Contents


```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        Homework6_main.m
% Author      : Zach Dischner
% Date        : 10/24/2013
% Description : Matlab script for all calculations required for
%               ASEN 5090 Homework 6
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-____________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.    """""  \\/
%                                    `"""""""""""`
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

Setup Work Space

```
clc;clear all;close all
screen_size = get(0,'ScreenSize');
sw = screen_size(3);    % Screen Width
sh = screen_size(4);    % Screen Height
% figColor = [0.99 0.99 0.98];
addpath HW6_files
soln_format = '|  %2.0f | %15.3f | %7.3f | %12.3f | %15.3f | %7.3f \t\n';
```

Setup Problem

```
%------Define Navigation and Observation File
nav_msg  = 'brdc2640.12n';
obs_file = 'darw264x.12o';
fprintf('1) Navigation File: %s\n2)Observation File: %s\n\n',nav_msg,obs_file);

%------Define Orbit determination parameters
```

```
params.mu = 3.986005e14;        % Gravitational param [m^3/s^2]
params.we = 7.2921151467e-5;    % Earth's rotation rate [rad/s]

%------Define speed of light
params.c = 299792458; % [m/s]
```

```
1) Navigation File: brdc2640.12n
2)Observation File: darw264x.12o
```

Read Files

```
%------Read navigation message content
fprintf('3) Read Navigation File\n\n')
nav_data = read_GPSbroadcast(nav_msg); % Returns [n x 25] matrix of sat orbit information
%                   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 l
atitude
%                  col13: Crc, amplitude of the cosine harmonic correction term to the orbit radiu
s
%                  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 in
clination
%                  col16: Cis, amplitude of the cosine harmonic correction term to the angle of in
clination
%                  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: blank (zero)
%                  col25: health, satellite health (0=good and usable)

%------Read a-priori receiver position from header of RINEX obs file
fprintf('4) Get a-priori from RINEX file\n\n')
[ fid, rec_xyz, observables ] = read_rinex_header( obs_file );

%------Read Observation file
obs_data = read_rinex_obs3(obs_file);
```

```
Week_col = 1;
SOW_col = 2;    % Simple indicator for clarification
PRN_col = 3;    % Simple indicator for clarification
C1_col = 6;
rows = find(obs_data.data(:,SOW_col)==min(obs_data.data(:,SOW_col)));
PRNS = obs_data.data(rows,PRN_col);
GPS_Secs = obs_data.data(rows,SOW_col);
GPS_Weeks = obs_data.data(rows,Week_col);
```

```
3) Read Navigation File

4) Get a-priori from RINEX file


ans =

    25    13
```

Calculate Geometric Range for First Epoch Satellites

```
fprintf('5) Get ephemeris data for first epoch in rinex file\n\n')
[epochData,rows] = findNearestEphem(PRNS,GPS_Weeks(1),GPS_Secs(1),nav_data);

fprintf(['6)For all the PRNs in the first epoch, make (and call)', ...
         'a function \n\tthat calculates the geomet- ric range (use instructions',...
                    '\n\tat the end of this assignment). Since your broadcast ',...
                    '\n\tephemeris has the information needed, calculate the ',...
                    '\n\trelativity correction.\n\n'])
type('getSatGeomRange')
fprintf('7) Write a function that calculates satellite clock correction\n\n')
type('getSatClockCorrection.m')
fprintf('8) Access values for C1\n\t[>>C1(ii) = obs_data.data(ii,C1_col);]\n\n')
fprintf('9) Output values in readable format\n')

%------Allocate
Tt = zeros(length(rows),1);
R=Tt; sat_clk_t_corr=Tt; satcorr=Tt; rel_corr=Tt; C1=Tt;
fprintf('|_PRN_|___geomRange_____|___rel___|____satClk____|________C1_______|__C1-R+satcorr\n')
for ii = 1:length(rows)
    %------Setup Range Finding
    GPS_SOW = epochData(ii,17);
    GPS_Week = GPS_Weeks(1);
    params.Secs = GPS_Secs(1);  % Seconds used to calculate seconds since epoch

    %------Calculate Geometric Range
    [R(ii), rel_dt] = getSatGeomRange(rec_xyz', GPS_Week, GPS_Secs(1), PRNS(ii), nav_data, params
);
    rel_corr(ii) = rel_dt*params.c;
    %------Get clock correction
    sat_clk_t_corr(ii) = getSatClockCorrection(GPS_Week, GPS_Secs(1), PRNS(ii), nav_data);
```

```
    %------Get Satellite Correction
    satcorr(ii) = sat_clk_t_corr(ii)*params.c;

    %------Retrieve C1
    C1(ii) = obs_data.data(ii,C1_col);


    %------Output Answers yo!
    fprintf(1,soln_format,PRNS(ii),...
        R(ii),rel_corr(ii),satcorr(ii),C1(ii),C1(ii)-R(ii)+satcorr(ii))
end
```

```
5) Get ephemeris data for first epoch in rinex file

6)For all the PRNs in the first epoch, make (and call)a function
        that calculates the geomet- ric range (use instructions
        at the end of this assignment). Since your broadcast
        ephemeris has the information needed, calculate the
        relativity correction.
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         getSatGeomRange.m
% Author      : Zach Dischner
% Date        : 10/22/2013
% Description : calculate satellite position from GPS ephemeris dataset
%
%
%                  __...___________________         ,
%                 `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                               _____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.    """""  \\/
%                                    `""""""""""`
% Inputs          : rStation - GPS Rx [x,y,z] coords in ECEF meters
%                   GPS_Weeks - GPS Week time
%                   GPS_SOW - Seconds into week
%                   PRN - Satellite PRN
%                   nav_data - nx25 array of sat data from broadcase
%                     ephemeris
%                   params - structure containing keplarian specs and extra
%                             calculations for sat position
%
% Outputs         : [rk]-3d ECI coordinates of satellite
%
% History         October 11 2013 - First Rev
%                 October 24 2013 - Reformatted output to [rk,tk]
%                                 - Added check for time field in params,
%                                 other that that in the ephemeris data
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [R, rel_dt] = getSatGeomRange(rStation, GPS_Weeks, GPS_SOW, PRN, nav_data, params)
```

```
%------Find Nearest Ephemeris
[epochData,rows] = findNearestEphem(PRN, GPS_Weeks, GPS_SOW, nav_data);
SOW_col = 20;
% Single Row in this case

%------Get Sat Position from Ephemeris data
[rSat,rel_dt] = calculateSatellitePosition(epochData, params);


%------Set up convergence limits
R = 0;
conv_limit = 1e-12;
max_iters = 100;
iter = 1;

%------Iterate and converge on Geometric Range
while(1)
    %------Calculate Geometric Range
    Rtmp = norm( rSat - rStation );

    %------Check for Convergence
    if(abs(Rtmp - R) < conv_limit)
        break
    end

    %------Assign new Range Value now that criterion are passed
    R = Rtmp;

    %------Check for iteration limit
    if(iter > max_iters)
        error('Range Calculation not converging!')
    end

    %------Increase iteration count
    iter = iter + 1;

    %------Calculate 'Tt', time of transmission
    dt = R/params.c;
%     Tr = epochData(SOW_col);
    Tr = GPS_SOW;
    Tt = Tr - dt;

    %------Recalculate Satellite position
    params.Secs = Tt;
    % to use new time value
    [rSat,rel_dt] = calculateSatellitePosition(epochData,params);

    %------Rotate Sat position at time Tr (account for earth's rotation)
    phi = params.we*dt;
    rSat = transpose(rot3(phi)*rSat');

end
```

7) Write a function that calculates satellite clock correction

```
function [tcorr] = getSatClockCorrection(GPS_Weeks, GPS_SOW, PRN, nav_data)
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        getSatClockCorrection.m
% Author      : Zach Dischner
% Date        : 10/24/2013
% Description : Function to return all emphimeris data from a nav data
%               array
%
%
%                __...___________________         ,
%               `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                 `"""""""""""""""""| |""` [_""_-____________"_/
%                                   | |  /..../`'-._.-'`
%                               ____| |__/::..'_
%                             |\ ".`"` '_____//\
%                             `"'-.   """"""  \\/
%                                  `""""""""""`
% Inputs        : PRN - Satellite PRN number
%                 GPSWeeks - GPS week number (modded or no?)
%                 GPSSOW - GPS Seconds of week
%                 navData - A full array of all emphimeris data, fetched
%                           from navigation file
% Outputs       : t_corr - Satellite clock correction
%
% History       Oct 24 2013 - First Rev
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

%------Get ephemeris dataset
[eph_data,tmp] = findNearestEphem(PRN, GPS_Weeks, GPS_SOW, nav_data);

%------Define readibility indices
Af0_col = 21;   %Af0, satellite clock bias (sec)
Af1_col = 22;   %Af1, satellite clock drift (sec/sec)
Af2_col = 23;   %Af2, satellite clock drift rate (sec/sec/sec)
SOW_col = 17;   %Toe, reference time ephemeris (seconds into GPS week)

%------Fetch Correction Constants
Af0 = eph_data(Af0_col);
Af1 = eph_data(Af1_col);
Af2 = eph_data(Af2_col);

t_eph = eph_data(SOW_col);
dt = GPS_SOW - t_eph;

%------Calculate clock correction
tcorr = Af0 + Af1*(dt) + Af2*(dt)^2;

end %function
```

```
8) Access values for C1
        [>>C1(ii) = obs_data.data(ii,C1_col);]
```

```
9) Output values in readable format
|_PRN_|___geomRange_____|___rel___|____satClk____|________C1_______|__C1-R+satcorr
|  22 |     23579224.918 |    3.615 |     44391.125 |     23534846.760 |  12.968
|  19 |     20735970.595 |   -3.023 |    -91561.324 |     20827530.400 |  -1.519
|  32 |     23108668.234 |   -7.571 |   -146813.326 |     23255482.600 |   1.040
|  30 |     24718340.387 |    1.192 |   -121274.090 |     24839641.980 |  27.503
|  11 |     23174554.067 |   -6.493 |    -91273.050 |     23265827.520 |   0.403
|  14 |     22306321.088 |    0.138 |     63867.107 |     22242460.940 |   6.959
|  16 |     24702119.850 |    4.184 |    -73858.542 |     24776002.040 |  23.648
|  23 |     24342413.027 |   -4.537 |     48800.686 |     24293621.640 |   9.299
|  31 |     22273098.557 |   -3.074 |     82598.046 |     22190500.780 |   0.269
|   6 |     21556964.807 |    3.992 |    -10214.738 |     21567188.380 |   8.835
|   3 |     21128808.735 |   -8.998 |     31140.308 |     21097664.960 |  -3.468
|   1 |     24399646.959 |   -0.721 |     82252.942 |     24317412.740 |  18.723
```

SUPPORTING FUNCTION - date2GPSTime.m

```
type('date2GPSTime.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         date2GPSTime.m
% Author      : Zach Dischner
% Date        : 10/11/2013
% Description : Convert a date type object into [GPS_Weeks, GPS_SOW] time
%
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--.._____
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% Inputs        : utcDate - Satellite PRN number
%
% Outputs       : [GPS_Weeks, GPS_SOW]-weeks and seconds of week
%
% TODOS         : Vectorize!
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [GPS_Weeks, GPS_SOW] = date2GPSTime(utcDate)

gps_week_start = 'January 6 1980 00:00:00';
modnum = 0; % modnum = 0 for no modulo
tmp = mod((datenum(utcDate) - datenum(gps_week_start))/7,modnum); % (Difference in days)/7 = diff
erence in weeks
GPS_Weeks = floor(tmp);
GPS_SOW = round((tmp-GPS_Weeks)*7*24*3600);
```

SUPPORTING FUNCTION - findNearestEphem.m

```
type('findNearestEphem.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         findNearestEmph.m
% Author      : Zach Dischner
% Date        : 10/11/2013
% Description : Function to return all emphimeris data from a nav data
%               array
%
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `"""""""""""`
% Inputs        : PRN - Satellite PRN number
%                 GPSWeeks - GPS week number (modded or no?)
%                 GPSSOW - GPS Seconds of week
%                 navData - A full array of all emphimeris data, fetched
%                           from navigation file
% Outputs       : emphData - Single row (struct?) of emphemeris data per
%                           sat PRN at time [gps_weeks, gps_seconds
%
% History       Oct 11 2013 - First Version
%               Oct 22 2013 - Added return for rownums
%               Oct 24 2013 - Changed PRN matching to ismember(), to allow
%                              for array matching of PRNs
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

function [ephemData,rownums] = findNearestEphem(PRN, GPS_Weeks, GPS_SOW, navData)

% weeknums = nav_ephem(:,19);
% secofweeks = nav_ephem(:,17)

rownums = find(navData(:,17)<=GPS_SOW & ismember(navData(:,1),PRN) & navData(:,19)==GPS_Weeks);
ephemData = navData(rownums,:);
```

SUPPORTING FUNCTION - calculateSatellitePosition.m

```
type('calculateSatellitePosition.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         calculateSatellitePosition.m
% Author      : Zach Dischner
% Date        : 10/24/2013
% Description : calculate satellite position from GPS ephemeris dataset
```

```
%
%
%                 __...____________________         ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% Inputs        : ephem - Satellite ephemeris dataset
%                 params - structure containing keplarian specs and extra
%                           calculations for sat position
%
% Outputs       : [rk]-3d ECI coordinates of satellite
%
% History       October 11 2013 - First Rev
%               October 24 2013 - Reformatted output to [rk,tk]
%                               - Added check for time field in params,
%                               other that that in the ephemeris data
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [rk,dt_rel] = calculateSatellitePosition(ephem,params)

%-----Extract all ephemeris components to make life easy and epicer
ephem = num2cell(ephem);
[prn,M0,delta_n,ecc,sqrt_a,Loa,incl,perigee,ra_rate,i_rate,Cuc,Cus,Crc,Crs,Cic,Cis,...
    Toc,IODE,GPS_week,Toc,Af0,Af1,Af2,nil,health] = deal(ephem{:});
A = sqrt_a^2;
%------Correct Mean Motion
n0  = sqrt(params.mu/(A)^3); % Calculated mean motion [rad/s]
n   = n0 + delta_n;                 % Corrected Mean Motion

%------Correct Time
tk  = params.Secs-Toc;

%------Mean Anomaly
Mk = M0 + n*tk; % Mean anomaly

%------Eccentric Anomaly
options=optimset('Display','off','TolFun',1e-15,'TolX',1e-15);
Ek = fsolve(@(Ek) (Ek)-ecc*sin(Ek)-Mk,4,options);

%------True Anomaly
vk = atan2(      (sqrt(1-ecc^2)*sin(Ek)/(1-ecc*cos(Ek))), ...
                      ((cos(Ek)-ecc)/(1-ecc*cos(Ek)))   );

%------Argument of Latitude
Phik = vk + perigee;

%------Second Harmonic Perturbations
del_uk = Cus*sin(2*Phik) + Cuc*cos(2*Phik);
del_rk = Crs*sin(2*Phik) + Crc*cos(2*Phik);
del_ik = Cis*sin(2*Phik) + Cic*cos(2*Phik);

%------Corrected argumet of latitude, radius, inclination
```

```
uk = Phik + del_uk;
rk = A*(1-ecc*cos(Ek)) + del_rk;
ik = incl + del_ik + i_rate*tk;

%------Position in Orbit Plane
xkp = rk*cos(uk);
ykp = rk*sin(uk);

%------Corrected Longitude of ascending node
Omegak = Loa + (ra_rate - params.we)*tk - params.we*Toc;

%------Earth Fixed Coordinates
xk = xkp * cos(Omegak) - ykp * cos(ik) * sin(Omegak);
yk = xkp * sin(Omegak) + ykp * cos(ik) * cos(Omegak);
zk = ykp * sin(ik);

%------Relativity time shift
dt_rel = 2*sqrt(params.mu)/params.c^2 * ecc * sqrt_a * sin(Ek);
rk = [xk,yk,zk];
```

SUPPORTING FUNCTION - findFirstEpoch.m

```
type('findFirstEpoch.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        findFirstEpoch.m
% Author      : Zach Dischner
% Date        : 10/24/2013
% Description : Function to return all emphimeris data from a nav data
%               array
%
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--.._____
%                  `"""""""""""""""""| |""` [_""_-___________"_/
```

```
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% Inputs         : navData - Navigation dataset.
% Outputs        : emphData - rows (struct?) of emphemeris data for
%                             the first epoch
%                  rows - row indices of the first epoch datasets
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

function [emphData,rows] = findFirstEpoch( navData )

weeknums = navData(:,19);
secofweeks = navData(:,17);

n_epochs = length(navData);
epochs = zeros(n_epochs,1);
for ii =1:n_epochs
    epochs(ii) = datenum(GPSTime2Date(weeknums(ii),secofweeks(ii)));
end

rows    = find(epochs==min(epochs));
emphData = navData(rows,:);
```

SUPPORTING FUNCTION - date2GPSTime.m

```
type('date2GPSTime.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         date2GPSTime.m
% Author      : Zach Dischner
% Date        : 10/11/2013
% Description : Convert a date type object into [GPS_Weeks, GPS_SOW] time
%
%
%                  __...____________________          ,
%                 `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                   `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% Inputs         : utcDate - Satellite PRN number
%
% Outputs        : [GPS_Weeks, GPS_SOW]-weeks and seconds of week
%
% TODOS          : Vectorize!
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [GPS_Weeks, GPS_SOW] = date2GPSTime(utcDate)
```

```
gps_week_start = 'January 6 1980 00:00:00';
modnum = 0; % modnum = 0 for no modulo
tmp = mod((datenum(utcDate) - datenum(gps_week_start))/7,modnum); % (Difference in days)/7 = diff
erence in weeks
GPS_Weeks = floor(tmp);
GPS_SOW = round((tmp-GPS_Weeks)*7*24*3600);
```

Published with MATLAB® R2013b