Siddula, Saimanoj

Astrodynamics II

Midterm

Table of Contents

[Problem 1 2](#_Toc445111191)

[Introduction 2](#_Toc445111192)

[Analysis 2](#_Toc445111193)

[Results: 4](#_Toc445111194)

[Solution: 5](#_Toc445111195)

[Problem 2 6](#_Toc445111196)

[Introduction 6](#_Toc445111197)

[Analysis 6](#_Toc445111198)

[Results 8](#_Toc445111199)

[Problem 3 9](#_Toc445111200)

[Introduction 9](#_Toc445111201)

[Analysis 9](#_Toc445111202)

[Results 11](#_Toc445111203)

[Problem 4 12](#_Toc445111204)

[Introduction 12](#_Toc445111205)

[Analysis 12](#_Toc445111206)

[Results 13](#_Toc445111207)

[Appendix 14](#_Toc445111208)

[Code for Problem 1 14](#_Toc445111209)

[Code for Problem 2/3 17](#_Toc445111210)

[Code for Problem 4 27](#_Toc445111211)

# Problem 1

## Introduction

Two spacecraft are in the same elliptical Earth orbit labeled 1, shown in Figure 1(not to scale). The perigee radius of orbit 1 is 7000 km, and its apogee radius is 10000 km, both lying on the P axis. At the instant shown, the spacecraft at B has a true anomaly of 90 degrees and the spacecraft at C is 30 degrees ahead of B. At this same moment, spacecraft B initiates a high-thrust maneuver to place it onto trajectory 2. Spacecraft B on orbit 2 intercepts s/c C still in orbit 1 exactly at the apogee of orbit 1, denoted A.

1. The semi-major axis of orbit 2.
2. The eccentricity vector e of orbit 2 stated in the {PQW} system shown below
3. The vector delta V needed at B to enter orbit 2
4. The vector delta V needed at A for s/c B to rendezvous with s/c C on orbit 1.
5. The total scalar delta-V of both maneuvers to allow the two craft to fly in formation with each other in orbit 1 from point A onwards. (Hint: deltaV approximately 5km/s)

## Analysis

Before even considering solving this problem, the best thing to do is to find out everything we can about orbit 1. Given the perigee and apogee, we can solve for everything we need.

With the basic constants found, we can use them in order to find the radius at point B, C and A using the fourth equation found above. With the ability to find the radius at any given ν, we can continue to find the velocity using the vice-viva equation.

Using Kepler’s equation, we can find the time it takes for s/c 1 to get from point C to point A, which ended up being 1704.290257 seconds. This is the same time it’ll take s/c 2 to use its orbit change and rendezvous with s/c 1 at point A.

It is important to note that at v=90 and 180 degrees is the velocity we need from orbit 1, since that is the starting and ending point respectively for the high thrust maneuver. From here we have the first vital information we need in order to solve the problem. From here, we find estimate values at orbit 2 using the Universal Variables solution to the Gauss problem.

To start about this method, only three things are needed to be known: initial radius at high thrust maneuver, final radius at high thrust and the delta v between starting and final radius. We start out initially estimating Z to be 0.

deltat\_t: time s/c 1 takes to get to A.

From here, since we know the time it takes s/c at point C to move to A, which should be the same time from s/c at point B to reach point A to rendezvous, a while loop is created to iterate through this phase recalculating t and all other values along with it until we finally get to a point where the difference margin between the times is negligible (below 10^-8).

Now that values have been optimized to a negligible degree, to continue using this method, the magnitudes of radius at the starting and finish of the maneuver must be calculated into vectors. To do this I simply used looked at the graph and noticed that point B is completely in the Q direction and point A is completely in the P direction. So the magnitudes were fully used in those directions and no other. From there I was able to use the following:

Here r1 and r2 are the initial and final points (B,A). From this the v1 and v2 given are the velocities in the second orbit 1 being the initial position and 2 being the final.

Before finding the delta\_v for each point I needed to convert the velocity magnitudes into vectors using the equation:

From there, it was just a matter of subtracting the velocities in orbit 1 from the v1 and v2 calculated a few steps above. Once the delta\_v vectors are found all I had to do was take the magnitudes to get the values for c and d, and e when I added them together.

Since part b needed the eccentricity for orbit 2, and the radius and velocity vectors are known, just need to use the formula to get the e vector:

To get the semi-major axis of the second orbit, use the vice viva equation since both radius and velocity magnitudes are known at two points. Either point can be used:

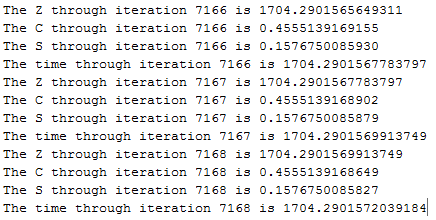
## Results:

After finding all the constants for the first orbit:

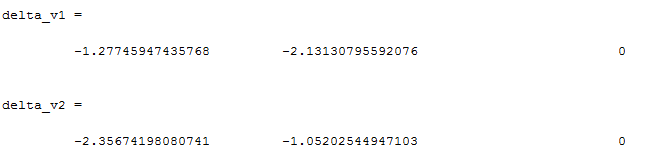


We can continue with solving for values in the second orbit.

After 7168 iterations the time converges to 1704.2915, which is very close to the actual value of 1704.290257 calculated with Kepler’s Equation.

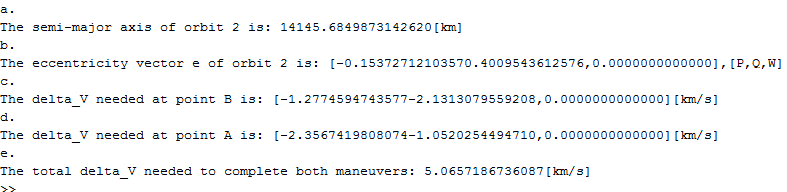


Values for Orbit 2:



The magnitudes are v\_1 = 2.48482923187838 and v\_2 = 2.58088944173027.

## Solution:



# Problem 2

## Introduction

The purpose of Gauss’ method is to be able to calculate an estimated value for the three radius vectors. Then the other method used to find the velocity vector is Gibbs method. The reason Gibbs method is used is to get a more accurate value for velocity vector. Gibbs method used three radius vectors to find the velocity vector.

Three sets of right ascension-declination observations in the Ropocentric-Horizon system have been made at a station located at 40 degrees N Latitude and 110 degrees West Longitude, and an altitude of 2000m on 20 August 2007. The data is shown below:

|  |  |  |  |
| --- | --- | --- | --- |
| Obs # | Hr:min:ss:ss(UT) | Right Ascension(deg) | Declination(deg) |
| 1 | 11:20:00:00 | -54.1249012 | -22.6156514 |
| 2 | 11:30:00:00 | -33.0588410 | -7.2056382 |
| 3 | 11:40:00:00 | -00.4172870 | 17.4626616 |

Find the Julian Day, Local Sidereal Times for each Observation, slant range unit vectors, slant range magnitudes, site position vectors.

## Analysis

First the Julian and Local Sidereal time are calculated using the process below given the day of observation:

Using the Local Sidereal Time found from the equations above they are then used with the times from the start of the problem then the radius of r1,r2 and r3 can be found using the following steps:

(Note: The East Latitude was needed, so instead of 110 degrees W, 250 degrees E was used)

Now r2 can be calculated using the equation:

After using the solve function, 8 solutions were found. In those solutions only on2 were real and from those only one was positive. I continued onwards with that r2 value:

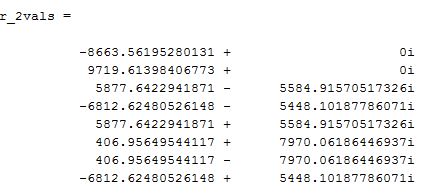
Once r2 is found it is then used to calculate the values of the vectors r1,r2,r3 with the following equations:

Now with the vectors for the radius at each observation Gibbs method is used to find a value and vector for v2 using the following equations:

## Results



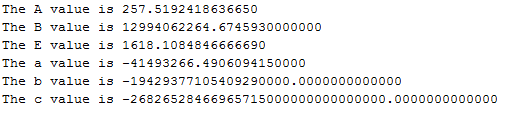
After this, plugging in the equation and solving for v2 gives us the following solutions:

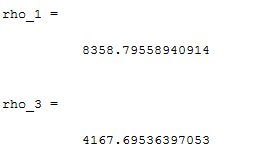


Out of these, only the real and positive was selected to be used for the rest of the process.











# Problem 3

## Introduction

With the Gauss Improvement method the values for the r2 vectors and the v2 vectors are used along with the respective magnitudes and the stumpff functions to better refine the error from the values to create a more accurate final value of r2 and v2. Once the values have been calculated then the classical orbital elements can be found through those vectors.

Problem Statement:

Use “improvement of the Gauss method to obtain better estimates of the ECI position and velocity vectors at the time of observation #2. Demonstrate that you have convergence in the improvement procedure. Using the “improved” results at Obs #2, estimate the following classical orbital elements (a,e,Ω).

## Analysis

Using the same results from problem 2 then the values for all the r vectors and v vectors can be better refined to ensure that the results are as accurate as possible. This is done by using the following equations in a loop iterating through until all the radius magnitudes have converged.

In this there are several equations that need to be numerically solved to find X for radius 1 and radius 3 these equations are listed below:

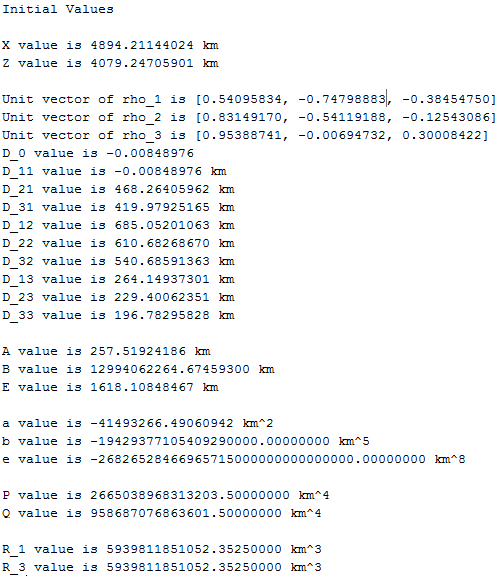
Once all the possible values for X1 and X2 are found only the value that is real and closest to 0 is kept and used later to find the values needed.

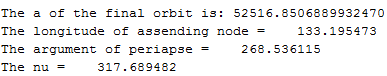
Next the values for rho1,rho2,rho3 are calculated along with the new r vectors and v vector

Now with a steady v2 value and r1, r2 and r3 the classical orbital elements can be calculated using the following equations:

, ,

## Results





# Problem 4

## Introduction

The theory behind Universal Variables is that it gives the ability to estimate accurately the location velocity of an object after a reasonable length time when given only the initial vectors of r and v at the initial observation.

At an instant of time an Earth spacecraft has the following radius and velocity vectors,

Find vectors r, v 40 minutes after this instant. Also report a, e for this orbit.

## Analysis

With the given equations first the magnitudes of both r and v are found and then the semi-major axis is calculated using the vice-viva equation:

Next is the Universal Variable X and Z are calculated using the following:

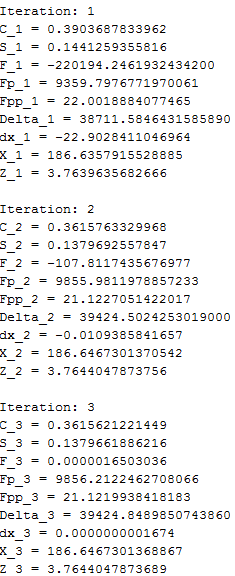
Now that the initial values for the Universal variables are found then the stumpff functions are then found along with the other needed functions as well as the delta functions and then recalculated for X and Z at the next step. The equations are shown below.

\*C

This process is repeated until the value of (dX^2/a)<10^-8 to ensure that the values for all coverage to a steady and appropriate value. Once they have been calculated then the values of f, f, fdot, gdot are found as well as the vectors of r and v. This is done with the equations listed below.

## Results

The values converged after 3 iterations:









# Appendix

## Code for Problem 1

%Saimanoj Siddula

%AerE 451 Astrodynamics II

%Problem 1

clear,clc

format longG

mu = 3.986e5;

%The values for Orbit 1 are given with:

r\_p1 = 7000; %km

r\_a1 = 10000; %km

new\_b0 = 90; %degrees

new\_c0 = new\_b0+30; %degrees

delta\_new0 = new\_c0-new\_b0;

%Calculations for the true values of the orbit 1

a1 = (r\_p1+r\_a1)/2;

e1 = 1-(r\_p1/a1);

p1 = a1\*(1-e1^2);

fprintf('a,e,p at orbit 1 are: %.13f,%.13f km,%.13f km',a1,e1,p1)

r\_b0 = p1/(1+e1\*cosd(new\_b0));

r\_c0 = p1/(1+e1\*cosd(new\_c0));

energy1 = -(mu)/(2\*a1);

v\_b0 = sqrt(2\*((-mu/(2\*a1))+(mu/r\_b0))); %Speed of s/c,1 at Initial

fprintf('Speed of s/c,1 at Initial %.13f\n',v\_b0)

v\_c0 = sqrt(2\*((-mu/(2\*a1))+(mu/r\_c0))); %Speed of s/c,2 at Initial

v\_fr = sqrt(2\*((-mu/(2\*a1))+(mu/r\_a1))); %Speed at Final Required.

fprintf('Speed of s/c,1 at Final %.13f\n',v\_fr)

%%

%Solving for the time from C at initial to 180 degres new.

E\_c0 = 2\*atan(sqrt((1-e1)/(1+e1))\*tan((new\_c0\*pi/180)/2));

E\_c1 = 2\*atan(sqrt((1-e1)/(1+e1))\*tan(pi/2));

delta\_t = sqrt((a1^3)/mu)\*((E\_c1-e1\*sin(E\_c1))-(E\_c0-e1\*sin(E\_c0))); %actual time in seconds.

%% (>^^)> <(^^)> <(^^<)

%Using Lambe't Equation:

delta\_newshort = 90;

DMshort = 1; %delta\_new<pi

r1 = r\_b0; %Defines starting position of manuvere

r2 = r\_a1; %Defines the ending position of manuver

%A=sqrt((r1\*r2))\*sind(delta\_new0)/(sqrt(1-cosd(delta\_new0)));

DM =1;

A=DM\*sqrt(r1\*r2\*(1+cosd(delta\_newshort)));

Z = 0;

C = 1/2-Z/factorial(4)+Z^2/factorial(6)-...

Z^3/factorial(8)+Z^4/factorial(10)-Z^5/factorial(12);

S = 1/factorial(3)-Z/factorial(5)+Z^2/factorial(7)-...

Z^3/factorial(9)+Z^4/factorial(11)-Z^5/factorial(13);

y = r1+r2-(A\*(1-Z\*S)/(sqrt(C)));

x = sqrt(y/C);

t = ((x^3)\*S+A\*sqrt(y))/sqrt(mu);

%1352

counter = 1;

while (abs(t-delta\_t)>(10^-4))

CP = (1/factorial(4))+ ((2\*Z)/factorial(6))...

-(3\*(Z^2)/factorial(8))+((4\*Z^3)/factorial(10));

SP = (1/factorial(5))+ ((2\*Z)/factorial(7))...

-(3\*(Z^2)/factorial(9))+((4\*Z^3)/factorial(11));

dtdz = (x^3)\*(SP-(3\*S\*CP/(2\*C)))+(A/8)\*(((3\*S\*sqrt(y))/C)+(A/x)); %Newton's Short Method to find next iteration of Z

Z= Z+(delta\_t-t)/dtdz;

%fprintf('The Z through iteration %g is %.13f\n',counter,t)

C = 1/2-Z/factorial(4)+Z^2/factorial(6)-...

Z^3/factorial(8)+Z^4/factorial(10)-Z^5/factorial(12);

% fprintf('The C through iteration %g is %.13f\n',counter,C)

S = 1/factorial(3)-Z/factorial(5)+Z^2/factorial(7)-...

Z^3/factorial(9)+Z^4/factorial(11)-Z^5/factorial(13);

%fprintf('The S through iteration %g is %.13f\n',counter,S)

y = r1+r2-(A\*(1-Z\*S)/(sqrt(C)));

x = sqrt(y/C);

t = ((x^3)\*S+A\*sqrt(y))/sqrt(mu);

% fprintf('The time through iteration %g is %.13f\n',counter,t)

counter = counter+1;

end

r1 = [0,r\_b0,0];

r2 = [-r\_a1,0,0];

f = 1- y/norm(r1);

g = A\*sqrt(y/mu);

gdot = 1- y/norm(r2);

v\_1 = (r2-f\*r1)/g;

disp(norm(v\_1))

v\_2 = (gdot\*r2-r1)/g;

disp(norm(v\_2))

%v\_b0 Speed of s/c,1 at Initial

%v\_fr Speed required to be in Orbit 1 at new= 180 degrees

%disp(v\_b0)

%disp(v\_fr)

v\_b0\_vector = (sqrt(mu/p1))\*[-(sind(90)),(e1+cosd(90)),0];

v\_fr\_vector = (sqrt(mu/p1))\*[-(sind(180)),(e1+cosd(180)),0];

delta\_v1 = v\_1-v\_b0\_vector;

delta\_v2 = v\_2-v\_fr\_vector;

delta\_v1\_mag = norm(delta\_v1);

delta\_v2\_mag = norm(delta\_v2);

sumdeltas = delta\_v1\_mag +delta\_v2\_mag;

%using e1,mu,p1,

%Finding the values for Orbit 2:

Hm\_2 = cross(r1,v\_1);

e\_vec = (1/mu)\*(((norm(v\_1)^2)-(mu/norm(r1)))\*r1-(dot(r1,v\_1)\*v\_1));

e\_mag = norm(e\_vec);

p = (norm(Hm\_2)^2)/mu;

Em\_2 = ((norm(v\_1)^2)/2)-(mu/norm(r1));

a\_2 = -(mu)/(2\*Em\_2)

fprintf('a.\n')

fprintf('The semi-major axis of orbit 2 is: %.13f[km]\n',a\_2)

fprintf('b.\n')

fprintf('The eccentricity vector e of orbit 2 is: [%.13f%.13f,%.13f],[P,Q,W]\n',e\_vec(1),e\_vec(2),e\_vec(3))

fprintf('c.\n')

fprintf('The delta\_V needed at point B is: [%.13f%.13f,%.13f][km/s]\n',delta\_v1(1),delta\_v1(2),delta\_v1(3))

fprintf('d.\n')

fprintf('The delta\_V needed at point A is: [%.13f%.13f,%.13f][km/s]\n',delta\_v2(1),delta\_v2(2),delta\_v2(3))

fprintf('e.\n')

fprintf('The total delta\_V needed to complete both maneuvers: %.13f[km/s]\n',sumdeltas)

## Code for Problem 2/3

%Saimanoj Siddula

%AerE 451 Astrodynamics II

%Midterm Problem 2/3

clear,clc

close all

%times = [11+(20/60),11+(35/60),12]; %hours

%alpha = [-54.1249012,-19.1381645,98.7739537]; %degrees

%delta = [-26.6156514,3.9172775,31.1314513]; %degrees

times = [11+(20/60),11+(30/60),11+(40/60)]; %hours

alpha = [-54.1249012,-33.0588410,-00.4172870]; %degrees

delta = [-22.6156514,-7.2056382,17.4626616]; %degrees

Y= 2007;

M=08;

D=20;

YT = 2000;

%Enter East Longitude angle in degrees

%EL = -110;

EL = 250;

%Finding the Julian Day Given UT---------------------------

J01 = 367\*Y;

J02 = (Y+floor((M+9)/12));

J03 = floor(7/4\*J02);

J04 = floor((275\*M)/9);

J05 = D + 1721013.5;

J0 = J01-J03+J04+J05;

%Julian Date at year 2000

J01T = 367\*YT;

J02T = (YT+floor((1+9)/12));

J03T = floor(7/4\*J02T);

J04T = floor((275\*1)/9);

J05T = 1 + 1721014;

%J0 is Julian Date at UT = 0

JT = J01T-J03T+J04T+J05T;

J01D = 367\*Y;

J02D = (Y+floor((1+9)/12));

J03D = floor(7/4\*J02D);

J04D = floor((275\*1)/9);

J05D = 1 + 1721013.5;

%J0 is the Julian Date at UT =0

J = J01D-J03D+J04D+J05D;

fprintf('The Julian Date at UT is: %.13f\n',J)

%To find Julian Date values after 2000

T0 = (J0-JT)/36525;

DAY = J0-J

%Finding Greenwhich Sidereal Time

G0 = 100.4606184+36000.77004\*T0+.000387933\*(T0^2)-(2.583\*10^-8)\*(T0^3);

G01 = G0;

n = 1;

while n==1 %Resetting answer to angle between 360 and 0.

if G0<0

G0 = G0+360;

elseif G0>360

G0 = G0-360;

elseif 0<=G0<=360

n=0;

end

end

GD = G0 + 360.98564724.\*times./24

GD1 = GD;

n=1;

while n==1

if GD<0

GD = GD+360;

elseif GD >360

GD = GD-360;

elseif 0<=GD<=360

n=0;

end

end

GR = GD.\*pi/180;

%finds local sidereal time

LSTD = GD+EL;

fprintf('The Local Sidereal Day is %.13f\n',LSTD)

LSTR = LSTD.\*pi/180;

%Defines Constants for the problem

L = 40;

ae = 6378.1;

ee = .08182;

mu = 3.986e5;

t = (times-(11+(20/60)))\*60\*60; %time in seconds

tau\_1 = t(1)-t(2);

tau\_3 = t(3)-t(2);

tau = tau\_3 - tau\_1;

theta = LSTD;

H = 2.0;

X\_val = (ae/(sqrt(1-ee^2\*sind(L)^2))+H)\*cosd(L);

Z = ((ae\*(1-ee^2))/(sqrt(1-ee^2\*sind(L)^2))+H)\*sind(L);

L\_1 = [cosd(alpha(1))\*cosd(delta(1));sind(alpha(1))\*cosd(delta(1));sind(delta(1))];

L\_2 = [cosd(alpha(2))\*cosd(delta(2));sind(alpha(2))\*cosd(delta(2));sind(delta(2))];

L\_3 = [cosd(alpha(3))\*cosd(delta(3));sind(alpha(3))\*cosd(delta(3));sind(delta(3))];

fprintf('The L(slant direction) values for time 1 is: [%13f,%13f,%13f][km] \n',L\_1)

fprintf('The L(slant direction) values for time 1 is: [%13f,%13f,%13f][km] \n',L\_2)

fprintf('The L(slant direction) values for time 1 is: [%13f,%13f,%13f][km] \n',L\_3)

R\_1 = [X\_val\*cosd(theta(1));X\_val\*sind(theta(1));Z];

R\_2 = [X\_val\*cosd(theta(2));X\_val\*sind(theta(2));Z];

R\_3 = [X\_val\*cosd(theta(3));X\_val\*sind(theta(3));Z];

fprintf('The R values for time 1 is: [%13f;%13f;%13f][km] \n',R\_1)

fprintf('The R values for time 2 is: [%13f;%13f;%13f][km] \n',R\_2)

fprintf('The R values for time 3 is: [%13f;%13f;%13f][km] \n',R\_3)

D\_0 = dot(L\_1,(cross(L\_2,L\_3))); %tripple scalar product (rho)

D\_11 = dot(R\_1, (cross(L\_2, L\_3)));

D\_21 = dot(R\_2, (cross(L\_2, L\_3)));

D\_31 = dot(R\_3, (cross(L\_2, L\_3)));

D\_12 = dot(R\_1,(cross(L\_1, L\_3)));

D\_22 = dot(R\_2,(cross(L\_1, L\_3)));

D\_32 = dot(R\_3,(cross(L\_1, L\_3)));

D\_13 = dot(R\_1,(cross(L\_1, L\_2)));

D\_23 = dot(R\_2,(cross(L\_1, L\_2)));

D\_33 = dot(R\_3,(cross(L\_1, L\_2)));

A = (1/D\_0)\*(-D\_12\*(tau\_3/tau)+D\_22+D\_32\*(tau\_1/tau));

B = (1/(6\*D\_0))\*(D\_12\*(tau\_3^2-tau^2)\*(tau\_3/tau)+D\_32\*(tau^2-tau\_1^2)\*(tau\_1/tau));

E = dot(R\_2, L\_2);

fprintf('The A value is %.13f\n',A)

fprintf('The B value is %.13f\n',B)

fprintf('The E value is %.13f\n',E)

a = -(A^2+2\*A\*E+norm(R\_2)^2);

b = -2\*mu\*B\*(A+E);

c = -mu^2\*B^2;

fprintf('The a value is %.13f\n',a)

fprintf('The b value is %.13f\n',b)

fprintf('The c value is %.13f\n',c)

syms r

r\_2vals = double(solve(r^8+a\*r^6+b\*r^3+c==0,r))

disp(length(r\_2vals))

for i = 1:length(r\_2vals)

if real(r\_2vals(i))>0

if imag(r\_2vals(i))==0

r\_2 = r\_2vals(i);

end

end

end

disp(r\_2)

rho\_2 = A+ (mu/(r\_2^3))\*B;

r\_2\_vector = R\_2 +rho\_2\*L\_2;

P = (6\*(D\_31\*(tau\_1/tau\_3)+D\_21\*(tau/tau\_3))\*r\_2^3)+(mu\*D\_31\*(tau^2-tau\_1^2)\*(tau\_1/tau\_3));

Q = (6\*(D\_13\*(tau\_3/tau\_1)-D\_23\*(tau/tau\_1))\*r\_2^3)+(mu\*D\_13\*(tau^2-tau\_3^2)\*(tau\_3/tau\_1));

R\_1 = (6\*(r\_2^3))+(mu\*(tau^2-tau\_3^2));

R\_3 = (6\*(r\_2^3))+(mu\*(tau^2-tau\_1^2));

rho\_1 = (1/D\_0)\*((P/R\_1)-D\_11)

rho\_3 = (1/D\_0)\*((Q/R\_3)-D\_33)

r\_1\_vector = R\_1 +rho\_1\*L\_1;

r\_3\_vector = R\_3 +rho\_3\*L\_3;

r\_1\_mag = norm(r\_1\_vector);

r\_2\_mag = norm(r\_2\_vector);

r\_3\_mag = norm(r\_3\_vector);

N = r\_1\_mag\*(cross(r\_2\_vector,r\_3\_vector))+r\_2\_mag\*(cross(r\_3\_vector,r\_1\_vector))+r\_3\_mag\*(cross(r\_1\_vector,r\_2\_vector));

magN = norm(N);

D = cross(r\_1\_vector,r\_2\_vector)+cross(r\_2\_vector,r\_3\_vector)+cross(r\_3\_vector,r\_1\_vector);

magD = norm(D);

S = r\_1\_vector\*(r\_2\_mag-r\_3\_mag)+r\_2\_vector\*(r\_3\_mag-r\_1\_mag)+r\_3\_vector\*(r\_1\_mag-r\_2\_mag);

v\_2\_vector = sqrt(mu/(magN\*magD))\*((cross(D,r\_2\_vector))/r\_2\_mag+S);

v\_2\_mag = norm(v\_2\_vector);

fprintf('The P value is: %13f \n',P)

fprintf('The Q value is: %13f \n',Q)

fprintf('The N value is: %13f \n',N)

fprintf('The D value is: %13f \n',D)

fprintf('The r\_1\_vector value is: [%13f,%13f,%13f][km] \n',r\_1\_vector)

fprintf('The r\_3\_vector value is: [%13f,%13f,%13f][km] \n',r\_3\_vector)

fprintf('The v\_2\_vector value is: [%13f,%13f,%13f][km] \n',v\_2\_vector)

%Start of Problem 3

%Stores initial values in the vectors

mag\_r1(1) = r\_1\_mag;

mag\_r2(1) = r\_2\_mag;

mag\_r3(1) = r\_3\_mag;

mag\_v2(1) = v\_2\_mag;

mag\_rho1(1) = rho\_1;

mag\_rho2(1) = rho\_2;

mag\_rho3(1) = rho\_3;

vect\_r1(1,:) = r\_1\_vector;

vect\_r2(1,:) = r\_2\_vector;

vect\_r3(1,:) = r\_3\_vector;

vect\_v2(1,:) = v\_2\_vector

error1 = 1;

error2 = 1;

error3 = 1;

i = 1;

errormin = 10^3;

syms X C S

f\_1(1) = 1-((mu/(2\*r\_2\_mag^3))\*tau\_1^2);

f\_3(1) = 1-(mu/(2\*r\_2\_mag^3))\*tau\_3^2;

g\_1(1) = tau\_1 -(mu/(6\*r\_2\_mag^3))\*tau\_1^3;

g\_3(1) = tau\_3 -(mu/(6\*r\_2\_mag^3))\*tau\_3^3;

%loop to converge the values of rho to createa accurate values

while error1>errormin && error2>errormin && error3>errormin

a\_invert(i) = (2/mag\_r2(i))-((mag\_v2(i)^2)/mu);

mag\_v2\_rad(i) = dot(vect\_v2(i,:),vect\_r2(i,:))/mag\_r2(i);

C = 1/(factorial(2))- (a\_invert(i)\*X^2)/factorial(4) +(a\_invert(i)\*X^2)^2/factorial(6)...

-(a\_invert(i)\*X^2)^3/factorial(8) + (a\_invert(i)\*X^2)^4/factorial(10) - (a\_invert(i)\*X^2)^5/factorial(12)...

+(a\_invert(i)\*X^2)^6/factorial(14);

S = 1/(factorial(3))- (a\_invert(i)\*X^2)/factorial(5) +(a\_invert(i)\*X^2)^2/factorial(7)...

-(a\_invert(i)\*X^2)^3/factorial(9) + (a\_invert(i)\*X^2)^4/factorial(11) - (a\_invert(i)\*X^2)^5/factorial(13)...

+(a\_invert(i)\*X^2)^6/factorial(15);

%solves for x1 and x3

X1\_val(i,:) = double(solve(sqrt(mu)\*tau\_1 == ((mag\_r2(i)\*mag\_v2\_rad(i))/sqrt(mu))\*X^2\*C+(1-a\_invert(i)\*mag\_r2(i))\*X^3\*S+mag\_r2(i)\*X,X));

X3\_val(i,:) = double(solve(sqrt(mu)\*tau\_3 == ((mag\_r2(i)\*mag\_v2\_rad(i))/sqrt(mu))\*X^2\*C+(1-a\_invert(i)\*mag\_r2(i))\*X^3\*S+mag\_r2(i)\*X,X));

%loops to make sure the x values are real and close to 0

k = 1;

for j = 1:length(X1\_val(i,:))

if X1\_val(i,j) == real(X1\_val(i,j))

X\_1\_val(i,k) = X1\_val(i,j);

k = k +1;

end

end

minabs = 1000000000000;

for l = 1:length(X\_1\_val(i,:))

minabs1 = abs(X\_1\_val(i:l));

if minabs1>0

if minabs1<minabs

X\_1(i) = X\_1\_val(i,l);

minabs = minabs1;

end

end

end

k = 1;

for j = 1:length(X3\_val(i,:))

if X3\_val(i,j) == real(X3\_val(i,j))

X\_3\_val(i,k) = X3\_val(i,j);

k = k+1;

end

end

minabs = 1000000000000;

for l = 1:length(X\_3\_val(i,:))

minabs1 = abs(X\_3\_val(i,l));

if minabs1>0

if minabs1<minabs

X\_3(i) = X\_3\_val(i,l);

end

end

end

%Calculates the new values for the new x1 and x2 values

C\_l(i) = 1/(factorial(2))- (a\_invert(i)\*X\_1(i)^2)/factorial(4) +(a\_invert(i)\*X\_1(i)^2)^2/factorial(6)

-(a\_invert(i)\*X\_1(i)^2)^3/factorial(8) + (a\_invert(i)\*X\_1(i)^2)^4/factorial(10) - (a\_invert(i)\*X\_1(i)^2)^5/factorial(12)

+(a\_invert(i)\*X\_1(i)^2)^6/factorial(14)

S\_l(i) = 1/(factorial(3))- (a\_invert(i)\*X\_1(i)^2)/factorial(5) +(a\_invert(i)\*X\_1(i)^2)^2/factorial(7)

-(a\_invert(i)\*X\_1(i)^2)^3/factorial(9) + (a\_invert(i)\*X\_1(i)^2)^4/factorial(11) - (a\_invert(i)\*X\_1(i)^2)^5/factorial(13)

+(a\_invert(i)\*X\_1(i)^2)^6/factorial(15)

C\_3(i) = 1/(factorial(2))- (a\_invert(i)\*X\_3(i)^2)/factorial(4) +(a\_invert(i)\*X\_3(i)^2)^2/factorial(6)

-(a\_invert(i)\*X\_3(i)^2)^3/factorial(8) + (a\_invert(i)\*X\_3(i)^2)^4/factorial(10) - (a\_invert(i)\*X\_3(i)^2)^5/factorial(12)

+(a\_invert(i)\*X\_3(i)^2)^6/factorial(14)

S\_3(i) = 1/(factorial(3))- (a\_invert(i)\*X\_3(i)^2)/factorial(5) +(a\_invert(i)\*X\_3(i)^2)^2/factorial(7)

-(a\_invert(i)\*X\_3(i)^2)^3/factorial(9) + (a\_invert(i)\*X\_3(i)^2)^4/factorial(11) - (a\_invert(i)\*X\_3(i)^2)^5/factorial(13)

+(a\_invert(i)\*X\_3(i)^2)^6/factorial(15)

f\_1(i+1) = 1-((X\_1(i)^2)/mag\_r2(i))\*C\_l(i);

g\_1(i+1) = tau\_1-(1/sqrt(mu))\*X\_1(i)^3\*S\_1(i);

f\_3(i+1) = 1-((X\_3(i)^2)/mag\_r2(i))\*C\_3(i);

g\_3(i+1) = tau\_3-(1/sqrt(mu))\*X\_3(i)^3\*S\_3(i);

f\_1\_bar(i) = (f\_1(i)+f\_1(i+1))/2;

f\_3\_bar(i) = (f\_3(i)+f\_3(i+1))/2;

g\_1\_bar(i) = (g\_1(i)+g\_1(i+1))/2;

g\_3\_bar(i) = (g\_3(i)+g\_3(i+1))/2;

c1(i) = (g\_3\_bar(i))/(f\_1\_bar(i)\*g\_3\_bar(i)-f\_3\_bar(i)\*g\_1\_bar(i));

c3(i) = (g\_1\_bar(i))/(f\_1\_bar(i)\*g\_3\_bar(i)-f\_3\_bar(i)\*g\_1\_bar(i));

mag\_rho1(i+1) = (1/D\_0)\*(-D\_11+(1/cl(i))\*D\_21-(c3(i)/c1(i))\*D\_31);

mag\_rho2(i+1) = (1/D\_0)\*(-c1(i)\*D\_12+D\_22-(c3(i))\*D\_32);

mag\_rho3(i+1) = (1/D\_0)\*(-(c1(i)/c3(i))\*D\_13+(1/c3(i))\*D\_23-D\_33);

vect\_r1(i+1,:) = R\_1 +mag\_rho1(i+1)\*L\_1;

vect\_r2(i+1,:) = R\_2 +mag\_rho2(i+1)\*L\_2;

vect\_r3(i+1,:) = R\_3 +mag\_rho3(i+1)\*L\_3;

mag\_r1(i+1) = nomr(vect\_r1(i+1,:));

mag\_r2(i+1) = nomr(vect\_r2(i+1,:));

mag\_r3(i+1) = nomr(vect\_r3(i+1,:));

vect\_v2(i+1,:) = (f\_1(i+1).\*vect\_r3(i+1,:)-f\_3(i+1).\*vect\_r1(i+1,:))./(f\_1(i+1)\*g\_3(i+1)-f\_3(i+1)\*g\_1(i+1));

mag\_v2(i+1) = norm(vect\_v2(i+1,:));

error1 = abs(mag\_rho1(i)-mag\_rho1(i+1));

error2 = abs(mag\_rho2(i)-mag\_rho2(i+1));

error3 = abs(mag\_rho3(i)-mag\_rho3(i+1));

i = i+1;

disp(i)

if i>150

error1 = 10^-9;

error2 = 10^-9;

error3 = 10^-9;

end

end

K = [0;0;1];

Hm = cross(vect\_r2(i,:),vect\_v2(i,:));

magHm = norm(Hm);

n = cross(K,Hm);

magn = norm(n);

evector = (1/mu)\*(((mag\_v2(i)^2-(mu/mag\_r2(i))).\*vect\_r2(i,:))-((dot(vect\_r2(i,:),vect\_v2(i,:))).\*vect\_v2(i,:)));

mage = norm(evector);

p\_final = (magHm^2)/mu;

a\_final = p\_final/(1-mage^2);

fprintf('The a of the final orbit is: %.13f \n',a\_final)

r\_p = p\_final/(1+mage);

r\_a = p\_final/(1-mage);

hk = Hm(3);

ni = n(1);

nj = n(2);

ek = evector(3);

rdv = dot(vect\_r2(i,:),vect\_v2(i,:));

inclin = acosd(hk/magHm);

if nj>0

Big\_Omega = acosd(ni/magn);

elseif nj<0

Big\_Omega = acosd(ni/mage)+180;

else

Big\_Omega = undefined;

end

fprintf('The longitude of assending node = %13f \n',Big\_Omega)

if ek>0

Little\_Omega = acosd(dot(n,evector)/(magn\*mage));

elseif ek<0

Little\_Omega = acosd(dot(n,evector)/(magn\*mage))+180;

else

Little\_Omega = 0;

end

fprintf('The argument of periapse = %13f \n',Little\_Omega)

if rdv>0

nu = acosd(dot(evector,vect\_r2(i,:))/(mage\*mag\_r2(i)));

elseif rdv<0

nu = acosd(dot(evector,vect\_r2(i,:))/(mage\*mag\_r2(i)))+180;

else

nu =0;

end

fprintf('The nu = %13f \n',nu)

disp('Initial Values')

disp(' ')

fprintf('X value is %.8f km\n',X\_val)

fprintf('Z value is %.8f km\n',Z)

disp(' ')

fprintf('Unit vector of rho\_1 is [%.8f, %.8f, %.8f]\n',L\_1)

fprintf('Unit vector of rho\_2 is [%.8f, %.8f, %.8f]\n',L\_2)

fprintf('Unit vector of rho\_3 is [%.8f, %.8f, %.8f]\n',L\_3)

disp('')

fprintf('D\_0 value is %.8f\n',D\_0)

fprintf('D\_11 value is %.8f km\n',D\_0)

fprintf('D\_21 value is %.8f km\n',D\_21)

fprintf('D\_31 value is %.8f km\n',D\_31)

fprintf('D\_12 value is %.8f km\n',D\_12)

fprintf('D\_22 value is %.8f km\n',D\_22)

fprintf('D\_32 value is %.8f km\n',D\_32)

fprintf('D\_13 value is %.8f km\n',D\_13)

fprintf('D\_23 value is %.8f km\n',D\_23)

fprintf('D\_33 value is %.8f km\n',D\_33)

disp(' ')

fprintf('A value is %.8f km\n',A)

fprintf('B value is %.8f km\n',B)

fprintf('E value is %.8f km\n',E)

disp(' ')

fprintf('a value is %.8f km^2\n',a)

fprintf('b value is %.8f km^5\n',b)

fprintf('e value is %.8f km^8\n',c)

disp(' ')

fprintf('P value is %.8f km^4\n',P(1))

fprintf('Q value is %.8f km^4\n',Q(1))

disp(' ')

fprintf('R\_1 value is %.8f km^3\n',R\_1(1))

fprintf('R\_3 value is %.8f km^3\n',R\_3(1))

for j = 1:i-1

fprintf('Iteration %g\n',j)

disp(' ')

fprintf('X\_1 value %.8 km^4\n',X1\_val(j,:))

fprintf('X\_3 value %.8 km^4\n',X3\_val(j,:))

disp(' ')

fprintf('f1 value is %.8f\n',f\_1(j))

fprintf('f3 value is %.8f\n',f\_3(j))

fprintf('g1 value is %.8f\n',g\_1(j))

fprintf('g3 value is %.8f\n',g\_3(j))

disp(' ')

fprintf('f1\_bar value is %.8f\n',f\_1\_bar(j))

fprintf('f3\_bar value is %.8f\n',f\_3\_bar(j))

fprintf('g1\_bar value is %.8f\n',g\_1\_bar(j))

fprintf('g3\_bar value is %.8f\n',g\_3\_bar(j))

disp(' ')

fprintf('c1 value is %.8\n',c1(j))

fprintf('c3 value is %.8\n',c3(j))

disp(' ')

fprintf('rho\_1 value is %.8f km\n',mag\_rho1(j))

fprintf('rho\_2 value is %.8f km\n',mag\_rho2(j))

fprintf('rho\_3 value is %.8f km\n',mag\_rho3(j))

disp(' ')

fprintf('r\_1 vector is [%.8f,%.8f,%.8f] [km]\n',mag\_rho1(j))

fprintf('r\_2 vector is [%.8f,%.8f,%.8f] [km]\n',mag\_rho1(j))

fprintf('r\_3 vector is [%.8f,%.8f,%.8f] [km]\n',mag\_rho1(j))

disp(' ')

fprintf('r\_1 value is %.8f km\n',mag\_r1(j))

fprintf('r\_2 value is %.8f km\n',mag\_r2(j))

fprintf('r\_3 value is %.8f km\n',mag\_r3(j))

disp(' ')

fprintf('v\_2 vector is [%.8f,%.8f,%.8f] [km/s]\n',vect\_v2(j,:))

fprintf('v\_2 value is %.8f km/s\n',mag\_v2(j))

disp(' ')

end

## Code for Problem 4

%Saimanoj Siddula

%AerE 451 Midterm

%Problem 4

clear,clc

close all

%Given Values---------------------

r0 = [1131.34,-2282.343,6672.423]; %Initial Radius in Km

v0 = [-6.643051,4.3033,2.42879]; %Initial Velocity in Km

%r0 = [10000,0,0]; %Initial Radius in Km

%v0 = [0,9.2,0]; %Initial Velocity in Km

r0\_mag = norm(r0); %km

v0\_mag = norm(v0); %km/s

%Defining the Gravitational Parameter of Earth

mu = 3.986012\*10^5; %km^3/sec^2

%Solving for a using a modified Vis-Viva Equation.

a= mu/((2\*mu/r0\_mag)-v0\_mag^2); %km

%---------------------------------

%Setting the Dt parameter as the time after the initial observatin

%t = 3\*60\*60;

t = 40\*60;

dt = t;

%Starts the initial error value at 1 to insure that the first iteration

%process begins

error = 1;

%Variable i is to count the number of iterations the program preforms and

%to use a s a storgate nu,ber of arrays

i=1;

%The first X and Z values are calculated and kept in the 1 cell of their

%respective arrays.

X(i) = (sqrt(mu)\*t)/abs(a);

Z(i) = (X(i)^2)/a;

%Checked

counter = 1;

%Starting the process of finding a more accurate X and Z values by checking

%whether the error is with in the tolerance range.

while (error > 10^-8)

C(i) = 1/2-Z(i)/factorial(4)+(Z(i)^2)/factorial(6)-(Z(i)^3)/factorial(8)+(Z(i)^4)/factorial(10)-(Z(i)^5)/factorial(12);

S(i) = 1/factorial(3)-Z(i)/factorial(5)+(Z(i)^2)/factorial(7)-(Z(i)^3)/factorial(9)+(Z(i)^4)/factorial(11)-(Z(i)^5)/factorial(13);

F(i) = (1-(r0\_mag/a))\*S(i)\*X(i)^3+ (dot(r0,v0)/(sqrt(mu)))\*C(i)\*X(i)^2 +r0\_mag\*X(i)-sqrt(mu)\*t;

F\_p(i) = C(i)\*X(i)^2+ (dot(r0,v0)/(sqrt(mu)))\*(1-S(i)\*Z(i))\*X(i)+r0\_mag\*(1-C(i)\*Z(i));

F\_pp(i) = (1-r0\_mag/a)\*(1-S(i)\*Z(i))\*X(i)+ (dot(r0,v0)/(sqrt(mu)))\*(1-C(i)\*Z(i));

delta(i) = 2\*(4\*F\_p(i)^2-5\*F(i)\*F\_pp(i))^.5;

dx(i) = 5\*F(i)/(F\_p(i)+sign(F\_p(i))\*delta(i));

X(i+1) = X(i)-dx(i);

Z(i+1) = (X(i+1)^2)/a;

error = abs(dx(i)^2/a);

i = i+1;

counter = counter+1;

end

C(i) = 1/2-Z(i)/factorial(4)+(Z(i)^2)/factorial(6)-(Z(i)^3)/factorial(8)+(Z(i)^4)/factorial(10)-(Z(i)^5)/factorial(12);

S(i) = 1/factorial(3)-Z(i)/factorial(5)+(Z(i)^2)/factorial(7)-(Z(i)^3)/factorial(9)+(Z(i)^4)/factorial(11)-(Z(i)^5)/factorial(13);

f = 1-((X(i)^2)/r0\_mag)\*C(i);

g = t-1/sqrt(mu)\*(X(i)^3)\*S(i);

fprintf('The f value at iteration is: %.13f\n',f)

fprintf('The g value at iteration is: %.13f\n',g)

r = f\*r0 + g\*v0;

r\_mag = norm(r);

f\_dot = (sqrt(mu)/(r\_mag\*r0\_mag))\*(S(i)\*Z(i)-1)\*X(i);

g\_dot = 1-((X(i)^2)/r\_mag)\*C(i);

fprintf('The fdot value at iteration is: %.13f /n',f\_dot)

fprintf('The gdot value at iteration is: %.13f /n',g\_dot)

v = f\_dot\*r0 +g\_dot\*v0

v\_mag = norm(v);

fprintf('The V Vector is [%.13f,%.13f,%.13f][km/s] \n',v)

e\_v = (1/mu).\*((((v\_mag^2)-(mu/r\_mag)).\*r)-(dot(r,v).\*v));

e\_mag = norm(e\_v);

nu = acosd((dot(e\_v,r))/(e\_mag\*r\_mag))

Hm = cross(r,v);

p = norm(Hm)^2/mu;

Em = ((norm(v)^2)/2)-(mu/norm(r))

a = -(mu)/(2\*Em)

e = sqrt(1-(p/a))

%Display the solutions to the problem

fprintf('X\_%g = %.13f\n',1,X(1))

fprintf('Z\_%g = %.13f\n',1,Z(1))

disp(' ')

for i=1:length(C)-1

fprintf('Iteration: %g\n',i)

fprintf('C\_%g = %.13f\n',i,C(i))

fprintf('S\_%g = %.13f\n',i,S(i))

fprintf('F\_%g = %.13f\n',i,F(i))

fprintf('Fp\_%g = %.13f\n',i,F\_p(i))

fprintf('Fpp\_%g = %.13f\n',i,F\_pp(i))

fprintf('Delta\_%g = %.13f\n',i,delta(i))

fprintf('dx\_%g = %.13f\n',i,dx(i))

%fprintf('error\_%g = %.13f\n',i,error(i))

fprintf('X\_%g = %.13f\n',i,X(i+1))

fprintf('Z\_%g = %.13f\n',i,Z(i+1))

disp(' ')

end

fprintf('f = %0.13f\n',f)

fprintf('g = %0.13f\n',g)

fprintf('R = %.8f I +%.8f J +%.8f K km\n',r)

disp(' ')

fprintf('Magnitude of R is %.8f km\n',r\_mag)

disp(' ')

fprintf('f\_dot = %.13f/n',f\_dot)

fprintf('g\_dot = %.13f/n',g\_dot)

disp(' ')

fprintf('V = %.8f I +%.8f J +%.8f K km/s\n',v)

fprintf('Magnitude of V is %.8f km/s\n',v\_mag)

disp(' ')

fprintf('eccentricity of the orbit is [%.8f,%.8f,%.8f]\n',e\_v)

disp(' ')

fprintf('The angle of nu is %8f [deg]\n',nu)