Table of Contents


```
function a = solveLambertsEq(mu, s, c, TOF, shortTOF, lt180, ellipse)
% Iteratively solves Lambert's Equation for a given instance of Lambert's
% Problem.
%   Inputs:
%       - mu: Gravitational parameter of the desired transfer
%       - s: Semi-perimeter of the space triangle for the desired transfer
%       - c: Chord length of the space triangle for the desired transfer
%       - TOF: Desired time of flight for the transfer
%       - shortTOF: Whether the TOF is shorter (1) or longer (0) than
%                   TOFmin
%       - lt180: Whether the desired transfer angle is less than (1) or
%                greater than (0) 180 degrees
%       - ellipse: Whether the transfer should be elliptical (1) or
%                  hyperbolic (0)
%   Outputs:
%       - a: Semi-major axis of the desired transfer
%
%   By: Ian Faber, 10/19/2024
%

amin = s/2; % Minimum energy ellipse semi-major axis
a0 = 1.01*amin; % Initial condition (amin + 1% offset)
tol = 1e-12; % Tolerance on abs(F - F_i)

fsolveOpt = optimoptions(@fsolve, 'MaxIterations', 999, 'FunctionTolerance',
tol, 'Display', 'none'); % Define stopping conditions and console interaction

if ellipse
    F_a = @(a)F_LambertsEq_Elliptical(a, mu, s, c, TOF, shortTOF, lt180); %
Define function for fsolve to use in terms of a
else
    F_a = @(a)F_LambertsEq_Hyperbolic(a, mu, s, c, TOF, shortTOF, lt180);
end

a = fsolve(F_a, a0, fsolveOpt);

end
```

Published with MATLAB® R2023b