Plotting Planck Distribution Law at varius temperatures

Verification of Wien's Displacement Law and the Stefan Boltzmann Law

Importing the necessary Python Libraries

```
import numpy as np
import matplotlib.pyplot as pl
from scipy.integrate import simps as smp
```

Defining the constants

```
#h = 4.1357 * 10**(-15)     #Planck Constant in units of eV.s
h = 6.6261 * 10**(-34)     #Planck Constant in units of J.s
c = 2.9979 * 10**(8)       #Speed of light in vacuum in units of m.s^(-1)
#k = 8.6173 * 10**(-5)      #Boltzmann constant in units of eV.K^(-1)
k = 1.3806 * 10**(-23)     #Boltzmann constant in units of J.K^(-1)
b = 2.898 * 10**(-3)       #Wien constant in units of m.K
sigma = 5.670 * 10**(-8)   # Stefan- Boltzmann Constant in W m^(-2) K^(4)
#T = 5000                   # Temperature of the Blackbody in K
#lmbda_max = b/T            #Wien's Displacement Law
#print ('lambda max in m =', lmbda_max)
```

Defining the Planck Distribution Function

```
# Defining the Distribution functions
def Planck(T, lmbda):
    x = h*c/(lmbda*k*T)
    return (2 * h *c**2 /(lmbda**5)) * (1/(np.exp(x)-1))
```

Function to calculate the area under the Planck Curve

```
def Planck_area(lmbda,Temp):
  integrand = Planck(Temp, lmbda)
  #print('From MeanSpeed integrand =', integrand)
  return smp(integrand, lmbda)
```

Defining the wavelength scales and the different temperatures

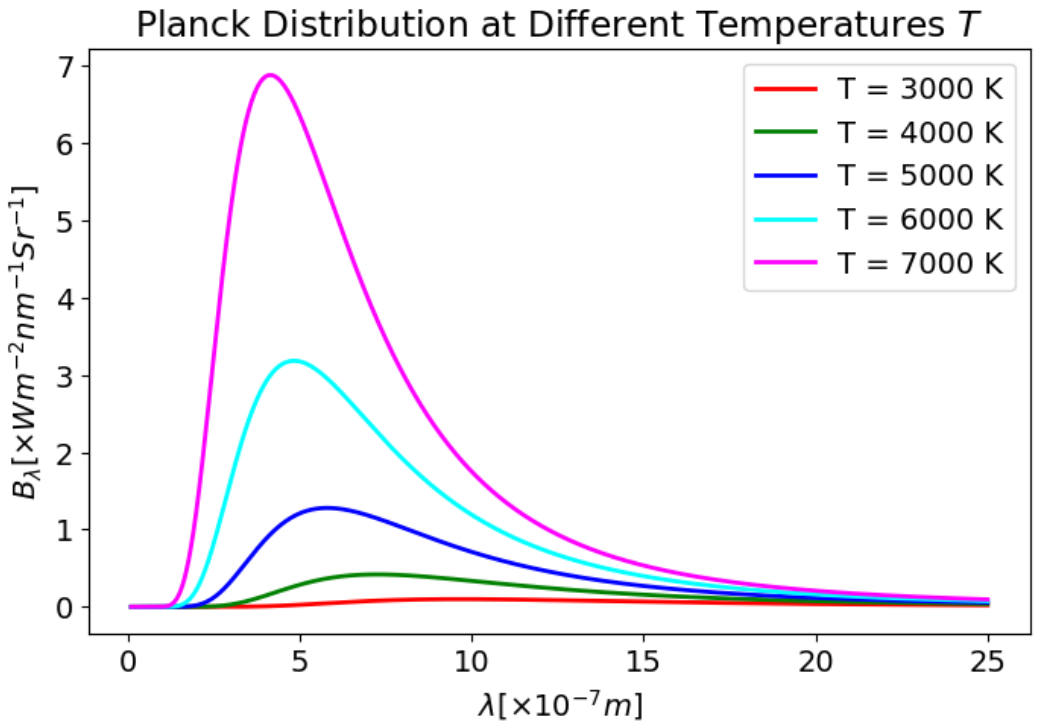
```
# Defining and setting the scales of the independent variables T and \lambda
lmbda = np.arange(0.1,25.0,0.001)   #defining the array of wavelength
lmbda *= 10**(-7)         # Wavelength lambda in micrometers
T = [3000, 4000, 5000, 6000, 7000]    # Range of temperatures in Kelvin
Temp = np.linspace(2000, 8000, 100)
T_Label = [ 'T = 3000 K', 'T = 4000 K', 'T = 5000 K', 'T = 6000 K', 'T = 7000 K']
T_Col = ['red', 'green', 'blue', 'cyan', 'magenta']
# lmbda_max_cal = np.zeros([len(Temp)])
# power_cal = np.zeros([len(Temp)])
# for i in range(len(Temp)):
#   lmbda_max_cal[i] = b/float(Temp[i])
#   power_cal[i] = sigma * (Temp[i]**4)
```

Calculating the Planckian for different temperatures

```
# Calulating the Planck Distribution Function
B_Plnck = np.zeros([len(T), len(lmbda)])
for i in range(len(T)):
    B_Plnck[i,:] = Planck(T[i], lmbda)
```

Plotting the Planckian for different temperatures

```
pl.rcParams["figure.figsize"]=[8,5]
pl.rcParams.update({'font.size': 14})
for i in range(len(T)):
  pl.plot(lmbda*(10**7), B_Plnck[i,:]/(10**13), linewidth = 2, label = T_Label[i], c = T_Col[
  pl.xlabel(r'$\lambda [\times 10^{-7} m]$')
  pl.ylabel(r'$B_\lambda [\times W m^{-2} nm^{-1} Sr^{-1}]$')
  pl.title('Planck Distribution at Different Temperatures $T$')
  pl.legend()
pl.show()
```



Calculation of theoretical values of λ_{max} and Power radiated by the blackbody surface at temperature T

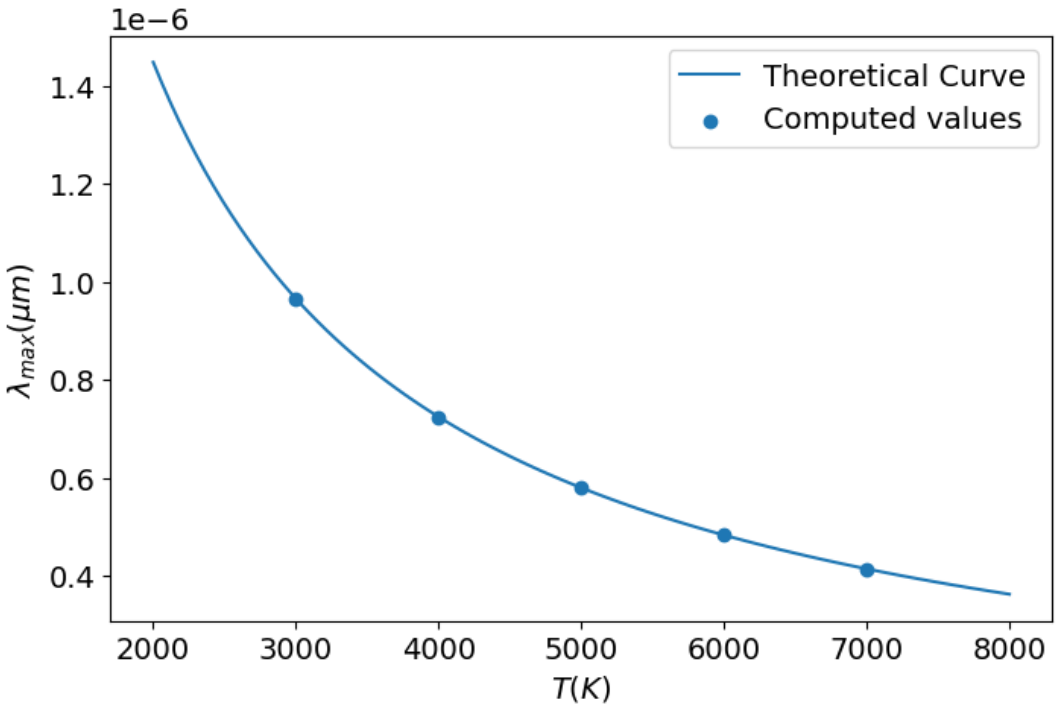
```
lmbda_max_cal = np.zeros([len(Temp)])
power_cal = np.zeros([len(Temp)])
for i in range(len(Temp)):
  lmbda_max_cal[i] = b/float(Temp[i])
  power_cal[i] = sigma * (Temp[i]**4)

power_cal = power_cal/np.max(power_cal)
```

Computation of λ_{max} from the plotted Planckians

```
maxval_B_index = np.argmax(B_Plnck, axis =1)
#print (maxval_B_index)
lmbda_max = lmbda[maxval_B_index]
#print(lmbda_max)

# print ("The" r"($\lambda_{max}$ at different temperatures")
# for i in range (len(T)):
#   print ("Temperature", T[i], "K:  " "%10.7e" %lmbda_max[i], "nm")
```

Verification of Wien's Displacement Law

```
pl.rcParams["figure.figsize"]=[8,5]
pl.rcParams.update({'font.size': 14})
pl.plot(Temp, lmbda_max_cal, label = 'Theoretical Curve')
pl.scatter (T, lmbda_max, label = 'Computed values')
pl.ylabel(r'$\lambda_{max} (\mu m)$ ')
pl.xlabel(r'$T(K)$')
pl.legend()
pl.show()
```



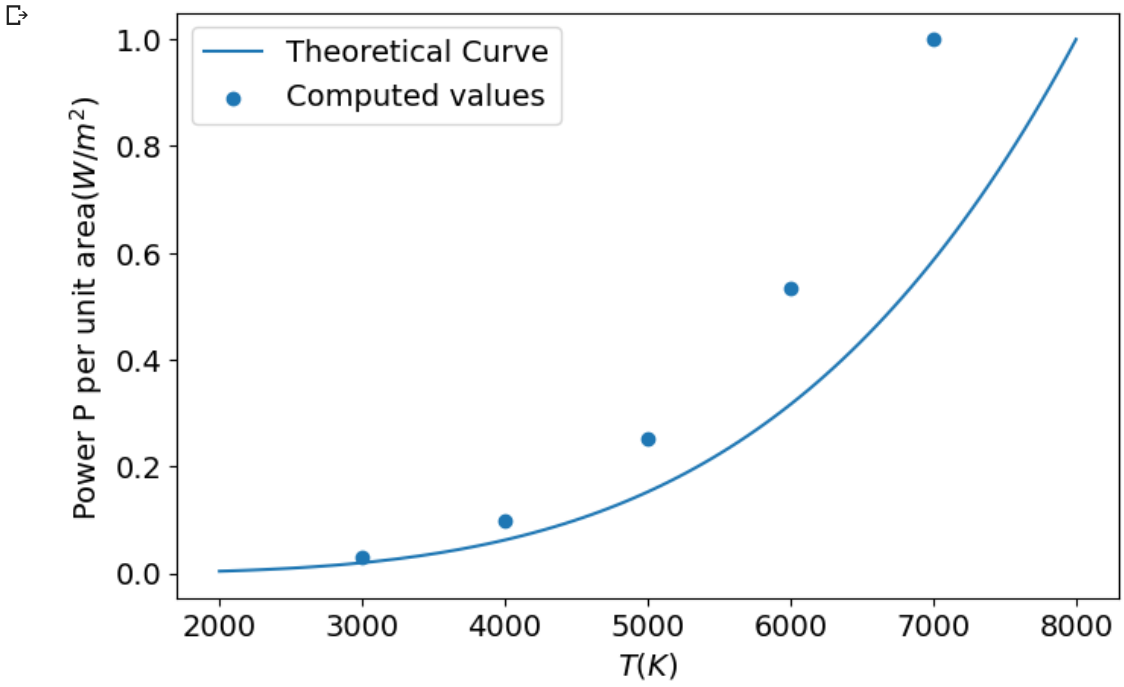
Computation of the area under the Planckians at different temperatures

```
power = np.zeros([len(T)])

for i in range(len(T)):
  power[i] = Planck_area(lmbda,T[i])

power = power/np.max(power)
#print ("The power per unit area radiated at different temperatures")
# for i in range (len(T)):
#   print ("Temperature", T[i], "K:  " "%10.7e" %power[i], "W")
```

Verification of the Stefan Boltzmann Law

- Difference because the Stefan Boltzmann Constant calculated and computed values are differing

```
pl.rcParams["figure.figsize"]=[8,5]
pl.rcParams.update({'font.size': 14})
pl.plot(Temp, power_cal, label = 'Theoretical Curve' )
pl.scatter(T, power, label = 'Computed values')
pl.ylabel('Power P per unit area' r'$(W/m^2)$ ')
pl.xlabel(r'$T(K)$')
pl.legend()
pl.show()
```




```
# log_T = np.log(T)
# log_power = np.log(power)
# fit = np.polyfit(log_T, log_power,1)
# print(fit[0])
# print(fit[1])
# sigma_cal = np.exp(fit[1])
# print(sigma_cal)
```

```
4.186755891207756
-19.48773967839921
3.440188128980995e-09
```

✓ 0s completed at 8:24 PM