```matlab
function [v1, v2, e, rp] = solve_lambert(r1, r2, TOF, mu, long_way)
%SOLVE_LAMBERT Solve Lambert's problem
% Solver for Lambert's problem using non-rigorous stumpff method
    arguments
        r1 double
        r2 double
        TOF double
        mu = SFD.mu_Earth
        long_way = false
    end
    r1_norm = norm(r1);
    r2_norm = norm(r2);
    cos_dth = dot(r1, r2)/(r1_norm*r2_norm);
    dth = acos(min(max(cos_dth, -1),1));
    if long_way
        if dth < pi
            dth = 2*pi - dth;
        end
    else
        if dth > pi
            dth = 2*pi - dth;
        end
    end
    A = sin(dth)*sqrt(r1_norm*r2_norm/(1 - cos(dth)));
    if A == 0
        error('Cannot compute Lambert solution: A = 0');
    end
    F = @(z) (( (r1_norm + r2_norm + A*(z.*SFD.stumpff_C3(z) -
1)./sqrt(SFD.stumpff_C2(z)))./SFD.stumpff_C2(z) ).^(3/2).*SFD.stumpff_C3(z)
+ A*sqrt(r1_norm + r2_norm + A*(z.*SFD.stumpff_C3(z) - 1)./
sqrt(SFD.stumpff_C2(z))) ) / sqrt(mu) - TOF;
    z = 0;
    for iter = 1:200
        Fz = F(z);
        if abs(Fz) < 1e-8
            break;
        end
        delta = 1e-6;
        dF = (F(z + delta) - F(z - delta))/(2*delta);
        z = z - Fz/dF;
    end
    C3 = SFD.stumpff_C3(z);
    C2 = SFD.stumpff_C2(z);
    y = r1_norm + r2_norm + A*(z*C3 - 1)/sqrt(C2);
    f = 1 - y/r1_norm;
    g = A*sqrt(y/mu);
    gdot = 1 - y/r2_norm;
    v1 = (r2 - f*r1)/g;
    v2 = (gdot*r2 - r1)/g;
    h_vec = cross(r1, v1);
    e_vec = (1/mu)*((norm(v1)^2 - mu/r1_norm)*r1 - dot(r1,v1)*v1);
    e = norm(e_vec);
```

```
    energy = norm(v1)^2/2 - mu/r1_norm;
    a = -mu/(2*energy);
    rp = a*(1 - e);
end
```

Published with MATLAB® R2024b