Numerical Methods Mid Sem Lab Exam

Jeevan Koshy

1740256

Question 1

Write a python program to plot the relationship between gravitational force and distance between two bodies. Force is given by

$$F = \frac{Gm1m2}{r^2}$$

$m1 = 1.5Kg;$

$m2 = 2.3Kg;$

$G = 6.674X10^{-}11Nm^{2}Kg^{-}2.$

Plot it for 20 different distances between 100 to 1000m.

In [12]:

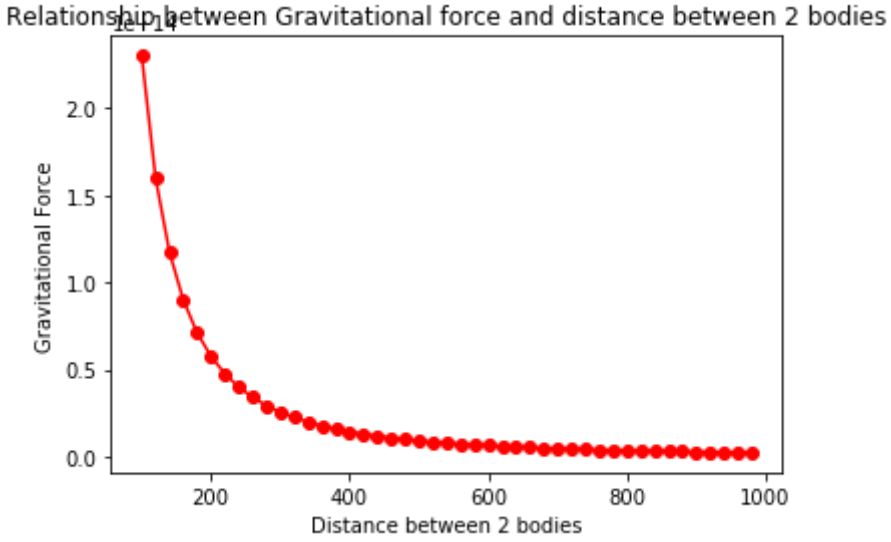
```
import numpy as np
import matplotlib.pyplot as plt
import math
```

In [32]:

```
F = []

dash = "-"
print(dash*45)
print("Distance  | \tTime")
print(dash*45)
def fun(r):
    G = 6.674 * (10**-11)
    m1 = 1.5
    m2 = 2.3
    return (G*m1*m2)/(r**2)

for i in range(100,1000,20):
    Force = fun(i)
    print("|  {0} \t  | \t  {1}  ".format(i,fun(i)))
    F.append(Force)

r = range(100,1000,20)
plt.plot(r,F,color = "red",marker="o")
plt.xlabel('Distance between 2 bodies')
plt.ylabel('Gravitational Force')
plt.title("Relationship between Gravitational force and distance between 2 bodies")
plt.show()
```

```
---------------------------------------------
Distance  |     Time
---------------------------------------------
|  100    |       2.3025299999999995e-14
|  120    |       1.5989791666666663e-14
|  140    |       1.1747602040816324e-14
|  160    |       8.994257812499998e-15
|  180    |       7.106574074074073e-15
|  200    |       5.756324999999999e-15
|  220    |       4.757293388429751e-15
|  240    |       3.997447916666666e-15
|  260    |       3.4061094674556208e-15
|  280    |       2.936900510204081e-15
|  300    |       2.5583666666666663e-15
|  320    |       2.2485644531249995e-15
|  340    |       1.9918079584775082e-15
|  360    |       1.7766435185185182e-15
|  380    |       1.5945498614958446e-15
|  400    |       1.4390812499999997e-15
|  420    |       1.3052891156462582e-15
|  440    |       1.1893233471074377e-15
|  460    |       1.0881521739130433e-15
|  480    |       9.993619791666664e-16
|  500    |       9.210119999999998e-16
|  520    |       8.515273668639052e-16
|  540    |       7.896193415637858e-16
|  560    |       7.342251275510203e-16
|  580    |       6.844619500594529e-16
|  600    |       6.395916666666666e-16
|  620    |       5.989932362122787e-16
|  640    |       5.621411132812499e-16
|  660    |       5.285881542699724e-16
|  680    |       4.979519896193771e-16
|  700    |       4.69904081632653e-16
|  720    |       4.441608796296296e-16
```

```
|  740  |       4.204766252739225e-16
|  760  |       3.9863746537396116e-16
|  780  |       3.7845660749506894e-16
|  800  |       3.597703124999999e-16
|  820  |       3.4243456276026167e-16
|  840  |       3.2632227891156456e-16
|  860  |       3.1132098431584635e-16
|  880  |       2.9733083677685943e-16
|  900  |       2.842629629629629e-16
|  920  |       2.720380434782608e-16
|  940  |       2.605851063829787e-16
|  960  |       2.4984049479166666e-16
|  980  |       2.3974698042482295e-16
```



Question 2

The saturation concentration of dissolved oxygen in freshwater can be calculated with the equation

$$lno_{sf} = -139.34411 + \frac{1.575701*10^5}{T_\alpha} - \frac{6.642308*10^7}{T_\alpha^2} + \frac{1.243}{}$$

where, o = the saturation concentration of dissolved oxygen in freshwater $1atm(mgL^-1)$ and T_α = absolute temperature (K). Remeber that $T_\alpha = T + 273.15$ where T temperature $(degreeC)$. According to this equation, saturation decreases with increasing temperature. For typical natural waters in temperature climates, the equa-tion can be used to determine that oxygen concentration ranges from $14.621mg/L$ to 0^0C to $6.949mg/L$ at 35^0C. Given a value of oxygenconcentration, this formula and the bisection method can be used tosolve for temperature in $(degreeC)$.

a) If the initial guesses are set as 0 and 35^0C, how many bisection iterations would be required to determine temperature to an absolute error of 0.05^0C?

b) Based on a) develop and test a bisection program to determine T as a function of a given oxygen concentration. Test your program for $o_{sf} = 8, 10$ and $14mg/L$. Check your results.

In [42]:

```
def func(x,os):
    return -139.34411+((1.575701*10**5)/(x+273.15))-(6.642308*(10**7)/(x+273.15)**2)+(1.243

dash = '-' * 75

def bisection():
    f = []
    a=int(input("Enter a"))
    b=int(input("Enter b"))
    os=float(input("Enter oxygen concentration: "))
    if(os<=14.621 and os>=6.949):
        break
    if (func(a,os) * func(b,os) >= 0):
        print("You have not assumed right a and b\n")
        return

    while (True):
        if fun(a,os)*fun(b,os)<0:
            break
        # Find middle point
        c = (a+b)/2
        absol.append(abs(b-a))
        i=i+1
        print('{:>12d}{:>12.6f}{:>12.6f}{:>12.6f}{:>12.6f}{:>12.6f}'.format(i+1,a,b,abs(b-a
        # Check if middle point is root



    pres=float(input("Enter precision limit: "))
    while (True):

    print(dash)
    print("|{:^12s}|{:^12s}|{:^12s}|{:^12s}|{:^12s}|{:^12s}|".format("iteration","a","b","c
    print(dash)
    i=0


    while(True):
        c=(a+b)/2
        i=i+1
        e.append(abs(b-a))
        print("|{:^12d}|{:^12.6f}|{:^12.6f}|{:^12.6f}|{:^12.6f}|{:^12.6f}|".format(i,a,b,c,
        if abs(b-a)<pres:
            break
        if fun(a,os)*fun(c,os)<0:
            b=c
        if fun(b,os)*fun(c,os)<0:
            a=c
        if abs(fun(c,os))==0:
            break


root=bisection()
print("The root using bisection method is %.6f"%root)
```

```
  File "<ipython-input-42-bcf80a1cb384>", line 32
    print(dash)
        ^
```

```
IndentationError: expected an indented block
```

In [ ]:

```

```