```
function [a,inc,W,w,e,th,h,Nvec,evec] = OrbitalElementsFromRV( r, v,
 mu )
%
% Compute 6 orbital elements from position and velocity vectors
%
% Inputs:
%   r     (3,1)   Position vector     [km]
%   v     (3,1)   Velocity vector     [km/s]
%
% Outputs:
%   a             Semi major axis     [km]
%   inc           Inclination         [rad]
%   W             Right ascension     [rad]
%   w             Argument of perigee [rad]
%   e             Eccentricity
%   th            True anomaly        [rad]
%

tol = 2*eps;

% Default value for mu (if not provided)
if( nargin<3 )
  mu = 398600.44;
end

% compute the magntiude of r and v vectors:
rMag  = sqrt(r'*r);
vMag  = sqrt(v'*v);

% radial component of velocity
vr    = v'*r/rMag;  % (Note that v'*r is equivalent to dot(v,r)

% Compute the specific angular momentum
hvec  = cross(r,v);
h     = sqrt(hvec'*hvec);

% inclination
inc = acos( hvec(3)/h ); % equivalent to acos( dot(hvec/h,[0;0;1]) )

% node line
Nvec  = cross([0;0;1],hvec);
N     = sqrt(Nvec'*Nvec);

% if the node line is not well defined from the angular momentum
 vector
% (this occurs at i=0 and i=pi) then define it to be along the
% inertial x vector
if( N<tol )
  Nvec = [1;0;0];
  N = 1.0;
end
```

```matlab
% right ascension
if( abs(inc)<tol || abs(inc-pi)<tol )
  W = 0; % R.A. not defined for zero-inclination orbits. Use W=0 by
 default.
elseif( Nvec(2)>=0 )
  W = acos(Nvec(1)/N);
else
  W = 2*pi-acos(Nvec(1)/N);
end

% Eccentricity
evec = 1/mu*( (vMag^2-mu/rMag)*r-rMag*vr*v );
e = sqrt(evec'*evec);

% Look out for special cases
equatorial = ( abs(inc)<tol || abs(inc-pi)<tol );
circular   = e < tol;

% Argument of perigee
%  - The angle between the eccentricity vector and the line of nodes
if( circular )
  % circular case
  % there is no definition for Arg. of perigee in this case.
  % just set it to zero
  w = 0;
else
  % non-circular cases...

  if( equatorial )
    % equatorial

    arg = evec(1)/ e;
    w = acos( arg );
    if( abs(inc) < tol && evec(2) < 0 )
      w = 2*pi - w;
    elseif (abs(inc - pi) < tol) && evec(2) > 0
      % retrograde: change sign of test
      w = 2*pi - w;
    end
  else
      % non-equatorial

    arg = Nvec'*evec/N/e;
    % check the argument to prevent issues with numerical rounding
    if( arg>=1 )
      w = 0;
    elseif( arg<= -1 )
      w = pi;
    else
      w = acos(arg);
    end

    % check for retrograde case!
    if( evec(3)<0 )
```

```
        w = 2*pi-w;
      elseif( abs(inc-pi)<tol && evec(2)>0 )
        w = 2*pi-w;
      end

    end

  end


  % True anomaly
  %  - The angle between the eccentricity vector and the position vector
  if( circular && equatorial )
    % circular and equatorial case
    arg = r(1)/rMag;
    th = acos(arg);
    if( r(2)<0 )
      th = 2*pi-th;
    end
  else
    if( circular )
      % circular and inclined case
      % (measure TA from line of nodes)
      evecDir = Nvec/N;
    else
      % non-circular case
      evecDir = evec/e;
    end
    arg = evecDir'*r/rMag;
    if( arg>= 1.0 )
      arg = 1.0;
    elseif( arg <= -1.0 )
      arg = -1.0;
    end

    if( vr>=0 )
      th = acos(arg); % T.A. is defined to lie between 0 and 2*PI
    else
      th = 2*pi-acos(arg);
    end

  end


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

  % return all six elements into a vector if only one output is
   requested
  if( nargout==1 )
    a = [a,inc,W,w,e,th];
  end
```

Published with MATLAB® R2019b