GPS Satellite Visibility

John Clouse*
University of Colorado, Boulder, CO 80309

*Student, Aerospace Engineering Department.

I. Summary

GPS satellite visiblility was calculated from ephemeris data and compared to the results of a proprietary program. Coverage over the course of a day was computed for the locations of the North Pole, the equator, and Boulder, CO; "holes" where satellites never passed through were explained. Satellites were found to take one sidereal day to be visible at the same point in the sky since a given time. GPS satellite visibility was found to be dependent on the latitude of the observation site.

II. The YUMA Almanac

The YUMA almanac filename for the September 20, 2014 is yuma0787.503808.alm from CelesTrak. Each PRN's orbit plane can be determined by its right ascension of the ascending node.Table 1 below shows the PRNs in each GPS plane, and was found to be in agreement with the almanac from GPSWorld:[1]

PRNs $i \approx -150°$ (Plane E)	PRNs $i \approx -90°$ (Plane F)	PRNs $i \approx -30°$ (Plane A)	PRNs $i \approx 30°$ (Plane B)	PRNs $i \approx 90°$ (Plane C)	PRNs $i \approx 150°$ (Plane D)
20	15	24	25	29	11 (actually $\sim 130°$)
18	9	7	12	17	2
22	23	31	16	29	4
5	14	30	28	19	21
10	26	8	–	–	6
32	13	–	–	–	1

Table 1: PRN Planes

III. Satellite Azimuth and Elevation Computation

A Matlab function was written to take propagated ephemeris data in cartesian coordinates and output the resulting azimuth/elevation data for a point on the earth. A masking capability was also created to disregard satellites below a given elevation. The results were compared with ASTER Labs' Sidera software to verify the correct functionality. Figure 1 shows the resulting plots side by side.

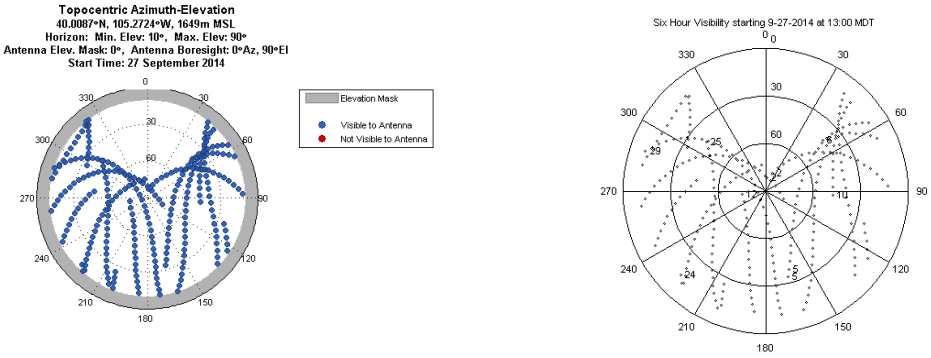

Figure 1: Left to right: Sidera output, computed output.

Satellites visible on September 27 at 1PM MDT were calculated from the YUMA almanac; the results are in Table 2. Receiver data is shown in Figure 2, qualitatively verifying the prediction.

PRN	Azimuth	Elevation
2	6.15°	81.65°
5	165.20°	33.88°
6	48.57°	41.72°
10	93.49°	47.20°
12	−97.24°	75.14°
17	95.80°	8.09°
24	−134.77°	14.42°
25	−50.16°	41.43°
29	−71.73°	10.05°

Table 2: PRN Azimuth/Elevation on September 27, 2014 at 1PM MDT

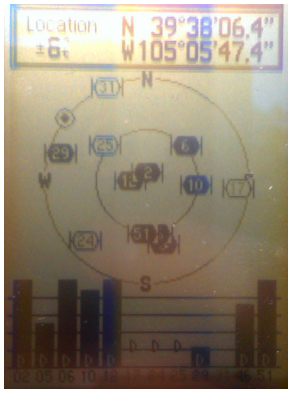

Figure 2: Satellites visible to receiver on September 27, 2014 at 1PM MDT.

IV. GPS Satellite Visibility

To see daily coverage of multiple points on the earth, the ephemerides were propagated for a day. Locations were chosen to be at the North pole, the equator, and Boulder, CO. The azimuth/elevation plots are shows in Figure 3.

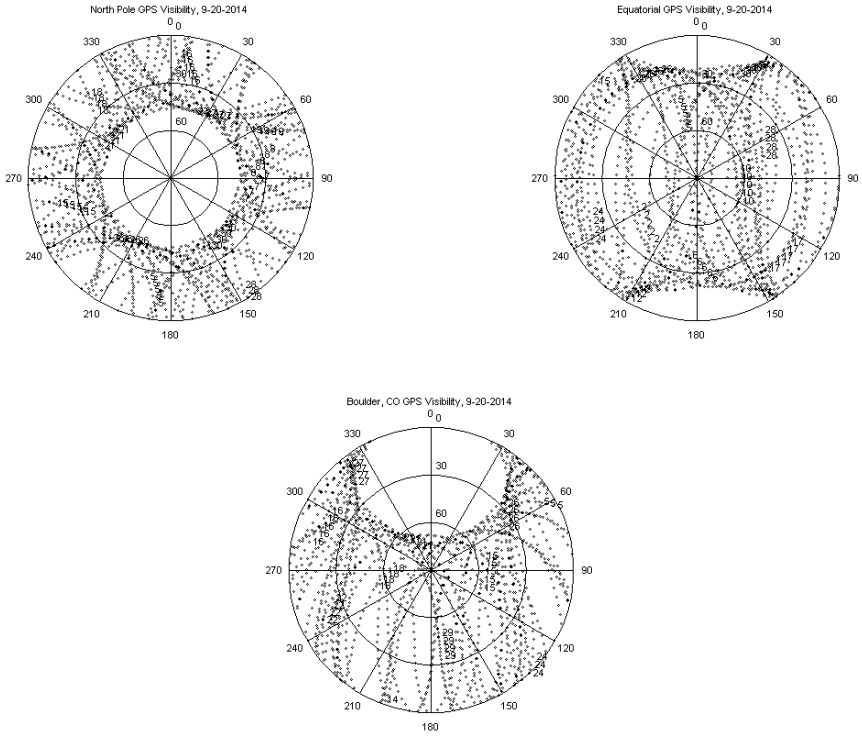

Figure 3: GPS satellite visibility from different points on the earth.

The equatorial plot shows two holes: one to the north and one to the south. This result makes sense, as the GPS satellites have inclinations of approximately $55°$. That limits the elevation to the north and the south that satellites can be seen. Boulder, 40 degrees north of the equator, shows a hole to the north for the same reason; there is no hole in visibility to the south due to the latitude of the location and the satellite inclination. At the North pole, the hole is directly above the point. This hole again is a result of the satellite inclinations, which are not polar. The maximum elevation for the North pole case is demonstrated in Figure 4.

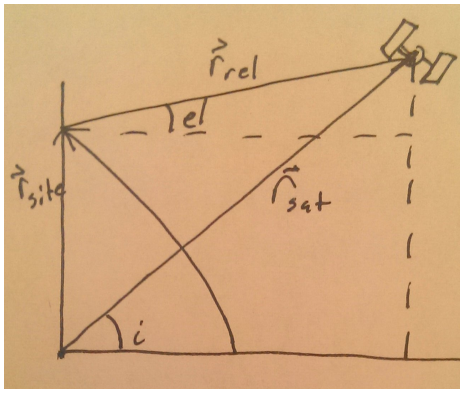

Figure 4: Planar view of satellite at its highest latitude being viewed from the North Pole.

The maximum elevation be roughly analytically calculated as

$$el = \arctan\left(\frac{r_{sat}\sin i - r_{Earth}}{r_{sat}\cos i}\right) \tag{1}$$

where $r_{sat} = 26,600$km, $i = 55°$, and $r_{Earth} = 6378$km. The result of Equation 2 is a maximum elevation of approximately $45°$.

The number of satellites visible from Boulder on September 27, 2014 are shown in Figure 5. The results are shown for 24 hours.

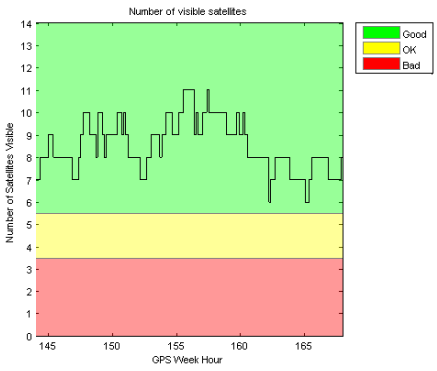

Figure 5: Number of visible satellites from Boulder, CO over 24 hours.

For a satellite to be visible in the same point in the sky on two consecutive days, the satellite period multiplied by an integer number of orbits must be a sidereal day (86164 seconds).[2] The orbit plane is essentially inertially fixed over the course of two days, and the change in other orbital elements are assumed to be negligible except for the true anomaly angle (although it must be the same as it previously was). To determine how many orbits a satellite must complete to satisfy this requirement,

$$n = \text{sidereal}/P \tag{2}$$

where n must be an integer. It takes two orbits for GPS satellites to be in the same position in the sky one sidereal day later. This is simulation results are shown in Table 3 for selected PRNs.

PRN	ΔAzimuth	ΔElevation
2	$0.31°$	$-0.06°$
5	$-0.02°$	$0.07°$
6	$0.03°$	$-0.10°$

Table 3: PRN Azimuth/Elevation differences between 9/27/2014 19:00 UTC and 9/28/2014 18:56:4 UTC

V. Conclusions and Recommendations

GPS satellite visibility for an observation site is latitude-dependent due to the satellite inclination. The number of satellites and the spacing of the constellation's orbit planes ensures multiple satellites should be visible from any point on Earth (notwithstanding topography). Each satellite was found to also occupy the same area in the sky one sidereal day later. Visibility computations were qualitatively compared to proprietary code and looked accurate. Unfortunately there was no known way to get computed satellite azimuth/elevation data from the receiver itself, so the comparisons had to be eyeballed (a potential error source in higher-fidelity research). Future investigations should look into the satellite-visibility geometry, based on computed visibility, and its connection to the accuracy of a receiver's measurement.

References

[1]"GPS World Almanac, August 2014," `http://gpsworld.com/wp-content/uploads/2014/07/GPSWorld_Almanac_August2014.pdf`, Accessed: 2014-09-28.

[2]Misra, P. and Enge, P., *Global Positioning System Signals, Measurements, and Performance*, Ganga-Jumana Press, Lincoln, MA, revised second ed., 2012.

Appendix: Matlab Code

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

%==========================================================================
%==========================================================================

clear, close all

% Enter the time of interest
% UTC = [2014 9 27 19 0 0];
UTC = [2014 9 28 18 56 4];
% UTC = [2014 9 20 19 0 0];
GPSvec = utc2gpsvec(UTC);  % This will adjust for leap second offset
[WN2, TOW, WN1] = gpsvec2gpstow(GPSvec) ; %WN1 is full week, WN2 is mod1024
check = gpstow2gpsvec(WN2,TOW,2); % See if it converts back correctly, 2 for WN2


% 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 =====================================================
% latgd = 39+38/60+07.340/3600;%90.0;  % latitude, deg
% lon   = -105+05/60+47.552/3600;%-105.0;  % longitude, deg
latgd = 40;  % latitude, deg
lon   = -105;  % longitude, deg
alt   = 1600.0;  % 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(1,:)',GPSdata.topo_el(1,:)',GPSdata.prn(1,:)')



% 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,{'\fontsize{11}\bfElevation Angle';'\bfof Satellites Seen by Antenna'});
```

CLOUSE: Load 9/27 data and subtract to find the satellite az/el diff

```
GPSdata_new=GPSdata;
load('GPS927.mat')
%Display the az/el diffs
GPSdata_new.prn(1,GPSdata_new.topo_el(1,:)>0)
GPSdata_new.topo_az(1,GPSdata_new.topo_el(1,:)>0)-GPSdata.topo_az(1,GPSdata_new.to
GPSdata_new.topo_el(1,GPSdata_new.topo_el(1,:)>0)-GPSdata.topo_el(1,GPSdata_new.to
```

Published with MATLAB® R2013b

```
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

```
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
% Modified to remove calculations for ASEN5090 assignments
%
% 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
%
%
% Coupling:
%
%  none
%
%
%=========================================================================
%=========================================================================

% Satellite pos rel to site
r_site2sat_ecef = r_sat - r_site;
% sines and cosines used later
sinp = sind(latgd);
cosp = cosd(latgd);
sinl = sind(lon);
cosl = cosd(lon);
% Rotation from ECEF to ENU
R_ecef2local = ...
    [-sinl cosl 0;
    -sinp*cosl -sinp*sinl cosp;
    cosp*cosl cosp*sinl sinp];

% Rotate the rel pos into ENU
r_site2sat_enu = r_site2sat_ecef* R_ecef2local';
```

```
% ENU coords give you az/el/range
az = atan2(r_site2sat_enu(1), r_site2sat_enu(2))*180/pi;
range = norm(r_site2sat_enu);
el = asin(r_site2sat_enu(3)/range)*180/pi;
end
```

Published with MATLAB® R2013b

```
%*******************************************************
% function hpol = plotAzEl(az,el,svs,varargin)
%
% DESCRIPTION:
%
%  Creates an az-el plot of satellites
%
% ARGUMENTS:
%
%  az - vector of azimuth angles, in degrees
%  el - vector of elevation angles, in degrees
%  svs - vector of satellite PRN numbers
%    NOTE: To avoid printing PRN numbers on the plot, make 'svs' a vector
%    of zeros.
%  varargin - axes handle for plot on previous
%
% OUTPUT:
%
%  hpol - handle to polar plot axes
%
% CALLED BY:
%
%  createAzElMap
%
% FUNCTIONS CALLED:
%
%  None
%
% MODIFICATIONS:
%
%             ??  :  P. Axelrad - Original
%       02-05-02  :  Lisa Reeh
%       05-17-04  :  Stephen Russell - minor modifications to allow plot
%           overlaying using new code (i.e. varargin with axes handle)
%
%
% Colorado Center for Astrodynamics Research
% Copyright 2004 University of Colorado, Boulder
%*******************************************************
function hpol = plotAzEl(az,el,svs,varargin)

line_style = 'auto';

if nargin < 1
 error('Requires 3 input arguments.')
end

if isstr(az) | isstr(el)
 error('Input arguments must be numeric.');
end
if any(size(az) ~= size(el))
 error('AZ and EL must be the same size.');
```

```
end

% get hold state
if(nargin > 3)
    axes(varargin{1});
end
cax = newplot;
next = lower(get(cax,'NextPlot'));
hold_state = ishold;

% get x-axis text color so grid is in same color
tc = get(cax,'xcolor');

% Hold on to current Text defaults, reset them to the
% Axes' font attributes so tick marks use them.
fAngle  = get(cax, 'DefaultTextFontAngle');
fName   = get(cax, 'DefaultTextFontName');
fSize   = get(cax, 'DefaultTextFontSize');
fWeight = get(cax, 'DefaultTextFontWeight');
set(cax, 'DefaultTextFontAngle',  get(cax, 'FontAngle'), ...
 'DefaultTextFontName',   get(cax, 'FontName'), ...
 'DefaultTextFontSize',   get(cax, 'FontSize'), ...
 'DefaultTextFontWeight', get(cax, 'FontWeight') )

% only do grids if hold is off
if ~hold_state

   % make a radial grid
 hold on;
 hhh=plot([0 2*pi],[0 90],'-','linewidth',0.5);
 v = [get(cax,'xlim') get(cax,'ylim')];
 ticks = length(get(cax,'ytick'));
 delete(hhh);

   % check radial limits and ticks
 rmin = 0; rmax = v(4); rticks = ticks-1;

 if rticks > 5   % see if we can reduce the number
  if rem(rticks,2) == 0
   rticks = rticks/2;
  elseif rem(rticks,3) == 0
   rticks = rticks/3;
  end
 end

   % define a circle
 th = 0:pi/50:2*pi;
 xunit = cos(th);
 yunit = sin(th);

   % now really force points on x/y axes to lie on them exactly
   inds = [1:(length(th)-1)/4:length(th)];
   xunits(inds(2:2:4)) = zeros(2,1);
   yunits(inds(1:2:5)) = zeros(3,1);
```

```
 rinc = (rmax-rmin)/rticks;
 for i=(rmin+rinc):rinc:rmax
  plot(yunit*i,xunit*i,'-','color',tc,'linewidth',0.5);
  text(0,i+rinc/20,['   ' num2str(90-i)],'verticalalignment','bottom' );
 end

    % plot spokes
 th = (1:6)*2*pi/12;
 cst = cos(th); snt = sin(th);
 cs = [cst; -cst];
 sn = [snt; -snt];
 plot(rmax*sn,rmax*cs,'-','color',tc,'linewidth',0.5);

    % annotate spokes in degrees
 rt = 1.1*rmax;
 for i = 1:max(size(th))
  text(rt*snt(i),rt*cst(i),int2str(i*30),'horizontalalignment','center' );
  if i == max(size(th))
   loc = int2str(0);
  else
   loc = int2str(180+i*30);
  end
  text(-rt*snt(i),-rt*cst(i),loc,'horizontalalignment','center' );
 end

    % set viewto 2-D
 view(0,90);
    % set axis limits
 axis(rmax*[-1 1 -1.1 1.1]);
end

% Reset defaults.
set(cax, 'DefaultTextFontAngle', fAngle , ...
 'DefaultTextFontName',   fName , ...
 'DefaultTextFontSize',   fSize, ...
 'DefaultTextFontWeight', fWeight );

set(gcf, 'color', 'white');

% transform data to Cartesian coordinates.
yy = (90-el).*cos(az*pi/180);
xx = (90-el).*sin(az*pi/180);

% plot data on top of grid
q = plot(xx,yy,'ok','MarkerSize',2);

% Place satellite PRN numbers with satellite position
for i = 1:length(svs)
    if(svs(i)~=0)
        text(xx(i)+3,yy(i),int2str(svs(i)));
    end
end
```

```
if nargout > 0
 eval(['hpol = gca;']);
end

if ~hold_state
 axis('equal');axis('off');
end

% set hold state
if ~hold_state
    hold on;
end
```

Published with MATLAB® R2013b