Table of Contents


```
%-------- HW 5 MATLAB code --------%
% Romeo Perlstein, section 0101 %

% Chat it's over... I'm cooked!!! %
mew_earth = 0.39860*10^6; % km^3/s^2
earth_rad = 6378; % km
```

Q1

Lamberts problem solver - Implementing solution from lecture:

Case 1

First, find deltaV of two orbits

```
TOF1 = 3600;
r1_vec1 = [8000;0;0];
r2_vec1 = [7000;7000;0];
r1_1 = norm(r1_vec1);
r2_1 = norm(r2_vec1);
cos_deltaV1 = (dot(r1_vec1,r2_vec1))/(r1_1*r2_1);
DM1 = 1; % Given that the thang is short way
A1 = DM1*sqrt(r1_1*r2_1*(1+cos_deltaV1));

% Adding these for eventual migration to function
if DM1==0
    fprintf("YOU GOT AN ERROR BOY");
end
if A1==0
    fprinf("YOU GOT AN ERROR BOY");
end

% Weird starter numbers I guess?
trident1 = 0;
C2_1 = 1/2;
C3_1 = 1/6;
trident_hp1 = 4*(pi^2);
trident_low1 = -4*pi;
deltat1 = 0;
```

```
tolerance = 10^-6; % good tolerance
while(abs(TOF1-deltat1) >= tolerance)
    y1 = r1_1 + r2_1 + (A1*((trident1*C3_1)-1)/sqrt(C2_1));
    if(A1>0 && y1 <0)
        trident_low1 = trident_low1 + 0.1;
    end
    x1 = sqrt(y1/C2_1);
    deltat1 = (((x1^3)*C3_1)+(A1*sqrt(y1)))/sqrt(mew_earth);
    if deltat1 < TOF1
        trident_low1 = trident1;
    else
        trident_hp1 = trident1;
    end
    trident1 = (trident_hp1 + trident_low1)/2;
    if trident1 > tolerance
        C2_1 = (1-cos(sqrt(trident1)))/trident1;
        C3_1 = (sqrt(trident1)-sin(sqrt(trident1)))/sqrt(trident1^3);
    elseif trident1 < -tolerance
        C2_1 = (1-cosh(sqrt(-trident1)))/trident1;
        C3_1 = (sinh(sqrt(-trident1))-sqrt(-trident1))/sqrt(-trident1^3);
    else
        C2_1 = 1/2;
        C3_1 = 1/6;
    end
end
% Now find other stuff (what is going ON with these variables maine)
f1 = 1 - (y1/r1_1);
g1 = A1*sqrt(y1/mew_earth);
g_dot1 = 1 - (y1/r2_1);
v1_vec1 = (r2_vec1 - (f1*r1_vec1))/g1;
v2_vec1 = ((g_dot1*r2_vec1) - r1_vec1)/g1;
```

Case 2

Now do it again for case 2 copy and pasted from above!!!

```
TOF2 = 16135;
r1_vec2 = [0.5;0.6;0.7]*earth_rad;
r2_vec2 = [0;-1;0]*earth_rad;
r1_2 = norm(r1_vec2);
r2_2 = norm(r2_vec2);
cos_deltaV2 = (dot(r1_vec2,r2_vec2))/(r1_2*r2_2);
DM2 = -1; % Given that the thang is short way
A2 = DM2*sqrt(r1_2*r2_2*(1+cos_deltaV2));

% Adding these for eventual migration to function
if DM2==0
    fprintf("YOU GOT AN ERROR BOY");
end
if A2==0
    fprinf("YOU GOT AN ERROR BOY");
end
```

```
% Weird starter numbers I guess?
trident2 = 0;
C2_2 = 1/2;
C3_2 = 1/6;
trident_hp2 = 4*(pi^2);
trident_low2 = -4*pi;
deltat2 = 0;

while(abs(TOF2-deltat2) >= tolerance)
    y2 = r1_2 + r2_2 + (A2*((trident2*C3_2)-1)/sqrt(C2_2));
    if(A2>0 && y2 <0)
        trident_low2 = trident_low2 + 0.1;
    end
    x2 = sqrt(y2/C2_2);
    deltat2 = (((x2^3)*C3_2)+(A2*sqrt(y2)))/sqrt(mew_earth);
    if deltat2 < TOF2
        trident_low2 = trident2;
    else
        trident_hp2 = trident2;
    end
    trident2 = (trident_hp2 + trident_low2)/2;
    if trident2 > tolerance
        C2_2 = (1-cos(sqrt(trident2)))/trident2;
        C3_2 = (sqrt(trident2)-sin(sqrt(trident2)))/sqrt(trident2^3);
    elseif trident2 < -tolerance
        C2_2 = (1-cosh(sqrt(-trident2)))/trident2;
        C3_2 = (sinh(sqrt(-trident2))-sqrt(-trident2))/sqrt(-trident2^3);
    else
        C2_2 = 1/2;
        C3_2 = 1/6;
    end
end
% Now find other stuff (what is going ON with these variables maine)
f2 = 1 - (y2/r1_2);
g2 = A2*sqrt(y2/mew_earth);
g_dot2 = 1 - (y2/r2_2);
v1_vec2 = (r2_vec2 - (f2*r1_vec2))/g2;
v2_vec2 = ((g_dot2*r2_vec2) - r1_vec2)/g2;
```

Q2

Use the 2bodyprop to get orbit data

```
%%%% Case 1
%--- ODE func values from HW1---%
tall_er_ant = (10^-13); % Tolerance
step_size = 0.01; % step size
max_time = 3600; % max time (0->max_time)
t = [0:step_size:max_time]; % timestep

% ODE options
ODE_options = odeset("RelTol", tall_er_ant, "AbsTol", tall_er_ant);
```

```
% get the inital state
case1_initial_state = [r1_vec1(1); r1_vec1(2); r1_vec1(3); v1_vec1(1);
 v1_vec1(2); v1_vec1(3);];

[T1, Y1] = ode45(@myodefun, t, case1_initial_state, ODE_options, mew_earth);
% checks out!
```

Case 2

```
%%%% Case 1
%--- ODE func values from HW1---%
tall_er_ant = (10^-13); % Tolerance
step_size = 0.01; % step size
max_time = 16135; % max time (0->max_time)
t = [0:step_size:max_time]; % timestep

% get the inital state
case1_initial_state = [r1_vec2(1); r1_vec2(2); r1_vec2(3); v1_vec2(1);
 v1_vec2(2); v1_vec2(3);];

[T2, Y2] = ode45(@myodefun, t, case1_initial_state, ODE_options, mew_earth);
% checks out!
```

Q3

Using the data from case 1, find the deltaV find orbital parameters at r2

```
[i3, omega3, w3, true_anom3, e_x3, e_y3, e_z3, a3, spef_energy3] =
 cartToOrbitalElements(r2_vec1, v2_vec1, mew_earth, "rad");
e3 = norm([e_x3, e_y3, e_z3]);

% find rp, ra
rp3 = a3*(1-e3);
ra3 = a3*(1+e3);

% now find the stuff considering Hohmann transfer
v_tf_peri3 = sqrt((2*mew_earth)/rp3 - (2*mew_earth)/(rp3 + ra3));
v_tf_apo3 = sqrt((2*mew_earth)/ra3 - (2*mew_earth)/(rp3 + ra3));
v_circ_i3 = sqrt(mew_earth/rp3);
v_circ_f3 = sqrt(mew_earth/ra3);

deltaV1_3 = v_tf_peri3 - v_circ_i3;
deltaV2_3 = v_circ_f3 - v_tf_apo3;
deltaV_3 = deltaV1_3 + deltaV2_3; % Got it!



% my two body prop
function ydot = myodefun(t, y, mew)
    r_mag = norm(y(1:3));
    ydot(1,1) = y(4);
    ydot(2,1) = y(5);
    ydot(3,1) = y(6);
```

```
    ydot(4,1) = (-mew/r_mag^3)*y(1);
    ydot(5,1) = (-mew/r_mag^3)*y(2);
    ydot(6,1) = (-mew/r_mag^3)*y(3);
end
```

Published with MATLAB® R2022b