```
function [h_mag,e,RA,incl,w,TA,a] = states2oe(r,v,mu)
%STATES2OE Converts states of satellite to orbital elements
%   Inputs are:
%   r       :a numeric array of 3x1 current position vector in m
%   v       :a numeric array of 3x1 current velocity vector in m/s
%   mu      :an optional scalar gravitational parameter in m^3/s^2 (default
%            earth)
%
%   Outputs are:
%   h       :a scalar specific angular momentum in m^2/s

    arguments
        r (3,1) {mustBeNumeric, mustBeReal}
        v (3,1) {mustBeNumeric, mustBeReal}
        mu {mustBeNumeric, mustBeReal} = 3.986004418e14
    end

%   Magnitudes of postion and velocity vectors
    r_mag = norm(r);
    v_mag = norm(v);

%   Radial velocity
    v_rad = dot(r,v)/r_mag;

%   Specific angular momentum
    h = cross(r,v);
    h_mag = norm(h);

%   Inclination
    incl = acos(h(3)/h_mag);

%   Node line and magnitude of node line
    N = cross([0,0,1],h);
    N_mag = norm(N);

%   Right Ascension
    if N(2) >= 0
        RA = acos(N(1)/N_mag);
    elseif N(2) < 0
        RA = 2*pi - acos(N(1)/N_mag);
    else
        error('Node line undefined')
    end

%   Eccentricity
    e_vec = (1/mu)*((v_mag^2 - (mu/r_mag))*r - r_mag*v_rad*v);
    e = sqrt(dot(e_vec,e_vec));

%   Argument of perigee
    w = real(acos(dot(N,e_vec)/N_mag/e));
    if e_vec(3) < 0
```

```
        w = 2*pi - w;
    end

%   True anomaly
    if e > eps
        TA = acos(dot(e_vec,r)/e/r_mag);
        if v_rad < 0
            TA = 2*pi - TA;
        end
    else
        cp = cross(N,r);
        if cp(3) >= 0
            TA = acos(dot(N,r)/N_mag/r_mag);
        else
            TA = 2*pi - acos(dot(N,r)/N_mag/r_mag);
        end
    end

%   Semi-major axis
    a = h_mag^2/mu/(1 - e^2);
end
```