```
%************Orbital Mech. HW 4, Alan Tsai and Vivek Suthram**********
function [v0, v1] = lambert(r0, r1, dt, direction)
    mu = 1.0;
    r0_mag = norm(r0);
    r1_mag = norm(r1);
    theta = acos(dot(r0, r1) / (r0_mag * r1_mag));

    if direction == 1
        % Short way
        signDir = 1;
    else
        % Long way
        theta = 2 * pi - theta;
        signDir = -1;
    end

    % Calculate A parameter
    A = signDir * sqrt(r0_mag * r1_mag * (1 + cos(theta)));
    z = 0;
    relErr = 1;
    tol = 1e-5;
    n = 0;
    nMax = 200;

    while relErr > tol
        if z == 0
            S = 1/6;
            S_prime = -1/120;
            C = 1/2;
            C_prime = -1/24;
        else
            C = (1 - cos(sqrt(z))) / z;
            S = (sqrt(z) - sin(sqrt(z))) / (z^1.5);
            S_prime = (1 / (2 * z)) * (C - 3 * S);
            C_prime = (1 / (2 * z)) * (1 - z * S - 2 * C);
        end

        % Solve for y, X, and error terms
        y = r0_mag + r1_mag - A * (1 - z * S) / sqrt(C);
        X = sqrt(y / C);
        U = (1 / sqrt(mu)) * (X^3 * S + A * sqrt(y)) - dt;
        V = (1 / sqrt(mu)) * (X^3 * (S_prime - (3 * S * C_prime) ...
            / (2 * C)) + (A / 8) * ((3 * S * sqrt(y)) / C + (A / X)));

        % Update z and relative error
        z_new = z - U / V;
        relErr = abs((z_new - z) / z);
        z = z_new;
        n = n + 1;

        if n > nMax
            break;
```

```
        end
    end

    % Compute coefficients for velocity
    f = 1 - y / r0_mag;
    g = A * sqrt(y / mu);
    g_dot = 1 - y / r1_mag;

    % Calculate initial and final velocity vectors
    v0 = (r1 - f * r0) / g;
    v1 = (g_dot * r1 - r0) / g;
end
```

```
Not enough input arguments.

Error in lambert (line 4)
    r0_mag = norm(r0);
```

Published with MATLAB® R2023b