The following is code for matlab converting the position of a vehicle in earth centered earth fixed coordinates to Latitude Longitude and altitude.
Figure 1 shows the position in ECEF coordinates
Figure 2 shows the position in latitude, longitude and altitude

From the website http://www.geocode.com/eagle.html
We were able to find that the approximate address where the data was take was approximately 1701 E. Empire St. in Bloomington Il.

```
function [lat, lon, alt] = ecef2ned(gtime, ecefpos)

% ECEF2NED  Convert ecef coordinates to NED coordinates and diplay.
%
% [latitude, longitude, altitude] = ecef2ned(gtime, ECEF position)
%  ECEF Position Array = [x, y, z]
%
%  Example:
%     array.x
%     array.y
%     array.z
%  *** The array parameters must have *.x, *.y, and *.z.
%
% Enter the time, and the x,y,z positions in ecef coordinates.
% This function will display two figures:
%           1:  x,y,z position in ecef coordinates and
%           2:  latitude, longitude, and altitude
% A negative longitude is the Western hemispere.
% A negative latitude is in the Southern hemisphere.
%
% Written  By: Brian Bleeker
%              Rob MacMillan

b = size(gtime);
gtime = gtime(:,1) - gtime(1,1);

figure(1), subplot(311), plot(gtime, ecefpos.x), grid
title('ECEF X Position')

subplot(312), plot(gtime, ecefpos.y), grid
title('ECEF Y Position')

subplot(313), plot(gtime, ecefpos.z), grid
title('ECEF Z Position')

a = 6378137.0;                                      %semi-major axis
(equatorial) radius
b = 6356752.3142;                                   %semi-minor axis
(polar) radius
f = (a-b)/a;
e = sqrt(f*(2-f));                                  %eccentricity of
ellipsoid
len = length(ecefpos.x);                            %get length of data
```

```
for i = 1:len
    lon(i) = atan2(ecefpos.y(i), ecefpos.x(i));     %long = atan(y,x) -
direct
    lon(i) = lon(i)*180/pi;                         %convert to degrees

    h = 0;                                          %initialize
    N = a;
    flag = 0;
    j = 0;
    p = sqrt(ecefpos.x(i)^2 + ecefpos.y(i)^2);

    sinlat = ecefpos.z(i)/(N*(1-e^2)+h);            %First iteration
    lat(i) = atan((ecefpos.z(i)+e^2*N*sinlat)/p);
    N = a/(sqrt(1 - (e^2)*(sinlat^2)));
    prevalt = (p/cos(lat(i)))-N;
    prevlat = lat(i)*180/pi;

    while (flag < 2)                                %do at least 100
iterations
        flag = 0;
        sinlat = ecefpos.z(i)/(N*(1-e^2)+h);
        lat(i) = atan((ecefpos.z(i)+e^2*N*sinlat)/p);
        N = a/(sqrt(1 - (e^2)*(sinlat^2)));
        alt(i) = (p/cos(lat(i)))-N;
        lat(i) = lat(i)*180/pi;
        if abs(prevalt-alt(i)) < .00000001
            flag = 1;
        end
        if abs(prevlat-lat(i)) < .00000001
            flag = flag + 1;
        end
        j = j+1;
        if j == 100
            flag = 2;
        end
        prevalt = alt(i);
        prevlat = lat(i);
    end
end

figure(2), subplot(311), plot(gtime, lon), grid
title('NED Longitude')

subplot(312), plot(gtime, lat), grid
title('NED Latitude')

subplot(313), plot(gtime, alt), grid
title('NED Altitude')
```

Figure 1

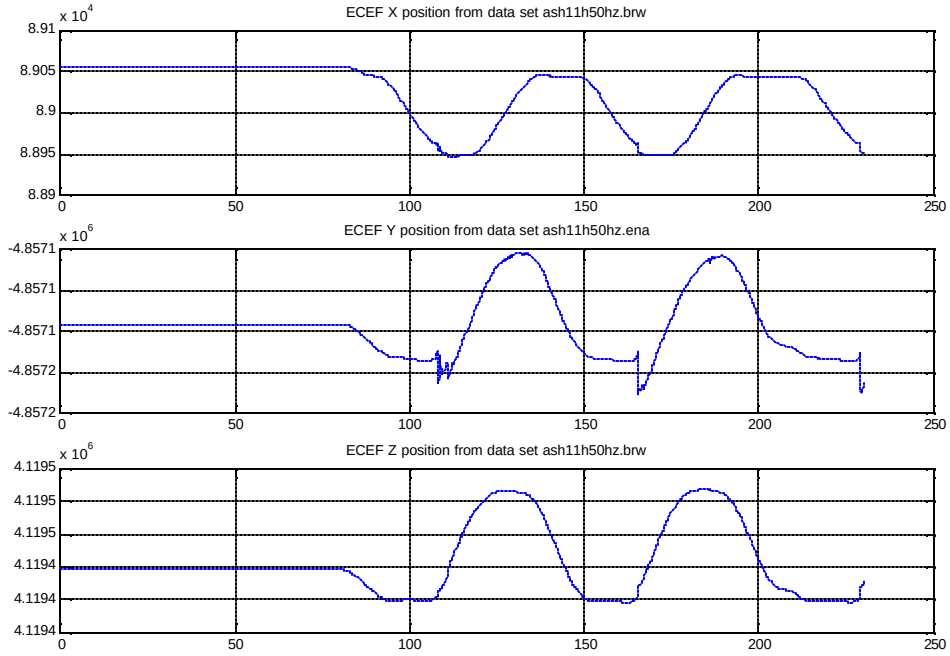
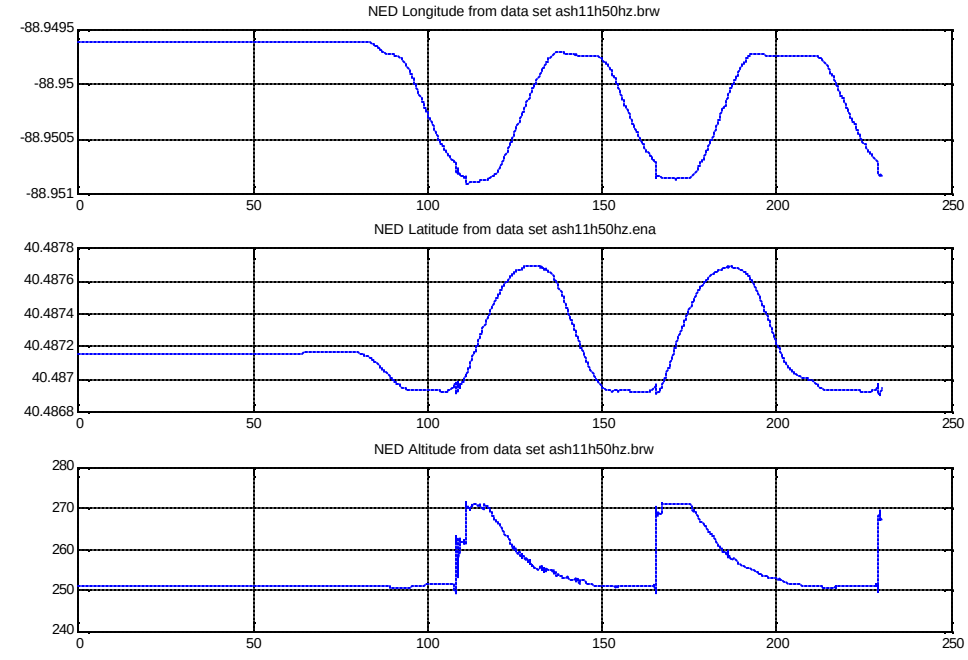


Figure 2



The following code converts North, East, Down coordinates to Earth Centered Earth Fixed (ECEF) coordinates.

```
function [ecef_pos] = ned2ecef2(ned)
% NED2ECEF2 Convert NED coordinates to ECEF coordinates and diplay.
%
% [ECEF position] = ned2ecef2(ned position)
%  NED Position Array [n,e,d]
%
%  Example:
%     array.n
%     array.e
%     array.d
%  *** The array parameters must have *.n, *.e, and *.d
%
% The NED positions are latitude longitude and altitude in decimal
degrees.
% A negative longitude is the Western hemispere.
% A negative latitude is in the Southern hemisphere.
%
% Written By: Brian Bleeker
%             Rob MacMillan

%a = earth_shape; %  call earth_shape to get earth data
a = 6378137 ; %a(1);
b = 6356752.3142; %a(2);
lat=ned.n*pi/180;
lon=ned.e*pi/180;
f = (a-b)/a;
e = sqrt(f*(2-f));
N = a /(sqrt(1-e^2*(sin(lat))^2));


%lat = ned.pos(1)/b + ned.geo_ref(1); % compute the current latitude
coslat = cos(lat);
sinlat = sin(lat);
coslon = cos(lon);
sinlon = sin(lon);

x = (N + ned.d)*coslat*coslon;

y = (N + ned.d)*coslat*sinlon;  % compute the current longitude

z = (N*(1 - e^2)+ ned.d)*sinlat;

%r0 = r0 + ned.geo_ref(3)*coslat;

ecef_pos.x = x;% assign positions
ecef_pos.y = y;
ecef_pos.z = z;
```

The following sample data takes the NED coordinates for Dr. Ahns house and converts it into ECEF coordinates. It then converts it back. From this we are able to find the error in the strictly mathematical model for processing the data.

Sample Data:

```
 » ned
ned =
    n: 40.752169
    e: -89.672332
    d: 250

» ecef=ned2ecef2(ned)
ecef =
    x: 27672.3433890974
    y: -4838712.35630367
    z: 4141776.28953698

» [lat, lon, alt] = ecef2ned(1, ecef)
lat =
         40.752176473724
lon =
             -89.672332
alt =
         249.999985948205
```

The error for the longitude calculation is very close to 0 as it is strictly an inverse tangent operation. When converting the latitude the error is on the order of .00002% which is caused by the itteration process. The error at this altitude is on the order of .00000006%. As coordinates and altitude change the error will increase however we calculated that the error at 2400 meters is approximately 2 milimeters and the error 12000 meters is aproximately 33 centimeters. We feel that this is reasonable error as 12000 meters is approximately the ceiling for commercial air travel (35000 ft.).