Computational Physics Assignment 1

Dara Corr - 18483836

In Task 1 of this assignment I wrote a programme which prints a values of Rayleigh-Jeans Law for the spectrum of a blackbody

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

for user-inputted frequency $v(Hz)$ and Temperature $T(K)$ where $k = 1.38x10^{-23} JK^{-1}$ and $c = 3.0x10^8 ms^{-1}$

In Task 2: Rayleigh-Jeans Law was plotted over a range of frequencies from 10^{13} to 10^{15} to at $T = 2000K$ to show that this theory leads to the Ultraviolet Catastrophy

In Task 3: Planck's formula is graphed for a few temperatures alongside the formula for Rayleigh-Jeans Law and we see that it will solve the Ultra-Violet Catastrophy

$$U(v,T) = \frac{8\pi hv^3}{c^3} \frac{1}{exp(hv/kT) - 1}$$

Where h = $6.63x10^{-34} Js$

In Task 4: I found the Peak of the Planck Function U for each temperature and from this evaluated the peak wavelength λ_{peak} for U at each temperature. From this I then verified that

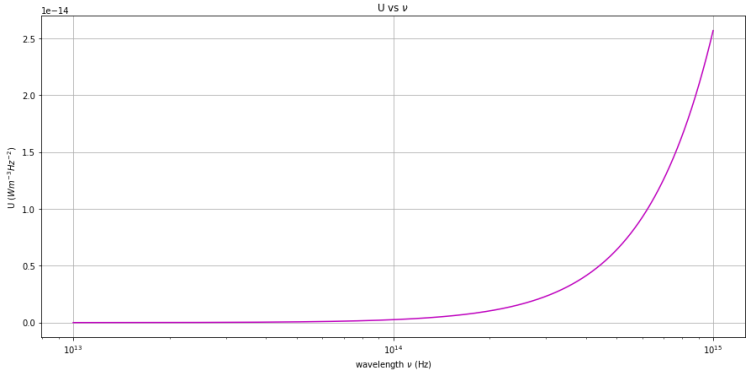
$$\lambda_{peak} T = Constant$$

In task 5: I carried out Numerical Integration of Planck's function using the Rectangle Rule and The Trapezium Rule. Using this I showed that the Area under the curve is proportional to T^4 and I evaluated the proportionality constant and decided which value for the proportionality constant was most accurately evaluated.

In [15]:
```
#task 1
import numpy as np
import matplotlib.pyplot as plt #import numpy and matplotlib libraries

k = 1.38*10**(-23) #define boltzman constant k and the speed of light c
c = 3.0*10**8
nu = eval(input("Enter a value for the frequency Nu (Hz):")) #user is prompted to enter a value for nu
T = eval(input("Enter a value for the Temperature T (K):")) #user prompted to enter value for T
U = ((8*np.pi*nu**2)/(c**3))*k*T #definition of expression for U
print("your value of U is {0:5.2e}".format(U) + " Wm^-3Hz^-2") #prints value of U for the values
```

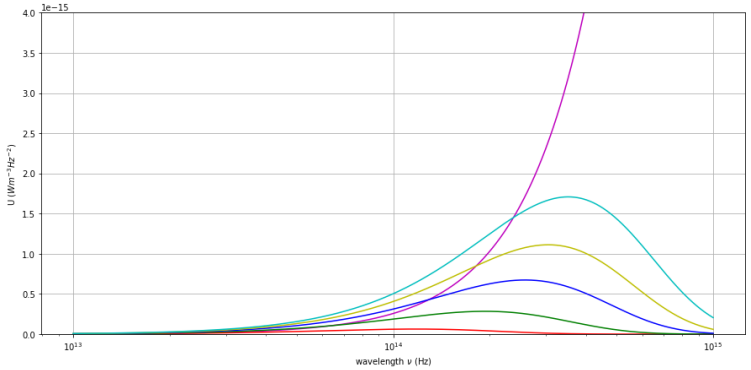
```
Enter a value for the frequency Nu (Hz):10**12
Enter a value for the Temperature T (K):2000
your value of U is 2.57e-20 Wm^-3Hz^-2
```

In [48]:
```
##Task 2
            #define constants
k = 1.38*10**(-23)
c = 3.0*10**8
T = 2000

nu = np.linspace(10**13, 10**15, 2000) #array of frequencies from 10**13 to 10**15
U =((8*np.pi*nu**2)/(c**3))*k*T #U defined here for Rayleigh-Jeans law


fig= plt.figure(figsize=(15,7)) #plot graph
plt.plot((nu), (U), "m-")
plt.xscale("log") #make x on log scale
plt.title("U vs $\\nu$")
plt.xlabel("wavelength $\\nu$ (Hz)")
plt.ylabel("U ($Wm^{-3}{Hz}^{-2}$)")
plt.grid()
plt.show()
```



This graph shows that Rayleigh-Jeans law leads to the ultraviolet catastrophy as the equation for U (the spectral radiance) agrees with physical predictions and behaviours of classical physics for low frequencies but once high frequencies around the ultraviolet spectrum are reached, U diverges and disagrees with classical observations as this graph implies (under classical theories) that a blackbody emits an unlimited amount of energy which contradicts the conservation of energy.

In [36]:
```
##Task 3

import numpy as np
import matplotlib.pyplot as plt

k = 1.38*10**(-23) #define boltzman constant k, the speed of light c and planck's constant h
c = 3.0*10**8
h = 6.63*10**-34

nu = np.linspace(10**13, 10**15, 2000) #array of frequencies from 10**13 to 10**15

U =((8*np.pi*nu**2)/(c**3))*k*T #U from Task 2 (Rayleigh-Jeans) to be plotted alongside new U plots (planck)
U1 = ((8*np.pi*h*nu**3)/(c**3))*(1/(np.exp((h*nu)/(k*2000))-1)) #U for T = 2000
U2 = ((8*np.pi*h*nu**3)/(c**3))*(1/(np.exp((h*nu)/(k*3300))-1)) #U for T = 3300
U3 = ((8*np.pi*h*nu**3)/(c**3))*(1/(np.exp((h*nu)/(k*4400))-1)) #U for T = 4400
U4 = ((8*np.pi*h*nu**3)/(c**3))*(1/(np.exp((h*nu)/(k*5200))-1)) #U for T = 5200
U5 = ((8*np.pi*h*nu**3)/(c**3))*(1/(np.exp((h*nu)/(k*6000))-1)) #U for T = 6000

fig= plt.figure(figsize=(15,7)) #plot graph
plt.plot(nu, U, "m-")
plt.plot(nu, U1, "r")
plt.plot(nu, U2, "g")
plt.plot(nu, U3, "b")
plt.plot(nu, U4, "y")
plt.plot(nu, U5, "c")
plt.xscale("log") #make x on log scale
plt.ylim(0,4*10**-15) #zoom into graph
plt.xlabel("wavelength $\\nu$ (Hz)")
plt.ylabel("U ($Wm^{-3}{Hz}^{-2}$)")
plt.grid()
plt.show()
```



We see here that Planck's Formula fixes Rayleigh-Jean's Law and thus solves the Ultra-Violet Catastrophy

In [39]:
```
#task 4

Nu = nu.tolist() #convert np arrays to python lists to use the .index() method
u1 = U1.tolist()
u2 = U2.tolist()
u3 = U3.tolist()
u4 = U4.tolist()
u5 = U5.tolist()

U_peak_values = np.array([np.max(u1),np.max(u2),np.max(u3),np.max(u4),np.max(u5)]) #array of max values for the Planck function

lambda_peak_values =  [0,0,0,0,0] #find the index of each peak lambda value
lambda_peak_values[0] = u1.index(np.max(u1))
lambda_peak_values[1] = u2.index(np.max(u2))
lambda_peak_values[2] = u3.index(np.max(u3))
lambda_peak_values[3] = u4.index(np.max(u4))
lambda_peak_values[4] = u5.index(np.max(u5))

#find the wavelength of each Lambda peak index
lambda_peak_values[0] = (3*10**8)/(Nu[lambda_peak_values[0]]) #c= f*Lambda => lambda = c/f
lambda_peak_values[1] = (3*10**8)/(Nu[lambda_peak_values[1]])
lambda_peak_values[2] = (3*10**8)/(Nu[lambda_peak_values[2]])
lambda_peak_values[3] = (3*10**8)/(Nu[lambda_peak_values[3]])
lambda_peak_values[4] = (3*10**8)/(Nu[lambda_peak_values[4]])


T_values = np.zeros(5) #make array of T values from task 3 to multiply by each U value
T_values[0] = 2000
T_values[1] = 3300
T_values[2] = 4400
T_values[3] = 5200
T_values[4] = 6000

lambdaT = np.zeros(5) #compute array of values for T*Lambda
for j in range(5):
    lambdaT[j] = lambda_peak_values[j]*T_values[j]

#print average value I got as my estimate for Wien's Displacement Constant
print("The value I found for Wien's Displacement Constant is {0:5.3e}".format((np.sum(lambdaT))/5) + " mK")
```

```
The value I found for Wien's Displacement Constant is 5.108e-03 mK
```

In [47]:
```
#task 5


#compute T^4:

T4vals = [0,0,0,0,0] #array to hold values of T^4
for j in range(5):
    T4vals[j] = (T_values[j])**4

#rectangle rule for numerical integration
#sum of f_i (U) by h ((b-a)/n) (from i = 0 to n-1)
#a = 10**11 b = 10**15

n= 2000 #no of steps
a= 10**13
b= 10**15

h_1 = (b-a)/(n)


#Loop to find Areas with rectangle rule
Areas = [0,0,0,0,0]
total1, total2, total3, total4, total5 = 0,0,0,0,0
for i in range(n-1):
    total1 += U1[i]*h_1
    total2 += U2[i]*h_1
    total3 += U3[i]*h_1
    total4 += U4[i]*h_1
    total5 += U5[i]*h_1
Areas[0] = total1 #Areas using Rectangle Rule
Areas[1] = total2
Areas[2] = total3
Areas[3] = total4
Areas[4] = total5

const1 = np.zeros(5)

for i in range(5):
    const1[i] = Areas[i]/T4vals[i]  #compute array to hold proportionality constant

print("Using the Rectangle Rule: The Area under the curve U is proportional to T**4 by the proportionality constant {0:8.8e}".format(np.sum(const1)/5))


#Trapezoidal Rule
#Area of each Trapezium is h/2(fi + fi+1)


#Loop to find Areas using Trapezoidal rule
Areas2 = [0,0,0,0,0]
Total1, Total2, Total3, Total4, Total5 = 0,0,0,0,0
for i in range(n-1):
    Total1 += (U1[i]+U1[i+1])*(h_1/2)
    Total2 += (U2[i]+U2[i+1])*(h_1/2)
    Total3 += (U3[i]+U3[i+1])*(h_1/2)
    Total4 += (U4[i]+U4[i+1])*(h_1/2)
    Total5 += (U5[i]+U5[i+1])*(h_1/2)
Areas2[0] = Total1 #Areas using Rectangle Rule
Areas2[1] = Total2
Areas2[2] = Total3
Areas2[3] = Total4
Areas2[4] = Total5

const2 = np.zeros(5)

for i in range(5):
    const2[i] = Areas2[i]/T4vals[i]  #compute array to hold proportionality constant values

#print value of average proportionality constant for trapezoidal rule

print("Using the Trapezoidal Rule: The Area under the curve U is proportional to T**4 by the proportionality constant {0:8.8e}".format(np.sum(const2)/5))
print("difference in values between Rectangle and Trapezoidal Rule that I obtained is {0:7.5e}".format(np.sum(const2)/5 - np.sum(const1)/5))
```

```
Using the Rectangle Rule: The Area under the curve U is proportional to T**4 by the proportionality constant 7.96739264e-16
Using the Trapezoidal Rule: The Area under the curve U is proportional to T**4 by the proportionality constant 7.97108525e-16
difference in values between Rectangle and Trapezoidal Rule that I obtained is 3.69262e-19
```

Here I used two numerical integration methods to approximate the value of the Proportionality constant for T^{-4} vs the Area of U.

The first method I used is called the Rectangle Rule. It approximates the Area under a curve from a to b by dividing it into n number of rectangles. The height of each rectangle is multiplied by the width of each rectangle $\frac{b-a}{n} = h$ and the area of each of these little triangles is summed to find the total estimate of the area under the given curve f.

The second Method I used is called the Trapezium rule. The trapezium rule works similarly to the rectangle rule except it accounts for the gaps between rectangles which are roughly triangular in shape. This makes the Trapezium rule more accurate than the rectangle rule and it is why I believe that the second estimate I found for the proportionality constant is more accurate than the first estimate.

In [ ]:
```

```