HW 8

Table of Contents

John Clouse

read_GPSbroadcast, broadcast2xv, adjust year, and read_rinex functions provided from class are used in this homework.

Initialize

```
clearvars -except function_list pub_opt
close all
c = 2.99792458e8; %m/s
f_L1 = 1575.42; % MHz
f_L2 = 1227.60; % MHz

print_vec3 = @(x) ...
    fprintf(sprintf('%13.2f\n%13.2f\n%13.2f\n',x(1),x(2),x(3)));
print_vec4 = @(x) ...
    fprintf(sprintf('%13.2f\n%13.2f\n%13.2f\n%13.2f\n',x(1),x(2),x(3),x(4)));
```

Read the data files

```
eph = read_GPSbroadcast('brdc2550.14n');
obs_data = read_rinex_obs('test.14o');
P1 = 6;
P2 = 7;
C1 = 8;
approx_rx_pos = [-1248596.2520 -4819428.2840  3976506.0340]'; % m
GPS_Week = eph(1,19);
GPS_TOD = [1 03 00];
TOW = eph(1,20)+GPS_TOD(1)*3600 + GPS_TOD(2)*60 + GPS_TOD(3);
fprintf('RINEX location:\n')
print_vec3(approx_rx_pos)
fprintf('\n')
```

```
Parsing RINEX file test.14o
Returning data for all PRNs

ans =

        1049          13
```

```
Total epochs: 131
Finished.

RINEX location:
  -1248596.25
  -4819428.28
   3976506.03
```

Observations at the given epoch

```
obs_at_epoch = obs_data.data(obs_data.data(:,2) == TOW, :);
prn_list = obs_at_epoch(:,3);
num_sats = length(prn_list);

% Ionosphere
iono_free_pseudorange = ...
    (f_L1*f_L1*obs_at_epoch(:,P1) - f_L2*f_L2*obs_at_epoch(:,P2))./ ...
    (f_L1*f_L1-f_L2*f_L2);

% Geometric range, satellite clock correction, relativity correction
geo_range = zeros(num_sats,1);
relativityCorr = zeros(num_sats,1);
satClkCorr = zeros(num_sats,1);
r_sat = zeros(num_sats,3);
for ii = 1:num_sats
    [~, geo_range(ii), tmp] = compute_range(eph, prn_list(ii), TOW, approx_rx_pos)
    r_sat(ii,:) = tmp';
    [~,~,~,relativityCorr(ii),satClkCorr(ii)] = ...
        broadcast2xv(eph,[GPS_Week TOW],prn_list(ii));
end

% Azimuth, Elevation
GPSvec = [2014 09 12 1 03 00];
% navfilename = generate_GPSyuma_name(GPSvec);
[navfilename,statusflag] = download_GPSyuma(GPSvec);
durationhrs = 1;
dt_sec = 3601;
ant_enu = [0 0 1];
mask_min = 0;  % deg
mask_max = 90; % deg
[latgd, lon, alt] = ECEF2ellipsoidal(approx_rx_pos);

[time_wntow, GPSdata] = ...
    ASEN5090_GPSvis(navfilename, 1, GPSvec,...
    durationhrs, dt_sec, latgd*180/pi, lon*180/pi, alt,...
    mask_min, mask_max, mask_min, ant_enu, 0, []);
% rearrange the data to be in the same order as prn list
prn_el = zeros(num_sats,1);
prn_az = zeros(num_sats,1);
for ii = 1:num_sats
    prn_el(ii) = GPSdata.topo_el(prn_list(ii));
```

```
    prn_az(ii) = GPSdata.topo_az(prn_list(ii));
end

T = table(prn_list, iono_free_pseudorange, geo_range, satClkCorr, ...
    relativityCorr, prn_el, prn_az);
fprintf('PRNs and corrections (distances in meters, angles in degrees):\n')
disp(T)
fprintf('\n\n\n\n\n')
```

```
Downloading GPS YUMA file from: http://www.celestrak.com/GPS/almanac/Yuma/
File name: yuma0785.405504.alm

PRNs and corrections (distances in meters, angles in degrees):
    prn_list    iono_free_pseudorange    geo_range     satClkCorr
    ________    _____________________    __________    __________

    18          2.1011e+07               2.1114e+07     1.032e+05
    15          2.0893e+07               2.0833e+07        -60652
    16          2.4701e+07               2.4642e+07        -58603
    26          2.2555e+07                2.254e+07        -15544
    22          2.3265e+07               2.3345e+07         80061
    21          2.1027e+07               2.0912e+07    -1.1497e+05
    29          2.2088e+07               2.2256e+07     1.6781e+05
    27          2.4188e+07               2.4192e+07        4115.5

    relativityCorr    prn_el    prn_az
    ______________    ______    _______

     2.6998           63.304    -83.622
     2.5463           57.302     79.488
      3.994           11.896    -72.048
     1.0851           26.304     46.523
      3.645           25.315    -104.39
    -9.2211           70.773    -26.078
    -0.6098           37.032     174.32
    0.94731           14.817    -40.155
```

Observation/Geometry matrix

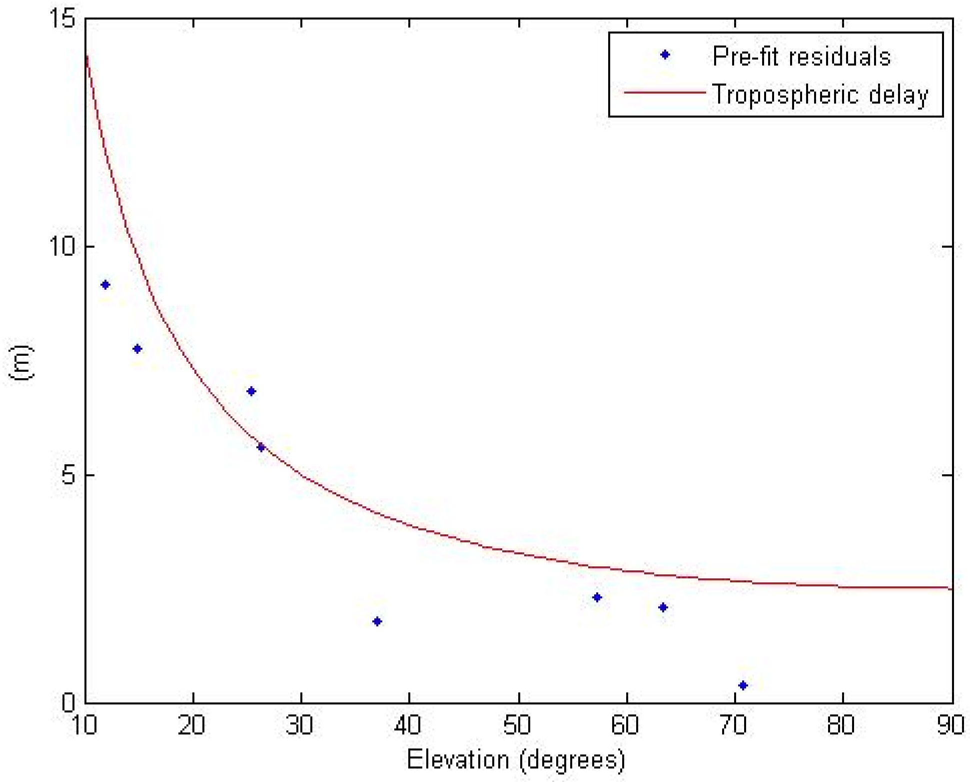
```
A = ones(num_sats, 4); % x, y, z, rx time error
for ii = 1:num_sats
    A(ii,1:3) = -(r_sat(ii,:)-approx_rx_pos')./geo_range(ii);
end
A
```

```
A =

    0.5998    0.5341   -0.5958    1.0000
   -0.3648    0.7107   -0.6015    1.0000
    0.8947   -0.2596   -0.3636    1.0000
   -0.6398    0.1260   -0.7581    1.0000
    0.9666    0.2398   -0.0904    1.0000
    0.2792    0.4998   -0.8199    1.0000
    0.1659    0.9550    0.2457    1.0000
    0.5384   -0.4093   -0.7366    1.0000
```

Prefit residuals

Plotting the residuals and Lecture 15's simple Tropo model vs. elevation

```
dy = iono_free_pseudorange - geo_range + satClkCorr - relativityCorr;

plot(prn_el, dy, '.')
hold on
T_el = 10:90;
plot(T_el, 2.5./sin(T_el*pi/180), 'r')
ylabel('(m)')
xlabel('Elevation (degrees)')
legend('Pre-fit residuals','Tropospheric delay')
```



Least squares solution for position deviation

Find the position deviation from the RINEX location, dx

```
est_deviation = (A'*A)\A'*dy;
est_deviation_with_T = (A'*A)\A'*(dy - 2.5./sin(prn_el*pi/180));

fprintf('Estimated Deviation (m):\n')
print_vec4(est_deviation)
fprintf('Estimated Deviation with Troposphere model (m):\n')
print_vec4(est_deviation_with_T)
fprintf('\n')
```

```
Estimated Deviation (m):
        -0.99
        -7.87
         4.60
         9.28
Estimated Deviation with Troposphere model (m):
        -0.31
         0.27
        -0.03
        -1.27
```

Starting from incorrect location

Choosing the center of the earth as the initial guess.

```
approx_rx_pos = [0 0 0]'; % m
b = 0;
for iter = 1:5
    for ii = 1:num_sats
        [~, geo_range(ii), tmp] = ...
            compute_range(eph, prn_list(ii), TOW, approx_rx_pos);
        r_sat(ii,:) = tmp';
        A(ii,1:3) = -(r_sat(ii,:)-approx_rx_pos')./geo_range(ii);
    end
    dy = iono_free_pseudorange - geo_range + satClkCorr - relativityCorr ;
    est_deviation = (A'*A)\A'*dy;
    approx_rx_pos = approx_rx_pos + est_deviation(1:3);
    b = est_deviation(4) + b;
    fprintf('Iteration %d\n',iter)
    fprintf('Estimated Deviation (m):\n')
    print_vec4(est_deviation)
    fprintf('New Position (m, ECF):\n')
    print_vec3(approx_rx_pos)
    fprintf('\n')
end
```

```
Iteration 1
Estimated Deviation (m):
  -1551257.50
  -5743499.33
```

```
   4823759.80
   1337058.30
New Position (m, ECF):
  -1551257.50
  -5743499.33
   4823759.80

Iteration 2
Estimated Deviation (m):
    289790.62
    895707.68
   -816001.34
     47666.39
New Position (m, ECF):
  -1261466.88
  -4847791.65
   4007758.46

Iteration 3
Estimated Deviation (m):
     12850.96
     28321.67
    -31206.10
        68.62
New Position (m, ECF):
  -1248615.92
  -4819469.98
   3976552.35

Iteration 4
Estimated Deviation (m):
        18.67
        33.83
       -41.72
         9.28
New Position (m, ECF):
  -1248597.24
  -4819436.16
   3976510.63

Iteration 5
Estimated Deviation (m):
        -0.00
         0.00
         0.00
         9.28
New Position (m, ECF):
  -1248597.24
  -4819436.16
   3976510.63
```

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 [range0, range1, r_gps] = compute_range(eph, PRN, t, userpos)

fcnPrintQueue(mfilename('fullpath'))
c = 2.99792458e8; %m/s
w = 7292115.0e-11; %rad/s Earth spin rate

GPS_Week = eph(1,19);

% Get the GPS SV vector
[~, r_gps,~,~,~] = broadcast2xv(eph,[GPS_Week t],PRN);
r_gps = r_gps'; % Vertical vectors
R = norm(r_gps-userpos);
range0 = R; % Uncorrected

% Iterate to correct the range for TOF
tol = 1e-6;
oldR = R + 1;
while abs(R-oldR) > tol

Tt = t - R/c;

[~, r_gps_Tt,~,~,~] = broadcast2xv(eph,[GPS_Week Tt],PRN);
r_gps_Tt = r_gps_Tt'; % Vertical vectors

phi = w*(t-Tt);

C = [cos(phi) sin(phi) 0;
    -sin(phi) cos(phi) 0;
    0 0 1];

r_gps = C*r_gps_Tt;

oldR = R;

R = norm(r_gps-userpos);

end

range1 = R;
```

Published with MATLAB® R2013b