```
# Plotting the Wien_Rayleigh-Jeans_Planck Distribution Law and compare
```

Plotting the Wien Distribution and Rayleigh-Jeans Distribution Laws as functions of Wavelength at a given temperature

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

Defining the constants

```
h = 4.1357 * 10**(-15)     #Planck Constant in units of eV.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)
b = 2.898 * 10**(-3)       #Wien constant in units of m.K
T = 500      # in K
lmbda_max = b/T
print ('lambda max in m =', lmbda_max, 'at temperature T =', T, 'K')
```

```
lambda max in m = 5.796000000000001e-06 at temperature T = 500 K
```

Defining the wavelength scale

```
lmbda = np.arange(0.1,20,0.01)   #defining the array of wavelength
lmbda *= 10**(-6)         # Wavelength lambda in micrometers
```

Defining the distribution functions

```
def Wien(lmbda, T):
    x = h*c/(lmbda*k*T)
    return (2 * h * c**2 / (lmbda**5))* np.exp (-x)

def Rayleigh_Jeans(lmbda, T):
    return (2 * c /(lmbda**4)) * k * T
```

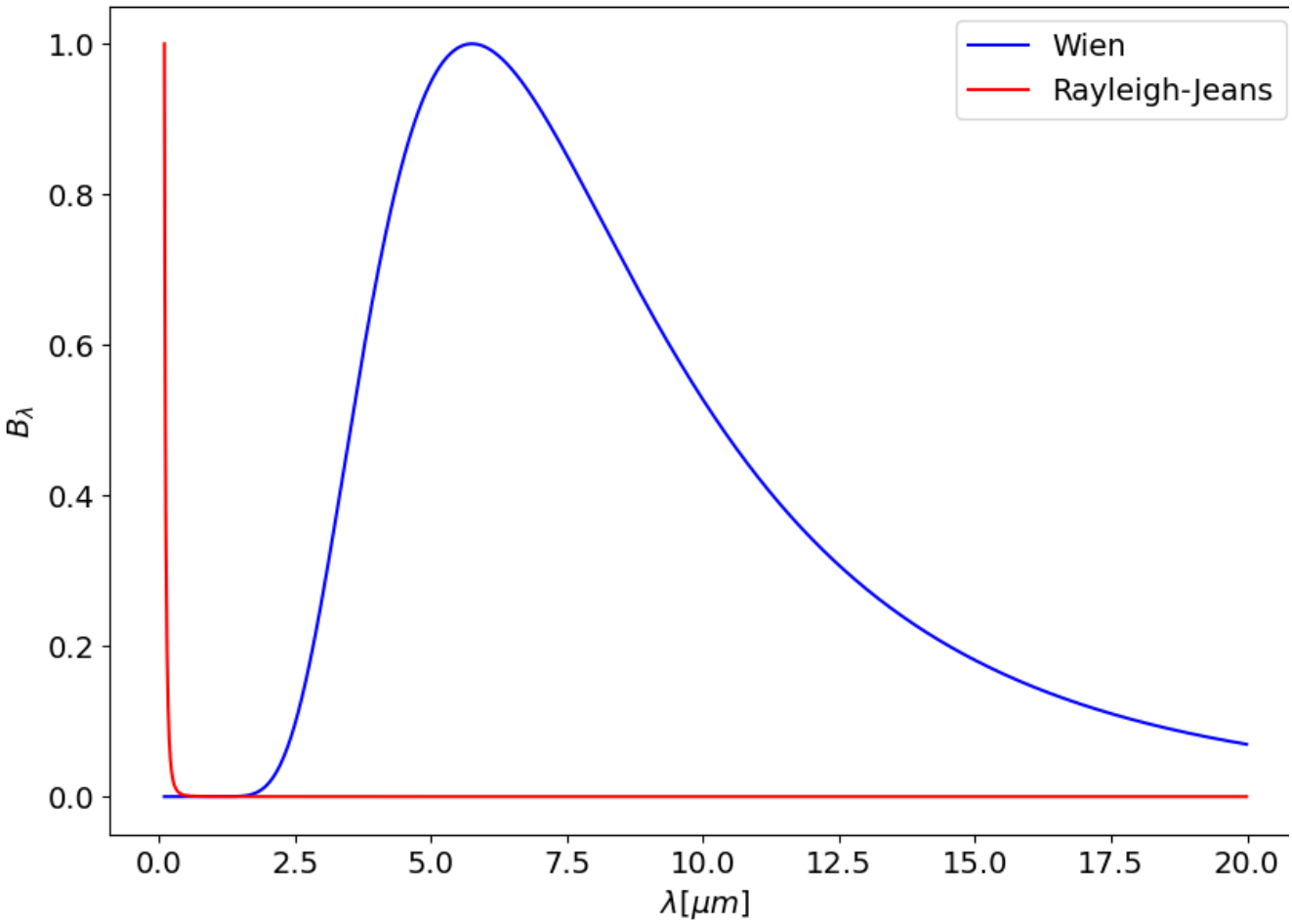
Plotting the Wein's Distribution and Rayleigh-Jeans Distribution Functions; The Ultraviolet Catastrophy

```
pl.close()
pl.rcParams.update({'font.size': 14})
pl.rcParams["figure.figsize"] = [10,7]
```

```
B_Wien = Wien(lmbda,T)
B_RJ = Rayleigh_Jeans(lmbda,T)
```

```
B_Wien_max = max(B_Wien)
B_RJ_max = max(B_RJ)
pl.plot(lmbda*(10**6), B_Wien/B_Wien_max, label='Wien', color = 'b')
pl.plot(lmbda*(10**6), B_RJ/B_RJ_max, label='Rayleigh-Jeans', color = 'r')

pl.xlabel('$\lambda [\mu m]$')
pl.ylabel('$B_\lambda$')
pl.legend()
pl.show()
```



✓ 0s completed at 4:08 PM