Table of Contents


```
% HW5 Script
% Orbital Mechanics
% Garrett Ailts

clear vars, close all
```

Constants

```
mu = 398600.44;
```

Define Cases

Types

```
direc = {'prograde','retrograde'};

% Case 1
cases(1).r1 = [-3472.57238396015;8042.69586228059;2196.56246176354];
cases(1).r2 = [-8710.68403512538;-3815.7502616566;885.874473990578];
cases(1).dT = 2157.063791;
% Case 2
cases(2).r1 = [-5192.64707332683;-2730.67520032545;-1312.84187347459];
cases(2).r2 = [-1396.17652892469;-2398.88103277394;7822.986845429];
cases(2).dT = 1288.841280;
% Case 3
cases(3).r1 = [506.717068361234;-470.869269481925;138.096930726997];
cases(3).r2 = [-10298.3068750911;6203.57304746482;-16637.5263693708];
cases(3).dT = 1800.000000;
```

Solve Lambert for Each Case

```
for i = 1:length(cases)
    for j = 1:2
    [v1, v2, dTheta] = solveLambert(cases(i).r1,cases(i).r2, ...
                                    cases(i).dT,mu,direc{j});
    cases(i).dir(j).v1 = v1; % dir(1) is prograde, dir(2) is
 retrograde
    cases(i).dir(j).v2 = v2;
    cases(i).dir(j).dTheta =dTheta;
    end
end
```

Obtain COE for Each Case

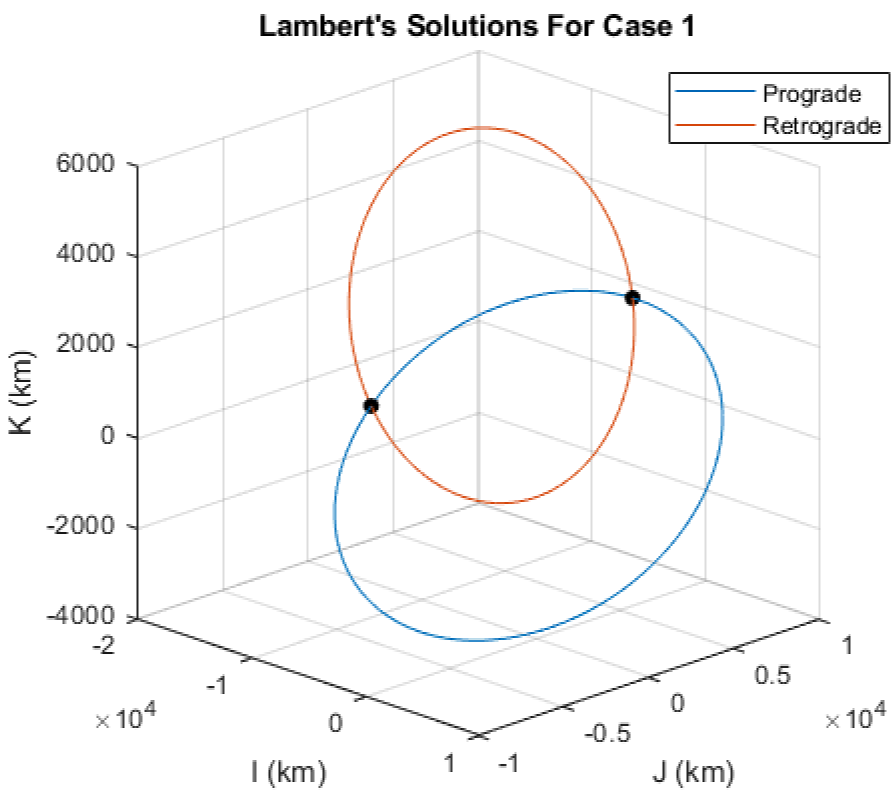
```
for i = 1:length(cases)
    for j = 1:2
    [a, e, theta, OMEGA, omega, inc, h] = rv2OrbEl(cases(i).r1, ...

 cases(i).dir(j).v1,mu);
    cases(i).dir(j).a = a;
    cases(i).dir(j).e =e ;
    cases(i).dir(j).theta = theta;
    cases(i).dir(j).OMEGA = OMEGA;
    cases(i).dir(j).omega = omega;
    cases(i).dir(j).inc = inc;
    cases(i).dir(j).h = h;
    if cases(i).dir(j).a < 0
        thetaInf = acos(-1/e);
        cases(i).dir(j).thetaVec = linspace(-thetaInf+0.05, ...
                                     thetaInf-0.05,1000);
    else
        cases(i).dir(j).thetaVec = linspace(0,2*pi,1000);
    end
    end
end
```

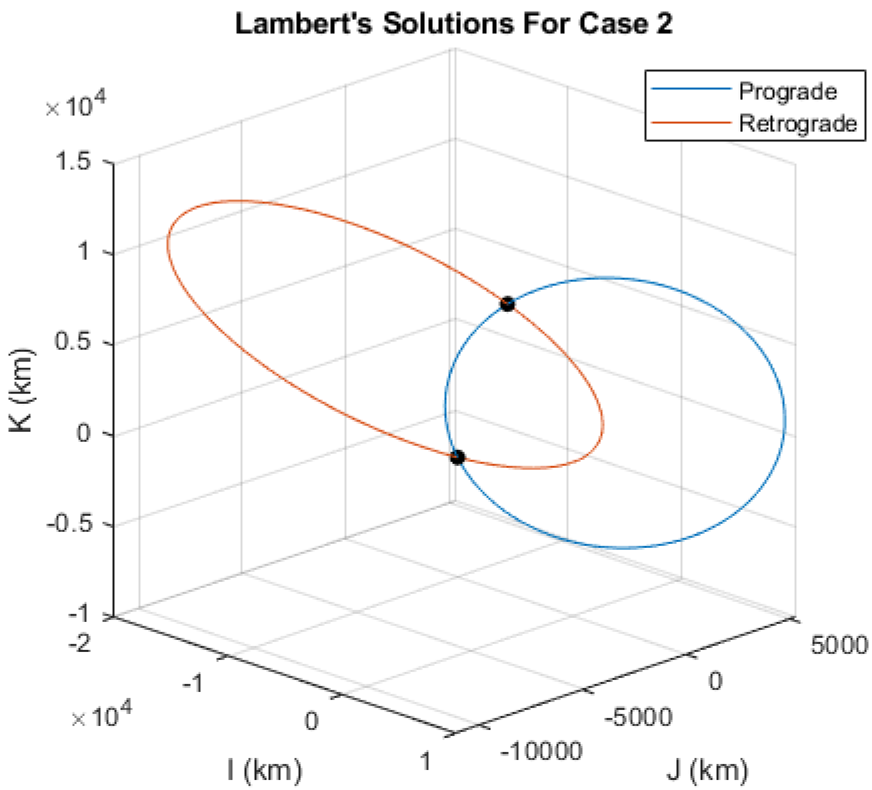
Display and Plot Lambert Solutions

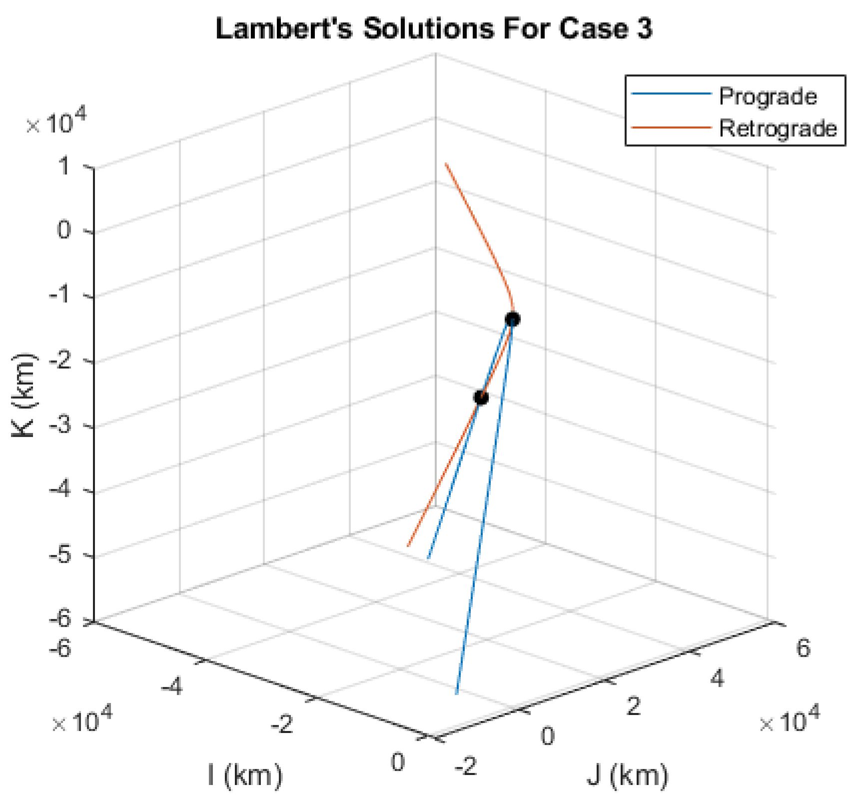
```
for i = 1:length(cases)
    fprintf("For case %u: \n",i);
    fprintf("The prograde solution (km/s) is:\n")
    fprintf("v1x = %.3f, v1y = %.3f, v1z = %.3f\n", ...
            cases(i).dir(1).v1(1),cases(i).dir(1).v1(2), ...
            cases(i).dir(1).v1(3));
    fprintf("v2x = %.3f, v2y = %.3f, v2z = %.3f\n", ...
            cases(i).dir(1).v2(1),cases(i).dir(1).v2(2), ...
            cases(i).dir(1).v2(3));
    fprintf("The retrograde solution (km/s) is:\n")
    fprintf("v1x = %.3f, v1y = %.3f, v1z = %.3f\n", ...
            cases(i).dir(2).v1(1),cases(i).dir(1).v1(2), ...
            cases(i).dir(2).v1(3));
    fprintf("v1x = %.3f, v1y = %.3f, v1z = %.3f\n", ...
            cases(i).dir(2).v2(1),cases(i).dir(1).v2(2), ...
            cases(i).dir(2).v2(3));
    figure(i);
    hold on
    plotOrbTraj(cases(i).dir(1), cases(i).dir(1).thetaVec, mu);
    plotOrbTraj(cases(i).dir(2), cases(i).dir(2).thetaVec, mu);

 scatter3(cases(i).r1(1),cases(i).r1(2),cases(i).r1(3),'k','filled');

 scatter3(cases(i).r2(1),cases(i).r2(2),cases(i).r2(3),'k','filled');
    hold off, axis equal, axis square, grid on
    xlabel('I (km)'), ylabel('J (km)'), zlabel('K (km)');
```

```
    title(sprintf("Lambert's Solutions For Case %u",i));
    legend('Prograde', 'Retrograde'), view(45,20);
end
```

```
For case 1:
The prograde solution (km/s) is:
v1x = -6.183, v1y = -2.775, v1z = 0.616
v2x = 2.080, v2y = -5.904, v2z = -1.525
The retrograde solution (km/s) is:
v1x = 6.200, v1y = -2.775, v1z = -1.785
v1x = -6.545, v1y = -5.904, v1z = 1.517
For case 2:
The prograde solution (km/s) is:
v1x = -1.541, v1y = -2.667, v1z = 8.730
v2x = 5.360, v2y = 2.303, v2z = 3.888
The retrograde solution (km/s) is:
v1x = 7.896, v1y = -2.667, v1z = -3.130
v1x = -4.381, v1y = 2.303, v1z = 5.485
For case 3:
The prograde solution (km/s) is:
v1x = -14.874, v1y = 20.430, v1z = 23.099
v2x = -3.990, v2y = 2.078, v2z = -7.782
The retrograde solution (km/s) is:
v1x = 3.884, v1y = 20.430, v1z = -31.958
v1x = -5.067, v1y = 2.078, v1z = -6.562
```




Lambert's Solutions For Case 2
Prograde
Retrograde
×10^4
1.5
1
0.5
0
-0.5
-1
K (km)
-2
-1
0
1
×10^4
I (km)
-10000
-5000
0
5000
J (km)


Lambert's Solutions For Case 3
Prograde
Retrograde
×10^4
1
0
-1
-2
-3
-4
-5
-6
K (km)
-6
-4
-2
0
×10^4
I (km)
-2
0
2
4
6
×10^4
J (km)

Published with MATLAB® R2018b