Space Flight Mechanics: Assignment-1

A Report submitted

For internal assessment for Coarse

Space Flight Mechanics

in

Aerospace Engineering

by

Shingala Vaidik Pareshbhai

(SC21B054)

pursued in

Department of Aerospace Engineering

Indian Institute of Space Science and Technology

to

INDIAN INSTITUTE OF SPACE SCIENCE AND TECHNOLOGY

THIRUVANANTHAPURAM

October 15, 2023

Contents

Chapter 1

ABSTRACT

Outlined in Curtis' "Orbital Mechanics for Engineering Students" third edition, is the Universal Variable approach and principles of orbital mechanics. The objective of this task is to use Algorithm 3.4 from the book to establish the velocity and position vectors of a space object, given the basic conditions, two hours ahead.

Following Algorithm 3.4, we carefully calculate the space object's final velocity (v) and position (r), utilizing the initial conditions provided (vo and ro), which are relative to an Earth-centered inertial frame of reference. We employ the Universal Variable approach throughout the procedure.

Introducing a unique set of initial conditions for position and velocity in Part (ii) of the task. We then apply the Universal Variable approach to determine the vectors of position and velocity at a point in time. Furthermore, this approach is utilized to determine the future location and speed of the space object.

Chapter 2

Problem 1 : Initial Condition Given

2.1 Problem

At a given instant, a space object has the following position and velocity vectors relative to an earth centered inertial frame of reference (i, j, k):

ro= 20,000i- 105000j- 19,000k (km)

vo= 0.9000i-3.4000j-1.5000k (km/s)

Refer Exercise Problem 3.20 in Curtis Use Algorithm 3.4 (Curtis 3rd edition) to find r and v, 2 hours later. Show the detailed steps involved and the matlab code used to arrive at the answer

2.2 The detailed steps involved

Step 1: Given Data

- We are provided with the initial position vector ($\mathbf{r}_0$) and initial velocity vector ($\mathbf{v}_0$) of a space object in an Earth-centered inertial frame.
- $\mathbf{r}_0 = 20,000\hat{i} - 105,000\hat{j} - 19,000\hat{k}$ (in kilometers)
- $\mathbf{v}_0 = 0.9000\hat{i} - 3.4000\hat{j} - 1.5000\hat{k}$ (in kilometers per second)

Step 2: Calculate Initial Values

- Calculate the magnitudes of the initial position and velocity vectors (r_0 and v_0) using the Pythagorean theorem.

Step 3: Find Radial Velocity (v_{r0})

- Calculate the component of velocity along the radial direction (v_{r0}) by taking the dot product of $\mathbf{r}_0$ and $\mathbf{v}_0$, divided by the magnitude of r_0.

Step 4: Calculate Parameter α

- Calculate the parameter α, which characterizes the type of orbit (hyperbolic, parabolic, or elliptical).
- $\alpha = \frac{2}{r_0} - \frac{v_0^2}{\mu}$, where μ is the gravitational parameter.

Step 5: Determine Orbit Type

- Based on the value of α, determine the type of orbit: hyperbolic ($\alpha < 0$), parabolic ($\alpha = 0$), or elliptical ($\alpha > 0$).

Step 6: Iterative Process for χ

- Initiate an iterative process to find the Universal Variable χ.
- Begin with an initial estimate $\chi_0 = \sqrt{\mu|\alpha|\Delta t}$.

Step 7: Stumpff Functions (C and S)

- Calculate the Stumpff functions $C(z)$ and $S(z)$, which are related to circular and hyperbolic trig functions.
- These functions are used to define the parameter $z_i = \alpha\chi_i^2$.

Step 8: Define $f(\chi)$ and $f'(\chi)$

- Define the functions $f(\chi_i)$ and $f'(\chi_i)$ based on the derived values.
- $f(\chi_i) = \frac{r_0 v_{r0}}{\sqrt{\mu}}\chi_i^2 C(z_i) + (1 - \alpha r_0)\chi_i^3 S(z_i) + r_0\chi_i - \sqrt{\mu}\Delta t$
- $f'(\chi_i) = \frac{r_0 v_{r0}}{\sqrt{\mu}}\chi_i[1 - \alpha\chi_i^2 S(z_i)] + (1 - \alpha r_0)\chi_i^2 C(z_i) + r_0$

Step 9: Calculate the Ratio and Iterate

- Calculate the ratio between $f(\chi)$ and $f'(\chi)$.
- If the ratio is above a specified tolerance (e.g., 10^{-8}), iterate to find the next χ using $\chi_{i+1} = \chi_i - ratio_i$.

Step 10: Final χ Value

- Once the ratio is below the tolerance limit, choose the final χ value (χ_{final}).

Step 11: Calculate f and g

- Use the final χ, α, r_0, Δt, and the Stumpff functions to calculate f and g.
- $f = 1 - \frac{\chi_{final}^2 C(z_{final})}{r_0}$
- $g = \Delta t - \frac{\chi_{final}^3 S(z_{final})}{\sqrt{\mu}}$

Step 12: Calculate $\dot{f}$ and $\dot{g}$

- Use the final χ, α, r_0, Δt, and the Stumpff functions to calculate f and g.
- $\dot{f} = \frac{\sqrt{\mu}}{r r_o}[\alpha\chi^3 S(z) - \chi]$
- $\dot{g} = 1 - \frac{\chi^2}{r} C(z)$

Step 13: Calculate Final Position and Velocity

- Use f and g to calculate the final position ($\mathbf{r}$) and velocity ($\mathbf{v}$) vectors of the space object.
- $\mathbf{r} = f\mathbf{r}_0 + g\mathbf{v}_0$
- $\mathbf{v} = \dot{f}\mathbf{r}_0 + \dot{g}\mathbf{v}_0$

2.3 Matlab code

```
% Given data
mu = 398600;                % Gravitational parameter (km^3/s^2)
R0 = [20000, -105000, -19000]; % Initial position vector (km)
V0 = [0.9000, -3.4000, -1.5000]; % Initial velocity vector (km/s)
t = 7200;                   % Elapsed time (seconds)

% Calculate initial values
r0 = norm(R0);              % Initial position magnitude (km)
v0 = norm(V0);              % Initial velocity magnitude (km/s)
vr0 = dot(R0, V0) / r0;     % Initial radial velocity (km/s)
alpha=2/r0-v0^2/mu;
% Calculate the initial eccentricity vector E0
E0 = 1 / mu * ((v0^2 - mu / r0) * R0 - r0 * vr0 * V0);

% Calculate the initial eccentricity e0
e0 = norm(E0);

% Call the function to find the final position and velocity vectors
[R, V] = Find_R_V(R0, V0, t);

% Calculate final values
r = norm(R);                % Final position magnitude (km)
v = norm(V);                % Final velocity magnitude (km/s)
vr = dot(R, V) / r;         % Final radial velocity (km/s)

% Calculate the final eccentricity vector E
E = 1 / mu * ((v^2 - mu / r) * R - r * vr * V);
```

```
28
29 % Calculate the final eccentricity e
30 e = norm(E);
31
32 H0 = cross(R0, V0);          % Initial angular momentum vector
33 H = cross(R, V);             % Final angular momentum vector
34
35 % Display results
36 disp('Initial Data:');
37 fprintf('Initial position vector (km):\n');
38 fprintf('r0 = (%g, %g, %g)\n', R0(1), R0(2), R0(3));
39
40 fprintf('Initial velocity vector (km/s):\n');
41 fprintf('v0 = (%g, %g, %g)\n', V0(1), V0(2), V0(3));
42
43 fprintf('\nElapsed time = %g s\n', t);
44 v=Universal_Variable(t,r0,vr0,alpha);
45 fprintf('\n Universal anomaly (km^0.5) = %g',v);
46
47 disp('Angular Momentum:');
48 fprintf('H0 = (%g, %g, %g)\n', H0(1), H0(2), H0(3));
49
50 disp('Final Data:');
51 fprintf('Final position vector (km):\n');
52 fprintf('r = (%g, %g, %g)\n', R(1), R(2), R(3));
53
54 fprintf('Final velocity vector (km/s):\n');
55 fprintf('v = (%g, %g, %g)\n', V(1), V(2), V(3));
56
57 fprintf('Final angular momentum vector (H):\n');
58 fprintf('H = (%g, %g, %g)\n', H(1), H(2), H(3));
59
60
61
62 function c = stumpC(z)
63     % This function evaluates the Stumpff function C(z) according to Step-7.
64     if z > 0
65         c = (1 - cos(sqrt(z))) / z;
66     elseif z < 0
67         c = (cosh(sqrt(-z)) - 1) / (-z);
68     else
69         c = 1/2;
70     end
71 end
72
73 % This function evaluates the Stumpff function S(z) according to Step-7.
74 function s = stumpS(z)
75     if z > 0
76         s = (sqrt(z) - sin(sqrt(z))) / (sqrt(z))^3;
77     elseif z < 0
78         s = (sinh(sqrt(-z)) - sqrt(-z)) / (sqrt(-z))^3;
79     else
80         s = 1/6;
81     end
82 end
83
84 % This function uses Newton s method to solve the universal Kepler equation for the universal
       anomaly
85 function X = Universal_Variable(t, r0, vr0, alpha)
```

```
    % Universal Variable Equation solver
    mu = 398600;
    error = 1e-8; % error tolerance
    nMax = 1000; % limit on the number of iterations
    X = sqrt(mu) * abs(alpha) * t; % Starting value for x

    % Iterate until convergence occurs within the error tolerance:
    n = 0;
    ratio = 1;
    while abs(ratio) > error && n <= nMax
        n = n + 1;
        z = alpha * X^2;
        C = stumpC(z);
        S = stumpS(z);
        F = r0 * vr0 / sqrt(mu) * X^2 * C + (1 - alpha * r0) * X^3 * S + r0 * X - sqrt(mu) * t
            ;
        dFdx = r0 * vr0 / sqrt(mu) * (1 - alpha * X^2 * S) + (1 - alpha * r0) * X^2 * C + r0;
        ratio = F / dFdx;
        X = X - ratio;
    end
end

% This function calculates the Lagrange f and g coefficients.
function [f, g] = Lagrange_coefficient(X, t, r0, alpha)
    mu=398600;
    z = alpha * X^2;
    C = stumpC(z);
    S = stumpS(z);
    f = 1 - X^2 / r0 * C; %...Equation 3.69a:
    g = t - 1 / sqrt(mu) * X^3 * S; %...Equation 3.69b:
end

% This function calculates the time derivatives of the Lagrange f and g coefficients.
function [fdot, gdot] = Lagrange_coefficient_Dot(X, r, r0, alpha)
    mu=398600;
    z = alpha * X^2;
    fdot = sqrt(mu) / (r * r0) * (z * stumpS(z) - 1) * X; %...Equation 3.69c:
    gdot = 1 - X^2 / r * stumpC(z); %...Equation 3.69d:
end

% This function computes the state vector (R,V) from the initial state vector (R0,V0) and the
    elapsed time.
function [R,V]=Find_R_V(R0,V0,t)
    mu=398600;
    r0=norm(R0); % Magnitudes of R0
    v0=norm(V0); % Magnitudes of v0
    vr0=dot(R0,V0)/r0; % Initial radial velocity
    alpha=2/r0-v0^2/mu; % Reciprocal of the semimajor axis
    X=Universal_Variable(t,r0,vr0,alpha); % Compute the universal anomaly
    [f,g]=Lagrange_coefficient(X,t,r0,alpha); % Compute the f and g functions
    R=f*R0+g*V0; % Compute the final position vector
    r=norm(R); % Compute the magnitude of R
    [fdot,gdot]=Lagrange_coefficient_Dot(X,r,r0,alpha); % Compute the derivatives of f and g
    V=fdot*R0+gdot*V0; % Compute the final velocity
end
```

Listing 2.1: Example MATLAB Code

2.4 Matlab code output

```
Initial Data:
Initial position vector (km):
r0 = (20000, -105000, -19000)
Initial velocity vector (km/s):
v0 = (0.9, -3.4, -1.5)
Elapsed time = 7200 s
 Universal anomaly (km^0.5) = 37.4852
Angular Momentum:
H0 = (92900, 12900, 26500)
Final Data:
Final position vector (km):
r = (26337.8, -128752, -29655.9)
Final velocity vector (km/s):
v = (0.862796, -3.2116, -1.46129)
Final angular momentum vector (H):
H = (92900, 12900, 26500)
```

we can see that **angular momentum vector not changing** from this we can say that **object not changing it's orbit** because for same orbit angular momentum vector remain constant so from that **we can see our result accuracy**.

Chapter 3

Problem 2 : Random Initial Condition

3.1 problem

Use a different initial condition (ro, vo) of your choice and find the 'r' and 'v' at a later time t (could be 1 hr , 2hr or 3hr, you may choose).

3.2 Assumed initial condition

Here I am assuming values of ro and vo randomly and I am taking all three cases 1 hr , 2hr or 3hr.

$\vec{r_o} = 20000\hat{i} - 13000\hat{j} - 700\hat{k}$ in km

$\vec{v_o} = 0.7\hat{i} - 7.3\hat{j} - 1.3\hat{k}$ in km/s

3.3 Result for t = 1 hr

```
Initial Data:
Initial position vector (km):
r0 = (20000, -13000, -7000)
Initial velocity vector (km/s):
v0 = (0.7, -7.3, -1.3)
Elapsed time = 3600 s
 Universal anomaly (km^0.5) = 68.2219
Angular Momentum:
H0 = (-34200, 21100, -136900)
Final Data:
Final position vector (km):
r = (20545.3, -37414.8, -10899.2)
Final velocity vector (km/s):
v = (-0.160718, -6.37065, -0.941739)
Final angular momentum vector (H):
H = (-34200, 21100, -136900)
```

3.4 Result for t = 2 hr

```
Initial Data:
Initial position vector (km):
r0 = (20000, -13000, -7000)
Initial velocity vector (km/s):
v0 = (0.7, -7.3, -1.3)
Elapsed time = 7200 s
 Universal anomaly (km^0.5) = 110.728
Angular Momentum:
H0 = (-34200, 21100, -136900)
Final Data:
Final position vector (km):
r = (19544.9, -59445.2, -14044.8)
Final velocity vector (km/s):
v = (-0.357051, -5.91841, -0.82299)
Final angular momentum vector (H):
H = (-34200, 21100, -136900)
```

3.5 Result for t = 3 hr

```
Initial Data:
Initial position vector (km):
r0 = (20000, -13000, -7000)
Initial velocity vector (km/s):
v0 = (0.7, -7.3, -1.3)
Elapsed time = 10800 s
 Universal anomaly (km^0.5) = 141.591
Angular Momentum:
H0 = (-34200, 21100, -136900)
Final Data:
Final position vector (km):
r = (18116.5, -80261.9, -16896.4)
Final velocity vector (km/s):
v = (-0.426374, -5.66767, -0.767026)
Final angular momentum vector (H):
H = (-34200, 21100, -136900)
```

we can see that **angular momentum vector not changing** from all cases from that we can say that **object not changing it's orbit** because for same orbit angular momentum vector remain constant so from that **we can see our result accuracy**.

3.6 conclusion

In this report, we used the Universal Variable approach to find the future position and velocity of a space object. We followed a step-by-step process to perform these calculations, ensuring accuracy and reliability.

In the first part, we worked with given initial conditions to find where the object would be two hours later. In the second part, we tried different initial conditions and calculated the object's new position and velocity after one hour.

Our calculations are based on sound orbital mechanics principles, making our predictions trustworthy. We also observed that the angular momentum remained constant, indicating the object's orbit stability.

In essence, this report demonstrates our ability to predict a space object's position and velocity accurately using the Universal Variable approach, a crucial skill in orbital mechanics.