# Assignment 1
# AE313 - Spaceflight Mechanics

Subrahmanya V Bhide (SC18B030)
*Indian Institute of Space Science and Technology*
*Department of Aerospace Engineering*
(Dated: 25 October 2020)

*This is a report on the Assignment 1 given in the course, AE313 - Spaceflight Mechanics. The transformation from state vector to orbital elements and vice-versa is implemented using python code.*

## Question 1

The orbital parameters provided are listed in the Tab. I.

TABLE I: Orbital Parameters

| Parameter | Value |
|---|---|
| Semi Major Axis ($a$) | $25000km$ |
| Eccentricity ($e$) | 0.1 |
| Inclination ($i$) | $40^\circ$ |
| Argument of Periapsis ($\omega$) | $120^\circ$ |
| Right Ascension of Acsending Node ($\Omega$) | $250^\circ$ |
| True Anomaly ($\nu$) | $0^\circ$ |

## Question 2

State vectors of two orbits/spacecrafts at time $t_0$ are given which are summarized in Tabs. II and III.

TABLE II: State Vector at $t = t_0$ for Spacecraft 1

| Parameter | Value ($km$ and $km/s$) |
|---|---|
| $x$ | $-8503.8558701333313$ |
| $y$ | 14729.110427313784 |
| $z$ | 6190.3008264368991 |
| $\dot{x}$ | $-4.3229148869407341$ |
| $\dot{y}$ | $-2.3293249804847838$ |
| $\dot{z}$ | $5.24855402558600526 \times 10^{-2}$ |

TABLE III: State Vector at $t = t_0$ for Spacecraft 2

| Parameter | Value ($km$ and $km/s$) |
|---|---|
| $x$ | $-13686.889393418738$ |
| $y$ | $-13344.772667428870$ |
| $z$ | 10814.629905439588 |
| $\dot{x}$ | 0.88259108105901152 |
| $\dot{y}$ | 1.9876415852134037 |
| $\dot{z}$ | 3.4114313525042017 |

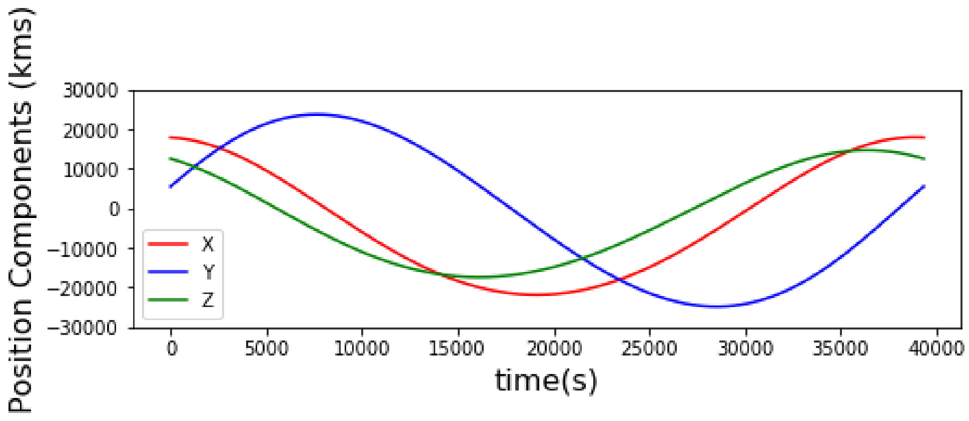

FIG. 1: Position Components in Geocentric frame vs time.

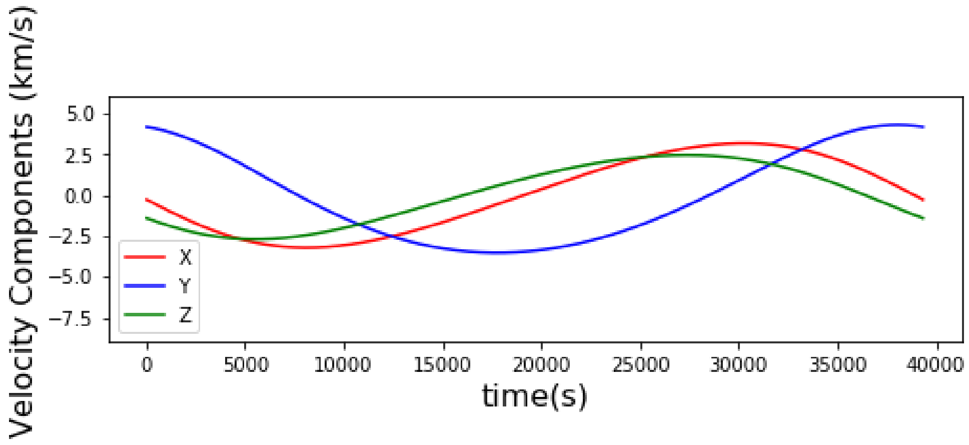

FIG. 2: Velocity Components in Geocentric frame vs time.

## RESULTS & DISCUSSIONS

### Question 1

Figure 1 shows the variation of Position Components in Geocentric frame with time. In the plots we can see that since the plot is for one complete orbit, the values at the start and end are the same for all the components.
Figure 2 shows the variation of Velocity Components in Geocentric frame with time. As in the previous plot i.e. Fig. 1, the values at the start and end are the same for all the components of velocity.

Figure 3 shows the variation of Radial distance of the spacecraft from the origin of the Geocentric frame of refrence with time. In the plot we can see that the least radial distance is at perigee and the maximum radial distance occurs at about half the period which corresponds to apogee.
From the graphs we obtain $r_p = 22499.999999999996\ km$ and $r_a = 27499.999993410962\ km$. From calculations $r_p = a(1-e) = 22500\ km$ and $r_a = a(1+e) = 27500\ km$ which are almost same as the values obtained from the graphs .

Figure 4 shows the variation of Velocity of the spacecraft with time. From Keplers Second Law we infer that the velocity of the spacecraft is the highest at perigee and decrases and reaches a minimum value at apogee. This is observed in the Fig. 4. From the plot the minimum and maximum velocity i.e $V_a$ and $V_p$ are $3.611798847670885 km/s$ and $4.414420812644748 km/s$ respectively and from the vis-viva equation

$$V^2 = \mu\left(\frac{2}{r} - \frac{1}{a}\right)$$

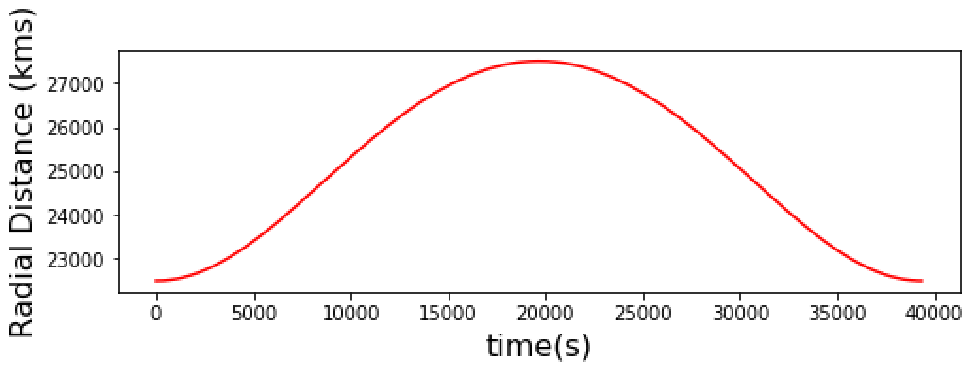

FIG. 3: Radial distance of the spacecraft vs time.

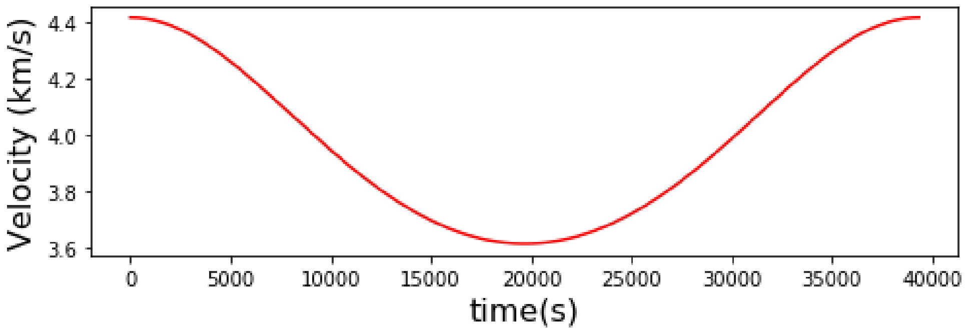

FIG. 4: Velocity of the spacecraft vs time.

substituting $r_a$ and $r_p$ we obtain $V_a = 3.611798847 km/s$ and $V_p = 4.414420813 km/s$ which match with the values obtained from the graphs.

Figure 5 shows the variation of Flight Path angle ($\gamma$) with True Anomaly ($\nu$). The expression for Flight Path Angle ($\gamma$) is given as

$$tan\gamma = \frac{esin\nu}{1 + ecos\nu}$$

The maximum value of $\gamma$ occurs when

$$\nu = -cos^{-1}e$$

In our case it corresponds to $\nu = 95.739°$ and $\gamma = 5.739170477°$. From the graph too the maximum value for $\gamma$ is $5.739170474854725°$. Similarly the minimum value of $\gamma$ is $-5.739170474854725°$ and it occurs at $\nu = 360° - 95.739° = 264.261°$

Figure 6 shows the variation of Eccentric Anomaly ($E$) with True Anomaly ($\nu$). The curve appears to be a straight line in the plot but it is not so, which can be noticed upon close observations through the help of grid lines, since the variation is not linear but involves trignometric quantities. Figure 7 shows the variation of Mean Anomaly ($M$) with True Anomaly ($\nu$).

Figure 8 shows the variation of Radial and Tangential velocity with True Anomaly ($\nu$) . Here in this plot we can observe that $V_r$ is zero and $V_t$ is maximum at perigee. $V_t$ attains its minimum value at apogee. $V_r$ is zero at the perigee and apogee and attains maximum and minimum values in between these two points.

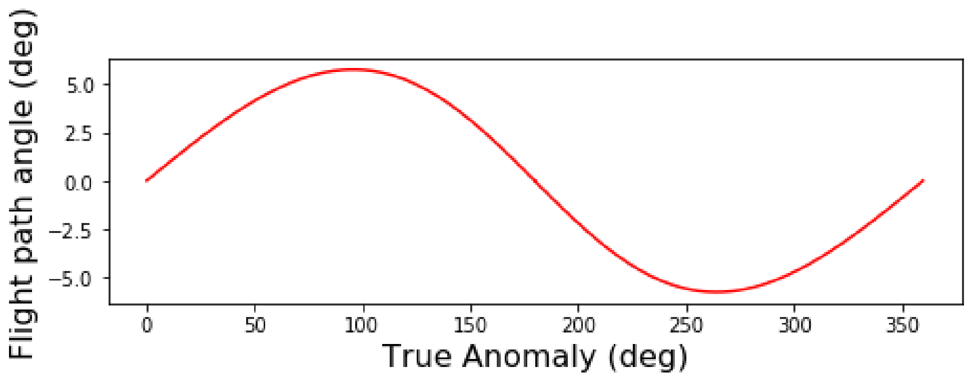

FIG. 5: Flight Path angle ($\gamma$) vs True Anomaly ($\nu$).

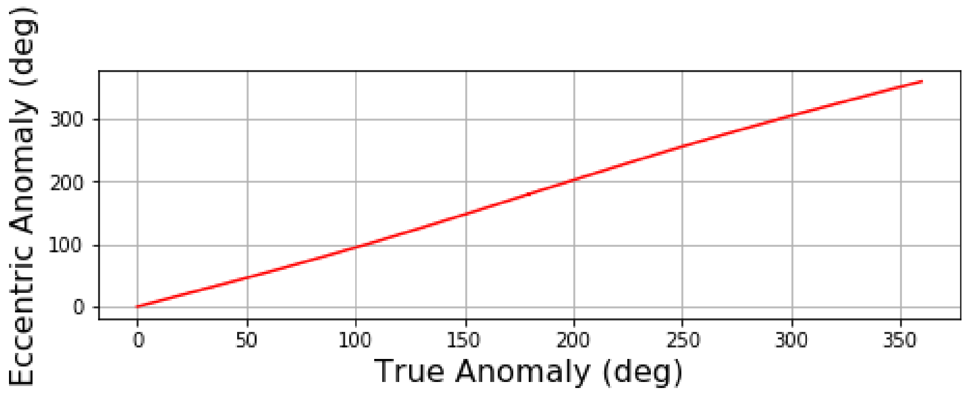

FIG. 6: Eccentric Anomaly ($E$) vs True Anomaly ($\nu$).

Figure 9 shows the variation of Radial velocity with Tangential velocity.
Figure 10 shows the variation of Angular Momentum of the spacecraft with time. From the plot we can see that the angular momentum is conserved.

Fig. 11 shows the 3D plot of x, y and z. The plot is an ellipse which is the orbit shape.

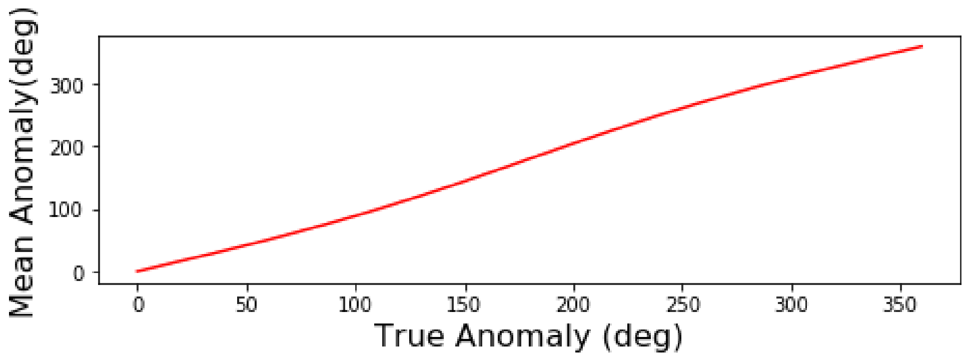

FIG. 7: Mean Anomaly ($M$) vs True Anomaly ($\nu$).

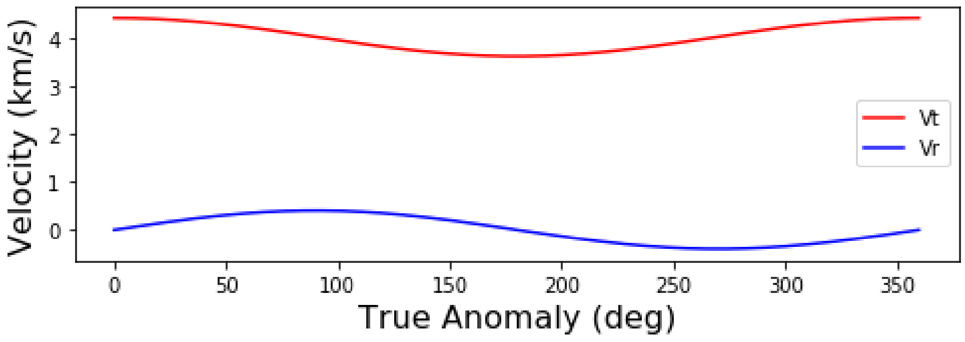

FIG. 8: Radial and Tangential velocity vs True Anomaly ($\nu$).

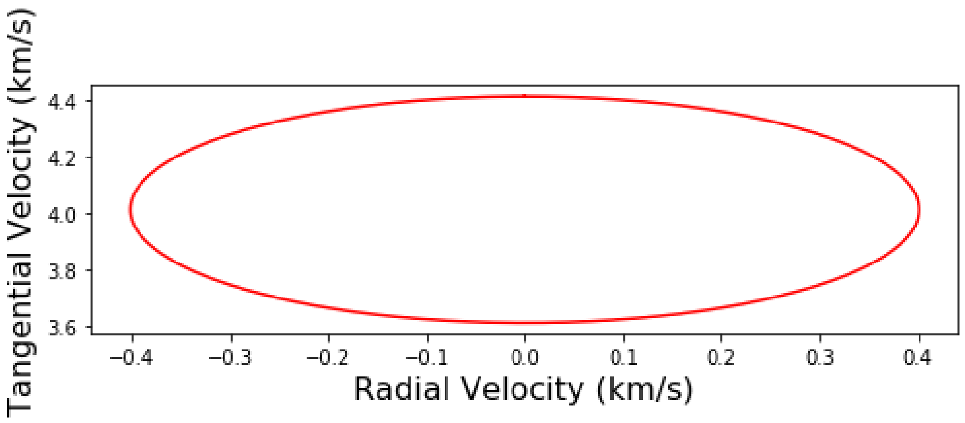

FIG. 9: Radial velocity vs Tangential velocity.

**Question 2**

The orbital parameters for Spacecraft 1 are found and tabulated in Tab. IV.
The Period of this orbit is $28148.61951s$. Table V shows the state vector after $1000s$.

TABLE IV: Orbital Parameters for Spacecraft 1

| Parameter | Value |
|---|---|
| Semi Major Axis ($a$) | $20000.027200108972km$ |
| Eccentricity ($e$) | 0.10000117003650112 |
| Inclination ($i$) | 19.999999999999975° |
| Argument of Periapsis ($\omega$) | 70.00022024697651° |
| Right Ascension of Acsending Node ($\Omega$) | 29.999999999999993° |
| True Anomaly ($\nu(t_0)$) | 19.999779753023482° |
| True Anomaly ($\nu(t_0 + 1000)$) | 35.37648852191359° |

The orbital parameters for Spacecraft 2 are found and tabulated in Tab. VI.
The Period of this orbit is $28148.60106s$. Table VII shows the state vector after $1000s$
Figure 12 shows both the orbits.
We can observe that in the same time i.e. 1000s Spacecraft 1 has a increment of about 15° in $\nu$ where as the Spacecraft 2 has an increment of about 10°. This implies that Spacecraft 1 is nearer to its perigee than Spacecraft 2. An animation of the two orbits along with the positions of spacecrafts at the two time instants $t_0$ and $t_0 + 1000$ can

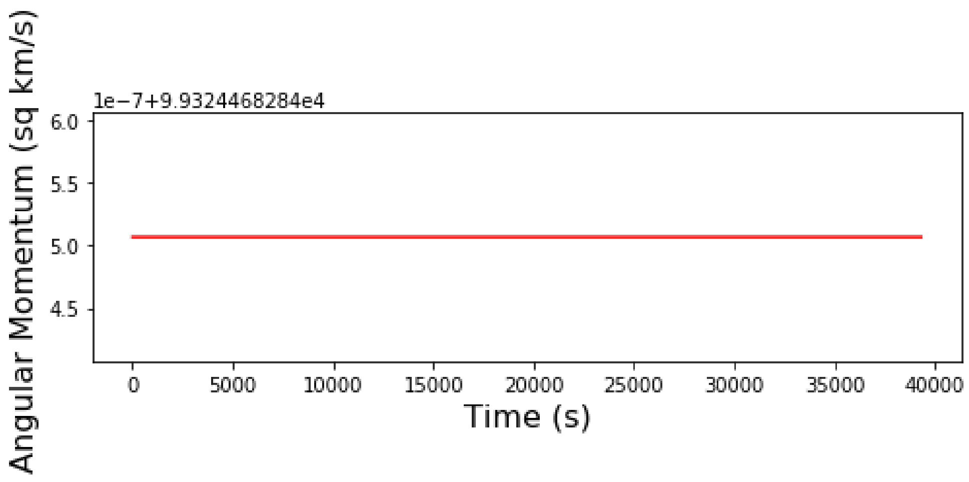

FIG. 10: Angular Momentum of the spacecraft vs time.

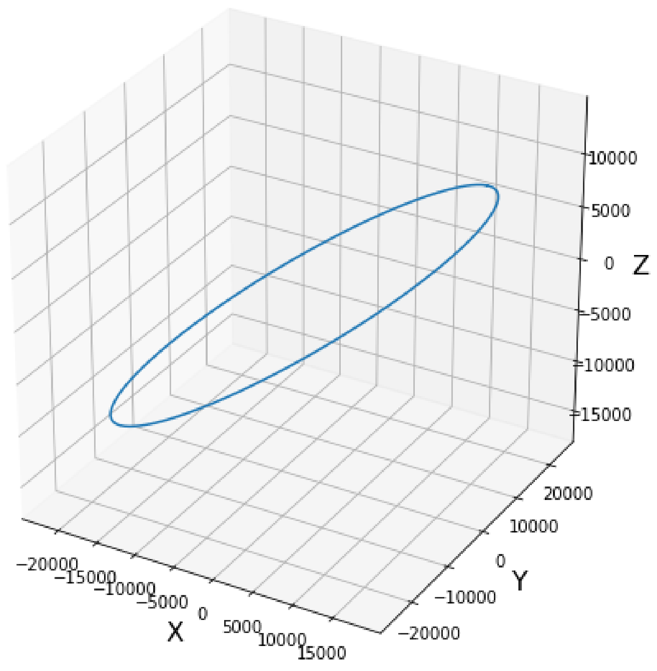

FIG. 11: 3D plot of X, Y and Z.

TABLE V: State Vector at $t = t_0 + 1000$ for Spacecraft 1

| Parameter | Value ($km$ and $km/s$) |
|---|---|
| $x$ | $-12497.7704005766853$ |
| $y$ | $11937.89734463553$ |
| $z$ | $6037.322618580829$ |
| $\dot{x}$ | $-3.624827305630527$ |
| $\dot{y}$ | $-3.217223319918579$ |
| $\dot{z}$ | $-0.354428198385424$ |

TABLE VI: Orbital Parameters for Spacecraft 2

| Parameter | Value |
|---|---|
| Semi Major Axis ($a$) | 20000.01846065068$km$ |
| Eccentricity ($e$) | 0.09999900702499653 |
| Inclination ($i$) | 100.0° |
| Argument of Periapsis ($\omega$) | 199.99988817526062° |
| Right Ascension of Acsending Node ($\Omega$) | 230.0° |
| True Anomaly ($\nu(t_0)$) | 190.00011182473935° |
| True Anomaly ($\nu(t_0 + 1000)$) | 200.6029534° |

TABLE VII: State Vector at $t = t_0 + 1000$ for Spacecraft 2

| Parameter | Value ($km$ and $km/s$) |
|---|---|
| $x$ | $-12552.041450214183$ |
| $y$ | $-11118.28352305635$ |
| $z$ | 14000.8449257959 |
| $\dot{x}$ | 1.3812752652579983 |
| $\dot{y}$ | 2.45251446469216 |
| $\dot{z}$ | 2.9395822864428554 |

be found here. The directions of motion of the spacecrafts are along the smaller arcs in the corresponding orbits.

## Python Codes

### Question 1

The .py file for this code can be found here.

```
import numpy as np
from numpy import sin, cos,sqrt,pi,arctan,tan,degrees
from numpy import radians
import matplotlib.pyplot as plt
from mpl_toolkits import mplot3d

#function for Newton Raphson Method
def raph_f(E,M,e):
return E - e*sin(E) -M
#Derivative of the previous function
def raphdash_f(E,e):
return 1- e*cos(E)
#Recursive solver to find Eccentric Anomaly
def solve_kepler(E0,M,e,error):
E = E0
delta = raph_f(E,M,e)
c =abs(delta)
if c-error ¿ 0:
E1 = E - (raph_f(E,M,e)/raphdash_f(E,e))
return solve_kepler(E1,M,e,error)
else:
return E
#Function to find TA given EA and e
def new(E,e):
```

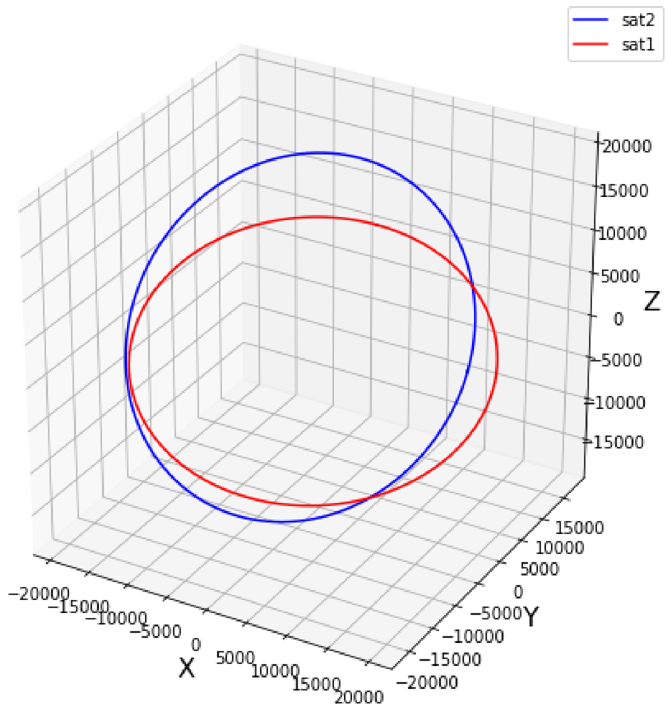

FIG. 12: 3D plots of the orbit followed by Spacecraft 1 and 2.

```
return 2*arctan((sqrt((1+e)/(1-e)))*tan(E*0.5))
#Function to convert orbital elements to state vector
def orbit_2_state(xp,xpd,yp,ypd,zp,zpd,a,e,i,o,O,n):
R11 = (cos(O)*cos(o)) - (sin(O)*sin(o)*cos(i))
R12 = (cos(O)*sin(o)) + (sin(O)*cos(o)*cos(i))
R13 = sin(i)*sin(O)
R21 = (sin(O)*cos(o)) + (cos(O)*sin(o)*cos(i))
R22 = (cos(O)*cos(o)*cos(i)) - (sin(O)*sin(o))
R23 = -sin(i)*cos(O)
R31 = (sin(o)*sin(i))
R32 = (cos(o)*sin(i))
R33 = cos(i)
x = R11*xp - R12*yp + R13*zp
y = R21*xp + R22*yp + R23*zp
z = R31*xp + R32*yp + R33*zp
xd = R11*xpd - R12*ypd + R13*zpd
yd = R21*xpd + R22*ypd + R23*zpd
zd = R31*xpd + R32*ypd + R33*zpd
return [x,y,z,xd,yd,zd]
#defining variables for the orbit
a = 25000 #km
e = 0.1
error = 10**(-6) #order of acceptable error in EA
mu = 398600 #km3/s
H = sqrt(mu*a*(1 - e**2))
n = sqrt(mu/a**3)
T = sqrt((4*(pi**2)*(a**3))/mu)
t = np.linspace(0,T,int(T))
inc = radians(40) #Inclination
omega = radians(120) #Angle of Periapsis
```

```
Omega = radians(250) #Right Ascension of Ascending node
#Finding M values for the orbit
M = np.zeros(len(t))
M[0] = 0.0
for i in range(1,len(t)):
M[i] = n*(t[i] - t[0])
#Using Recursive function finding EA values
E = np.zeros(len(t))
E[0] = 0.0
for i in range(1,len(t)):
E[i] = solve_kepler(M[i],M[i],e,error)
#TA array
nur = np.zeros(len(t))
for i in range(0,len(t)):
if new(E[i],e) ¿ 0 :
nur[i] = new(E[i],e)
else :
nur[i] = 2*3.14 + new(E[i],e)

   #TA in degrees
nud = np.zeros(len(t))
for i in range(0,len(t)):
if new(E[i],e) ¿ 0 :
nud[i] = degrees(new(E[i],e))
else :
nud[i] = degrees(2*3.14 + new(E[i],e))

   #Arrays of components of the state vectors at different instants during the orbit
x=[]
y=[]
z=[]
xd=[]
yd=[]
zd=[]
for i in range(0,len(t)):
r = (a*(1 - (e**2)))/(1 + e*cos(nur[i]))
xp = r*cos(nur[i])
xpd = (-mu/H)*sin(nur[i])
yp = r*sin(nur[i])
ypd = (mu/H)*(e + cos(nur[i]))
zp = 0
zpd = 0
soln = orbit_2_state(xp,xpd,yp,ypd,zp,zpd,a,e,inc,omega,Omega,nur[i])
x.append(soln[0])
y.append(soln[1])
z.append(soln[2])
xd.append(soln[3])
yd.append(soln[4])
zd.append(soln[5])

   #Finding some other relevant quantities using the state vector
Md = []#Mean Anomaly in degrees
Ed =[]# Eccentric Anomaly in degrees
r = []#Radial distance v = []#Velocity
gamma = []#Flight path angle
vr = []#radial velocity
vt = []#tangential velocity
Hexp = []# Value of angluar momentum from the state vectors
```

```
for i in range(0,len(t)):
r.append(sqrt(x[i]**2 + y[i]**2 + z[i]**2))
v.append(sqrt(xd[i]**2 + yd[i]**2 + zd[i]**2))
gamma.append((180/pi)*arctan((e*sin(nur[i]))/(1 + e*cos(nur[i]))))
vr.append(v[i]*sin(gamma[i]))
vt.append(v[i]*cos(gamma[i]))
Hexp.append(r[i]*vt[i])
Ed.append(E[i]*180/pi)
Md.append(M[i]*180/pi)

  #Plotting relevant figures
fig = plt.figure(1, figsize=(8,8))
ax = plt.axes(projection='3d')
ax.plot3D(x, y, z)
ax.set_xlabel('X',size='16')
ax.set_ylabel('Y',size='16')
ax.set_zlabel('Z',size='16')
```

## Question 2

The .py file for this code can be found here.

```
import numpy as np
from numpy import sin, cos,sqrt,pi,arctan,tan,degrees,arccos,arctan2
from numpy import radians
import matplotlib.pyplot as plt
from IPython.display import HTML
from matplotlib import animation
from mpl_toolkits import mplot3d
#Defining general functions for some vector operations
#dot product
def dot(a,b):
return a[0]*b[0] + a[1]*b[1] + a[2]*b[2]
#magnitude of the vector
def modulus(a):
return sqrt(a[0]**2 + a[1]**2 + a[2]**2)
#Scalar multiplication of a vector
def sm(a,b):
return [a*b[0] , a*b[1] , a*b[2] ]
#negatie of a vector
def n(a):
return [-a[0] , -a[1] , -a[2]]
#sum of 3 vectors
def Sum(a,b,c):
return [(a[0] + b[0] +c[0]) , (a[1] + b[1] +c[1]) , (a[2] + b[2] +c[2]) ]
#cross product def cross(a,b):
return [(a[1]*b[2] - a[2]*b[1]) , (a[2]*b[0] - a[0]*b[2]) , (b[1]*a[0] - a[1]*b[0])]

  #Function to convert state vector to orbital elements
def state_2_orbit(x,y,z,xd,yd,zd):
rvec = [x,y,z]
vvec = [xd,yd,zd]
r = modulus(rvec)
v = modulus(vvec)
rc = sm((1/modulus(rvec)),rvec)
a = (mu*r)/(2*mu - r*v**2)
evec = sm(1/mu , Sum(sm(v**2,rvec),n(sm(dot(rvec,vvec),vvec)), n(sm(mu/r,rvec))))
```

```
e = modulus(evec)
ec = sm((1/modulus(evec)),evec)
wc = sm((1/modulus(cross(rvec,vvec))),cross(rvec,vvec))
Nc = sm((1/modulus(cross(kc,wc))),cross(kc,wc))
pc = cross(ec,rc)
if arccos(dot(wc,kc)) ¿ 0:
inc = arccos(dot(wc,kc))
else :
inc = 2*pi + arccos(dot(wc,kc))
if arctan2(dot(Nc,jc),dot(ic,Nc)) ¿ 0:
O = arctan2(dot(Nc,jc),dot(ic,Nc))
else :
O = 2*pi + arctan2(dot(Nc,jc),dot(ic,Nc))
if arctan2(dot(wc,cross(Nc,ec)),dot(Nc,ec)) ¿ 0:
o = arctan2(dot(wc,cross(Nc,ec)),dot(Nc,ec))
else :
o = 2*pi + arctan2(dot(wc,cross(Nc,ec)),dot(Nc,ec))
if arctan2(dot(pc,wc),dot(ec,rc)) ¿ 0:
nu = arctan2(dot(pc,wc),dot(ec,rc))
else :
nu = 2*pi + arctan2(dot(pc,wc),dot(ec,rc))
return [a,e,inc,O,o,nu]

  def raph_f(E,M,e):
return E - e*sin(E) -M

  def raphdash_f(E,e):
return 1- e*cos(E)

  def solve_kepler(E0,M,e,error):
E = E0
delta = raph_f(E,M,e)
c =abs(delta)
if c-error ¿ 0:
E1 = E - (raph_f(E,M,e)/raphdash_f(E,e))
return solve_kepler(E1,M,e,error)
else:
return E
def new(E,e):
return 2*arctan((sqrt(1+e)/sqrt(1-e))*tan(E*0.5))
def Ecc(n,e):
return 2*arctan((sqrt(1-e)/sqrt(1+e))*tan(n*0.5))

  def Mean(E,e):
return E - e*sin(E)

  def orbit_2_state(xp,xpd,yp,ypd,zp,zpd,a,e,i,o,O,n):
R11 = cos(O)*cos(o) - sin(O)*sin(o)*cos(i)
R12 = -cos(O)*sin(o) - sin(O)*cos(o)*cos(i)
R13 = sin(i)*sin(O)
R21 = sin(O)*cos(o) + cos(O)*sin(o)*cos(i)
R22 = -sin(O)*sin(o) + cos(O)*cos(o)*cos(i)
R23 = -sin(i)*cos(O)
R31 = sin(o)*sin(i)
R32 = cos(o)*sin(i)
R33 = cos(i)
R = np.array([[R11,R12,R13] , [R21,R22,R23] ,[R31,R32,R33]])
```

```
   rp = np.array([xp,yp,zp])
rpd = np.array([xpd,ypd,zpd])

   r = np.matmul(R,rp)
rd = np.matmul(R,rpd)

   return [r[0],r[1],r[2],rd[0],rd[1],rd[2]]

   #defining given conditions and relevant constants
mu = 398600 #km3/s
ic = [1,0,0]
jc = [0,1,0]
kc = [0,0,1]

   x1 = -8503.8558701333313
y1 = 14729.110427313784
z1 = 6190.3008264368991
x1d = -4.3229148869407341
y1d = -2.3293249804847838
z1d = 5.24855402558600526*0.01

   x2 = -13686.889393418738
y2 = -13344.772667428870
z2 = 10814.629905439588
x2d = 0.88259108105901152
y2d = 1.9876415852134037
z2d = 3.4114313525042017

   #Obtaining orbital elemants
orbit_element_1 = state_2_orbit(x1,y1,z1,x1d,y1d,z1d)
orbit_element_2 = state_2_orbit(x2,y2,z2,x2d,y2d,z2d)

   a1 = orbit_element_1[0]
e1 = orbit_element_1[1]
i1 = orbit_element_1[2]
O1 = orbit_element_1[3]
o1 = orbit_element_1[4]
nu1 = orbit_element_1[5]
a2 = orbit_element_2[0]
e2 = orbit_element_2[1]
i2 = orbit_element_2[2]
O2 = orbit_element_2[3]
o2 = orbit_element_2[4]
nu2 = orbit_element_2[5]

   E1 = Ecc(nu1,e1)
E2 = Ecc(nu2,e2)
M1 = Mean(E1,e1)
M2 = Mean(E2,e2)

   n1 = sqrt(mu/a1**3)
n2 = sqrt(mu/a2**3)
#Obtaining MA,TA,EA after 1000s
M1d = n1*1000 + M1
M2d = n2*1000 + M2

   E1d = solve_kepler(M1d,M1d,e1,0.000001)
E2d = solve_kepler(M2d,M2d,e2,0.000001)
```

```
  nu1d = new(E1d,e1)
nu2d = new(E2d,e2)

  H1 = sqrt(mu*a1*(1 - e1**2))
H2 = sqrt(mu*a2*(1 - e2**2))

  r1 = a1*(1 - e1**2)/(1 + e1*cos(nu1d))
xp1 = r1*cos(nu1d)
xpd1 = -(mu/H1)*sin(nu1d)
yp1 = r1*sin(nu1d)
ypd1 = (mu/H1)*(e1 + cos(nu1d))
zp1 = 0
zpd1 = 0
r2 = a2*(1 - e2**2)/(1 + e2*cos(nu2d))
xp2 = r2*cos(nu2d)
xpd2 = -(mu/H2)*sin(nu2d)
yp2 = r2*sin(nu2d)
ypd2 = (mu/H2)*(e2 + cos(nu2d))
zp2 = 0
zpd2 = 0
#Finding state vectors after 1000s
state_vec_d_1 = orbit_2_state(xp1,xpd1,yp1,ypd1,zp1,zpd1,a1,e1,i1,o1,O1,nu1d)
state_vec_d_2 = orbit_2_state(xp2,xpd2,yp2,ypd2,zp2,zpd2,a2,e2,i2,o2,O2,nu2d)
```

[1] https://elainecoe.github.io/orbital-mechanics-calculator/calculator.html
[2] https://janus.astro.umd.edu/orbits/elements/convertframe.html
[3] http://www2.arnes.si/~gljsentvid10/ele2vec.html