ASEN 5090-Intro to GNSS Homework 3

Zach Dischner

9-24-2013

1 Problem 1 - YUMA Almenac File

The name of the YUMA file for September 20, 2012 is: **yuma0682.319488.alm**

The different PRNs corresponding to each GPS satellite and their respective orbit planes (normalized, by planes) is in Table 1.

PRN Number	**Normalized Plane**
20, 5, 18, 22, 10, 32	1
15, 23, 14, 26, 13	2
27, 9, 7, 31, 8	3
30, 25, 12, 16, 28	4
3, 6, 17, 29, 19	5
11, 2, 1, 4, 21	6

Table 1: PRN Satellites sorted by orbital plane

2 Problem 2 - ECEF2AzEL

I modified the function appropriately and verified that it works. The demonstration of this will come in later exercises as the function is called.

3 Problem 3 - Sat Masks

I created a function to return topographical and antenna counts of the number of satellites visible, given user-defined Az-El boundaries and antenna specifications. See Figure 1. It shows the total number of visible satellites to the antenna in time, given stats provided in the main script for this assignment.

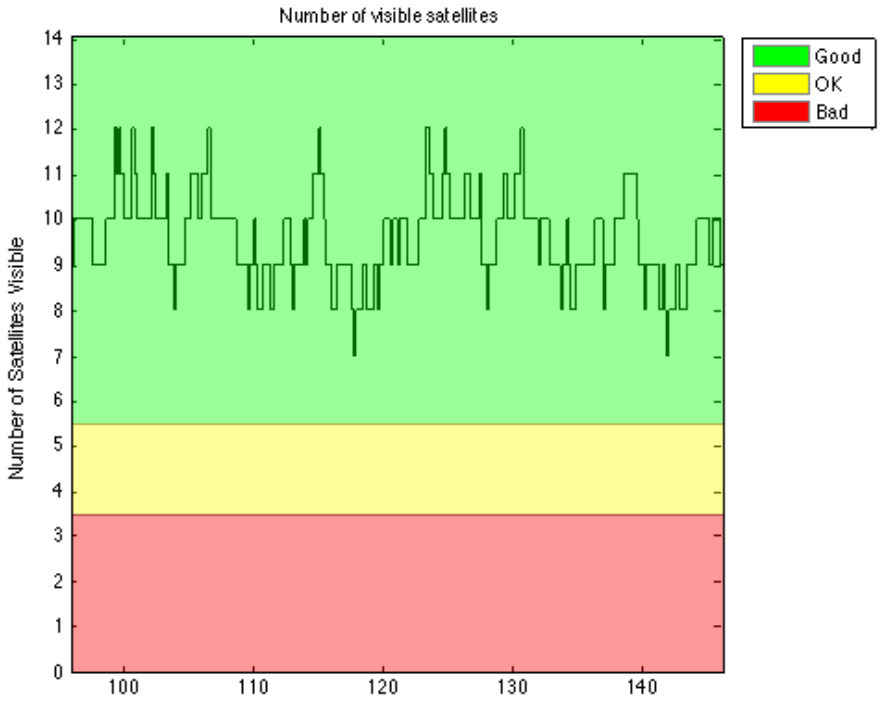

Figure 1: Number of visible satellites

4 Problem 4 - Visible Satellites in Boulder

Using the same masks and specifications as provided in previous problems (assuming them to be relevant to Boulder's masking constraints), I was able to obtain visible satellites on September 20, 2012 at 12:00 PM. I did this by inserting breakpoints in the masking code and reporting relevant stats directly from there. These stats are summarized in Table 2, below. This was done where 12:00 local time means 18:00 GMT.

PRN Number	Az	El
3	29.72	29.72
5	264.88	264.88
6	265.86	265.86
16	350	350
18	151.56	151.56
21	234.2	234.2
25	322.07	322.07
29	109.64	109.64
30	107.87	107.87
31	298.49	298.49

Table 2: PRN and Az, El for visible Satellites

5 Problem 5 - Visible Satellite Az El plots

Next, I plotted the Az and El of visible satelites at three different coordinates on the earth. I added to the *GPSdata* structure to include masking indices to make the process easier.

5.1 a) 0 N, 0 E

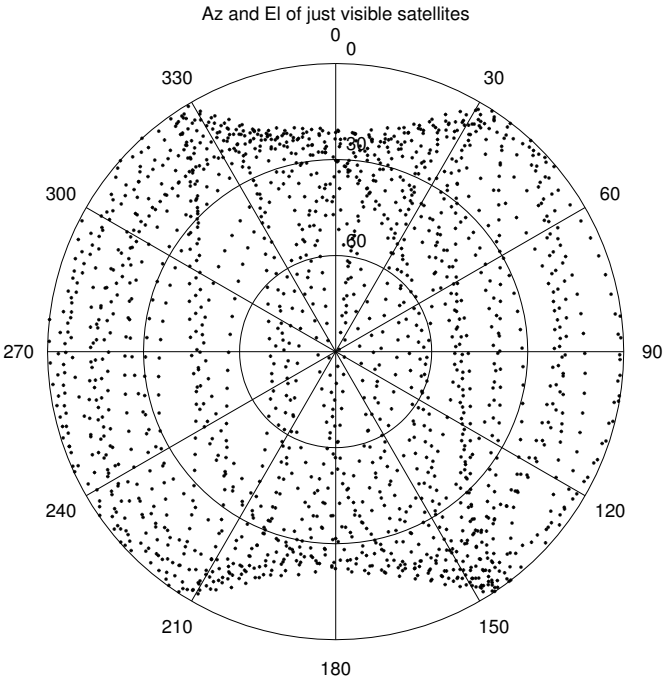

Figure 2: Az El for visible satellites at 0N 0E

At 0,0, on the equator, I see that satellites are visible along all longitudes, and only become invisible at high latitudes. The satellite orbits are designed to focus their concentration along the equator, and not so much at high latitudes. So this plot makes intuitive sense.

5.2 b) 90 N, 0 E

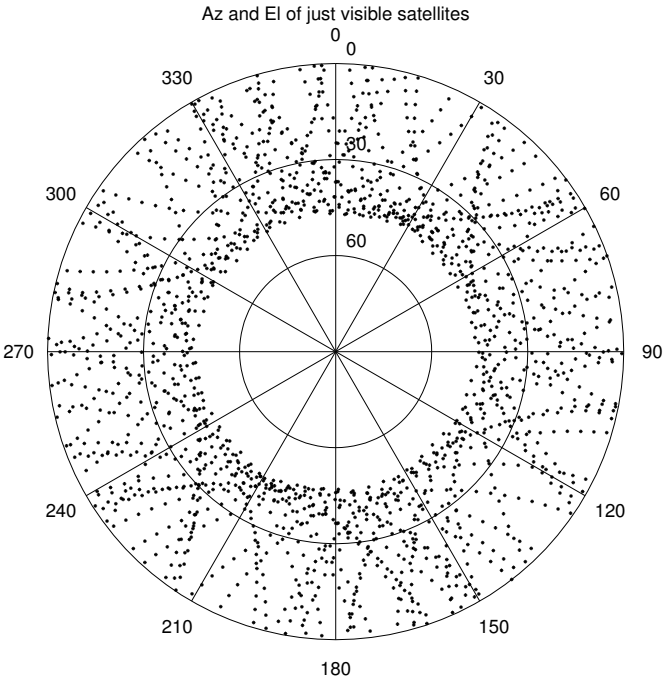

Figure 3: Az El for visible satellites at 9N 0E

This plot (at the north pole) features a large spot in the center of the field where no satellite ground tracks go. This makes sense because, as described before, the GPS satellites are in non-polar orbits. So their Az El paths should never traverse over the pole. They all come within a certain radius, based on the orbit inclinations

5.3 b) 90 N, 0 E

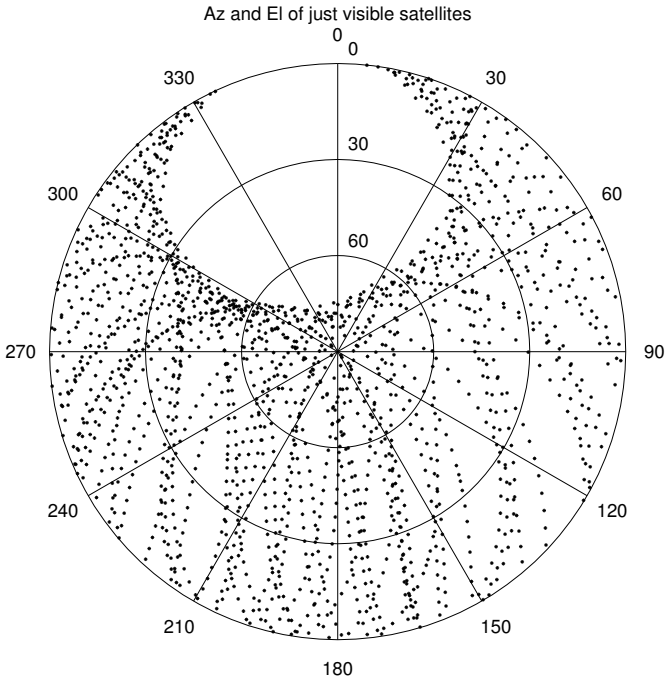

Figure 4: Az El for visible satellites at Boulder

The track over Boulder shows that we can see the "hole" over the North Pole where satellite tracks avoid. This is enlightening, because it shows that there are certain areas in our sky that we would not want to point an antenna.

6 Problem 6 - Highest North Pole Elevation

To calculate the highest elevation visible for the satellites, I used some simple trig. The GPS specifications require a 55 degree inclination, and a 26500 Km orbit radius. From there, I was able to construct a trigonometric equation for finding the maximum possible elevation.

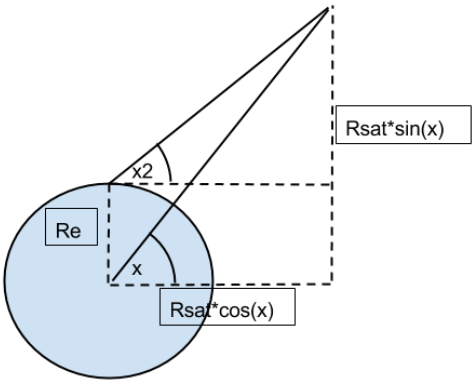

Figure 5: Visualization for Max Elevation Calculation

Using that drawing, I came up with an equation for the maximum angle allowable:

$$x_2 = \tan^{-1}\left(\frac{R_{orbit}\sin x - h}{R_{orbit} * \cos x}\right) \tag{1}$$

With that calculation, I found the maximum elevation angle to be: **45.27 degrees**.

7 Problem 7 - Visible Satellites over Boulder at 10 Degree Elevation Mask

Below is the plot of the visible satellites' track over Boulder with a mask of 10 degrees in elevation.

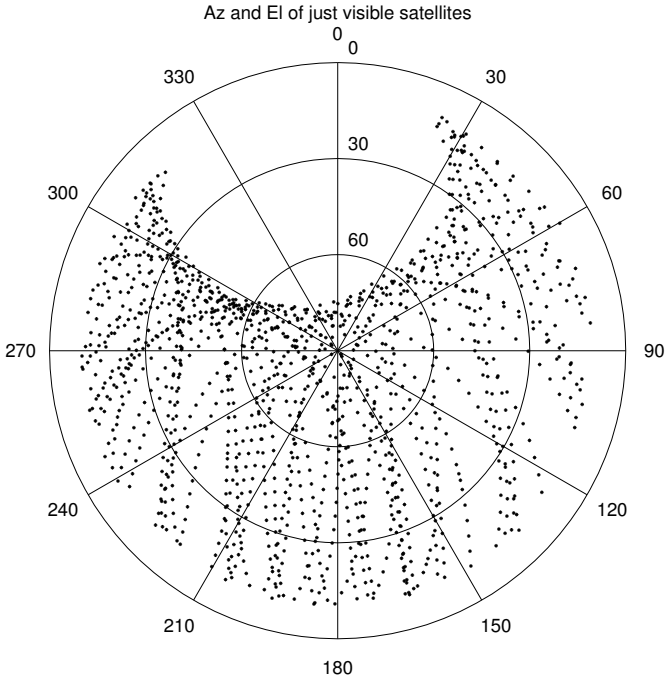

Figure 6: Az El for visible satellites at Boulder with Elevation Mask

8 Problem 8 - Repeatibility

To see how often the visibility correlates, I plotted the correlation between the number of visible sats on 9/20/2013 with that on 9/21/2013. This way, I could see what kind of offset occurs that shows repeatability in the availability pattern.

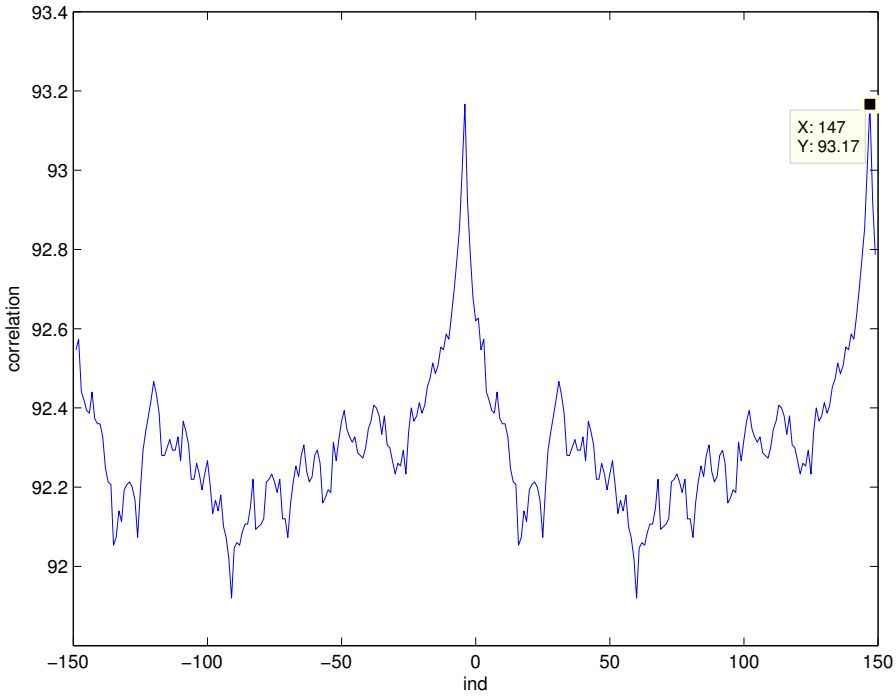

Figure 7: Visibility Correlation Pattern

From the plot, you can see that there are strong correlations between days at -4 and 147. As I went in 10 minute increments, this indicates that the pattern repeats roughly every 24.5 hours. The plot of the days themselves reinforce that conclusion.

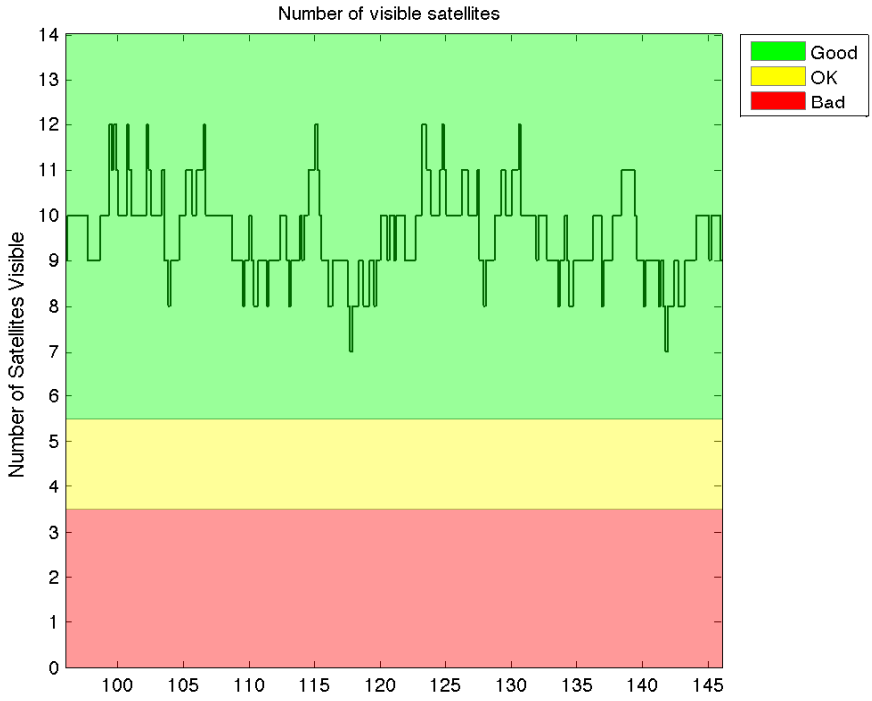

Figure 8: Visibility Plot

9 Conclusion/Recomendations

Overall, this assignment was pretty interesting. I thought the code provided was well written and easy enough to follow. It shows how simple codesets can yield powerful results. The GPS information and dataset is very accessible, and it is cool to see that first hand.

As far as things I didn't like about this assignment, and recommendations for the future are as follows.

- Some questions worth points ask you to simply modify code and verify that it works. Do you want proof? What kind? It is hard to know how you will apply points, and how I am to obtain them (since that is what this all)
- The most helpful thing would be a summarizing PDF of the data structures, and overall code flow. It is very difficult to discern what code suites do just by reading through them function-by-function. A single reference place would be very helpful.

- I like open source information. Especially in a learning environment. I personally think giving students protected **.p** files is completely counter productive. I don't pay ever increasing tuition to go to class, only to have information hidden from me. If there is some reason we cannot use code written for/by this university, the asssignment needs to be adjusted accordingly. I would rather write something and learn along the way than have a black box I have no clue how to change and interact with.

- Describe things in a more real-world way. Not with just aerospace terminology that only means something to the code maintainer. Not the first time user.

```
%=========================================================================
% Simplified driver for GPS Visbility Codes
%
% Based on GPSVisibility_GUI by Ben K. Bradley and calling functions used
% in that GUI.
% P. Axelrad 9/12
%
% Uses:
% generate_GPSyuma_name
% download_GPSyuma
% utc2gpsvec
%=========================================================================
%=========================================================================

clear, close all

% Enter the time of interest
UTC = [2012 9 20 0 0 0];
GPSvec = utc2gpsvec(UTC);


% Construct YUMA almanac file name since this is default setting
navfilename = generate_GPSyuma_name(GPSvec);
[navfilename,statusflag] = download_GPSyuma(GPSvec);


% Grab duration and time step ===============================================
durationhrs = 24;

dt_sec = 600;


% Input Antenna Location ====================================================
% North Pole
% latgd = 90;  % latitude, deg
% lon   = 0;  % longitude, deg
% alt   = 0;  % altitude, m

% Equator
% latgd = 0;  % latitude, deg
% lon   = 0;  % longitude, deg
% alt   = 0;  % altitude, m


latgd = 40.0;  % latitude, deg
lon   = -105.0;  % longitude, deg
alt   = 1631.0;  % altitude, m
% % Make antenna pointed straight up
ant_enu = [0 0 1];
```

```
% Set minimum and maximum mask angles
mask_min = 10;  % deg
mask_max = 90; % deg

[time_wntow,GPSdata] = ASEN5090_GPSvis(navfilename, 1, GPSvec,...
    durationhrs, dt_sec, latgd, lon, alt,...
    mask_min, mask_max, mask_min, ant_enu, 0, []);
hrofweek = time_wntow(:,2)/3600;

% Plot results ==========================================================
% ======================================================================


% Number of Satellites Visible ---------------------------------
[ax2] = plot_GPSnumsats(hrofweek,GPSdata.ant_numsats);

title(ax2, 'Number of visible satellites')

% Topocentric: AzEl Plot ---------------------------------------------------
[rows,cols] = size(GPSdata.topo_el);

az_vec     = reshape(GPSdata.topo_az,rows*cols,1);
el_vec     = reshape(GPSdata.topo_el,rows*cols,1);
GPSdata.prn = repmat([1:32],rows,1);


% plot the sats visible to antenna only
%fig3 = figure; ax3 = axes;
figure
%plotAzEl(GPSdata.topo_az',GPSdata.topo_el',GPSdata.prn')
plotAzEl(GPSdata.topo_az',GPSdata.topo_el',zeros(rows,cols))

figure
%plotAzEl(GPSdata.topo_az',GPSdata.topo_el',GPSdata.prn')
plotAzEl(GPSdata.topo_az(GPSdata.ant_mask)',GPSdata.topo_el(GPSdata.ant_mask)',zer
title('Az and El of just visible satellites')


% Antenna-centric: Elevation Plot ------------------------------------------

time_mat = repmat(hrofweek,1,cols);
time_vec = reshape(time_mat,rows*cols,1);

fig4 = figure; ax4 = axes;
plot(ax4,time_vec,el_vec,'ob','markerfacecolor','b','markersize',4);

ylabel('Elevation (deg)');
xlabel('Time (hr)');
```

```
grid(ax4,'on');


title(ax4,{'Elevation Angle';'of Satellites Seen by Antenna'});



function GPSdata = GPSMask(GPSdata, sz, topomaskmin,topomaskmax, antmask, ant_enu)

% Function written by Zach Dischner with help from Nick Truesdale to
% compute the GPS masking to tell how many satellites are visible to the
% antenna given a set of as and el masks

%% Compute Antenna Az, El vectors
azAnt = wrapTo360( atan2d( ant_enu(2), ant_enu(1) ) );
elAnt = asind(ant_enu(3)/norm(ant_enu));

%% Trig Calculations for antenna
SazAnt = sind(azAnt);
CazAnt = cosd(azAnt);

SelAnt = sind(90 - elAnt);
CelAnt = cosd(90 - elAnt);

%% Trig Calculations for satellite
Saz2 = sind(GPSdata.topo_az);
Caz2 = cosd(GPSdata.topo_az);

Sel2 = sind(90 - GPSdata.topo_el);
Cel2 = cosd(90 - GPSdata.topo_el);

%% Angle between sat and ant, used by dot product
alpha = acosd((CazAnt.*Caz2 + SazAnt.*Saz2).*SelAnt.*Sel2 + CelAnt.*Cel2);

%% Satellite visibility matrix
topo_numsats = zeros(sz,32);
ant_numsats = zeros(sz,32);

%% Create Topographical Mask
topo=GPSdata.topo_el;
topo(topo ~= topo) = 0;  % Take care of nans
topo(topo < topomaskmin) = 0;
topo(topo > topomaskmax) = 0;
topo(topo ~= 0) = 1;
GPSdata.topo_numsats = sum(topo,2);

%% Create Antenna Mask
```

```
c1 = GPSdata.topo_el > topomaskmin;
c2 = GPSdata.topo_el < topomaskmax;
c3 = alpha < 90 - antmask;              % Adjust for spherical Trig

% Apply conditions -- satellite is visible if all are true
ant_numsats( c1 & c2 & c3) = 1;

% Sum to get number of sats visible
GPSdata.ant_numsats  = sum(ant_numsats,2);
GPSdata.topo_mask = (c1 & c2);
GPSdata.ant_mask = (c1 & c2 & c3);


end %function

function [time_wntow,GPSdata,GLNSS,BOTH] = ASEN5090_GPSvis(navfilename,ephemtype,
    gpsvecstart,durationhrs,dt_sec,latgd,lon,alt,topomaskmin,topomaskmax,...
    antmask,ant_enu,junk1,junk2)

%=========================================================================
%=========================================================================
% [time_wntow,GPSdata,GLNSS,BOTH] = GPSvis(navfilename,ephemtype,...
%   gpsvecstart,durationhrs,dt_sec,latgd,lon,alt,topomaskmin,topomaskmax,...
%   antmask,ant_enu,glnss,tlefilename)
%
% Predicts the number of GPS and/or Glonass satellites that will be
%  visible for a specific observation site and antenna. Either YUMA
%  almanacs or broadcast ephemerides can be used to propagate GPS orbits.
%  Two Line Elements (TLE) sets are used to propagate Glonass satellites.
%  The default Glonass TLE filename that is used is Glonass.tle.
%
%
% Author: Ben K. Bradley
% date: 02/20/2011
% Modified to remove calculations for ASEN 5090 Class 9/12
%
%
% INPUT:              Description
Units
%
%  navfilename   - filename of IGS broadcast ephemeris file to use
string
%  ephemtype     - ephemeris type of navfilename:       1=YUMA, 2=Broadcast
%  gpsvecstart   - GPS date/time vector to start analysis     [y m d h m s]
%  durationhrs   - duration of analysis
hours
%  dt_sec        - time step
seconds
%  lat_gd        - geodetic latitude of antenna site           [-90,90] deg
```

```
 %  lon           - longitude of antenna site                  [-180,180] deg
 %  alt           - WGS84 ellipsoidal height (altitude) of antenna
meters
 %  topomaskmin   - topographic minimum elevation (wrt horizon)
deg
 %  topomaskmax   - topographic maximum elevation (wrt horizon)
deg
 %  antmask       - antenna elevation mask (wrt antenna)
deg
 %  ant_enu       - boresight direction of antenna in east-north-up unit vector [E
 %  glnss         - boolean:  1=include Glonass, 0=don't include Glonass
 %  tlefilename   - filename of Glonass TLE file to use
 %
 %
 % OUTPUT:
 %
 %  NOTE: for outputs with 32 columns, the column number corresponds to
 %         the PRN number of the GPS satellite (i.e. each row is a specific
 %         time and each column is a specific satellite)
 %
 %  time_wntow    - week number and tow for all computations  [WN TOW] (nx2)
 %  GPSdata       - structure of GPS results. structure is below
 %  GLNSS         - structure of Glonass results. structure is below
 %  BOTH          - structure of GPS/Glonass combined results
 %
 %
 %       topo_numsats: number of satellites above topomask
(nx1)
 %            topo_az: topocentric azimuth of satellites
(nx32)
 %            topo_el: topocentric elevation of satellites
(nx32)
 %                vis: flag for satellite visible (nx32)
 %        ant_numsats: number of satellites above antenna mask
(nx1)
 %             ant_el: satellite elevation wrt antenna
(nx32)
 %                DOP: structure containing DOPs:  .GDOP  .HDOP
each (nx1)
 %                                                 .VDOP  .PDOP
 %                                                 .TDOP
 %
 %
 % Coupling:
 %
 %  lla2ecef         read_GPSbroadcast
 %  gpsvec2gpstow    broadcast2xva
 %  ecef2azelrange2  sez2ecef
 %  gpsvec2utc       utc2utc
```

```
%  init_EOP          get_EOP
%  alm2xv            read_TLE
%  tle2rv            teme2ecef
%
% References:
%
%   none
%
%==========================================================================
%==========================================================================



% Compute ECEF Position of Antenna Location ================================
r_site = lla2ecef(latgd, lon, alt*0.001);

r_site = r_site' * 1000;  % [x y z] meters


% Compute boresight direction of antenna in ECEF ===========================
ant_ecef = sez2ecef(latgd,lon, [-ant_enu(2) ant_enu(1) ant_enu(3)]);



% Open GPS Ephemeris File and create ephemeris matrix ======================

if (ephemtype == 1) % YUMA
    ephem_all = read_GPSyuma(navfilename);
elseif (ephemtype == 2) % Broadcast
    ephem_all = read_GPSbroadcast(navfilename);
end



% Create GPS time series ===================================================

% Convert GPS date/time start to week number and time of week
[WN0, TOW0] = gpsvec2gpstow( gpsvecstart );

    % WN0  = week number count from 22Aug99
    % TOW0 = time of week, seconds


TOWseries = (TOW0: dt_sec : TOW0+durationhrs*3600)';
elapsed   = TOWseries - TOW0;

sz = length(TOWseries);

time_wntow = [WN0*ones(sz,1)   TOWseries];
```

```
    % time_wntow = [WN  TOW]  % Week numbers and Time of Weeks


% Initialize Variables ================================================

% GPS ------------------------------------------------
prnlist = 1:32;

Xpos = zeros(sz,32) * NaN;
Ypos = Xpos;
Zpos = Xpos;

GPSdata = struct;

GPSdata.topo_numsats = zeros(sz,1);
GPSdata.topo_az      = zeros(sz,32) * NaN;
GPSdata.topo_el      = zeros(sz,32) * NaN;
GPSdata.ant_numsats  = zeros(sz,1);
GPSdata.ant_el       = zeros(sz,32) * NaN;
GPSdata.DOP.HDOP     = zeros(sz,1)  * NaN;
GPSdata.DOP.VDOP     = zeros(sz,1)  * NaN;
GPSdata.DOP.PDOP     = zeros(sz,1)  * NaN;
GPSdata.DOP.TDOP     = zeros(sz,1)  * NaN;
GPSdata.DOP.GDOP     = zeros(sz,1)  * NaN;
% ----------------------------------------------------


GLNSS = [];
BOTH  = [];



% Compute GPS Visibility ===============================================
% =====================================================================

for nn = prnlist  % Loop through satellite list ------------------------

    if (ephemtype == 1)     % YUMA
        [health,state] = alm2xv(ephem_all,time_wntow,nn);
    elseif (ephemtype == 2) % Broadcast
        [health,state] = broadcast2xva(ephem_all,time_wntow,nn);
    end

    Xpos(:,nn) = state(:,1);  % column number = PRN number
    Ypos(:,nn) = state(:,2);  % meters ECEF
    Zpos(:,nn) = state(:,3);
```

```
    for tt = 1:sz % loop through all times ————————————————————————

        if (health(tt) ~= 0)   % satellite is unhealthy

            Xpos(tt,nn) = NaN; % get rid of unhealthy states
            Ypos(tt,nn) = NaN;
            Zpos(tt,nn) = NaN;

        else

            % Compute topocentric azimuth and elevation
            [GPSdata.topo_az(tt,nn),GPSdata.topo_el(tt,nn)] = ASEN5090_ecef2azelra

            % az,el = degrees
            % If the satellite is not visible, set the az and el to NaNs or zero h


        end % END of satellite health


    end % END of time loop ————————————————————————————————————————


end % END of PRN loop ————————————————————————————————————————————————




% PUT IN YOUR CALCULATIONS FOR VISIBLE SATELLITES HERE:

% ========================================================================
mysize = size(GPSdata.topo_el);
GPSdata = GPSMask(GPSdata,mysize(1), topomaskmin, topomaskmax, antmask, ant_enu);
%========================================================================
%========================================================================







function [az,el,range] = ASEN5090_ecef2azelrange(r_sat,r_site,latgd,lon)
%========================================================================
% [az,el,range] = ecef2azelrange(r_sat,r_site,latgd,lon)
```

```
 %
 % Calculates the azimuth, elevation, and range of a satellite with respect
 %  to an observation site.
 %
 % Author: Ben K. Bradley
 % Date: 11/15/2010
 % Homework changes made by Zach Dischner and Nick Truesdale, 9/24/13
 %
 % INPUT:          Description
Units
 %  r_sat       - position of satellite in ECEF frame
[x y z]
 %  r_site      - position of observing site in ECEF frame
[x y z]
 %  latgd       - geodetic latitude of observation site          [-90,90] deg
 %  lon         - longitude of observation site     [-180,180] or [0,360] deg
 %
 % OUTPUT:
 %  az          - azimuth (degrees clockwise from North)
[0,360] deg
 %  el          - elevation (degrees up from horizon)            [-90,90] deg
 %  range       - distance from observation site to satellite
 %=========================================================================


 % Range vector, ECEF
 rhoECF = r_sat - r_site;

 % Convert to SEZ
 rhoSEZ = rotmat(2, pi/2 - latgd*pi/180)*rotmat(3, lon*180/pi)*rhoECF';

 % Range
 range = norm(rhoSEZ);

 % Azimuth
 az = wrapTo360( atan2d(rhoSEZ(2), -rhoSEZ(1)) );

 % Elevation
 el = asind(rhoSEZ(3)/range);


 end %function

 function B = rotmat(dim, theta, A)
 %
 % File: rotation_matrix.m
 % Author: Nick Truesdale, my partner on this homework assignment. This is a
 % helpful rotation matrix function for use.
 % Date: 9/21/2012
```

```
%
% Description: This function calculates a rotation matrix in either two or
% three dimensions. DIM = 0 yields a 2D matrix, while DIM = 1,2 or 3 yields
% one of the three 3D matrices. THETA is the angle of rotation. If A is
% given as a vector or array, B will be the rotated vector/array. If A is
% unspecified, B will be the rotation matrix.
%
% Note: The size of A must match the given dimension. If DIM = 0, A must be
% a 2x2 or 2x1 matrix. If DIM = 1,2,3, A must be a 3x3 or 3x1 matrix.

% Parse inputs
mat_dim = logical(dim) + 2;
if(nargin < 3), A = []; end

% Sine and cosine
S = sin(theta);
C = cos(theta);

% Calculate matrix
switch dim
    case 0
        B = [C, S;
            -S, C];
    case 1
        B = [1  0 0;
            0  C S;
            0 -S C];
    case 2
        B = [C 0 -S;
            0 1  0;
            S 0  C];
    case 3
        B = [C  S  0;
            -S  C  0;
            0  0  1];
    otherwise
        error('Dim must be 0,1,2 or 3, denoting a 2D, X, Y or Z rotation')
end

% If A is supplied
if(~isempty(A))
    if(isequal(size(A), [mat_dim, mat_dim]) || isequal(size(A), [mat_dim, 1]))
        B = B*A;
    else
        error(['A must be a column vector or square matrix with dimension ',...
            '2 for DIM = 0, or dimension 3 for DIM = 1,2,3'])
    end
end
end
```