SPACE FLIGHT MECHANICS

Project - 1

MAYANK AHUJA

AIM:

This report aims to implement the methods to transform the state vectors to Orbital Elements and vice-versa.

METHODOLOGY:

The Rotation matrix and reverse transformation methods as studied in the Class were implemented using Python Code. Also, some of the parameters were also plotted using Python's matplotlib.pyplot library. In the plots some of the key observations were noted and are gone through at the last of this report.

Also, the solution of Kepler's Equation was done using Newton-Raphson method implicitly using the code. For that, a Thousand iterations were performed to find the Eccentric anomaly from the Mean anomaly.

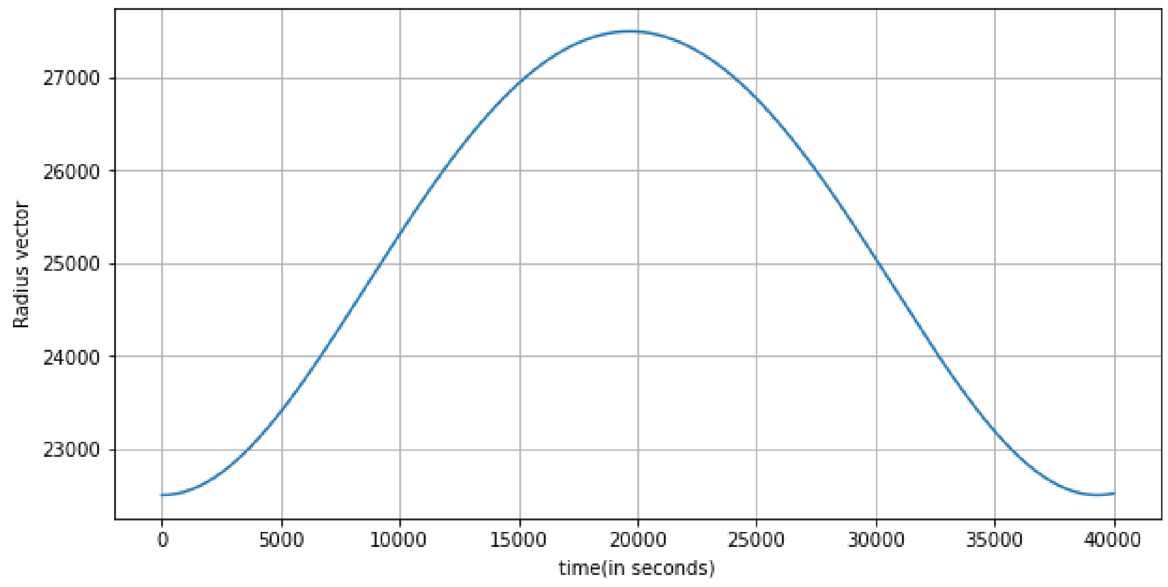
Finally, when given the initial state vectors, final state vectors were found using the seven steps as were discussed in the class. Python codes for all of these are provided at the end of this report.

Q1: When the following parameters were given:

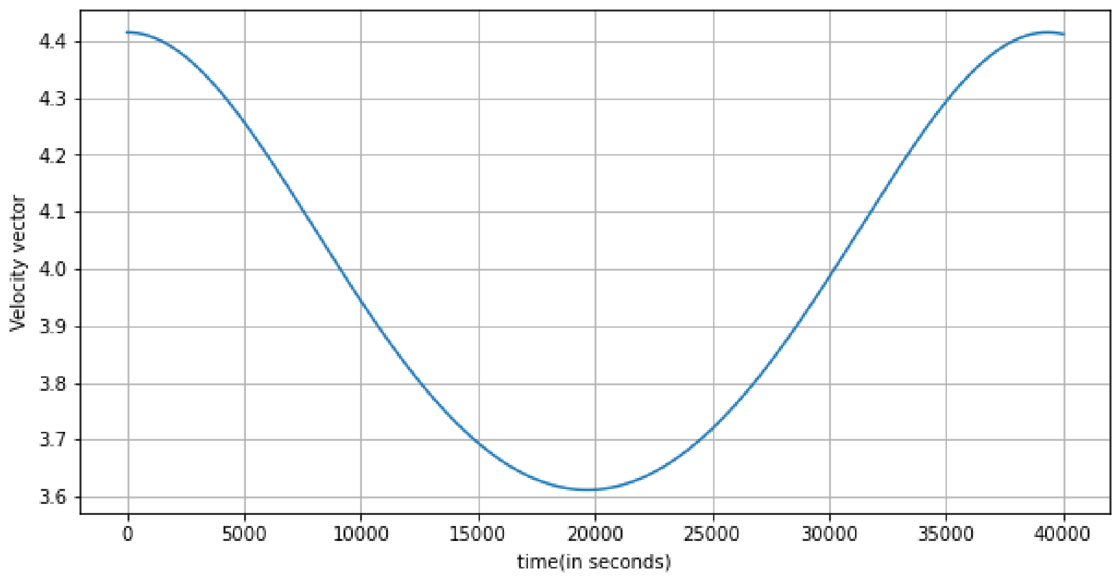
MA = 25000 km; ECC = 0.1 ; INC =40 deg; AoP = 120 deg; RAAN = 250 deg; TA = 0.0 deg

A rotation matrix was calculated using the above parameters and using that the state vectors were calculated.

Firstly, the radius vector w.r.t time was plotted as follows:

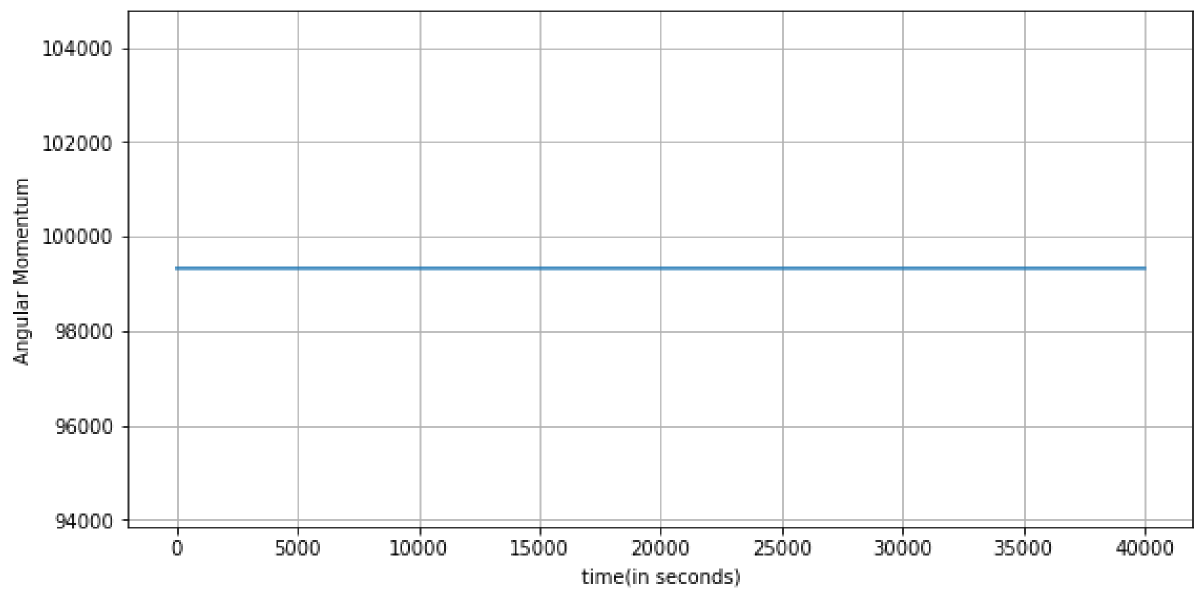


After this, the velocity vector (magnitude) was also plotted as follows:



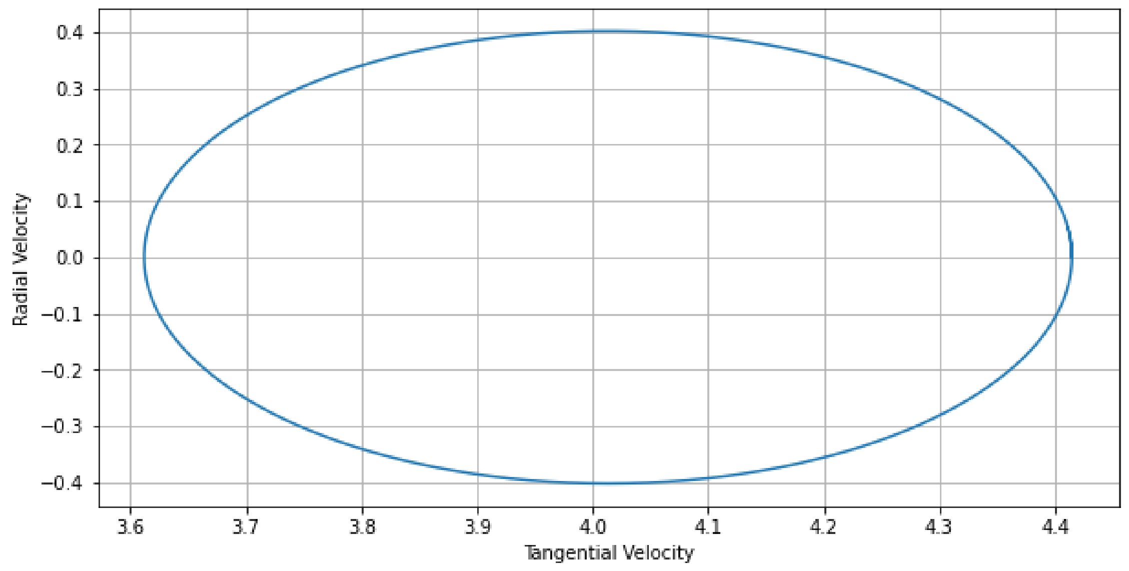
Both of these plots seem to be consistent as when the spacecraft is moving away from the focus, it's speed starts to slow down and when it's moving towards the planet it's speed starts to increase as it's angular momentum is constant. This can be seen in the above plots as well. Also, when calculating the time period of one full motion, it came somewhere near 39000 seconds which can also be seen in the above plots.

After this, Angular momentum w.r.t time was plotted:



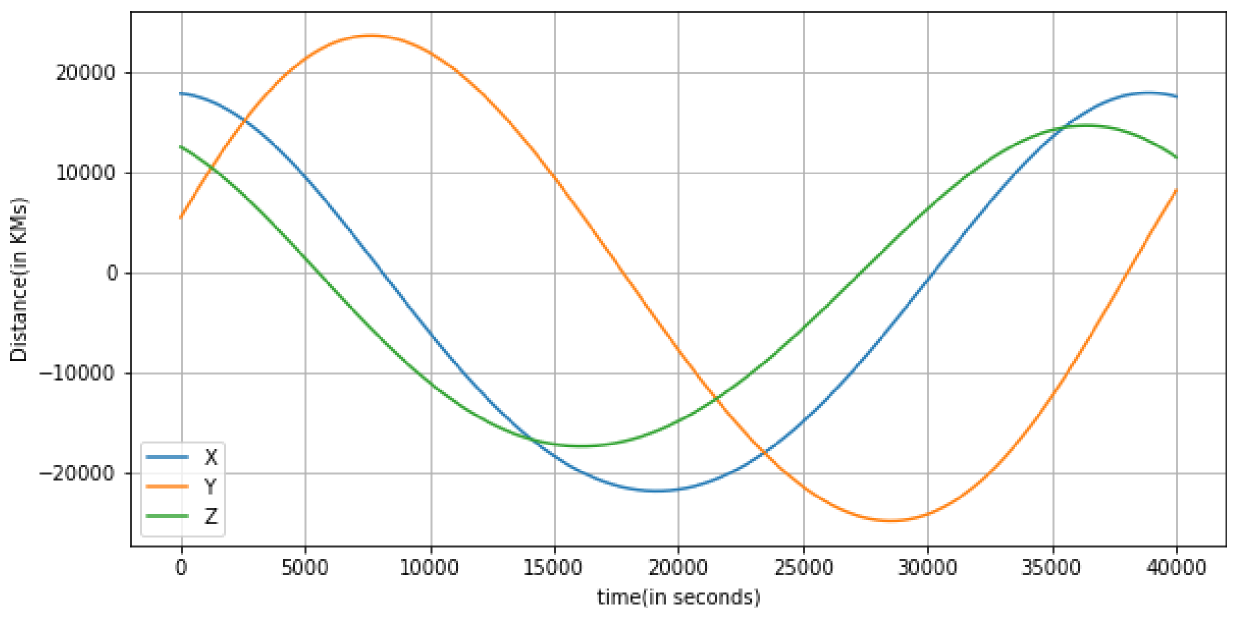
As Angular momentum should be constant throughout this motion, this can be seen in the above plot w.r.t time.

After this is shown the plot of radial velocity v/s Tangential velocity:

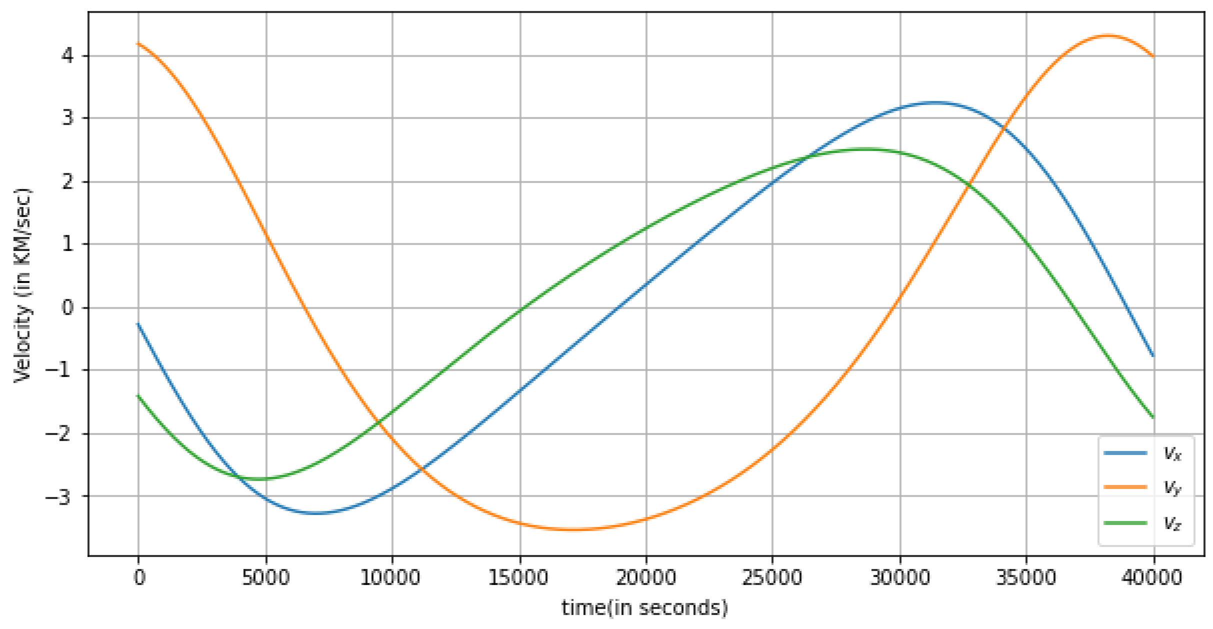


This plot also looks consistent as the zero Radial speed corresponds to minimum tangential speed at apogee as well as maximum tangential speed at perigee. Also, at min. Tangential speed, maximum radial velocity (magnitude) is achieved.

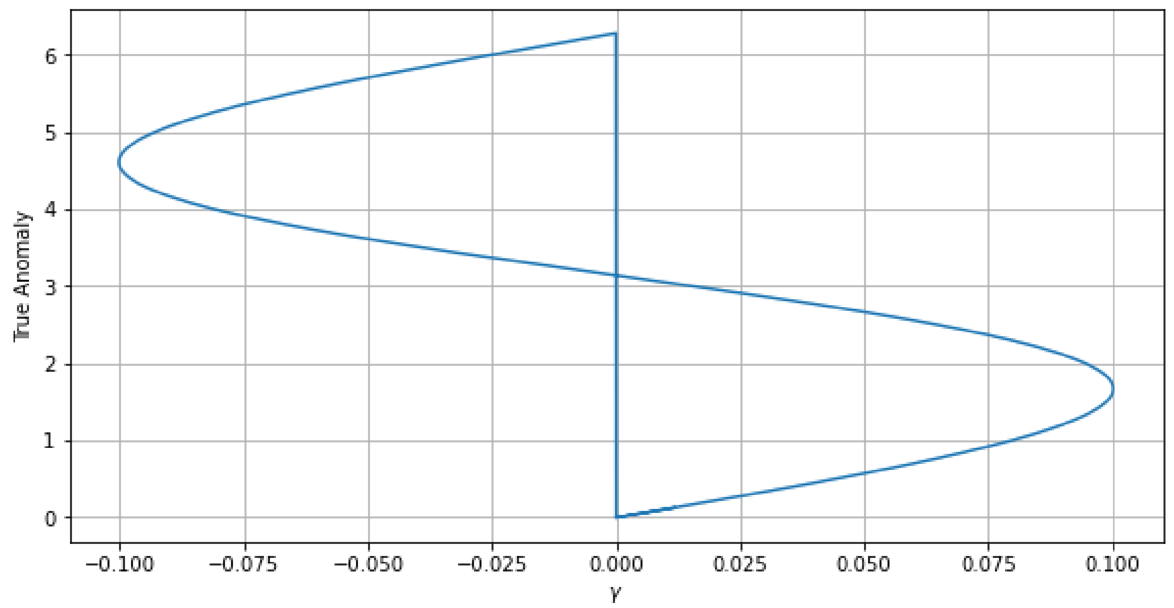
After this is plotted state vectors (both position and velocity) with respect to time:
The plot for Position vectors w.r.t time:



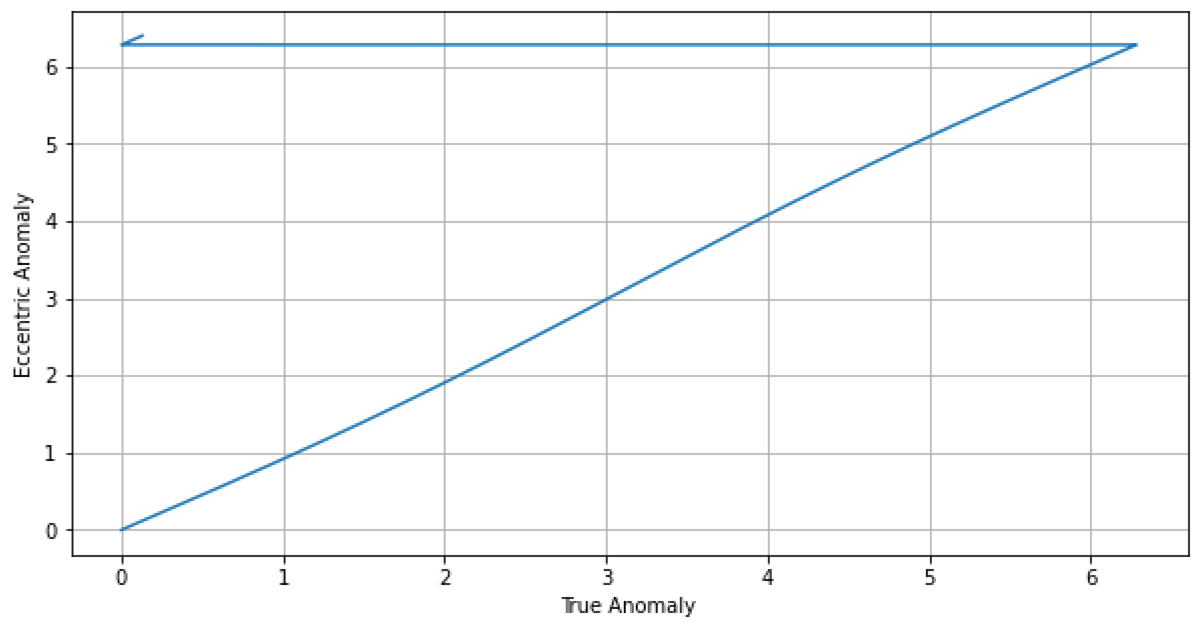
The plot for Velocity vectors w.r.t time:



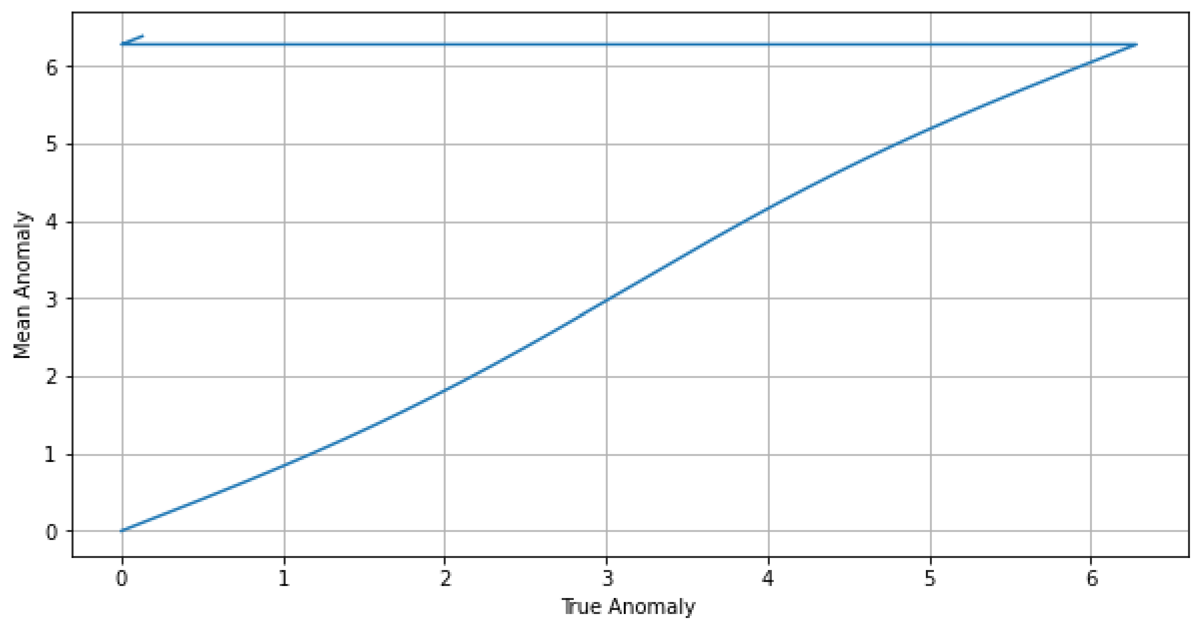
After this is plotted Flight path angle(gamma) v/s True anomaly:



After this is plotted True anomaly v/s Eccentric Anomaly:



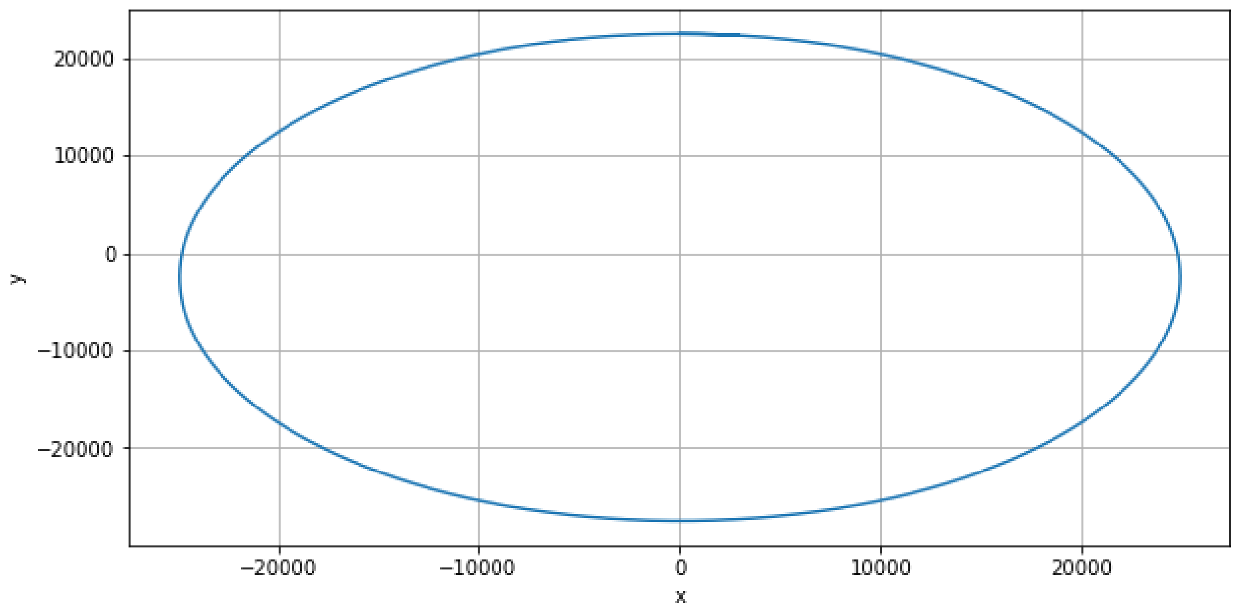
After this is plotted True anomaly v/s Mean Anomaly:



The Mean anomaly and Eccentric anomaly's plots look very similar as Mean anomaly is very closely related with Eccentric anomaly as:

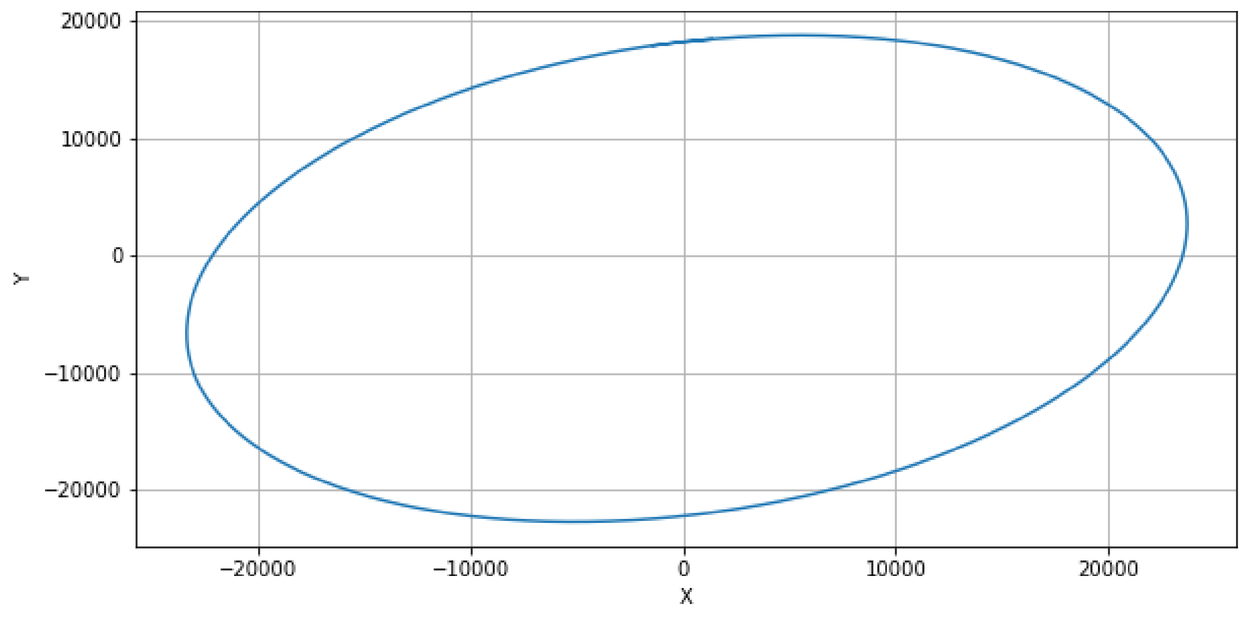
Mean anomaly(M) = Eccentric anomaly(E) - e * sin(E)
For small values of eccentricity which is the case here (0.1) they'll be very close to each other, there is always a slight change in their values as I confirmed using my code.
At, Mean anomaly = 0.3194714562862482 radians
Corresponding Eccentric anomaly = 0.35415086107496313 radians

Apart from what asked in the question I also tried plotting X and Y coordinates of the spacecraft in both orbital and Earth's Reference Frame and they Came out as:
In Orbital Reference Frame:



As expected as the ellipse was plotted in this representing the elliptical orbit the spacecraft is going on.

In Earth's Reference Frame:



As expected, a tilted ellipse came out in Earth's reference frame as it's inclined at a certain angle w.r.t Earth's reference frame.

Q2: Case 1:

The initial state vectors which were given in this case were:

X = -8503.8558701333313

Xdot = -4.3229148869407341

Y=14729.110427313784

Ydot = -2.3293249804847838

Z = 6190.3008264368991

Zdot = 5.24855402558600526e-2

It was asked to find final state vectors after 10000 seconds of the spacecraft. This is a typical true body problem and it demanded to go through all the steps we went through in the class for first transforming the state vectors to orbital elements and then apply the two- body problem steps to find out the final True Anomaly and finally transform them to state vectors once again:

Using the code given at the end of this report we found out the final state vectors of this problem and they came out as:

Position vectors = (-12497.77039794 , 11937.89734697 , 6037.32261884) Kms

Velocity vectors = (-3.62482731 -3.21722332 -0.3544282) Km/sec.

Also, in order to calculate these state vectors, orbital elements were calculated and they came out as:

Eccentricity=0.10000117003650137

AOP =70.00022024697653 Degrees

SMA = 20000.027200108972 Kms

RAAN = 29.999999999999993 Degrees

Inc. = 19.999999999999975 Degrees

Similarly for the second case:

Final Position vectors: (-14307.97155129 , 15070.41311265 , -7222.19181875) Kms

Velocity vectors: (-0.35613091 , 1.45329428 , 3.75068776) Km/sec.

The codes used in this report are given as:

Python code for first Question:

```
import numpy as np

import math

from sympy import *

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

from scipy.optimize import *

E=[]

r=[]

v=[]

nu=[]

H=[]
```

```
gamma=[]

vh=[]

vt=[]

t=np.linspace(0,40000,10000)

#M=2.099438697


F=0

Fprime=0

e=0.1

I=0.698132

omega=2.0944

Comega=4.36332


#print(E[100])

a=25000

TA=[]

TA.append(0)

X=[]

Y=[]

Z=[]

vx=[]

vy=[]

vz=[]
```

```
VX=[]

VY=[]

VZ=[]

mu=398600

r11 = math.cos(Comega)*math.cos(omega) -
math.sin(Comega)*math.sin(omega)*math.cos(I)

r12 = -math.cos(Comega)*math.sin(omega) -
math.sin(Comega)*math.cos(omega)*math.cos(I)

r13 = math.sin(Comega)*math.sin(I)

r21 = math.sin(Comega)*math.cos(omega) +
math.cos(Comega)*math.sin(omega)*math.cos(I)

r22 = -math.sin(Comega)*math.sin(omega) +
math.cos(Comega)*math.cos(omega)*math.cos(I)

r23 = -math.cos(Comega)*math.sin(I)

r31 = math.sin(omega)*math.sin(I)

r32 = math.cos(omega)*math.sin(I)

r33 = math.cos(I)


#R=(R1,R2,R3)

#XP=([r*np.cos(TA), r*np.sin(TA), 0])

R1=[r11,r12,r13]

R2=[r21,r22,r23]

R3=[r31,r32,r33]

xp=[]

yp=[]
```

```
M=[]

mm=np.sqrt(mu/(a**3))


for i in range(0,len(t)):

 M.append(mm*t[i])

 E.append(M[i]-0.1)


 for j in range(100):

   F=E[i]-e*np.sin(E[i])-M[i]

   Fprime=1-e*np.cos(E[i])

   E[i]=E[i]-F/Fprime




#print(E[0])

for i in range(0,len(t)):

 r.append(a*(1-e*np.cos(E[i])))

 v.append(np.sqrt(mu*(2/r[i]-1/a)))

 nu.append(2 * np.arctan(np.sqrt((1+e)/(1-e)) * np.tan(E[i]/2)))

 if (nu[i]>=0):

   nu[i]=nu[i]

 else:

   nu[i]=nu[i]+2*np.pi

 gamma.append(np.arctan(e*np.sin(nu[i])/(1+e*np.cos(nu[i]))))
 H.append(np.sqrt(mu*a*(1-e**2)))

 vt.append(mu/H[i]*(1+e*np.cos(nu[i])))

 vh.append(mu/H[i]*e*np.sin(nu[i]))




for i in range(len(t)):

 mat1 = (R1, R2, R3)

 mat2 = ([r[i]*np.cos(nu[i]), r[i]*np.sin(nu[i]), 0])

 res = np.dot(mat1,mat2)

 X.append(res[0])

 Y.append(res[1])

 Z.append(res[2])

 xp.append(r[i]*np.cos(nu[i]))

 yp.append(r[i]*np.sin(nu[i]))

 vx.append(-v[i]*np.sin(gamma[i]+nu[i]))

 vy.append(v[i]*np.cos(gamma[i]+nu[i]))

 vz.append(0)

 vaft=([-v[i]*np.sin(gamma[i]+nu[i]),v[i]*np.cos(gamma[i]+nu[i]),0])

 res2=np.dot(mat1,vaft)

 VX.append(res2[0])

 VY.append(res2[1])

 VZ.append(res2[2])
```

```
print(M[500])

print(E[500])

plt.figure(figsize=(10,5))

plt.plot(t,X,label=r'X')

plt.plot(t,Y,label=r'Y')

plt.plot(t,Z,label=r'Z')

plt.legend(loc='best')

plt.ylabel('Distance(in KMs)')

plt.xlabel('time(in seconds)')

plt.grid()

plt.show()

plt.figure(figsize=(10,5))

plt.plot(t,VX,label=r'$v_x$')

plt.plot(t,VY,label=r'$v_y$')

plt.plot(t,VZ,label=r'$v_z$')

plt.legend(loc='best')

plt.ylabel('Velocity (in KM/sec)')

plt.xlabel('time(in seconds)')

plt.grid()

plt.show()

plt.figure(figsize=(10,5))

plt.plot(t,r)

plt.grid()
plt.xlabel('time(in seconds)')

plt.ylabel('Radius vector')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(t,v)

plt.xlabel('time(in seconds)')

plt.ylabel('Velocity vector')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(gamma,E)

plt.xlabel(r'$\gamma$')

plt.ylabel('Eccentric Anomaly')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(gamma,nu)

plt.xlabel(r'$\gamma$')

plt.ylabel('True Anomaly')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()
plt.plot(nu,E)

plt.xlabel('True Anomaly')

plt.ylabel('Eccentric Anomaly')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(nu,M)

plt.xlabel('True Anomaly')

plt.ylabel('Mean Anomaly')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(t,H)

plt.xlabel('time(in seconds)')

plt.ylabel('Angular Momentum')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(vt,vh)

plt.xlabel('Tangential Velocity')

plt.ylabel('Radial Velocity')

plt.show()

plt.figure(figsize=(10,5))
plt.grid()

plt.plot(Y,X)

plt.xlabel('X')

plt.ylabel('Y')

plt.show()

plt.figure(figsize=(10,5))

plt.grid()

plt.plot(yp,xp)

plt.xlabel('x')

plt.ylabel('y')

plt.show()
```

Python code for second Question:

```
import numpy as np

import math

from sympy import *

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

from scipy.optimize import *
```

```
e=0.1

I=0.698132

omega=2.0944

Comega=4.36332


x0=-8503.8558701333313

xdot0 = -4.3229148869407341

y0=14729.110427313784

ydot0 = -2.3293249804847838

z0= 6190.3008264368991

zdot0 =  5.2485540255860052 6e-2

r0=np.sqrt(x0**2+y0**2+z0**2)

v0=np.sqrt(xdot0**2 + ydot0**2 + zdot0**2)

mu=398600

a=1/(2/r0-(v0**2)/mu)

r=np.array([x0, y0, z0])

rcap=r/r0

v=np.array([xdot0,ydot0,zdot0])

vcap=v/v0

e=v0**2/mu*r-np.dot(r,v)/mu*v-r/r0

e0=np.sqrt(e[0]**2+e[1]**2+e[2]**2)

ecap=e/e0

w=np.cross(r,v)
```

```
w0=np.sqrt(w[0]**2+w[1]**2+w[2]**2)

wcap=w/w0

kcap=[0,0,1]

I=np.arccos(np.dot(wcap,kcap))

n=np.cross(kcap,wcap)

n0=np.sqrt(n[0]**2+n[1]**2+n[2]**2)

ncap=n/n0

icap=[1,0,0]

Comega=np.arccos(np.dot(icap,ncap))

Comega2=np.arcsin(np.dot(np.cross(icap,ncap),kcap))

omega=np.arccos(np.dot(ncap,ecap))

nu=np.arccos(np.dot(rcap,ecap))

nu0=np.arcsin(np.dot(np.cross(ecap,rcap),wcap))

print(nu0)



E0=2*np.arctan(np.sqrt((1-e0)/(1+e0))*np.tan(nu/2))

M0=E0-e0*np.sin(E0)

mm=np.sqrt(mu/(a**3))

#after 1000 seconds

M1=M0+mm*(1000)

E1=M1-0.1

for j in range(100):

   F=E1-e0*np.sin(E1)-M1
   Fprime=1-e0*np.cos(E1)

   E1=E1-F/Fprime


nu1=np.arctan(np.sqrt((1+e0)/(1-e0))*np.tan(E1/2))*2

omegahai=omega*180/np.pi

r1=a*(1-e0*np.cos(E1))

v1=mu*(2/r1-1/a)

Comega5=180/np.pi*Comega

x1=r1*np.cos(nu1)

y1=r1*np.sin(nu1)

z1=0

final=[x1,y1,z1]

gamma1=np.arctan(e0*np.sin(nu1)/(1+e0*np.cos(nu1)))

H = np.sqrt(mu*a*(1-(e0**2)))

v1x=-mu/H*np.sin(nu1)

v1y=mu/H*(e0+np.cos(nu1))

finalspeed=[v1x,v1y,0]


r11 = np.cos(Comega)*np.cos(omega) -
np.sin(Comega)*np.sin(omega)*np.cos(I)

r12 = -np.cos(Comega)*np.sin(omega) -
np.sin(Comega)*np.cos(omega)*np.cos(I)

r13 = np.sin(Comega)*np.sin(I)
```

```
r21 = np.sin(Comega)*np.cos(omega) +
np.cos(Comega)*np.sin(omega)*np.cos(I)

r22 = -np.sin(Comega)*np.sin(omega) +
np.cos(Comega)*np.cos(omega)*np.cos(I)

r23 = -np.cos(Comega)*np.sin(I)

r31 = np.sin(omega)*np.sin(I)

r32 = np.cos(omega)*np.sin(I)

r33 = np.cos(I)



R = np.array([[r11,r12,r13],[r21,r22,r23],[r31,r32,r33]])

finaltransformed=np.dot(R,final)

finalspeedT=np.dot(R,finalspeed)

#print(a)

print(finaltransformed)

print(finalspeedT)

#print(e0)

#print(I2)

#print(nu0)

#print(Comega5)

#print(omegahai)
```

```
import numpy as np

import math

from sympy import *

import matplotlib.pyplot as plt

from matplotlib.pyplot import figure

from scipy.optimize import *


e=0.1

I=0.698132

omega=2.0944

Comega=4.36332


x0=-13686.889393418738

xdot0 = 0.88259108105901152

y0=-13344.772667428870

ydot0 = 1.9876415852134037

z0= 10814.629905439588

zdot0 =  3.4114313525042017
```

```
r0=np.sqrt(x0**2+y0**2+z0**2)

v0=np.sqrt(xdot0**2 + ydot0**2 + zdot0**2)

mu=398600

a=1/(2/r0-(v0**2)/mu)

r=np.array([x0, y0, z0])

rcap=r/r0

v=np.array([xdot0,ydot0,zdot0])

vcap=v/v0

e=v0**2/mu*r-np.dot(r,v)/mu*v-r/r0

e0=np.sqrt(e[0]**2+e[1]**2+e[2]**2)

ecap=e/e0

w=np.cross(r,v)

w0=np.sqrt(w[0]**2+w[1]**2+w[2]**2)

wcap=w/w0

kcap=[0,0,1]

I=np.arccos(np.dot(wcap,kcap))

n=np.cross(kcap,wcap)

n0=np.sqrt(n[0]**2+n[1]**2+n[2]**2)

ncap=n/n0

icap=[1,0,0]

Comega=np.arccos(np.dot(icap,ncap))

Comega2=np.arcsin(np.dot(np.cross(icap,ncap),kcap))

omega=np.arccos(np.dot(ncap,ecap))
```

```
nu=np.arccos(np.dot(rcap,ecap))

nu0=np.arcsin(np.dot(np.cross(ecap,rcap),wcap))

print(nu0)


E0=2*np.arctan(np.sqrt((1-e0)/(1+e0))*np.tan(nu/2))

M0=E0-e0*np.sin(E0)

mm=np.sqrt(mu/(a**3))

#after 1000 seconds

M1=M0+mm*(1000)

E1=M1-0.1

for j in range(100):

   F=E1-e0*np.sin(E1)-M1

   Fprime=1-e0*np.cos(E1)

   E1=E1-F/Fprime


nu1=np.arctan(np.sqrt((1+e0)/(1-e0))*np.tan(E1/2))*2

omegahai=omega*180/np.pi

r1=a*(1-e0*np.cos(E1))

v1=mu*(2/r1-1/a)

Comega5=180/np.pi*Comega

x1=r1*np.cos(nu1)

y1=r1*np.sin(nu1)

z1=0
```

```
final=[x1,y1,z1]

gamma1=np.arctan(e0*np.sin(nu1)/(1+e0*np.cos(nu1)))

H = np.sqrt(mu*a*(1-(e0**2)))

v1x=-mu/H*np.sin(nu1)

v1y=mu/H*(e0+np.cos(nu1))

finalspeed=[v1x,v1y,0]



r11 = np.cos(Comega)*np.cos(omega) -
np.sin(Comega)*np.sin(omega)*np.cos(I)

r12 = -np.cos(Comega)*np.sin(omega) -
np.sin(Comega)*np.cos(omega)*np.cos(I)

r13 = np.sin(Comega)*np.sin(I)

r21 = np.sin(Comega)*np.cos(omega) +
np.cos(Comega)*np.sin(omega)*np.cos(I)

r22 = -np.sin(Comega)*np.sin(omega) +
np.cos(Comega)*np.cos(omega)*np.cos(I)

r23 = -np.cos(Comega)*np.sin(I)

r31 = np.sin(omega)*np.sin(I)

r32 = np.cos(omega)*np.sin(I)

r33 = np.cos(I)



R = np.array([[r11,r12,r13],[r21,r22,r23],[r31,r32,r33]])

finaltransformed=np.dot(R,final)

finalspeedT=np.dot(R,finalspeed)

#print(a)
```

```
print(finaltransformed)

print(finalspeedT)

#print(e0)

#print(I2)

#print(nu0)

#print(Comega5)

#print(omegahai)
```