Table of Contents


```
clear all;close all;clc
```

Problem 1

Write program that takes initial S/C position and velocity at initial time and predict future/past position of S/C at time t Part a -- Calculate orbital elements from arbitrary position vector, velocity vecor, and time

```
mu = 4e5; %Kg^3/s^2
r0 = [6e3 6e3 6e3];
v0 = [-5 5 0];
t0 = 0;
h = cross(r0,v0);
[a,e,i,w,W,tau,P,ehat,ehatperp] = compOE(r0,v0,t0,mu);
n = sqrt(mu/a^3);
T = 2*pi*sqrt(a^3/mu);
t = [0:60:2*T];
for i = 1:numel(t)
    M = n*(t(i) - tau);
    E = solvekep(M,norm(e));
    [r(i,:),v(i,:)] = calcRV(E,P,norm(e),ehat,ehatperp,mu);
end
figure
plot3(r(:,1),r(:,2),r(:,3))
grid on
grid minor
title('Oribt Trajectory')
figure
plot3(v(:,1),v(:,2),v(:,3))
grid on
grid minor
title('Orbit Velocity')
% part ii
clear all
mu = 4e5; %Kg^3/s^2
e = [0 0.25 0.5 0.75 0.99];
rp = 10000;
i = 135;
Omega = 45;
omega = -90;
tau = 0;
nhat_Omega = cosd(Omega)*[1 0 0] + sind(Omega)*[0 1 0];
nhat_Omega_perp = -cosd(i)*sind(Omega)*[1 0 0] +
 cosd(i)*cosd(Omega)*[0 1 0] + sind(i)*[0 0 1];
```

```
ehat = cosd(omega)*nhat_Omega + sind(omega)*nhat_Omega_perp;
ehat_perp = -sind(omega)*nhat_Omega + cosd(omega)*nhat_Omega_perp;
figure
hold on
for i = 1:5
    P = rp*(1+e(i));
    a = P/(1 - e(i)^2);
    T = 2*pi*sqrt(a^3/mu);
    t = [0:60:2*T];
    n = sqrt(mu/a^3);
    for j = 1:numel(t)
        M = n*(t(j) - tau);
        E = solvekep(M,e(i));
        [R(j,:),~] = calcRV(E,P,e(i),ehat,ehat_perp,mu);
    end
    plot3(R(:,1),R(:,2),R(:,3))
    grid on
    grid minor
end
title('Orbit Trajectories for varying e')
legend('e = 0','e = 0.25','e = 0.5','e = 0.75','e = 0.99')
```

Question 2

```
clear all;close all;clc
% Write a script that will solve Lambert's problem: Given two position
% vectors and a specified interval of time, compute the necessary
 initial
% velocity
% Part a -- Determine if requested transfer is elliptic or hyperbolic
t1 = 0;
t2 = 4.6;
% rtest = [-1.4641e4 0.4478e4 -0.5081e4];
% vtest = [-1.6848 -3.5827 -2.6338];
mu = 1;
r1 = [1 0 0];
r2 = [0 2 0];
tspan = [t1 t2(1)];
[astar,V,tp] = solvelamb(r1,r2,tspan,mu);
energy = -mu/(2*astar);
% for i = 2:numel(t2)
%     tspan = [t1 t2(i)];
%     [astar,V] = solvelamb(r1,r2,tspan,mu);
%     energy(i) = -mu/(2*astar);
%     if energy(i-1) < energy(i)
%         break
%     end
% end
%part iii
tspan = [0 2*tp];
[astar,V,tp] = solvelamb(r1,r2,tspan,mu);
```

Question 3

Write a script that numerically integrates the 2-body problem in cartesian coordinates Part a -- produce data files suitable for plotting traj. and vel. against each other in a 3D plot

```
clear all;close all;clc
mu = 4e5; %Kg^3/s^2
r0 = [6e3 6e3 6e3];
v0 = [-5 5 0];
T = 1.793395497395230e+04;
tspan = [0 2*T];
y0 = [r0 v0]';
opts = odeset('RelTol',1e-7,'AbsTol',1e-7);
[t,y] = ode45(@(t,y) odefun(t,y,mu),tspan,y0,opts);
figure
plot3(y(:,1),y(:,2),y(:,3))
grid on
grid minor
title('Orbit Trajectory ODE45')
figure
plot3(y(:,4),y(:,5),y(:,6))
grid on
grid minor
title('Orbit Velocity ODE45')
% Part b -- produce data files suitable for plotting containing
 computed
% values for orbital elements, energy, and angular momentum at each
 point
% in time.
for j = 1:numel(t)
    [a,e,i,w,W,tau,P,ehat,ehatperp] =
 compOE(y(j,1:3),y(j,4:6),t(j),mu);
    h(j) = norm(cross(y(j,1:3),y(j,4:6)));
    energy(j) = -mu/(2*a);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [a,e,i,w,W,tau,P,ehat,ehatperp] = compOE(r0,v0,t0,mu)
% mu = 4e5; %km^3/s^2
h = cross(r0,v0); %compute ang. mom.
hhat = h/norm(h);
P = norm(h)^2/mu;
i = acos(dot(hhat,[0 0 1])); %compute inclination
n_Omega = (cross([0 0 1],h))/(norm(cross([0 0 1],h)));
n_Omega_hat = n_Omega/norm(n_Omega);
W = atan2((dot(n_Omega,[0 1 0])),(dot(n_Omega,[1 0 0]))); %compute
 Omega
e = ((1/mu)*(cross(v0,h))) - (r0/norm(r0)); %compute omega
ehat = e/norm(e);
n_Omega_perp = cross(hhat,n_Omega_hat);
n_Omgegaperp_hat = n_Omega_perp/norm(n_Omega_perp);
w = atan2((dot(ehat,n_Omega_perp)),(dot(ehat,n_Omega)));
a = P/(1 - norm(e)^2);
```

```
ehatperp = cross(hhat,ehat);
n_Omgegaperp_hat = n_Omega_perp/norm(n_Omega_perp);
f = atan2((dot(r0,ehatperp)),(dot(r0,ehat)));
E = 2*atan2(sqrt(1-norm(e))*tan(f/2),sqrt((1+norm(e))));
n = sqrt(mu/a^3);
tau = t0 - (1/n)*(E-norm(e)*sin(E));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [E] = solvekep(M,e)
E = M;
tol = 0.001;
while (M - (E - e*sin(E))) > tol
    E = E - (M - (E - e*sin(E)))/(-(1 - e*cos(E)));
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [R,V] = calcRV(E,P,e,ehat,ehatperp,mu)
f = atan2(tan(E/2),sqrt((1-e)/(1+e)))*2;
R = (P/(1+e*cos(f)))*(cos(f)*ehat + sin(f)*ehatperp);
V = sqrt(mu/P)*(-sin(f)*ehat + (e+cos(f))*ehatperp);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function dydt = odefun(t,y,mu)
dydt = zeros(6,1);
r = sqrt(y(1)^2 + y(2)^2 + y(3)^2);
dydt(1) = y(4);
dydt(2) = y(5);
dydt(3) = y(6);
dydt(4) = (-mu*y(1))/r^3;
dydt(5) = (-mu*y(2))/r^3;
dydt(6) = (-mu*y(3))/r^3;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [astar,V,tp] = solvelamb(r1,r2,tspan,mu)
c = norm(r2 - r1);
s = .5*(norm(r1) + norm(r2) + c);
theta = acos(dot(r1,r2)/(norm(r1)*norm(r2)));
tp = (1/sqrt(mu))*(sqrt(2)/3)*((s^(3/2)) - sign(sin(theta))*(s -
 c)^(3/2));
beta_m = 2*asin(sqrt((s - c)/s));
if (theta >= pi) && (theta <= 2*pi)
    beta_m = -beta_m;
end
tm = (1/sqrt(mu))*sqrt(s^3/8)*(pi - beta_m + sin(beta_m));
syms a
alpha = 2*asin(sqrt(s/(2*a)));
beta = 2*asin(sqrt((s-c)/(2*a)));
if (theta >= pi)
    beta = -beta;
end
if (tspan(2) - tspan(1) > tm)
```

```
    alpha = 2*pi - alpha;
end
f = sqrt(mu)*(tspan(2) - tspan(1)) - a^(3/2)*(alpha - beta -
 (sin(alpha) - sin(beta)));
fprime = diff(f);
tol = 0.0001;
astar = (s/2)+1;
while abs(double(subs(f,astar))) > tol
    astar = astar - double(subs(f,astar))/double(subs(fprime,astar));
end
A = sqrt(mu/(4*astar))*cot(double(subs(alpha,astar))/2);
B = sqrt(mu/(4*astar))*cot(double(subs(beta,astar))/2);
u1 = r1/norm(r1);
u2 = r2/norm(r2);
uc = (r2 - r1)/norm(r2 - r1);
V = (B + A)*uc + (B - A)*u1;
%part i -- min energy transfer ellipse converged to a semi-major axis
 of
%1.309 with a transfer time of 4.5885 seconds or
beta_m = -beta_m;
tmflip = (1/sqrt(mu))*sqrt(s^3/8)*(pi - beta_m + sin(beta_m));
%part ii
min_e = (norm(r2) - norm(r1))/c;
min_e_a = (norm(r1) + norm(r2))/2;
alpha = 2*asin(sqrt(s/(2*min_e_a)));
beta = 2*asin(sqrt((s-c)/(2*min_e_a)));
tmin_e  = (min_e_a^(3/2)*(alpha - beta - (sin(alpha) - sin(beta))))/
sqrt(mu);
beta = -beta;
alpha = 2*pi - alpha;
tmin_e_flip = (min_e_a^(3/2)*(alpha - beta - (sin(alpha) -
 sin(beta))))/sqrt(mu);
%part iii -- V = [0.4829 1.0115 0]

end
```

Published with MATLAB® R2019b