```
function [elevation, azimuth] = elevazim(satLoc,obsLoc)
%-------------------------------------------------------------------------
% Calculates the elevation and azimuth from a reference position specified in ECEF coordinates (e.g. antenna
% location) to another position specified in ECEF coordinates (e.g. satellite location)
%-------------------------------------------------------------------------
%
% input:'satLoc'  matrix of rows that contain ECEF coordiantes of satellites [m]
%                 could be differentiated by different satelits or
%                 different times for the same satellite
%       'obsLoc' vector which contains the ECEF coordinates
%                (meters) of a reference position (Ground station) which elevation and azimuth
%                will be calculated from
%
% output: the elevation and azimuth look angles [degrees] to the satellite
% from the ground station
%


degrad = pi/180.0;
% define satellite locations in ECEF coordinates
    satX = satLoc(:,1); % meters
    satY = satLoc(:,2); % meters
    satZ = satLoc(:,3); % meters
% define observation location in ECEF coordinates
    obsX = obsLoc(1);       % meters
    obsY = obsLoc(2);       % meters
    obsZ = obsLoc(3);       % meters
% compute unit vector from observation station to satellite position
    r = sqrt((satX - obsX).^2 + (satY - obsY).^2 + ...
        (satZ - obsZ).^2);
    dx = (satX - obsX) ./ r;
    dy = (satY - obsY) ./ r;
    dz = (satZ - obsZ) ./ r;
% compute the observation latitude and longitude
    obsLoc = latlong(obsLoc);
    latOBS = obsLoc(1) * degrad;        % radians
    longOBS = obsLoc(2) * degrad;       % radians
% compute the rotated unit vectors in VEN from observation station to satellite position
    north = dz .* cos(latOBS) - sin(latOBS) .* ...
        (dx .* cos(longOBS) + dy .* sin(longOBS));
    east = dy .* cos(longOBS) - dx .* sin(longOBS);
    vertical = cos(latOBS) .* (dx .* cos(longOBS) + ...
        dy .* sin(longOBS)) + dz .* sin(latOBS);
% compute elevation
    elevation = (pi / 2 - acos(vertical)) ./ degrad;    % degrees
% compute azimuth; check for negative angles
    azimuth = atan2(east,north);                % radians
    idx = find(azimuth < 0);
    azimuth(idx) = azimuth(idx) + 2 * pi;
    azimuth = azimuth ./ degrad;                % degrees
```

```
    return;



function ecoord = latlong(location)
% get ECEF location to be converted to latitude-longitude-altitude
degrad = pi/180.0;
% coordinates
    ECEFx = location(:,1);
    ECEFy = location(:,2);
    ECEFz = location(:,3);
% compute the longitude which is an exact calculation
    long = atan2(ECEFy , ECEFx);    % radians
% compute the latitude using iteration
    p = sqrt(ECEFx.^2 + ECEFy.^2);
    % compute approximate latitude
    AA = 6378137.00000;                 % meters
    BB = 6356752.31425;                 % meters
    esquare=(AA^2 - BB^2) / AA^2;
    lat0 = atan((ECEFz ./ p) ./ (1 - esquare));
    stop = 0;
    while (stop == 0)
        N0 = AA^2 ./ (sqrt(AA^2 * (cos(lat0)).^2 + ...
             BB^2 .* (sin(lat0)).^2));
        altitude = (p ./ cos(lat0)) - N0;       % meters
        % calculate improved latitude
        term = (1 - esquare * (N0 ./ (N0 + altitude))).^(-1);
        lat = atan(ECEFz ./ p .* term);         % radians
        % check if result is close enough,
        if (abs(lat - lat0) <  1.0e-12)
            stop = 1;
        end
        lat0 = lat;
    end
% convert the latitude and longitude to degrees
    latitude = lat ./ degrad;       % degrees
    longitude = long ./ degrad;     % degrees
% return location in latitude-longitude-altitude coordinates
    ecoord = [ latitude longitude altitude ];
    return;
```