IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

TO VISUALIZE ORBITS OF GRAVITATING BODIES BY USING NUMERICAL APPROACH

[1] Lavli Rana, [2] Suresh Kumar

[1,2]Assistant Professor of Physics

[1]Department of Physics

[1]NSCBM Government College Hamirpur

Abstract: In recent years there has been an increasing demand for solutions of the general three-body problem in various astrophysical situations. For example, binaries and their interactions with single stars play a major role in the evolution of star clusters. Triple stellar systems are another obvious astrophysical three-body problem. Many other astrophysical bodies, ranging from compact bodies to galaxies, occur in triple systems. At present time, three body astrophysics is primarily motivated by the need to understand the role of binaries in the evolution of stellar systems because mathematically ideal solution for the three-body problem is of no use i.e., Sundman's series solution which has extremely slow convergence. In this paper we have implemented the numerical integration method in python for two body problem for understanding the implementation and then extended it to higher body problem (three and four body). Here we have used Odeint package of Scipy library in Python to solve the equation of motion for bodies under gravitation. By observations of the results of computer simulations, it is found that this problem got some patterns when observed over a very long interval of time. Particularly in problems of astrophysical importance, one may almost always identify a binary and a third body. A binary can be treated as a single entity with certain internal properties (component masses, energy and angular momentum, as well as orientation in space) and this system and third body makes themselves effective binary.

Index Terms - Binaries, Convergence, Star Clusters, Odeint Package, Python.

. I. INTRODUCTION

The Two –body central problem arises in different contexts in nature. The problem of motion of three celestial bodies under their mutual gravitational attraction is an old problem and logically follows from the two-body problem which was solved by Newton in his Principia in 1687. Newton also considered the three-body problem in connection with the Motion of the Moon under the influences of the Sun and the Earth, the consequences of which included a headache. There are good reasons to study the three-body gravitational problem. The motion of the Earth and other planets around the Sun is not strictly a two-body problem. The

gravitational pull by another planet constitutes an extra force which tries to steer the planet off its elliptical path. One may even worry, as scientists did in the eighteenth century, whether the extra force might change the orbital course of the Earth entirely and make it fall into the Sun or escape to cold outer space. This was a legitimate worry at the time when the Earth was thought to be only a few thousand years old, and all possible combinations of planetary influences on the orbit of the Earth had not yet had time to occur.

Another serious question was the influence of the Moon on the motion of the Earth. Would it have long term major effects? Is the Moon in a stable orbit about the Earth or might it one day crash on us? The motion of the Moon was also a question of major practical significance, since the Moon was used as a universal time keeping device in the absence of clocks which were accurate over long periods of time. After Newton, the lunar theory was studied in the eighteenth century using the restricted problem of three bodies (Euler 1772). In the restricted problem, one of the bodies is regarded as massless in comparison with the other two which are in a circular orbit relative to each other.

We can solve the two-body problem for its general solution. But when we add a third body to the system, something extraordinary happens. The system becomes chaotic and highly unpredictable. It has no analytical solution (except a few special cases) and its equations can only be solved numerically on a computer. They can turn abruptly from stable to unstable and vice versa. Here we have implemented the numerical integration method in python for two body problem for understanding the implementation and then extend it to higher body problem (three and four body).

II. Visualization of Orbits

Basic Approach: According to Newton's law of gravitation, the gravitational force between two point masses is directly proportional to product of masses of the two bodies and inversely proportional to the square of distance between them and acts along the line joining the center of two bodies.

$$\vec{F} = G\frac{m_1 m_2}{r^2}\hat{r}$$

Where m1 is the mass of the first body, m2 is the mass of the second body and r is the distance between them. G is the universal gravitational constant.

Equation of motion:

The equation of motion of two body system is given by Newton's second law of motion. According to Newton's second law of motion, the net force on an object produces a net change in momentum of the object — in simple terms, force is mass times acceleration. So, applying the above equation to the body system having mass m_1 and m2, we get the following differential equation of motion for the body.

$$m_1\frac{\mathrm{d}^2\vec{r_1}}{\mathrm{d}t^2} = G\frac{m_1 m_2}{r^2}\hat{r}$$

Or

$$m_1 \frac{\mathrm{d}^2 \vec{r_1}}{\mathrm{d}t^2} = \vec{F} = G \frac{m_1 m_2}{r^3} \vec{r_{12}}$$

Now, we have a second-order differential equation that describes the interaction between two bodies due to gravity. To simplify its solution, we can break it down into two first order differential equations. The acceleration of an object is the change in velocity of the object with time so the second order differential of position can be replaced with a first order differential of velocity. Similarly, the velocity can be expressed as a first order differential of the position.

$$m_1 \frac{d\vec{v_v}}{dt} = G \frac{m_i m_j}{r_{ij}^3} \vec{r_{l_j}}$$

$$\frac{d\vec{r_i}}{dt} = \vec{v_l}$$

The index i is for the body whose position and velocity is to be calculated whereas the index j is for the other body which is interacting with body i. Thus, for a two-body system, we will be solving two sets of these two equations.

Non dimensionalization:

Before implementing our problem in python, we have non-dimensionalized the physical quantities. We converted all the quantities in the equation (like position, velocity, mass and so on) that have dimensions (like m, m/s, kg respectively) to non-dimensional quantities that have magnitudes close to unity.

To non-dimensionalize the equations, we divided each quantity by a fixed reference quantity. For example, divide the mass terms by the mass of the sun, position (or distance) terms with the distance between the two stars in the Alpha Centauri system, time term with the orbital period of Alpha Centauri and velocity term with the relative velocity of the earth around the sun.

When you divide each term by the reference quantity, you will also need to multiply it to avoid changing the equation. All these terms along with G can be clubbed into a constant, say K_1 for equation 1 and K_2 for equation 2. Thus, the non-dimensionalized equations are as follows:

$$\bar{m_l} \frac{d\vec{v_l}}{dt} = k_1 \frac{\bar{m_l} \bar{m_j}}{\bar{r_{ij}^3}} \vec{\bar{r}_{ij}}$$

$$\frac{d\vec{\bar{r}_l}}{dt} = k_2 \vec{\bar{v}_l}$$

The bar over the terms indicates that the terms are non-dimensional. So these are the final equations that we'll be using in our simulation.

III. Numerical approach to visualize the Orbits

(a) Python Code for Two Body Problem

```
import scipy as sci
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import numpy
from mpl_toolkits.mplot3d import Axes3D
#defining constants and reference quantities
G=6.67408e-11 #N-m2/kg2
m_ref = 1.989e+30 #Kg
r_ref = 5.326e+12 #meter
v_ref = 30000 #m/s
t_ref = 79.91*365*24*3600*0.51 #s
#constants
K1 = G*t_ref*m_ref/(r_ref**2*v_ref)
    K2 = v_ref*t_ref/r_ref
#defining quantities
m1 = 1.1
m2 = 0.907
r1 = [-0.5, 0 ,0]
r2 = [0.5,0,0]
# conversion of position vectors to arrays
r1 = numpy.array(r1, dtype="float64")
r2 = numpy.array(r2, dtype="float64")
#finding center of mass
r_cm = (m1*r1+m2*r2)/(m1+m2)
#defning initial velocities
v1 = [0.01,0.01,0] #m/s
    v2 = [-0.05,0,-0.1] #m/s
50
#converting velocity vector to arrays
v1 = numpy.array(v1, dtype="float64")
v2 = numpy.array(v2, dtype="float64")
#finding velocity of center of mass
v_cm = (m1*v1+m2*v2)/(m1+m2)
#defining equations of motion in a function
    def two_body_equations(w,t,G,m1,m2):
```

```
r1=w[:3]
r2=w[3:6]
v1=w[6:9]
v2=w[9:12]
r = numpy.linalg.norm(r2-r1)
dv1bydt = K1*m2*(r2-r1)/r**3
dv2bydt = K1*m1*(r1-r2)/r**3
dr1bydt = K2*v1
    dr2bydt = K2*v2
derivs = numpy.concatenate((dr1bydt,dr2bydt,dv1bydt,dv2bydt))
return derivs
#creating figure for plotting
fig = plt.figure(figsize=(15,15))
#create 3d axes
    ax = fig.add_subplot(111,projection="3d")
import scipy.integrate
def update(frame):
ax.clear()
global r1,r2,v1,v2,K1,K2
init_params = numpy.array([r1,r2,v1,v2])
init_params = init_params.flatten() # flatten to 1d array
    time_span = numpy.linspace(0,1.7,403)
two_body_sol = sci.integrate.odeint(two_body_equations,init_params,time_span,args=(G,m1,m2))
r1_sol = two_body_sol[:,:3]
r2_sol = two_body_sol[:,3:6]
v1_sol = two_body_sol[:, 6:9]
v2_sol = two_body_sol[:, 9:12]
#Finding location of COM
rcom_sol=(m1*r1_sol+m2*r2_sol)/(m1+m2)
#Finding location of Alpha Centauri A w.r.t COM
r1com_sol= r1_sol-r1_sol # for frame of reference in one of the body
#r1com_sol= r1_sol-rcom_sol
#Finding location of Alpha Centauri B w.r.t COM
r2com_sol= r2_sol-r1_sol # for frame of reference in one of the body
#r2com_sol = r2_sol-rcom_sol
#Ploting the orbits in COM frame
ax.plot(r1com_sol[:,0],r1com_sol[:,1],r1com_sol[:,2],color="darkblue")
ax.plot(r2com_sol[:,0],r2com_sol[:,1],r2com_sol[:,2],color="tab:red")
```

```
#Ploting the final positions of the stars in COM Frame
ax.scatter(r1com_sol[-1,0],r1com_sol[-1,1],r1com_sol[-1,2],color="darkblue",marker="o",s=100,label="Alpha Centauri A")
ax.scatter(r2com_sol[-1,0],r2com_sol[-1,1],r2com_sol[-1,2],color="tab:red",marker="o",s=100,label="Alpha Centauri B")
#"""
r1 = [r1_sol[-1,0],r1_sol[-1,1],r1_sol[-1,2]]
r2 = [r2_sol[-1,0],r2_sol[-1,1],r2_sol[-1,2]]
v1 = [v1_sol[-1,0],v1_sol[-1,1],v1_sol[-1,2]]
v2 = [v2_sol[-1,0],v2_sol[-1,1],v2_sol[-1,2]]
r1 = numpy.array(r1, dtype="float64")
r2 = numpy.array(r2, dtype="float64")
v1 = numpy.array(v1, dtype="float64")
v2 = numpy.array(v2, dtype="float64")
ani = FuncAnimation(fig, update, interval=1000)
ax.set_xlabel("x-coordinate",fontsize=14)
ax.set_ylabel("y-coordinate",fontsize=14)
ax.set_zlabel("z-coordinate",fontsize=14)
ax.set_title("Visualization of orbits of stars in a two-body system\n",fontsize=14)
plt.show()
```

On running the above python code, for initial conditions

m1	*m2*	*v1*	*v2*	*r1*	*r2*
1.1	*0.907*	[0.01,0.01,0]	[-0.05,0, -0.1]	[-0.5, 0 ,0]	[0.5,0,0]

The trajectories of the two bodies can be seen in different frame of references. In the frame of and arbitrary point fixed in space, the trajectory looks like:

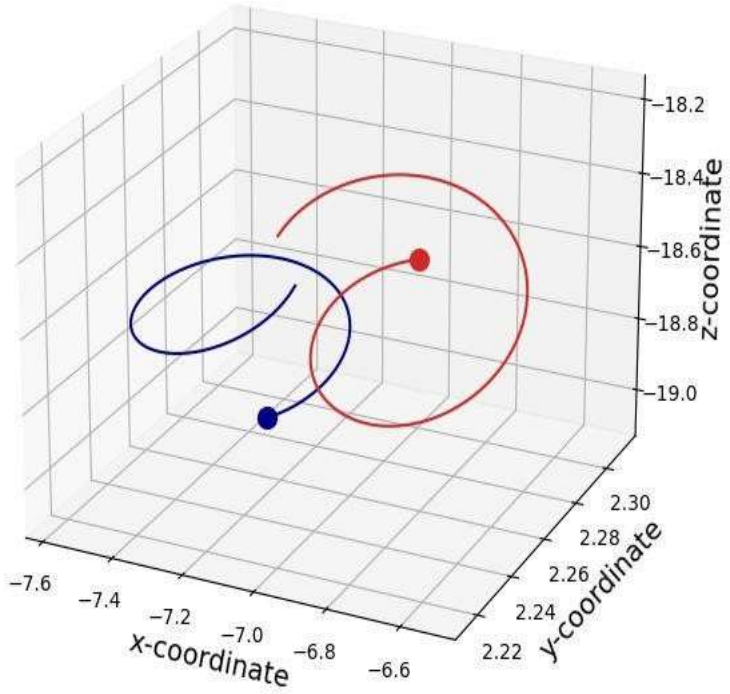

Fig1. Two body orbits in arbitrary frame

i.e. the bodies move in a spiral type path with the passage of time for stable initial conditions. If the conditions are not stable, bodies may be thrown apart from each other and keeps travelling away from each other.

When we observe orbits of two bodies in center of mass frame, the orbits are as:

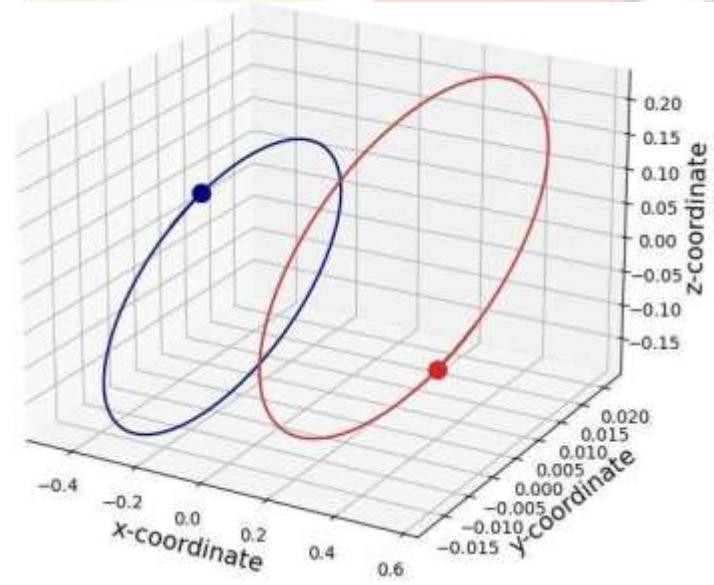

Fig2. Two body orbit in Center of Mass frame

i.e., bodies are found to be moving in elliptical orbits for stable initial conditions. If we shift frame of reference to one of the bodies then one body is observed revolving around reference body in elliptical orbit.

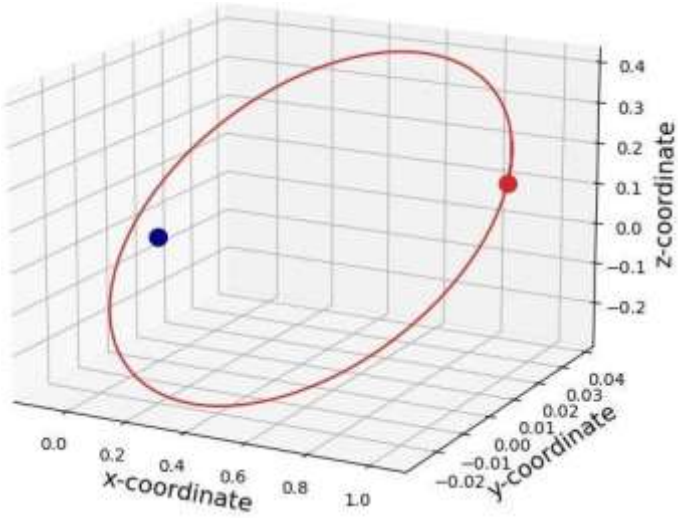

Fig3. Two body orbit in one of the Planet's frame

(b) Python Code for Three Body Problem:

```
import scipy as sci
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
import numpy
from mpl_toolkits.mplot3d import Axes3D
#defining constants and reference quantities
G=6.67408e-11 #N-m2/kg2
m_ref = 1.989e+30 #Kg
r_ref = 5.326e+12 #meter
v_ref = 30000 #m/s
    t_ref = 79.91*365*24*3600*0.51 #s
#constants
K1 = G*t_ref*m_ref/(r_ref**2*v_ref)
    K2 = v_ref*t_ref/r_ref
#defining quantities
m1 = 1.1
m2 = 0.907
m3 = 1.0
r1 = [-0.5, 0.2 ,0.0]
r2 = [0.5,0.0,0.1]
r3 = [0.2,0.5,0.0]
    # conversion of position vectors to arrays
r1 = numpy.array(r1, dtype="float64")
r2 = numpy.array(r2, dtype="float64")
```

```
r3 = numpy.array(r3, dtype="float64")
#finding center of mass
r_cm = (m1*r1+m2*r2+m3*r3)/(m1+m2+m3)
    #defning initial velocities
v1 = [0.01,0.01,0.0] #m/s
v2 = [-0.05,0.0,-0.1] #m/s
v3 = [0.0,-0.01,0.0] #m/s
#converting velocity vector to arrays
v1 = numpy.array(v1, dtype="float64")
v2 = numpy.array(v2, dtype="float64")
v3 = numpy.array(v3, dtype="float64")
#finding velocity of center of mass
    v_cm = (m1*v1+m2*v2+m3*v3)/(m1+m2+m3)
#defining equations of motion in a function
def three_body_equations(w,t,G,m1,m2,m3):
r1=w[:3]
r2=w[3:6]
r3=w[6:9]
v1=w[9:12]
    v2=w[12:15]
v3=w[15:18]
r12 = numpy.linalg.norm(r2-r1)
r13 = numpy.linalg.norm(r3-r1)
r23 = numpy.linalg.norm(r3-r2)
dv1bydt=K1*m2*(r2-r1)/r12**3+K1*m3*(r3-r1)/r13**3
dv2bydt=K1*m1*(r1-r2)/r12**3+K1*m3*(r3-r2)/r23**3
dv3bydt=K1*m1*(r1-r3)/r13**3+K1*m2*(r2-r3)/r23**3
dr1bydt=K2*v1
dr2bydt=K2*v2
dr3bydt=K2*v3
r12_derivs=numpy.concatenate((dr1bydt,dr2bydt))
r_derivs=numpy.concatenate((r12_derivs,dr3bydt))
v12_derivs=numpy.concatenate((dv1bydt,dv2bydt))
v_derivs=numpy.concatenate((v12_derivs,dv3bydt))
derivs=numpy.concatenate((r_derivs,v_derivs))
return derivs
#creating figure for plotting
    fig = plt.figure(figsize=(15,15))
```

```
#create 3d axes
ax = fig.add_subplot(111,projection="3d")
import scipy.integrate
def update(frame):
ax.clear()
global r1,r2,r3,v1,v2,v3,K1,K2
init_params = numpy.array([r1,r2,r3,v1,v2,v3])
init_params = init_params.flatten() # flatten to 1d array
time_span = numpy.linspace(0,0.1,20)
#time_span = numpy.linspace(0,4,500)
three_body_sol = sci.integrate.odeint(three_body_equations,init_params,time_span,args=(G,m1,m2,m3))
r1_sol = three_body_sol[:,:3]
r2_sol = three_body_sol[:,3:6]
    r3_sol = three_body_sol[:,6:9]
v1_sol = three_body_sol[:,9:12]
v2_sol = three_body_sol[:,12:15]
v3_sol = three_body_sol[:,15:18]
#Finding location of COM
#rcom_sol=(m1*r1_sol+m2*r2_sol+m3*r3_sol)/(m1+m2+m3)
#Finding location of first star w.r.t COM
r1com_sol= r1_sol#-rcom_sol
    #Finding location of second star w.r.t COM
r2com_sol= r2_sol#-rcom_sol
#Finding the location of third star w.r.t. COM
r3com_sol = r3_sol#-rcom_sol
#Ploting the orbits in COM frame
ax.plot(r1com_sol[:,0],r1com_sol[:,1],r1com_sol[:,2],color="darkblue")
ax.plot(r2com_sol[:,0],r2com_sol[:,1],r2com_sol[:,2],color="tab:red")
ax.plot(r3com_sol[:,0],r3com_sol[:,1],r3com_sol[:,2],color="green")
#Ploting the final positions of the stars in COM Frame
ax.scatter(r1com_sol[-1,0],r1com_sol[-1,1],r1com_sol[-
1,2],color="darkblue",marker="o",s=100,label="Alpha Centauri A")
ax.scatter(r2com_sol[-1,0],r2com_sol[-1,1],r2com_sol[-
1,2],color="tab:red",marker="o",s=100,label="Alpha Centauri B")
ax.scatter(r3com_sol[-1,0],r3com_sol[-1,1],r3com_sol[-1,2],color="green",marker="o",s=100,label="Third
Star")
#"""
r1 = [r1_sol[-1,0],r1_sol[-1,1],r1_sol[-1,2]]
```

```
r2 = [r2_sol[-1,0],r2_sol[-1,1],r2_sol[-1,2]]
r3 = [r3_sol[-1,0],r3_sol[-1,1],r3_sol[-1,2]]
v1 = [v1_sol[-1,0],v1_sol[-1,1],v1_sol[-1,2]]
v2 = [v2_sol[-1,0],v2_sol[-1,1],v2_sol[-1,2]]
    v3 = [v3_sol[-1,0],v3_sol[-1,1],v3_sol[-1,2]]
r1 = numpy.array(r1, dtype="float64")
r2 = numpy.array(r2, dtype="float64")
v1 = numpy.array(v1, dtype="float64")
v2 = numpy.array(v2, dtype="float64"
#ani.event_source.stop()
ani = FuncAnimation(fig, update, interval=500)
ax.set_xlabel("x-coordinate",fontsize=14)
ax.set_ylabel("y-coordinate",fontsize=14)
ax.set_zlabel("z-coordinate",fontsize=14)
ax.set_title("Visualization of orbits of stars in a two-body system\n",fontsize=14)
    plt.show()
```

On running the simulation for three body problem, with initial conditions:

m1	*m2*	*m3*
1.1	*0.907*	*1.0*

r1	*r2*	*r3*
[-0.5, 0.2 ,0.0]	[0.5,0.0,0.1]	[0.2,0.5,0.0]

v1	*v2*	*v3*
[0.01,0.01,0.0]	[-0.05,0.0, -0.1]	[0.0, -0.01,0.0]

The orbits of the body abruptly change with change of initial conditions and are chaotic in nature.

A typical chaotic orbit in our simulation is:

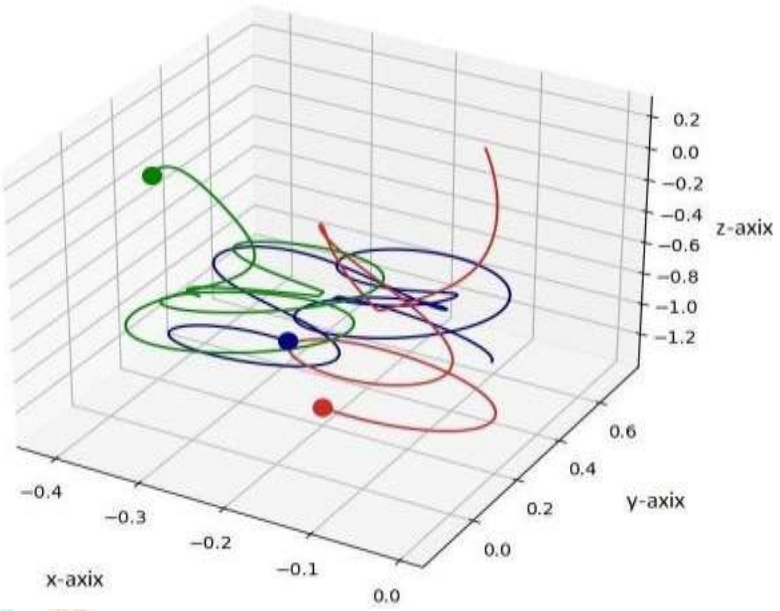

Fig4. Chaotic orbit of three body system

Over the long-term evolution of the system, it is found that one of the bodies is ejected outof the system and two bodies seems to orbit each other in a stable orbit.

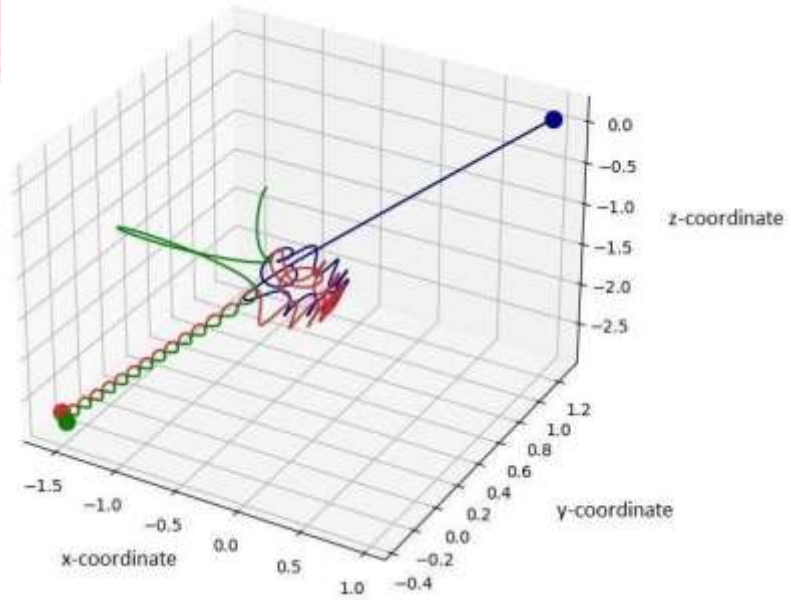

Fig5. Long term evolution of three body system

(c) Modeling four body problems:

On running the simulation for four bodies for some initial conditions, the orbits are very chaotic:

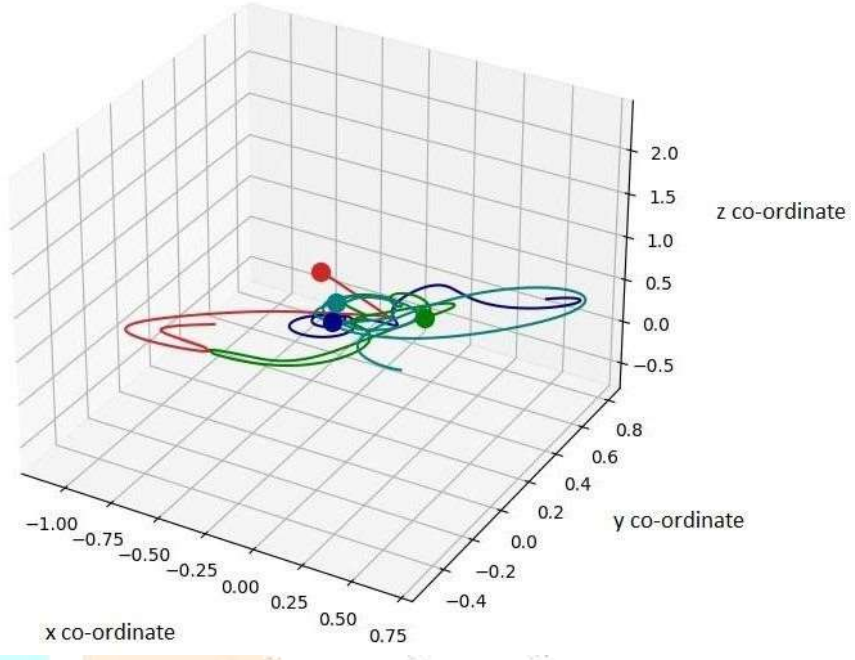

Fig6. Chaos in four body problem

IV. Conclusion

The results of above simulations are very important in understanding the interaction between different numbers of bodies and their trajectories. It can help us in understanding the formation of planetary systems, star clusters and binaries. Most work done on the *n*-body problem has been on the gravitational problem. But there exist other systems for which *n*-body mathematics and simulation techniques have proven useful. In large scale electrostatics problems, such as the simulation of proteins and cellular assemblies in structural biology, the Coulomb potential has the same form as the gravitational potential, except that charges may be positive or negative, leading to repulsiveas well as attractive forces.

V. References

1. E. I. Abouelmagd and S. M. El-Shaboury. Periodic orbits under combined effects of oblateness and radiation in the restricted problem of three bodies. Astrophys. Space Sci., 341:331–341, 2012.

2. 50 E. I. Abouelmagd, H. M. Asiri, and M. A. Sharaf. The effects of oblateness in the perturbed restrictive three-body problem. Meccanica, 48:2479–2490, 2013.

3. J. C. Adams. Explanation of the observed irregularities in the motion of uranus, on the hypothesis of disturbance by a more distant planet; with a determination of the mass, orbit, and position of the disturbing body. Mon. Not. Royal Astron. Soc., 7:149–152, November 1846.

4. E. Agol, J. Steffen, R. Sari, and W. Clarkson. On detecting terrestrial planets with timing of giant planet transits. Mon. Not. Royal Astron. Soc., 359:567–579, May 2005.

5. A. Ahmad. Stability of the periodic solutions of the restricted three-body problem representing analytical continuations of keplerian rectilinear periodic motions. Celestial Mechanics and Dynamical Astronomy, 65:181–196, 1995.

6. M. K. Ahmed, F. A. Abd El-Salam, and S. E. Abd El-Bar. On the stability of the triangular lagrangian equilibrium points in the relativistic restricted three-body problem. American Journal of Applied Sciences, 3:1993–1998, 2006.

7. R. Barnes and R. Greenberg. Stability limits in extrasolar planetary systems. ApJL, 647:L163–L166, August 2006.

8. J. Barrow-Green. Oscar ii's prize competition and the error in poincar´e's memoir on the three body problem. Archive for History of Exact Sciences, 48:107–131, December 1994.