Adhikari_Insan_HW_01

January 29, 2025

###

The Planck Law

$$B(\lambda, T) = \frac{2hc^2}{\lambda^5 \left(exp\left[\frac{hc}{\lambda kT}\right] - 1\right)}$$

"…where h is Planck's constant, c is the speed of light, k is Boltzmann's constant, is the emission wavelength, and T is the temperature of the blackbody…"

#==========#

Question 1:

Q1. Write a program that determines where the peak of the Planck function occurs in wavelength for a given temperature. You must use a while loop to do this.

(12 pts) a) If the temperature of an object is 13,000 K, what is the peak wavelength? Use 200 nm as your initial guess, and a step-size of +5 nm. You cannot use the 'max' function in Python, nor should you take a derivative. You are essentially writing your own max function. You must solve this problem with a while loop.

(8 pt) b) Print your list(s) of wavelength and B values. Make sure to format your output so that wavelength has zero decimal points and is expressed in nm, and B is expressed in scientific notation with 2 decimal points.

(1 pt) c) How many iterations did it take for your algorithm to find it?

#==========#

First we need to get the constants we will use in the formula. We will use the scipy library to do so. However, **scipy** constants are in SI and will need to be converted into cgs for our use.

We will also use **numpy** for general mathmatics.

[260]:
```
import numpy as np
from scipy import constants as const
```

I will setup h, k, and c as global variables in cgs for ease of use

[261]:
```
#Locating h and its value in scipy
print(const.find("Planck")) # find h is called "Planck constant" in scipy
print(const.physical_constants["Planck constant"],"\n") # h is in J * s
```

```
# since 1 J = 1e7 erg, h_cgs = h * 1e7
h_cgs = const.physical_constants["Planck constant"][0] * 1e7 # first element is␣
 ↪the value of the constant
print(f"h in cgs = {h_cgs} erg * s,\n") # 6.62607015e-27 erg * s

# ============================================================================
#Locating k and its value in scipy
print(const.find("Boltzman")) # find k is called "Boltzmann constant" in scipy
print(const.physical_constants["Boltzmann constant"],"\n") # k is in J/K

# since 1 J = 1e7 erg, k_cgs = k * 1e7
k_cgs = const.physical_constants["Boltzmann constant"][0] * 1e7 # first element␣
 ↪is the value of the constant
print(f"k in cgs = {k_cgs} erg/K,\n") # 1.380649e-16 erg/K

# ============================================================================
#Locating c and its value in scipy
print(const.find("light")) # find c is called "speed of light in vacuum" in␣
 ↪scipy
print(const.physical_constants["speed of light in vacuum"],"\n") # k is in m/s

# since 1 m = 100 cm, c_cgs = c * 100
c_cgs = const.physical_constants["speed of light in vacuum"][0] * 100 # first␣
 ↪element is the value of the constant
print(f"c in cgs = {c_cgs} m/s,\n") # 29979245800.0 m/s
```

```
['Planck constant', 'Planck constant in eV/Hz', 'Planck length', 'Planck mass',
'Planck mass energy equivalent in GeV', 'Planck temperature', 'Planck time',
'molar Planck constant', 'reduced Planck constant', 'reduced Planck constant in
eV s', 'reduced Planck constant times c in MeV fm']
(6.62607015e-34, 'J Hz^-1', 0.0)

h in cgs = 6.62607015e-27 erg * s,

['Boltzmann constant', 'Boltzmann constant in Hz/K', 'Boltzmann constant in
eV/K', 'Boltzmann constant in inverse meter per kelvin', 'Stefan-Boltzmann
constant']
(1.380649e-23, 'J K^-1', 0.0)

k in cgs = 1.380649e-16 erg/K,

['speed of light in vacuum']
(299792458.0, 'm s^-1', 0.0)

c in cgs = 29979245800.0 m/s,
```

I will create a funtion to calculate the equation for a given wavelength and temperature to make code easier to read.

[262]:
```
def blackBodyRad(lamda, temp):
    # function to calculate black body radiance for an object given wavelength␣
 ↪and temperature
    return (2 * h_cgs * c_cgs**2)/(lamda**5 * (np.exp((h_cgs * c_cgs) / (lamda␣
 ↪* k_cgs * temp) ) - 1))
```

Q1) Part A

– If the temperature of an object is 13,000 K, what is the peak wavelength? Use 200 nm as your initial guess, and a step-size of +5 nm. You cannot use the 'max' function in Python, nor should you take a derivative. You are essentially writing your own max function. You must solve this problem with a while loop.

[263]:
```
temp = 13000 # K
initLamda = 200 * 1e-7 # 200 nm = 200 * 1e-9 m = 200 * 1e-7 cm
stepSize = 5 * 1e-7 # 5 nm = 5 * 1e-9 m = 5 * 1e-7 cm
maxIter = 100 # maximum number of iterations for while loop

radiances = []
waveLengths = []

i = 1
lambda_ = initLamda # tracks the current wavelength (i think lambda is a␣
 ↪reserved keyword in python so i used "lambda_")
waveLengths.append(lambda_)
maxRad = blackBodyRad(lambda_, temp) # initial value of radiance at 200 nm
radiances.append(maxRad)
increasing = True # bool to check if radiance is increasing or decreasing
peakWavelength = lambda_ # store the wavelength corresponding to the peak␣
 ↪radiance

while i < maxIter + 1 and increasing:
    lambda_ += stepSize # increment wavelength by 5 nm
    bbRad = blackBodyRad(lambda_, temp) # bbRad = black body radiance,␣
 ↪calculate the next radiance value
    if bbRad > maxRad: # if the new radiance is greater than the previous peak␣
 ↪radiance
        maxRad = bbRad # update peak radiance
        peakWavelength = lambda_ # update peak wavelength
        increasing = True
    else:
        increasing = False

    waveLengths.append(lambda_)
    radiances.append(bbRad)
```

```
    i += 1

print("a) Peak values\n")
print("Peak Values:")
print(f"Peak Radiance = {maxRad:.3e} erg s^-1 cm^-2 sr^-1 Å^-1")
print(f"Peak Wavelength = {peakWavelength:.3e} cm = {peakWavelength * 1e7:.0f}␣
 ↪nm \n")

print("\nb) Table of wavelength and B  values\n")
# Print table header
print(f"#{'='*16}#{'='*30}#")
print(f"#{'Wavelength':^16}#{'Radiance [B ]':^30}#")
print(f"#{'(nm)':^16}#{"(erg s^-1 cm^-2 sr^-1 Å^-1)":^30}#")
print(f"#{'='*16}#{'='*30}#")

# Print data rows using a while loop
j = 0
while j < len(waveLengths):
    print(f"#{(waveLengths[j] * 1e7):^16.0f}#{radiances[j]:^30.2e}#")
    j += 1
# close off table
print(f"#{'='*16}#{'='*30}#\n")

# print the number of iteration to find the peak radiance
print("\nc) Number of iterations")
print(f"\nIterations:{i}")
```

```
a) Peak values

Peak Values:
Peak Radiance = 1.520e+16 erg s^-1 cm^-2 sr^-1 Å^-1
Peak Wavelength = 2.250e-05 cm = 225 nm


b) Table of wavelength and B  values

#================#==============================#
#   Wavelength   #        Radiance [B ]         #
#      (nm)      # (erg s^-1 cm^-2 sr^-1 Å^-1)  #
#================#==============================#
#      200       #           1.48e+16           #
#      205       #           1.49e+16           #
#      210       #           1.51e+16           #
#      215       #           1.52e+16           #
#      220       #           1.52e+16           #
#      225       #           1.52e+16           #
#      230       #           1.52e+16           #
#================#==============================#
```

c) Number of iterations

Iterations:7

Rounding the wavelengths to just 2 decimal points doesnt show but the values do peak at 225 nm.

B(220 nm, 13000 K) = 1.5200607648211218e+16 erg s^-1 cm^-2 sr^-1 Å^-1

B(225 nm, 13000 K) = 1.5203755562099896e+16 erg s^-1 cm^-2 sr^-1 Å^-1

B(230 nm, 13000 K) = 1.5171391963532882e+16 erg s^-1 cm^-2 sr^-1 Å^-1

#==========#

Question 2:

Q2. (2 pt) a) Calculate the peak wavelength of the blackbody curve using Wien's Displacement Law.

Wien's Law calculates the peak wavelength of a blackbody curve based on the temperature of the object in question.

$$\lambda_{max} = \frac{2.898 \cdot 10^{-3}}{T}$$

where max is the peak wavelength in meters [m], T is the blackbody temperature in Kelvin [K], and the constant is in [m · K]. Use 13,000 K for the temperature. Your answer should have zero decimal points (for example: "300 nm").

(3 pt) b) How close is your result in the previous question (through Planck's law) to the analytical result (Wien's Law)? Calculate the difference and print it.

(4 pts) c) Re-run your algorithm in Question 1 with a step-size of 1 nm. How many iterations did it take to find the peak? How close is your result to Wien's Law now? Again, print your list(s) of wavelength and B values.

#==========#

[264]:
```
 _max = 2.898e-3 / temp # Find the peak analytically using Wien's Law
print(f"a) Wien's Law\n")
print(f" _max = { _max:.3e} m = { _max * 1e9:.0f} nm\n")

print("\nb) Percent Error\n") # Find the percent error between the two methods
print(f"Iterative Peak Radiance = {peakWavelength} cm = {peakWavelength * 1e-2}␣
 ↪m")
Δ  = peakWavelength * 1e-2 -  _max
print(f"Δ  = {peakWavelength * 1e-2} m - { _max} m = {Δ }")
print(f"Error = Δ / _max = {Δ } m / { _max} m = {Δ / _max}")
print(f"\nPercent Error = {Δ / _max * 100:.3f}%\n")
```

a) Wien's Law

```
_max = 2.229e-07 m = 223 nm


b) Percent Error

Iterative Peak Radiance = 2.249999999999999e-05 cm = 2.249999999999999e-07 m
Δ  = 2.249999999999999e-07 m - 2.2292307692307692e-07 m = 2.0769230769229907e-09
Error = Δ / _max = 2.0769230769229907e-09 m / 2.2292307692307692e-07 m =
0.009316770186335017

Percent Error = 0.932%
```

(4 pts) c) Re-run your algorithm in Question 1 with a step-size of 1 nm. How many iterations did it take to find the peak? How close is your result to Wien's Law now? Again, print your list(s) of wavelength and B values.

[265]:
```
print("\nc) Rerun w/ iteration step of 1 nm:\n")
stepSize = 1 * 1e-7 # 1 nm = 1 * 1e-9 m = 1 * 1e-7 cm
i = 1
lambda_ = initLamda # still start @ 200 nm
# clear lists
waveLengths = []
radiances = []

i = 1
lambda_ = initLamda # reset lambda to 200 nm

# reset other variables
waveLengths.append(lambda_)
maxRad = blackBodyRad(lambda_, temp) # initial value of radiance at 200 nm
radiances.append(maxRad)
increasing = True
peakWavelength = lambda_

while i < maxIter + 1 and increasing:
    lambda_ += stepSize # increment wavelength by 5 nm
    bbRad = blackBodyRad(lambda_, temp) # bbRad = black body radiance,␣
 ↪calculate the next radiance value
    if bbRad > maxRad: # if the new radiance is greater than the previous peak␣
 ↪radiance
        maxRad = bbRad # update peak radiance
        peakWavelength = lambda_ # update peak wavelength
        increasing = True
    else:
        increasing = False

    waveLengths.append(lambda_)
```

```
    radiances.append(bbRad)
    i += 1

print("Peak Values:")
print(f"Peak Radiance = {maxRad:.3e} erg s^-1 cm^-2 sr^-1 Å^-1")
print(f"Peak Wavelength = {peakWavelength:.3e} cm = {peakWavelength * 1e7:.0f}␣
 ↪nm \n")

print("\n-- Table of wavelength and B  values (at 1 nm persicion)\n")
# Print table header
print(f"#{'='*16}#{'='*30}#")
print(f"#{'Wavelength':^16}#{'Radiance [B ]':^30}#")
print(f"#{'(nm)':^16}#{"(erg s^-1 cm^-2 sr^-1 Å^-1)":^30}#")
print(f"#{'='*16}#{'='*30}#")

# Print data rows using a while loop
j = 0
while j < len(waveLengths):
    print(f"#{(waveLengths[j] * 1e7):^16.0f}#{radiances[j]:^30.2e}#")
    j += 1
# close off table
print(f"#{'='*16}#{'='*30}#\n")

# print the number of iteration to find the peak radiance
print("\n-- Number of iterations")
print(f"Iterations:{i}")
```

c) Rerun w/ iteration step of 1 nm:

```
Peak Values:
Peak Radiance = 1.521e+16 erg s^-1 cm^-2 sr^-1 Å^-1
Peak Wavelength = 2.230e-05 cm = 223 nm


-- Table of wavelength and B  values (at 1 nm persicion)

#================#==============================#
#   Wavelength   #        Radiance [B ]         #
#      (nm)      # (erg s^-1 cm^-2 sr^-1 Å^-1)  #
#================#==============================#
#      200       #           1.48e+16           #
#      201       #           1.48e+16           #
#      202       #           1.48e+16           #
#      203       #           1.49e+16           #
#      204       #           1.49e+16           #
#      205       #           1.49e+16           #
#      206       #           1.50e+16           #
```

```
#      207       #           1.50e+16           #
#      208       #           1.50e+16           #
#      209       #           1.51e+16           #
#      210       #           1.51e+16           #
#      211       #           1.51e+16           #
#      212       #           1.51e+16           #
#      213       #           1.51e+16           #
#      214       #           1.51e+16           #
#      215       #           1.52e+16           #
#      216       #           1.52e+16           #
#      217       #           1.52e+16           #
#      218       #           1.52e+16           #
#      219       #           1.52e+16           #
#      220       #           1.52e+16           #
#      221       #           1.52e+16           #
#      222       #           1.52e+16           #
#      223       #           1.52e+16           #
#      224       #           1.52e+16           #
#================#==============================#


-- Number of iterations
Iterations:25
```

The new peak is more precise and higher than the previous peak with 5 nm increments

The raw peak with 1 nm increments is 1.5206946788337758e+16 @ 223 nm

The raw peak with 5 nm incremetns was 1.5203755562099896e+16 @ 225 nm

We have found a slightly higher peak when using 1 nm increments

[266]:
```
# Recalculate percent error

print("\nb) Percent Error (1 nm)\n")
print(f"Iterative Peak Radiance = {peakWavelength} cm = {peakWavelength * 1e-2}␣
  ↪m")
Δ  = peakWavelength * 1e-2 -  _max
print(f"Δ  = {peakWavelength * 1e-2} m - { _max} m = {Δ }")
print(f"Error = Δ / _max = {Δ } m / { _max} m = {Δ / _max}")
print(f"\nPercent Error = {Δ / _max * 100:.3f}%\n")
```

```
b) Percent Error (1 nm)

Iterative Peak Radiance = 2.2299999999999976e-05 cm = 2.2299999999999976e-07 m
Δ  = 2.2299999999999976e-07 m - 2.2292307692307692e-07 m = 7.69230769228404e-11
Error = Δ / _max = 7.69230769228404e-11 m / 2.2292307692307692e-07 m =
0.0003450655624558058
```

```
Percent Error = 0.035%
```