```matlab
function [vi, vf] = lambert(ri_vec, rf_vec, dt, DM, Sun)
%lambert Solve lambert problem using universal variables method
%   Output initial and final velocities given respective position vectors
%   Inputs in km, Results in km/s
%   DM = Direction of Motion (short way or long way)

fcnPrintQueue(mfilename('fullpath')) % Add this code to code app

tol = 1e-6;

ri = norm(ri_vec);
rf = norm(rf_vec);

cos_df = dot(ri_vec, rf_vec)/(ri*rf);

A = DM*sqrt(ri * rf * (1+cos_df));

psi = 0;
c2 = 1/2;
c3 = 1/6;
psi_up = 4*pi*pi;
psi_low = -4*pi;

dt_calc = 0;

while abs(dt_calc-dt) > tol
    y = ri + rf + A*(psi*c3-1)/sqrt(c2);
    if A > 0 && y < 0
        while y < 0
            psi = psi + 0.1;
            y = ri + rf + A*(psi*c3-1)/sqrt(c2);
        end
    end

    X = sqrt(y/c2);
    dt_calc = (X*X*X*c3 + A*sqrt(y))/sqrt(Sun.mu);

    if (dt_calc <= dt)
        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*psi*psi);
    elseif psi < -1e6
        c2 = (1-cosh(sqrt(psi)))/psi;
        c3 = (sinh(sqrt(-psi)) - sqrt(-psi))/sqrt(-psi*psi*psi);
    else
```

```
        c2 = 1/2;
        c3 = 1/6;
    end

end

f = 1-y/ri;
g_dot = 1-y/rf;
g = A*sqrt(y/Sun.mu);

vi = (rf_vec-f*ri_vec)/g;
vf = (g_dot*rf_vec-ri_vec)/g;


end
```

Published with MATLAB® R2013b