```matlab
clear all; clc; close all;

ro = [7000 2900 -3800];
vo = [7.11 1.52 2.69];

[h,e,i,w,W,true_ana] = stateVec2OrbElem(ro,vo);

fprintf('True Anomaly: %0.3f  degrees \n', true_ana)
fprintf('Specific Angular Momentum: %0.0f km^2/s \n', norm(h))
fprintf('Eccentricity: %0.3f \n', norm(e))
fprintf('Inclination: %0.3f  degrees \n', i)
fprintf('Right ascension of the ascending Node: %0.3f degrees \n', W)
fprintf('Argument of perigee %0.3f degrees \n', w)

function [h,e,i,w,W,true_ano,N] = stateVec2OrbElem(r,v)
%This function will take two state vectors r and v and compute the six
%orbital elements
% r and v must be 3-D vectors
% Currently this is only for Geocentric orbits
mu = 3.9860044189e5;



h = cross(r,v); % specific angular momentum
i = acosd(h(3)/norm(h));  % inclination is hz divided by the magnitude of h
N = cross([0 0 1],h); % line of nodes is the unit vector k cross h

    if N(2) >= 0
        % Right ascesnsion of the ascending Node
        W = acosd(N(1)/norm(N));
    else
        W = 360-acosd(N(1)/norm(N));
    end
 % eccentricity vector
e = ((norm(v)^2 - mu/norm(r))*r - (dot(r,v)*v))/mu;

    if e(3) >= 0
        % argument of perigee
        w = acosd(dot(N,e)/(norm(N)*norm(e)));
    else
        w = 360-acosd(dot(N,e)/(norm(N)*norm(e)));
    end

    if (dot(r,v)/norm(r)) >= 0
        %true anomaly
        true_ano = acosd(dot(e,r)/(norm(e)*norm(r)));
    else
        true_ano = 360-acosd(dot(e,r)/(norm(e)*norm(r)));
    end


end
```

True Anomaly: 114.815 degrees

Specific Angular Momentum: 48846 km^2/s

Eccentricity: 0.700

Inclination: 101.788 degrees

Right ascension of the ascending Node: 16.496 degrees

Argument of perigee 217.930 degrees

>>