```
%Attached is a compilation of all code used for this exam into one
 large
%script for the sake of brevity. Only modified code is included,
 unaltered
%code was used to plot (PLotAzEl)
clear all;close all;clc
c   = 2.99792458e8;    % GPS acceptd speed of light, m/s
%Create equator and near north pole user positions
EQUAECEF = 1000*[6380 0 0];
POLEECEF = 1000*[0 0 6380];
userpos = [EQUAECEF; POLEECEF];

%Create stand-in almanac entry to be used for exam
[gps_ephem,gps_ephem_cell] = read_GPSyuma('YUMA245.ALM',2);
ephem = gps_ephem(1,:);
ephem(4) = 0; %set eccen. to 0 for circ orbit
ephem(5) = sqrt(26560000); %Set square root of semi major axis as
 square root of circ. orbit radius
ephem(6) = 0;
ephem(7) = deg2rad(55); %Set inc, as exactly 55 degress
ephem(9) = 0; %Change rate of change or right ascension to 0 to
 account for assumption Eearth is not rotating.

% tvec = gps_ephem_cell{1,1}.Toe:60:gps_ephem_cell{1,1}.Toe+24*3600;
tvec = 0:30:86400;
tvec = tvec + 2*86400;

[health,pos] = broadcast2pos_alt(ephem,[2121*ones(length(tvec),1)
 tvec'],1);

for j = 1:2
    for i = 1:size(pos,1)
        [az(i,j),el(i,j),range(i,j)] =
 compute_azelrange_alt(userpos(j,:),pos(i,:));
    end
end
    figure
    for j = 1:size(pos,1)
        if el(j,1)>0
            plotAzEl(az(j,1),el(j,1),0)
        end
    end
    title('Equator Observer Sky Plot')
    figure
    for j = 1:size(pos,1)
        if el(j,2)>0
            plotAzEl(az(j,2),el(j,2),0)
        end
    end
    title('North Pole Observer Sky Plot')

%Determine min range vals.
```

```
EQ = 26560-6380;
temp = max(el(:,2));
temp = find(el(:,2) == temp);
Pole = range(temp,2);
%range at beginning of pass found when elevation first goes positive
EQ_passrange = range(735,1);
Pole_passrange = range(1107,2);
%TOF of range computed in part c
EQ_time = EQ_passrange*(1/c);
Pole_time = Pole_passrange*(1/c);
%max elevation for both observer positions
maxel = max(el);
%approximate duration of satellite pass
EQ_passtime = (tvec(1341) - tvec(735))/3600;
Pole_passtime = (tvec(1688) - tvec(1107))/3600;
%calculating max and min doppler shifts for obeserver positions
%minimum doppler for L1 for both positions is 0 because the range rate
 will
%always change from positive to negative at some point for both
 positions,
%nad this indicates a change from positive to negative doppler shift,
 so
%the minimum must be 0
ft = 1575.42;
rdotEQ = diff(range(:,1));
rdotPole = diff(range(:,2));
maxfdEQ = (-max(rdotEQ)/c)*ft;
maxfdPole = (-max(rdotPole)/c)*ft;
figure
for i = 1:numel(tvec)
    if el(i,1)
    plot(tvec-2*86400,el(:,1))
    end
end

function out = compute_LOS_ENU_alt(userECEF,satECEF)
vec = satECEF - userECEF;
lla_vec = ecef2lla(userECEF,0,6380000);
ECEF2ENU = calcECEF2ENU(lla_vec(1),lla_vec(2));
vec = ECEF2ENU*vec';
out = vec;
end

function [az,el,range] = compute_azelrange_alt(userECEF,satECEF)
LOS_ENU = compute_LOS_ENU_alt(userECEF,satECEF);
az = atan2d(LOS_ENU(1),LOS_ENU(2));
el = asind(LOS_ENU(3)/norm(LOS_ENU));
% range = LOS_ENU(3);
range = norm(satECEF - userECEF);
% out = [az el range];
end

function [health,x] = broadcast2pos_alt(ephem_all,t_input,prn)
```

```
%==========================================================================
%==========================================================================
% [health,x] = broadcast2pos(ephem_all,t_input,prn)
%
% Calculates the position from an ephemeris
%  matrix (see read_GPSbroadcast.m).  The input ephem_all can
%  be generated by the read_GPSbroadcast.m function.
%
%
% Modified by P. Axelrad 9/10/2018 to remove extra functionality
% Author: Ben K. Bradley
% Date: 07/19/2009
%
%
% INPUT:               Description
 Units
%
%  ephem_all    - matrix of gps satellite orbit parameters
 (nx25)
%
%                  col1: prn, PRN number of satellite
%                  col2: M0, mean anomaly at reference time, rad
%                  col3: delta_n, mean motion difference from computed
 value, rad/s
%                  col4: ecc, eccentricity of orbit
%                  col5: sqrt_a, square root of semi-major axis, m^0.5
%                  col6: Loa, longitude of ascending node of orbit
 plane at weekly epoch, rad
%                  col7: incl, inclination angle at reference time,
 rad
%                  col8: perigee, argument of perigee, rad
%                  col9: ra_rate, rate of change of right ascension,
 rad/s
%                 col10: i_rate, rate of change of inclination angle,
 rad/s
%                 col11: Cuc, amplitude of the cosine harmonic
 correction term to the argument of latitude
%                 col12: Cus, amplitude of the sine harmonic
 correction term to the argument of latitude
%                 col13: Crc, amplitude of the cosine harmonic
 correction term to the orbit radius
%                 col14: Crs, amplitude of the sine harmonic
 correction term to the orbit radius
%                 col15: Cic, amplitude of the cosine harmonic
 correction term to the angle of inclination
%                 col16: Cis, amplitude of the cosine harmonic
 correction term to the angle of inclination
%                 col17: Toe, reference time ephemeris (seconds into
 GPS week)
%                 col18: IODE, issue of data (ephemeris)
%                 col19: GPS_week, GPS Week Number (to go with Toe)
%                 col20: Toc, time of clock
%                 col21: Af0, satellite clock bias (sec)
%                 col22: Af1, satellite clock drift (sec/sec)
```

```
%                  col23: Af2, satellite clock drift rate (sec/sec/sec)
%                  col24: Timing Group Delay (TGD), seconds
%                  col25: health, satellite health (0=good and usable)
%
%  t_input      - GPS times to calculate values at                 [WN
 TOW] (nx2)
%  prn          - PRN to compute values for (one satellite only)
%
%
%
% OUTPUT:
%
%  health       - health of satellite (0=good)
      (nx1)
%  x            - position of satellite (ECEF)                  [x y
 z]   m (nx3)
%
%
%
% Coupling:
%
%   mean2eccentric.m
%
% References:
%
%   [1] Interface Control Document: IS-GPS-200D
%         < http://www.navcen.uscg.gov/gps/geninfo/IS-GPS-200D.pdf >
%
%   [2] Zhang, J., et.all. "GPS Satellite Velocity and Acceleration
%         Determination using the Broadcast Ephemeris". The Journal of
%         Navigation. (2006), 59, 293-305.
%            < http://journals.cambridge.org/action/
displayAbstract;jsess ...
%                 ionid=C6B8C16A69DD7C910989C661BAB15E07.tomcat1?
fromPage=online&aid=425362 >
%
%   [3] skyplot.cpp by the National Geodetic Survey
%          < http://www.ngs.noaa.gov/gps-toolbox/skyplot/skyplot.cpp >
%
%
% Last Updated:
%
%  2015/01/22  B.K. Bradley - the capability to look for updated ephem
%                              entries that occur at odd times within
 each
%                              2hr window has been commented out in
 this
%                              function and added to
 read_GPSbroadcast.m
%                              instead. This moves the computational
%                              overhead to the reading which only
 occurs
%                              once.
%
```

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


% NOTE: Numbered equations in the code (e.g., Eq. 21) correspond to
%  equations in the [2] reference.

%===========================================================================
% Load GPS Accepted WGS-84 Constants
%===========================================================================
muE = 3.986005e14;     % WGS-84 value, m^3/s^2
wE  = 0; % WGS-84 value, rad/s
c   = 2.99792458e8;    % GPS acceptd speed of light, m/s

%===========================================================================
% Initialize Output Variables for Speed
%===========================================================================
sz         = size(t_input,1);
x          = ones(sz,3) * NaN;
health     = ones(sz,1) * NaN;


%===========================================================================
% Pull Out Correct Ephemerides
%===========================================================================

% Pull out ephemerides for PRN in question
kk  = find(ephem_all(:,1) == prn);  % kk is vector containing row
 numbers of ephem_all that are for sat.no. 'index'
sat_ephem = ephem_all(kk,:);        % sat_ephem is matrix of all ephem
 data for each entry of sat.no. 'index'


% No matching PRN found, returning data will be NaNs
if isempty(kk),return,end




%===========================================================================
% Start Main Calculation Loop
%===========================================================================

% Compute elapsed times of each ephemeris epoch wrt first entry,
 seconds
dt_ephem = (sat_ephem(:,19) - sat_ephem(1,19))*604800 +
 (sat_ephem(:,17) - sat_ephem(1,17));


% Compute elapsed times of each input time wrt first ephemeris entry,
 seconds
dt_input = (t_input(:,1) - sat_ephem(1,19))*604800 + (t_input(:,2) -
 sat_ephem(1,17));
```

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


    % Pull out most recent ephemeris values
%     jj = max( find(dt_input(tt) >= dt_ephem) ); % sat_ephem(:,17) =
 toe (sec into GPS week) of each entry
                                                   % jj = row of specific
 sat. ephem. data with epoch closest to input time

    % Pull out nearest ephemeris values
    [mn,jj] = min(abs( dt_input(tt) - dt_ephem ));



    if isempty(jj),continue,end  % no matching ephemeris time found.
 continue to next input time


    % Pull out common variables from the ephemeris matrix

  %======================================================================
    %toe = sat_ephem(jj,17);           % time of ephemeris
    dt  = dt_input(tt) - dt_ephem(jj); % seconds difference from epoch

    a   = sat_ephem(jj,5)^2;           % semimajor axis, sqrt(a) =
 gps_ephem_all(:,5) (meters)
    ecc = sat_ephem(jj,4);             % eccentricity
    n0  = sqrt(muE/a^3);               % nominal mean motion (rad/s)
    n   = n0 + sat_ephem(jj,3);        % corrected mean motion,
 delta_n = gps_ephem_all(:,3)
    M   = sat_ephem(jj,2) + n*dt;      % mean anomaly, M0 =
 gps_ephem_all(:,2)


    % Compute perigee, true and eccentric anomaly...

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

    % Load argument of perigee to a local variable and add perigee
 rate, rad
    perigee  = sat_ephem(jj,8); % + perigee_rate * dt;

    % Compute Eccentric Anomaly, rad
    E    = mean2eccentric(M,ecc);
    cosE = cos(E);
    sinE = sin(E);

    % Compute true anomaly, rad
    nu    = atan2( sqrt(1 - ecc*ecc).*sinE,  cosE-ecc );

    % Compute the argument of latitude, rad
    u = nu + perigee;  % true anomaly + argument of perigee
```

```
   % Compute radius and inclination

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

   r   = a * (1 - ecc*cosE) ;                         % corrected
radius
   inc = sat_ephem(jj,7) ;   %  inclination
                                                              % i_dot
= sat_ephem(jj,10)

   cosu = cos(u);
   sinu = sin(u);

   % Compute satellite position in orbital plane (Eq. 13)

 %=======================================================================
   xo = r * cosu;    % satellite x-position in orbital plane
   yo = r * sinu;    % satellite y-position in orbital plane

   % Corrected longitude of ascending node for node rate and Earth
rotation

 %=======================================================================
   % Ascending node = ephem_all(jj,6)
   node = sat_ephem(jj,6);% + (sat_ephem(jj,9) - wE)*dt -  (wE *
sat_ephem(jj,17)); % Toe = gps_ephem_all(jj,17)

   % Calculate GPS Satellite Position in ECEF (m)

 %=======================================================================
   cosi = cos(inc);    sini = sin(inc);
   coso = cos(node);   sino = sin(node);


   % Satellite position in ECEF (m)
   x(tt,1) = xo*coso - yo*cosi*sino;  %x-position

   x(tt,2) = xo*sino + yo*cosi*coso;  %y-position

   x(tt,3) = yo*sini;                 %z-position


   % Keep track of health of each satellite

 %=======================================================================
   health(tt,1) = sat_ephem(jj,25); % satellite health (0.00 is
useable)
```

```
end % END of t_input loop
 =================================================
%===========================================================================
```

```
end
```

Published with MATLAB® R2020b