```
function [a,ecc,theta,inc,raan,aop,h,T,E] = joshCOE(R,V,u,magOrVec)

%%%%%%%%%%%%%%%%%%
% Revamped to do rads and fit new naming convention %
%%%%%%%%%%%%%%%%%%


%COESOATESJOSHUA Takes postion and velocity vector and returns COES, all in
%ECI frame of reffrenece, km and seconds as units and degrees
% a = semi major axis
% ecc = eccentricity
% i = inclination
% raan = right accention acending node
% aop = argument of periapsis
% theta = true anomaly

% will return T in s as a period
% and E (sometimes epsilon) in km^2/s^2 as specific mechanical energy
% h is agular momentum

% magOrVec is a parameter that can be set for vector inputs to return
% vector h and ecc

% scalar entry should only be used if the spacecraft is at apoapse or
% periapse

arguments
    R {mustBeNumeric, mustBeReal}
    V  {mustBeNumeric, mustBeReal}
    u (1,1) {mustBeNumeric, mustBeReal, mustBePositive} = 3.986004418 *
 (10^5) %km^3/s^2
    magOrVec {mustBeMember(magOrVec,{'magnitude','vector'})} = 'magnitude'
end

[m1,n1] = size(R);
[m2,n2] = size(V);

if joshIsOnes([m1 n1 m2 n2])
    magOrVec = 'magnitude';
    R = [0 0 R];
    V = [V 0 0];
    warning("joshCOE will assume that R and V are normal if the inputs are
 scalar ie: the craft is in a circular orbit or is at periapse or apoapse")
elseif (~joshIsOnes([m1 n1] == [m2 n2]))|~((n1==1&m1==3)|(n1==3&m1==1))
    throw(MException("COEsOatesJoshua:invalidInput","R and V must be either
 1x3 vectors or scalars"))
end

% uearth = 3.986004418(8) x 10^14 m^3/s^2
ihat=[1,0,0];
khat=[0,0,1];
```

```matlab
Rm = norm(R);
Vm = norm(V);

%calculate orbital constants
h = cross(R,V); %angular momentum vector
hm = norm(h);
E = ( ( Vm^2 ) / 2) - ( u / Rm ) ; %specfic mechanical energy
%calculate COEs

a = -u / (2 * E ); %semi major axis in km
T = 2*pi*sqrt((a^3)/u); %period in s

e = (1/u) * (((Vm^2)-(u/Rm) ) * R - (dot(R,V) * V)); %eccentricity vector
em = norm (e); %magnitude of e

inc = acos((dot(khat,h))/hm); %inclination
n = cross(khat,h); %node vector
nm = norm(n); %magnitude n

%raan
raan = acos(dot(ihat,n)/nm);
if n(2) < 0  %checks the vector relative to j to see if angle is positive or
 negative
    raan = 360 - raan;
end

%aop
aop = acos(dot(n,e)/(nm*em));
if e(3) < 0
    aop = 360 - aop;
end

%theta
theta = acos(dot(e,R)/(em*Rm));
if(dot(R,V) < 0) % cehck flight path angle to see if it is postive or negative
    theta = 360 -theta;
end

if strcmp(magOrVec, 'magnitude') %for magnitude mode, vectors will not be
 returned
    h = hm;
    e = em;
    if joshIsOnes([m1 n1 m2 n2]) % for scalar inputs it is not possible to
 calculate these values
        theta = NaN;
        inc = NaN;
        raan = NaN;
        aop = NaN;
    end
end
ecc = e;

end
```

Published with MATLAB® R2022a