Computational Physics I

Name: Sudip KC

Physics | Class of 2022 | Jacobs University Bremen

Solar system

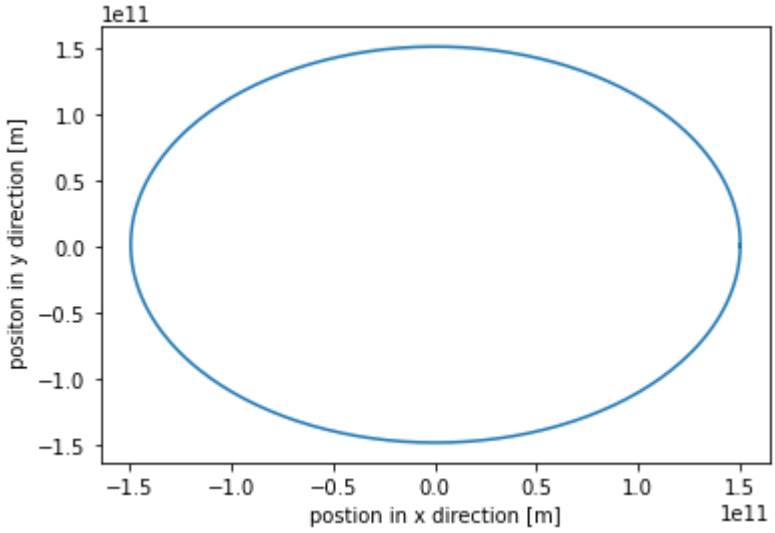
As all the planets in the solar system move in approximately in the same plane, a two-dimensional simulation is sufficient. We simulated the earths rotation for 1 year at a time step dt (1 day). Similarly, the Newton's equation of motion for earth under the influence of the gravitation of the Sun at a time step of 1 day is shown.

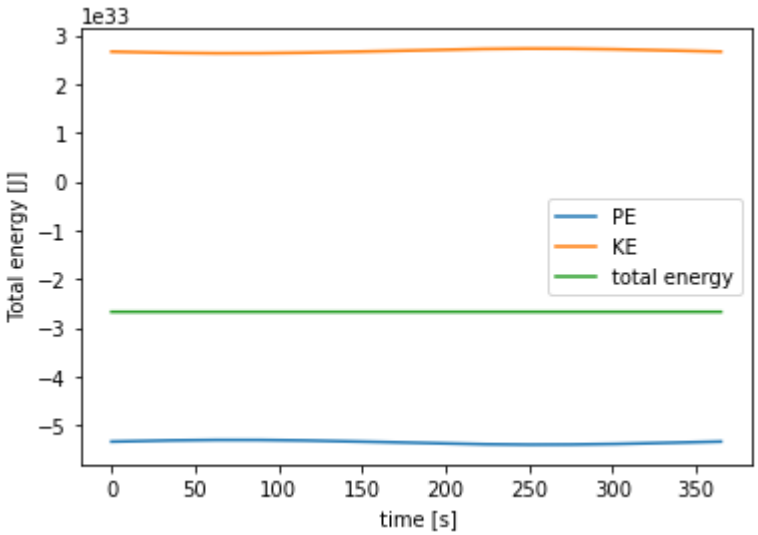
Validation

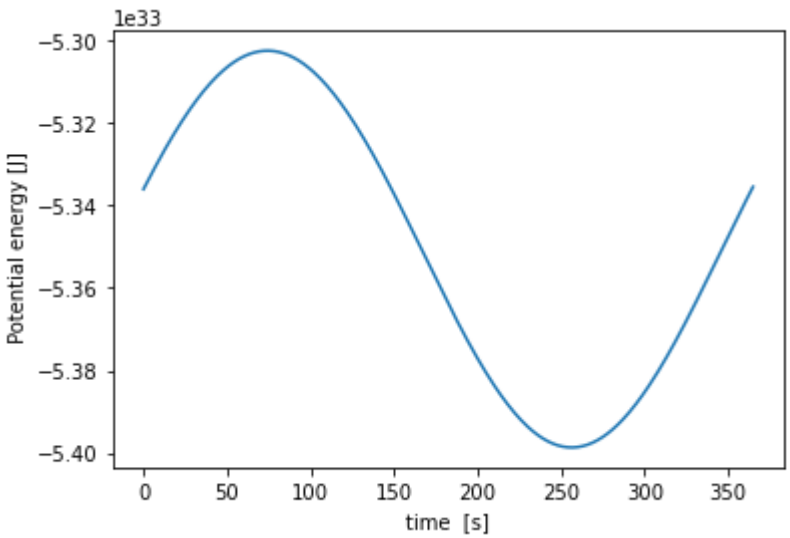
From the plotted graph of x-y position , an elliptical nature of the earths orbital around the sun was observed as expected. Furthermore, the graph of total energy as a function of time was constant, which proves the conservation of energy in the system. Furthermore, the change in KE is compensated by the change in PE at every time which results in the constant total energy of the system.

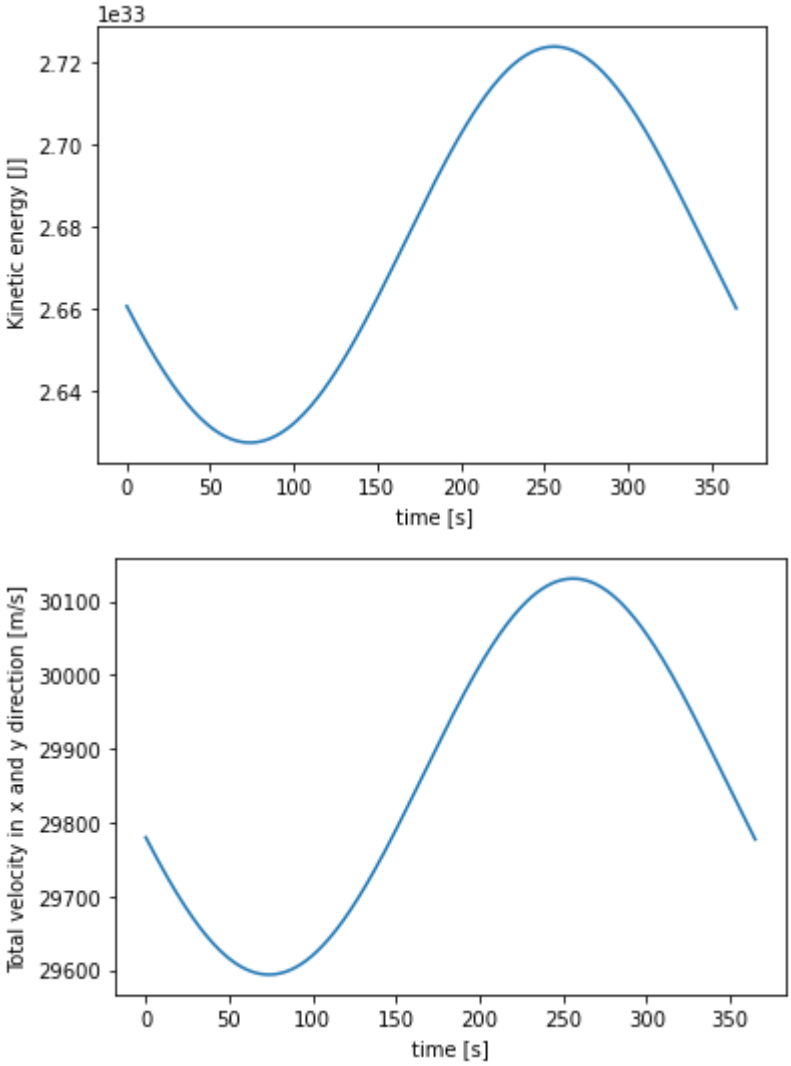
In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
G = 6.67*10**(-11)                          #Gravitational Constant
M = 2.0*10**(30)                            #Mass of sun
m = 6 * 10**24                              #Mass of earth

x0 = 150.0*10**9
y0, vx0, vy0,  = 0, 0, 29780
ax0, ay0 = accx(x0, y0), accy(x0, y0)
x, y, vx, vy, ax, ay, tarr, KEarr, PEarr = [x0], [y0], [vx0], [vy0], [ax0], [
dt = 86400
day = 0

def KE(vx,vy):
    v = (np.sqrt(vx**2 + vy**2))
    return(1/2*m*v**2)

def PE(x,y):
     r = np.sqrt(x**2+y**2)
     return(-G*M*m/r)


while day < 365:
    x0 = x0 + vx0 * dt
    y0 = y0 + vy0 * dt
    ax0 = accx(x0, y0)
    ay0 = accy(x0, y0)
    vx0=  vx0 + ax0*dt
    vy0 =  vy0 + ay0*dt
    x.append(x0)
    y.append(y0)
    vx.append(vx0)
    vy.append(vy0)
    ax.append(ax0)
```

```
    ay.append(ay0)
    day= day+1
    tarr.append(day)



for i in range(len(vx)):
    KEarr.append(KE(vx[i], vy[i]))
    PEarr.append(PE(x[i], y[i]))
Tenergy = [KEarr[i] + PEarr[i] for i in range(len(KEarr))]

v = [np.sqrt(vx[i]**2 + vy[i]**2) for i in range(len(vx))]
plt.plot(x,y)
plt.xlabel('postion in x direction [m]')
plt.ylabel(' positon in y direction [m]')
plt.show()

plt.plot(tarr, PEarr, label = 'PE')
plt.xlabel('time  [s]')
plt.ylabel(' Potential energy [J]')


plt.plot(tarr, KEarr, label = 'KE')
plt.xlabel(' time [s]')
plt.ylabel(' Kinetic energy [J]')


plt.plot(tarr, Tenergy, label = 'total energy')
plt.xlabel(' time [s]')
plt.ylabel(' Total energy [J]')
plt.legend()
plt.show()

plt.plot(tarr, PEarr, label = 'PE')
plt.xlabel('time  [s]')
plt.ylabel(' Potential energy [J]')
plt.show()

plt.plot(tarr, KEarr, label = 'KE')
plt.xlabel(' time [s]')
plt.ylabel(' Kinetic energy [J]')
plt.show()



plt.plot(tarr, v)
plt.xlabel('time [s]')
plt.ylabel('Total velocity in x and y direction [m/s]')
plt.show()
```


1e11
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
positon in y direction [m]
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
postion in x direction [m]
1e11


1e33
3
2
1
0
−1
−2
−3
−4
−5
Total energy [J]
PE
KE
total energy
0
50
100
150
200
250
300
350
time [s]


1e33
−5.30
−5.32
−5.34
−5.36
−5.38
−5.40
Potential energy [J]
0
50
100
150
200
250
300
350
time [s]



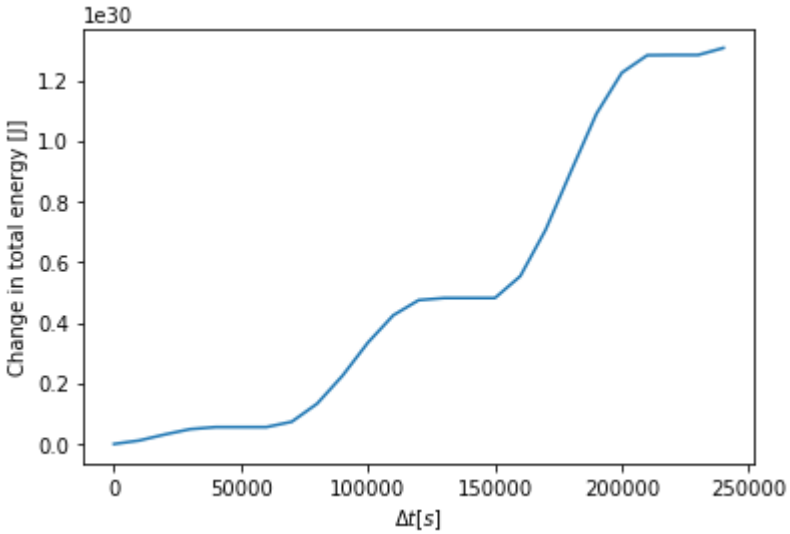
b)

In this part of the question, for earth and uranus the Δt has to be optimized using the requirement of total energy conservation. After simulating the systems (i.e Earth and Uranus) around the sun and plotting the change in total energy of the systems for different dt values. In our program, we had the system for one complete rotation of the Earth around the sun and we observed that the total change in the energy was almost stable for smaller value of dt. When the value of dt is more than 75000s, it violates the stabilty of the change in total energy. So the better approximatin in this case would be the value of dt lower than 75000 s.

In [16]:
```
import numpy as np
import matplotlib.pyplot as plt

G = 6.67*10**(-11)
M = 2.0*10**(30)        #Mass of sun
m = 6 * 10**24          #Mass of earth
mu = 8.8*10**(25)       #Mass of uranus
def accx(M,x1, y1, x2, y2):
    r = np.sqrt((x2-x1)**2+(y2-y1)**2)
    return (-G*M*x1/r**3)

def accy(M, x1, y1, x2, y2):
    r = np.sqrt((x2-x1)**2+(y2-y1)**2)
    return (-G*M*y1/r**3)
```

```
xe0 = 150.0*10**9
ye0, vxe0, vye0,  = 0, 0, 29780

xu0, yu0, vxu0, vyu0 = 19.19*xe0, 0, 0, 6800
axe0 = accx(M, xe0, ye0, 0, 0)+ accx(mu, xe0, ye0, xu0, yu0)
aye0 = accy(M, xe0, ye0, 0, 0)+ accy(mu, xe0, ye0, xu0, yu0)

axu0 = accx(M, xu0, yu0, 0, 0)+ accx(m, xu0, yu0, xe0, ye0)
ayu0 = accy(M, xu0, yu0, 0, 0)+ accy(m, xu0, yu0, xe0, ye0)
xe, ye, vxe, vye, axe, aye, tarr, KEarre, PEarre = [xe0], [ye0], [vxe0], [vye
xu, yu, vxu, vyu, axu, ayu, KEarru, PEarru = [xu0], [yu0], [vxu0], [vyu0], [a



delTE= []

def KE(vx,vy):
    v = (np.sqrt(vx**2 + vy**2))
    return(1/2*m*v**2)

def PE(x1,y1, x2, y2, m ,mu):
     r1 = np.sqrt(x1**2+y1**2)
     PEs= (-G*M*m/r1)
     r2 = np.sqrt((x2-x1)**2+(y2-y1)**2)
     PEu= (-G*mu*m/r2)

     return PEs+PEu
dtarr = [10000*i for i in range(25)]
day = 0
for dt in dtarr:
    while day < (365*dt/86400):
        xe0 = xe0 + vxe0 * dt
        ye0 = ye0 + vye0 * dt

        xu0 = xu0 + vxu0 * dt
        yu0 = yu0 + vyu0 * dt

        axe0 = accx(M, xe0, ye0, 0, 0)+ accx(mu, xe0, ye0, xu0, yu0)
        aye0 = accy(M, xe0, ye0, 0, 0)+ accy(mu, xe0, ye0, xu0, yu0)

        axu0 = accx(M, xu0, yu0, 0, 0)+ accx(m, xu0, yu0, xe0, ye0)
        ayu0 = accy(M, xu0, yu0, 0, 0)+ accy(m, xu0, yu0, xe0, ye0)

        vxe0=  vxe0 + axe0*dt
        vye0 =  vye0 + aye0*dt

        vxu0=  vxu0 + axu0*dt
        vyu0 =  vyu0 + ayu0*dt

        xe.append(xe0)
        ye.append(ye0)
        vxe.append(vxe0)
        vye.append(vye0)
        axe.append(axe0)
        aye.append(aye0)

        xu.append(xu0)
        yu.append(yu0)
        vxu.append(vxu0)
        vyu.append(vyu0)
        axu.append(axu0)
        ayu.append(ayu0)

        day= day+dt/86400
```

```
        tarr.append(day)

    ve = [np.sqrt(vxe[i]**2 + vye[i]**2) for i in range(len(vxe))]
    vu = [np.sqrt(vxu[i]**2 + vyu[i]**2) for i in range(len(vxu))]


    for i in range(len(vxe)):
        KEarre.append(KE(vxe[i], vye[i]))
        KEarru.append(KE(vxu[i], vyu[i]))
        PEarre.append(PE(xe[i], ye[i], xu[i], yu[i],m , mu  ))
        PEarru.append(PE(xu[i], yu[i], xe[i], ye[i],mu , m  ))
    TenergyE = [KEarre[i] + PEarre[i] for i in range(len(KEarre))]
    TenergyU = [KEarru[i] + PEarru[i] for i in range(len(KEarru))]
    totalEnergy = [(TenergyE[i] + TenergyU[i]) for i in range(len(TenergyE))]
    delTE.append(max(totalEnergy) - min (totalEnergy))


plt.plot(dtarr, delTE)
plt.xlabel('$\Delta t [s]$')
plt.ylabel('Change in total energy [J]')
plt.show()
```



In [ ]: