Table of Contents


```
%%%%%%%%%%%%%%%%%%%%%%%%%%
% ASEN 3200 Homework O-2
% Author: Caleb Bristol
% Date: 11/03/21
%
%%%%%%%%%%%%%%%%%%%%%%%%%%

clc
clear
close all;
```

Generalized Functions for Orbital Elements

Problem 1

This problem involves calculating various orbital elements based on given initial conditions

Initial Conditions

```
    r = [0 2 0]'; %[DU]
```

```
gamma = acos(2/sqrt(5)); %[radians]
% gamma given as f(f) function of true anomaly
h = sqrt(2); %[DU^2/TU]
hdotk = 0.5;
mu = 1; %[DU^3/TU^2]
```

Solutions

```
% a) Semi-latus rectum
p = h^2 / mu;

% b) inclination

% c) ascending node, argument of periapsis, true anomaly

% d) semi-major axis, eccentricity

% e) draw orbit, label apoapsis, perifocal frame, position
```

Problem 2

This problem converts position and velocity vectors from the ECI frame into the topocentric frame

Initial Conditions

```
r_a = [-1 -1.8 1]';
r_dot_a = [0.3 0.3 0.4]';
r_b = [2.4 -2.4 -2]';
r_dot_b = [0.5 -0.2 0.2]';

lambda_1 = 15; %[deg]
phi_1 = 25; %[deg]
lambda_2 = 65; %[deg]
phi_2 = 42; %[deg]
```

Define DCM Function

```
TE = @(lambda,phi) [-sind(lambda) cosd(lambda) 0; ...
    -sind(phi)*cosd(lambda) -sind(phi)*sind(lambda) cosd(phi); ...
    cosd(phi)*cosd(lambda) cosd(phi)*sind(lambda) sind(phi)];
```

Solve

```
TE_1 = TE(lambda_1,phi_1);
TE_2 = TE(lambda_2,phi_2);

r_a_t = TE_1 * r_a;
r_dot_a_t = TE_1 * r_dot_a;
r_b_t = TE_2 * r_b;
r_dot_b_t = TE_2 * r_dot_b;
```

Display Results

```
fprintf("Problem 2: \n")
fprintf("a) \n")
fprintf("r: \n")
disp(r_a_t)
fprintf("r_dot: \n")
disp(r_dot_a_t)
fprintf("b) \n")
fprintf("r: \n")
disp(r_b_t)
fprintf("r_dot: \n")
disp(r_dot_b_t)
```

```
Problem 2:
a)
r:
   -1.4798
    1.5114
   -0.8750

r_dot:
    0.2121
    0.2072
    0.5020

b)
r:
   -3.1894
   -0.7095
   -2.2009

r_dot:
   -0.5377
    0.1285
    0.1562
```

Problem 3

This problem involves calculating all of the orbital elements given a position and velocity vector

Initial Conditions

```
r_1 = [3 2 1]';
r_dot_1 = [-0.2 0.4 0.4]';
r_2 = [-2.5 -1.7 -2.5]';
r_dot_2 = [0.3 -0.3 0.4]';
```

Solve

See function definition at bottom of page for explanation on calculations in orbelements

```
[h_1,i_1,Omega_1,e_1,omega_1,f_1] = orbelements(r_1,r_dot_1,mu);
[h_2,i_2,Omega_2,e_2,omega_2,f_2] = orbelements(r_2,r_dot_2,mu);

% Convert Angles
i_1 = rad2deg(i_1);
i_2 = rad2deg(i_2);
Omega_1 = rad2deg(Omega_1);
Omega_2 = rad2deg(Omega_2);
omega_1 = rad2deg(omega_1);
omega_2 = rad2deg(omega_2);
f_1 = rad2deg(f_1);
f_2 = rad2deg(f_2);
```

Display Results

```
fprintf("Problem 3: \n")
fprintf("a) \n")
fprintf("momentum [DU^2/TU]: \n")
disp(h_1)
fprintf("inclination i [deg]: \n")
disp(i_1)
fprintf("Longitude of Ascending Node Omega [deg]: \n")
disp(Omega_1)
fprintf("eccentricity e: \n")
disp(e_1)
fprintf("argument of perigee omega [deg]: \n")
disp(omega_1)
fprintf("true anomaly f [deg]: \n")
disp(f_1)
fprintf("b) \n")
fprintf("momentum [DU^2/TU]: \n")
disp(h_2)
fprintf("inclination i [deg]: \n")
disp(i_2)
fprintf("Longitude of Ascending Node Omega [deg]: \n")
disp(Omega_2)
fprintf("eccentricity e: \n")
disp(e_2)
fprintf("argument of perigee omega [deg]: \n")
disp(omega_2)
fprintf("true anomaly f [deg]: \n")
disp(f_2)
```

```
Problem 3:
a)
momentum [DU^2/TU]:
    2.1633

inclination i [deg]:
   42.3026

Longitude of Ascending Node Omega [deg]:
   15.9454
```

```
eccentricity e:
    0.4281

argument of perigee omega [deg]:
  329.2602

true anomaly f [deg]:
   54.1363

b)
momentum [DU^2/TU]:
    1.9222

inclination i [deg]:
   49.0435

Longitude of Ascending Node Omega [deg]:
  260.0835

eccentricity e:
    0.6104

argument of perigee omega [deg]:
   37.9175

true anomaly f [deg]:
  264.5358
```

Problem 4

This problem involves the calculation of the eccentric anomaly and the true anomaly at corresponding times using numeric methods (tolerance 1e-9)

Initial Conditions

```
p = 2; %[DU]
e = 1/3;
t_a = 1e-3; %[TU]
t_b = 1; %[TU]
t_c = 5; %[TU]
tol = 1e-9;
```

Solve

```
a = p / (1 - e^2);
n = sqrt(mu/a^3);
P = 2*pi/n;
M = @(t) 2*pi/P*t;
```

```
M_a = M(t_a);
[E_a,f_a,it_a] = newtmeth(1,e,M_a,tol);
DeltaE_a = [0 diff(E_a)];

M_b = M(t_b);
[E_b,f_b,it_b] = newtmeth(1,e,M_b,tol);
DeltaE_b = [0 diff(E_b)];

M_c = M(t_c);
[E_c,f_c,it_c] = newtmeth(1,e,M_c,tol);
DeltaE_c = [0 diff(E_c)];
```

Display Results

```
Iteration = it_a';
E = E_a';
f = f_a';
Delta_E = DeltaE_a';
Table_a = table(Iteration,Delta_E,E,f);

Iteration = it_b';
E = E_b';
f = f_b';
Delta_E = DeltaE_b';
Table_b = table(Iteration,Delta_E,E,f);

Iteration = it_c';
E = E_c';
f = f_c';
Delta_E = DeltaE_c';
Table_c = table(Iteration,Delta_E,E,f);

fprintf("Problem 4: \n")
fprintf("a) \n")
disp(Table_a)
fprintf("b) \n")
disp(Table_b)
fprintf("c) \n")
disp(Table_c)
```

Problem 4:
a)

Iteration	Delta_E	E	f
1	0	1	1.3156
2	-0.8772	0.1228	0.17345
3	-0.12205	0.00074981	0.0010604
4	-0.00030537	0.00044444	0.00062854
5	-1.5107e-11	0.00044444	0.00062854

b)

Iteration   Delta_E   E   f

```
    1              0        1     1.3156
    2       -0.51618  0.48382    0.67147
    3      -0.046063  0.43776    0.60956
    4    -0.00022871  0.43753    0.60925
    5    -5.2933e-09  0.43753    0.60925
    6              0  0.43753    0.60925
```

```
c)
 Iteration    Delta_E       E         f
 _________    _______    ______    ______

    1               0         1    1.3156
    2         0.92935    1.9293    2.2301
```

Problem 5

This problem uses the same initial conditions as problem 1 in homework 1 but uses Kepler's time of flight equation to calculate position and velocity vectors

Initial Conditions

```
r_i = [7642;170;2186]; %[km]
r_dot_i = [0.32;6.91;4.29]; %[km/s]
mu = 3.986e14 * 1e-9; %[km^3/s^2] 1e-9 to convert from m^3 to km^3
tol = 1e-9;
```

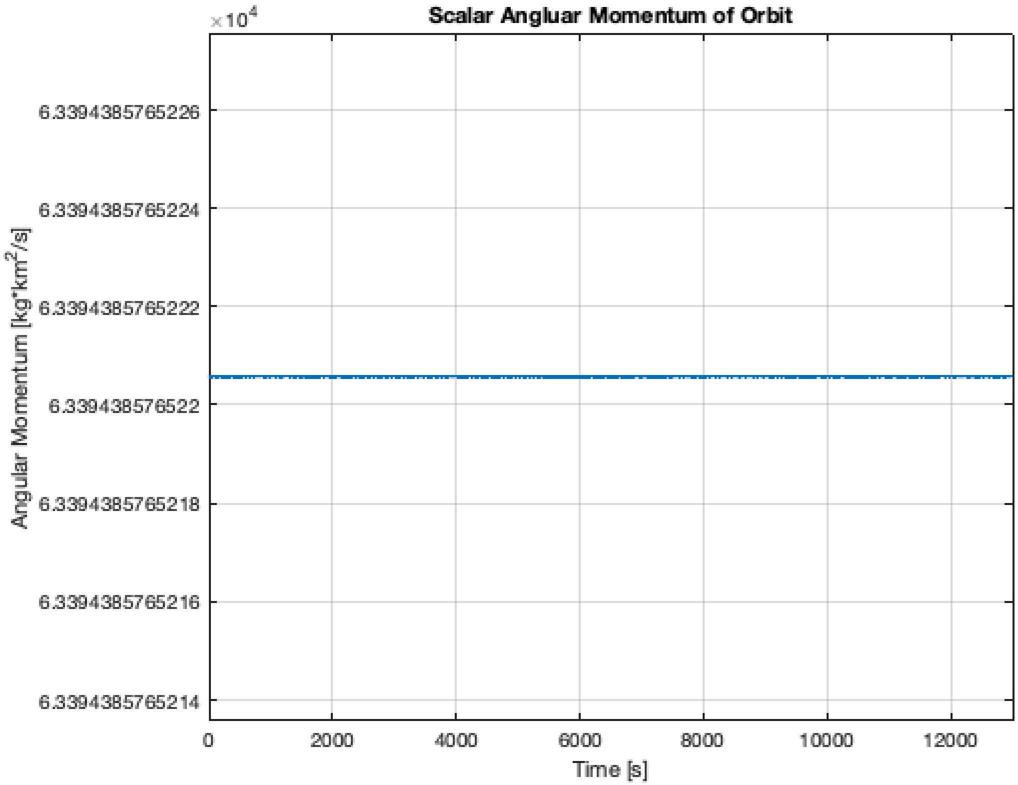
Make use of my functions

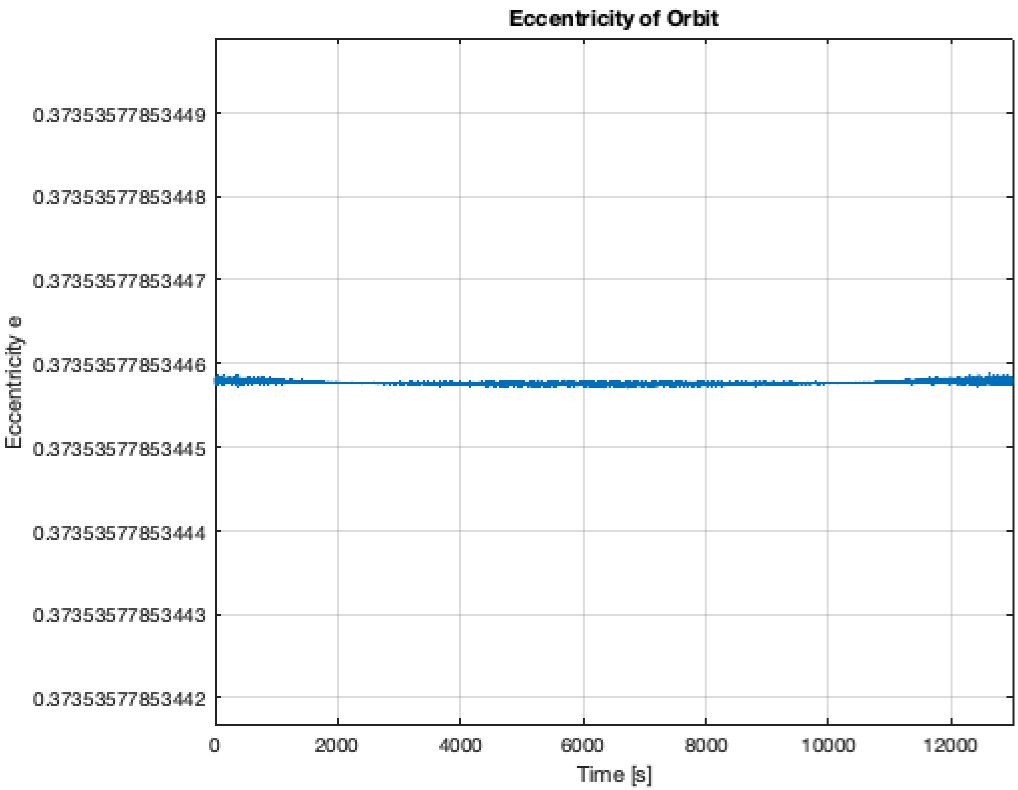
```
[h,i,Omega,e,omega,f] = orbelements(r_i,r_dot_i,mu);
p = h^2/mu;
a = p / (1 - e^2);
n = sqrt(mu/a^3);
P = 2*pi/n;
M = @(t) 2*pi/P*t;
t = 0:13000;
M_ = M(t);

% Preallocate
E_ = zeros(length(t),1);
f_ = zeros(length(t),1);

% Run time of flight integration
for i = 1:length(t)
    [E_(i),f_(i)] = keptof(2*pi*i/length(t),e,M_(i),tol);
end

% Extract position and velocity
%
```

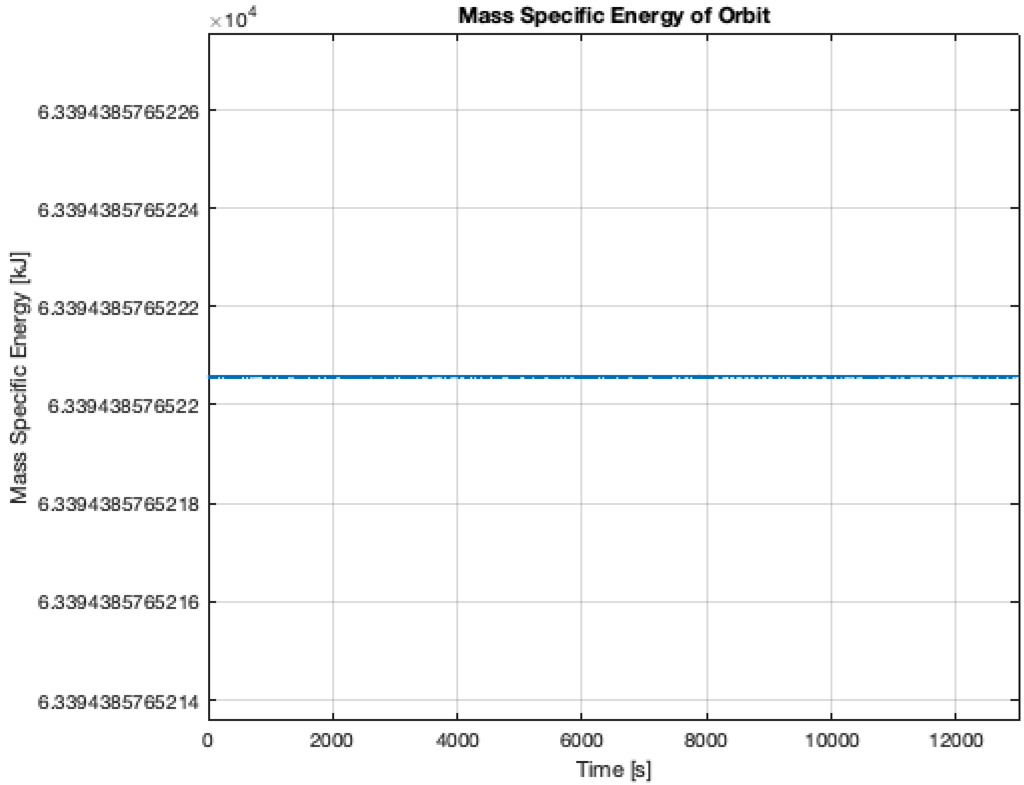
```
    % Note: these are in a different coordinate frame but because they
 are
    % in the same coordinate frame as each other, it won't matter when
 we
    % calculate scalar values
    r_ = a*(1 - e*cos(E_));
    r_vec = [r_.*cos(f_) r_.*sin(f_) zeros(length(r_),1)];
    r_dot_vec = mu/h * [-sin(f_) (e + cos(f_)) zeros(length(r_),1)];

    % Calculate Momentum Vector, Eccentricity Vector, Orbit Energy
    h_vec = zeros(length(t),3);
    h_s = zeros(length(t),1);
    e_vec = zeros(length(t),3);
    e_s = zeros(length(t),1);
    epsilon = zeros(length(t),1);
    for i = 1:length(t)
        h_vec(i,:) = cross(r_vec(i,:),r_dot_vec(i,:));
        h_s(i) = norm(h_vec(i,:));
        e_vec(i,:) = 1/mu*(cross(r_dot_vec(i,:),h_vec(i,:)) - mu *
 r_vec(i,:)/norm(r_vec(i,:)));
        e_s(i) = norm(e_vec(i,:));
        epsilon(i) = 0.5 * dot(r_dot_vec(i,:),r_dot_vec(i,:)) - mu/
norm(r_vec(i,:));
    end
```

Plot Results

```
    figure()
    plot(t,h_s); hold on
    title("Scalar Angluar Momentum of Orbit")
    xlabel("Time [s]")
    ylabel("Angular Momentum [kg*km^2/s]")
    xlim([t(1) t(end)])
    grid on
    hold off

    figure()
    plot(t,e_s); hold on
    title("Eccentricity of Orbit")
    xlabel("Time [s]")
    ylabel("Eccentricity e")
    xlim([t(1) t(end)])
    grid on
    hold off

    figure()
    plot(t,h_s); hold on
    title("Mass Specific Energy of Orbit")
    xlabel("Time [s]")
    ylabel("Mass Specific Energy [kJ]")
    xlim([t(1) t(end)])
    grid on
    hold off
```


Scalar Angluar Momentum of Orbit
×10^4
Angular Momentum [kg*km^2/s]
6.3394385765226
6.3394385765224
6.3394385765222
6.339438576522
6.3394385765218
6.3394385765216
6.3394385765214
0
2000
4000
6000
8000
10000
12000
Time [s]


Eccentricity of Orbit
Eccentricity e
0.37353577853449
0.37353577853448
0.37353577853447
0.37353577853446
0.37353577853445
0.37353577853444
0.37353577853443
0.37353577853442
0
2000
4000
6000
8000
10000
12000
Time [s]



Display Results

See notes on written section of homework

Functions

```
% Function for Problem 3
function [h,i,Omega,e,omega,f] = orbelements(r_,r_dot_,mu)
    % This is literally just a carbon copy of the method described in
    % Chapter 4.4 of the textbook
    %
    % If you want a better explanation, read the textbook, but it's
    % literally just plug and chug so here's a lot of vector math that
 I
    % didn't really bother to comment:
    r = norm(r_);
    r_dot = norm(r_dot_);
    v_r = dot(r_,r_dot_)/r;
    h_ = cross(r_,r_dot_);
    h = norm(h_);
    i = acos(h_(3)/h);
    N_ = cross([0;0;1],h_);
    N = norm(N_);

    if N_(2) >= 0
```

```
        Omega = acos(N_(1)/N);
    else
        Omega = 2*pi - acos(N_(1)/N);
    end

    e_ = 1/mu*(cross(r_dot_,h_) - mu*r_./r);
    e = norm(e_);

    if e_(3) >= 0
        omega = acos(dot((N_./N),(e_./e)));
    else
        omega = 2*pi - acos(dot((N_./N),(e_./e)));
    end

    if v_r >= 0
        f = acos(dot((e_./e),(r_./r)));
    else
        f = 2*pi - acos(dot((e_./e),(r_./r)));
    end
end


% Function for Problem 4
function [E,f,iteration] = newtmeth(E_0,e,M,tol)
f_calc = @(E) 2*atan(sqrt((1+e)/(1-e)) * tan(E/2));
ratio = 1;
E_ = E_0;
f_ = f_calc(E_0);

i = 2;
    while ratio > tol
        ratio = ((E_(i-1) - e*sin(E_(i-1)) - M)/(1 - e*cos(E_(i-1))));
        E_(i) = E_(i-1) - ratio;
        f_(i) = f_calc(E_(i));
        i = i+1;
    end
iteration = 1:i-1;
E = E_;
f = f_;
end


% Function for Problem 5
%
% Slightly modified funciton from problem 4, doesn't return vectors
 but
% rather returns scalar values of the final iteration of E
function [E,f] = keptof(E_0,e,M,tol)
f_calc = @(E) 2*atan(sqrt((1+e)/(1-e)) * tan(E/2));
ratio = 1;
E_ = E_0;
%f_ = f_calc(E_0);

%i = 2;
```

```
    while ratio > tol
        ratio = (E_ - e*sin(E_) - M)/(1 - e*cos(E_));
        E_ = E_ - ratio;
        %f_(i) = f_calc(E_(i));
        %i = i+1;
    end
%iteration = 1:i-1;
E = E_;
f = f_calc(E_);
end
```

Published with MATLAB® R2020a