```
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
            % CLOUSE: just NaN stuff with elevations below the min-mask
            GPSdata.topo_az(GPSdata.topo_el < topomaskmin) = NaN;
            GPSdata.topo_el(GPSdata.topo_el < topomaskmin) = NaN;

        end % END of satellite health


    end % END of time loop ----------------------------------------


end % END of PRN loop ----------------------------------------------------




% PUT IN YOUR CALCULATIONS FOR VISIBLE SATELLITES HERE:

% ========================================================================
for tt = 1:sz

    % GPS ------------------------------------------------------------

    % Number of Visible GPS Satellites
    % ----------------------------------------------------------------
    % Put in logic here to calculate the number visible
    % GPSdata.ant_numsats(tt) = ??;
    GPSdata.ant_numsats(tt) = sum(GPSdata.topo_el(tt, :)>topomaskmin);
```

```
end
%=========================================================================
```

CLOUSE: sort the planes the PRNs are on,

```
planes = []; planes2prns = zeros(32,2); tol = 0.1; for prn=1:31 RAAN = ephem_all(prn,6); plane_found = 0; for plane = 1:length(planes) if planes(plane)-tol < RAAN < planes(plane)+tol plane_found = 1; planes2prns(ephem_all(prn,1),1)=RAAN*180/pi; planes2prns(ephem_all(prn,1),2)=ephem_all(prn,1); break end end if plane_found continue end planes = [planes RAAN]; planes2prns(ephem_all(prn,1),1)=RAAN*180/pi; planes2prns(ephem_all(prn,1),2)=ephem_all(prn,1); end planes2prns sortrows(planes2prns,1)

%=========================================================================
```

Published with MATLAB® R2013b