```
%{
Lambert solver -- solves for traj. conecting two position vectors
 given a TOF.
Inputs: Initial pos. vector, final pos. vector, desired TOF, direction
 of
Motion (optional)
%}
function [v0,vf,dt] = solvelambert_MR(r0,rf,psi,DM,n)
mu = 1.32712440018e11;
% dt = 1;
nu1 = atan2(r0(2),r0(1));
nu2 = atan2(rf(2),rf(1));

delta_nu = nu2 - nu1;

if delta_nu < 0
    delta_nu = delta_nu + 2*pi;
end
if DM == 0
    if delta_nu < pi
        DM = 1;
    else
        DM = -1;
    end
end
cosdnu = (dot(r0,rf))/(norm(r0)*norm(rf));

A = DM*sqrt(norm(r0)*norm(rf)*(1 + cosdnu));

if delta_nu == 0 || A == 0
    error('Trajectory cannot be computed')
end

% C2 = (1/2); C3 = (1/6);
%         if (psi > 1e-6)
            C2 = (1 - cos(sqrt(psi)))/psi; C3 = (sqrt(psi) -
 sin(sqrt(psi)))/sqrt(psi^3);
%         elseif (psi < -1e-6)
%             C2 = (1 - cosh(sqrt(-psi)))/psi; C3 = (sinh(sqrt(-psi))
 - sqrt(-psi))/sqrt((-psi)^3);
%         end

% psi = 0;
% psi_up = 4*(n+1)^2*pi^2;
% psi_low = -4*n^2*pi;

% its = 1;

% while abs(dt - TOF) > 1e-1
    y = norm(r0) + norm(rf) + (A*(psi*C3 - 1))/sqrt(C2);

%     if A > 0.0 && y < 0.0
```

```
%         while y < 0.0
%             psi = psi + 0.1;
%         if (psi > 1e-6)
%             C2 = (1 - cos(sqrt(psi)))/psi; C3 = (sqrt(psi) -
 sin(sqrt(psi)))/sqrt(psi^3);
%         elseif (psi < -1e-6)
%             C2 = (1 - cosh(sqrt(-psi)))/psi; C3 = (sinh(sqrt(-psi))
 - sqrt(-psi))/sqrt((-psi)^3);
%         else
%             C2 = (1/2); C3 = (1/6);
%         end
%         y = norm(r0) + norm(rf) + (A*(psi*C3 - 1))/sqrt(C2);
%         end
%     end
    chi = sqrt(y/C2);
    dt = (chi^3*C3 + A*sqrt(y))/sqrt(mu);

%     if dt <= TOF
%         psi_low = psi;
%     else
%         psi_up = psi;
%     end
%
%     psi = (psi_up + psi_low)/2;
%
%     if (psi > 1e-6)
%         C2 = (1 - cos(sqrt(psi)))/psi; C3 = (sqrt(psi) -
 sin(sqrt(psi)))/sqrt(psi^3);
%     elseif (psi < -1e-6)
%         C2 = (1 - cosh(sqrt(-psi)))/psi; C3 = (sinh(sqrt(-psi)) -
 sqrt(-psi))/sqrt((-psi)^3);
%     else
%         C2 = (1/2); C3 = (1/6);
%     end
%     its = its + 1;
% end

f = 1 - (y/norm(r0));
gdot = 1 - (y/norm(rf));
g = A*sqrt(y/mu);

v0 = (rf - f*r0)/g;
vf = (gdot*rf - r0)/g;
end
```

Published with MATLAB® R2020b