713e+08
Power: 2.32259e+08

Temperature: 9000 K
Trapezoidal Rule Integral: 4.50806072e+08
Simpson's Rule Integral: 4.50806146e+08
Power: 3.72033e+08

Temperature: 10000 K
Trapezoidal Rule Integral: 7.01864200e+08
Simpson's Rule Integral: 7.01864445e+08
Power: 5.67037e+08
```

In this case, the trapezoidal rule is more accurate than the Simpsons ruile because there is a less of a difference from that value compared to the analytical power value calculated. However, the Simpsons rule is generally deemed more accurate, so there may be issues when teh integration deals with lower and higher wavelengths.