**The Lagrange Point**

**Part a)**

Calculating the centripetal acceleration:

Since the inward pull of the Earth and the outward pull of the Moon combines to equal the centripetal force (of the satellite so it orbits Earth):

Where ms is the mass the satellite, and g is the acceleration due to gravity.  
Similarly, for the satellite and the Moon:

Therefore, since the satellite is orbiting Earth:

**Part b)**

import numpy as np

import matplotlib.pyplot as plt

G = 6.674\*(10\*\*(-11))

M = 5.974\*(10\*\*(24))

m = 7.348\*(10\*\*(22))

R = 3.844\*(10\*\*(8))

w = 2.662\*(10\*\*(-6))

def f(r):

y = ((G\*M)/(r\*\*2)) - ((G\*m)/((R-r)\*\*2)) - ((w\*\*2)\*r)

return y

def df(r):

dy = ((-2\*G\*M)/(r\*\*3))-((2\*G\*m)/((R-r)\*\*3)) -(w\*\*2)

return dy

# def f(r):

# y = ((w\*\*2)\*r)

# return y

# def df(r):

# dy = (w\*\*2)

# return dy

accuracy = 1\*(10\*\*-5)

r0 = 750000

All\_r=[]

All\_r.append(r0)

# x\_array = [0]

# for i in range(1,10000):

# # x\_array.append(i)

# r1 = f(i)

# All\_r.append(r1)

# plt.figure()

# plt.plot(x\_array,All\_r)

# plt.show()

Have\_Solution = False

# Newton's Method

for i in range(100):

r1=r0-f(r0)/df(r0)

if abs(r1-r0)<accuracy:

Have\_Solution=True

print('Repetition: ',i)

break

else:

r0=r1

All\_r.append(r0)

print(All\_r)

if Have\_Solution:

plt.figure()

plt.plot(All\_r,'o-')

plt.xlabel("# of interation")

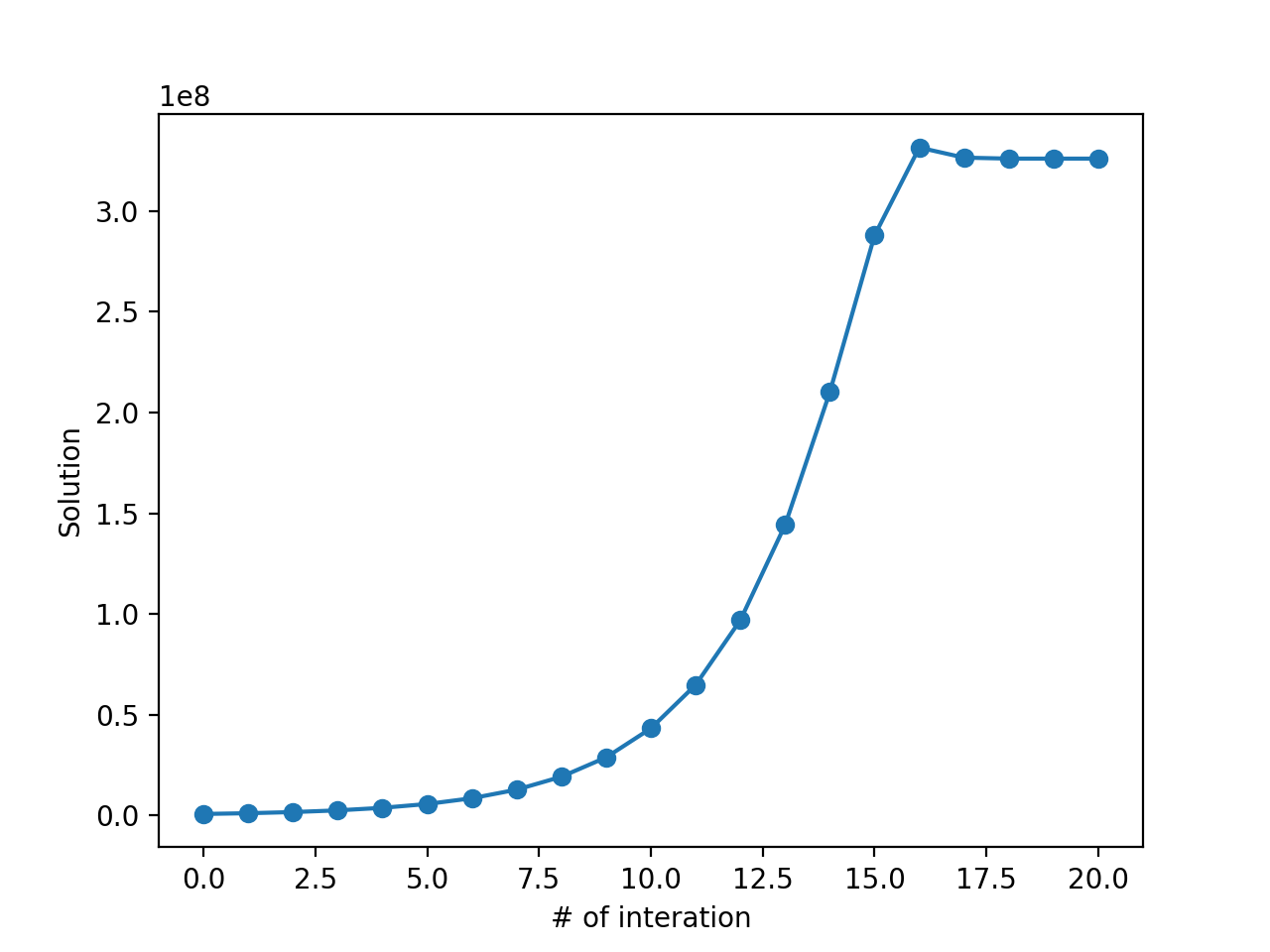
plt.ylabel("Solution")

plt.show()

print("Solution: ",All\_r[-1])

else:

print("No Solution")



Using Newton’s Method, the graph eventually converges at ~326045071 with an accuracy of 1e-5.



**The Temperature of a Light Bulb**

**Part a)**

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

def bulb(x):

y = x\*\*3/(np.exp(x)-1)

return y

accuracy = 1

z = (1 + np.sqrt(5))/2

hc = 1.23984193\*(10\*\*3)

lambda\_1 = 390

lambda\_2 = 750

kB = 8.6173303\*(10\*\*-5)

upper\_const = hc / (lambda\_1 \* kB)

lower\_const = hc / (lambda\_2 \* kB)

ans,error = integrate.quad(bulb,lower\_const,upper\_const)

result = []

T\_list = np.linspace(300,10000)

for Temperature in T\_list:

upper = upper\_const/Temperature

lower = lower\_const/Temperature

ans, error = integrate.quad(bulb, lower, upper)

result.append(ans)

for i in range(len(result)):

result[i] = result[i]\*15/(np.pi\*\*4)

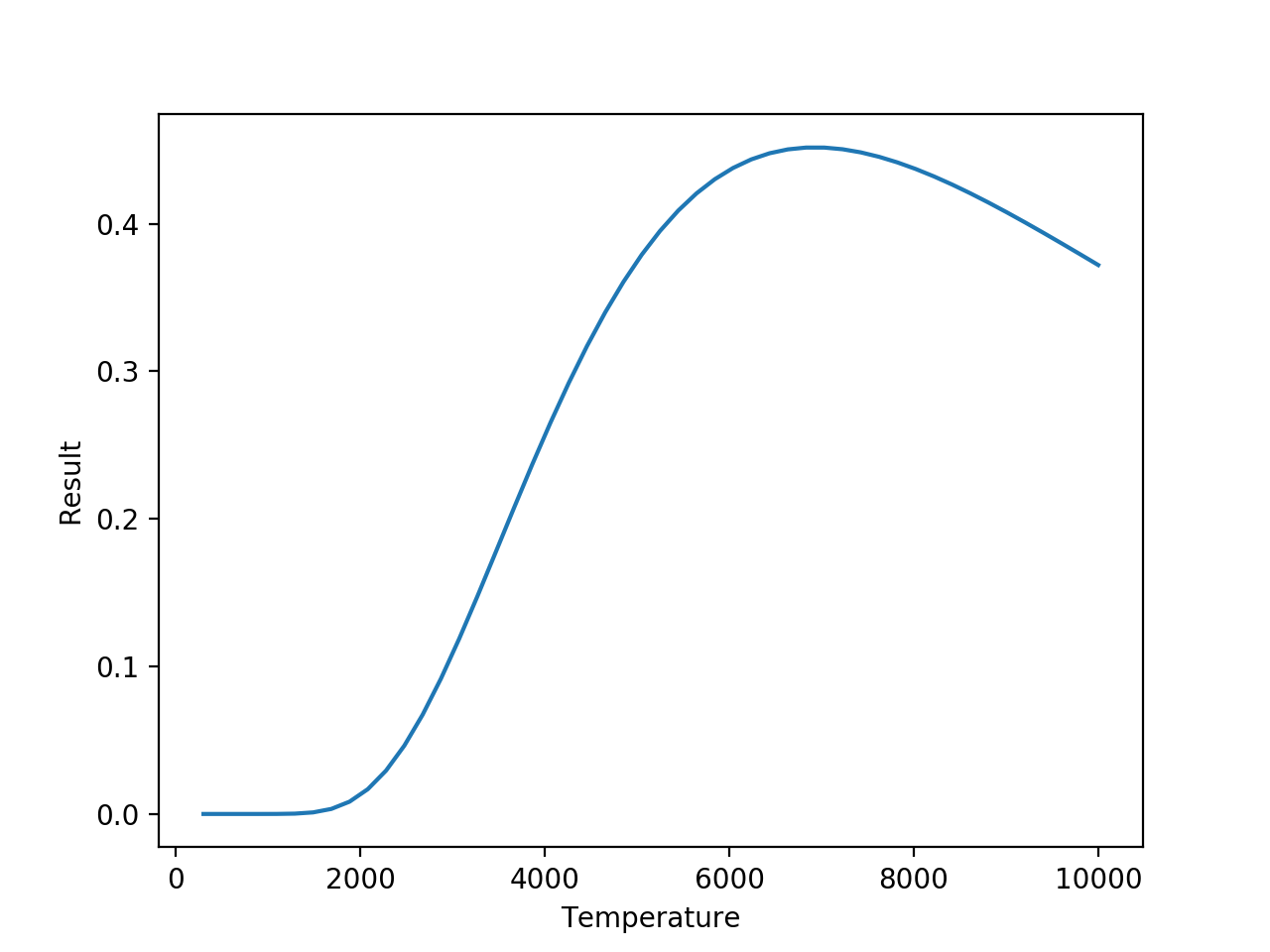
plt.figure()

plt.plot(T\_list,result)

plt.xlabel('Temperature')

plt.ylabel('Result')

plt.show()



The graph showed an obvious maxima at around 7000K, which could be used as a reference point to pick the T1 and T4 values for use using the Golden Ratio method.

**Part b)**

import numpy as np

import matplotlib.pyplot as plt

import scipy.integrate as integrate

def bulb(x):

y = x\*\*3/(np.exp(x)-1)

return y

accuracy = 1

z = (1 + np.sqrt(5))/2

hc = 1.23984193\*(10\*\*3)

lambda\_1 = 390

lambda\_2 = 750

kB = 8.6173303\*(10\*\*-5)

upper\_const = hc / (lambda\_1 \* kB)

lower\_const = hc / (lambda\_2 \* kB)

ans,error = integrate.quad(bulb,lower\_const,upper\_const)

result = []

T\_list = np.linspace(300,10000)

for Temperature in T\_list:

upper = upper\_const/Temperature

lower = lower\_const/Temperature

ans, error = integrate.quad(bulb, lower, upper)

result.append(ans)

for i in range(len(result)):

result[i] = result[i]\*15/(np.pi\*\*4)

def integral(temp):

upper = upper\_const/temp

lower = lower\_const/temp

ans= integrate.quad(bulb, lower, upper)[0]

return ans

# plt.figure()

# plt.plot(T\_list,result)

# plt.xlabel('Temperature')

# plt.ylabel('Result')

# plt.show()

# Max efficiency around 7000K from part\_a. Therefore choose T1 and T4 to fit 7000K in between.

T1 = 6000

T4 = 8000

T2 = T4-(T4-T1)/z

T3 = T4+(T4-T1)/z

# Initial values of eta

initial\_T1 = integral(T1)

initial\_T2 = integral(T2)

initial\_T3 = integral(T3)

initial\_T4 = integral(T4)

# golden ratio search loop

while T4 - T1 > accuracy:

if initial\_T2 < initial\_T3 :

T4, initial\_T4 = T3, initial\_T3

T3, initial\_T3 = T2, initial\_T2

T2 = T4 - (T4 - T1)/z

initial\_T2 = integral(T2)

else:

T1, initial\_T1 = T2, initial\_T2

T2, initial\_T2 = T3, initial\_T3

T3 = T1 + (T4 - T1)/z

initial\_T3 = integral(T3)

print('The temperature of max efficiency is', 0.5 \* (T1 + T4))

Using the Golden Ratio method, the temperature at maximum efficiency was output to be:



This value seems to agree with the graph in Part (a). But it seems too hot to be practical.

**Part c)**

The maximum efficiency temperature was calculated as 7235.961014189429 K. However, at this temperature, it is simply impossible to run the tungsten-filament light bulb because the melting point of tungsten is approximately 3695.15 K, which is about half the maximum efficiency temperature. Running the light bulb at its maximum efficient temperature would melt the tungsten. One should have noticed that 7235 K is even hotter than the surface temperature of the Sun (~5778 K)!, and therefore even before considering the melting point of tungsten, or the glass light bulb, one can predict it would be impossible for a light-bulb to create and withstand such extreme condition.