Contents


```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         Homework5_main.m
% Author      : Zach Dischner
% Date        : 10/10/2013
% Description : Matlat script for all calculations required for
%               ASEN 5090 Homework 5
%
%                 __...__________________________           ,
%                `(\ [ ===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 HW5_files
```

Setup Problem

```
%------Define Navigation and Observation File
nav_msg  = 'brdc2640.12n';
obs_file = 'test.12o';

%------Read navigation message content
nav_ephem = 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)
```

Get GPS Time from UTC time

```
%------Convert time of interest into GPS week and seconds of week
% Using standard for GPS Epoch: http://tycho.usno.navy.mil/gps_week.html
% Verified with online calendar: http://adn.agi.com/GNSSWeb/
emph_date      = 'September 20 2012 02:11:00';
[GPS_W, GPS_SOW] = date2GPSTime(emph_date);
fprintf('\n*^*^*^*^*^*^*^*^*^*^*^*^*^*^* HW5 Step 3 *^*^*^*^*^*^*^*^*^*^*^*^*^*^*\n')
fprintf('GPS Week and seconds of week: [ %d , %d ]\n',GPS_W, GPS_SOW)
```

```
*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^ HW5 Step 3 *^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*
GPS Week and seconds of week: [ 1706 , 353460 ]
```

Calculate Satellite Position

```
%------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]
params.Secs = GPS_SOW;      % Seconds used to calculate seconds since epoch

%------Define Satellite PRN Ranges
PRNs = [15, 17, 12];
fprintf('\n\n*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^ HW5 Step 6 *^*^*^*^*^*^*^*^*^*^*^*^*^*^*^*\n\n')
```

```
for PRN = PRNs
    %------Fetch corresponding ephemeris data
    ephem = findNearestEphem(PRN,GPS_W,GPS_SOW, nav_ephem);
    [x,y,z] = calculateSatellitePosition(ephem,params);
    fprintf('PRN: %d ===> [x,y,z]: [%10.3f , %10.3f , %10.3f]m \n',PRN,x,y,z)
    [h,pos,v,r,s,a]=broadcast2xva(nav_ephem,[GPS_W,GPS_SOW],PRN);
    fprintf('   Ben B Ref [x,y,z]: [%10.3f , %10.3f , %10.3f]m \n\n',pos)

end
fprintf('\n\n')
```

```
*^*^*^*^*^*^*^*^*^*^*^*^*^*^*^ HW5 Step 6 *^*^*^*^*^*^*^*^*^*^*^*^*^*^*^

PRN: 15 ===> [x,y,z]: [6407933.252 , -20631648.099 , 15294938.689]m
   Ben B Ref [x,y,z]: [6407933.252 , -20631648.099 , 15294938.689]m

PRN: 17 ===> [x,y,z]: [14276106.295 , 15238381.805 , -16373240.384]m
   Ben B Ref [x,y,z]: [14276106.295 , 15238381.805 , -16373240.384]m

PRN: 12 ===> [x,y,z]: [4674101.298 , -14021589.264 , -22087395.102]m
   Ben B Ref [x,y,z]: [4674101.298 , -14021589.264 , -22087395.102]m
```

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
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
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 for
%                           sat PRN at time [gps_weeks, gps_seconds
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

function emphData = findNearestEmph(PRN, GPS_Weeks, GPS_SOW, navData)

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

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

SUPPORTING FUNCTION - calculateSatellitePosition.m

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

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        calculateSatellitePosition.m
% Author      : Zach Dischner
% Date        : 10/11/2013
% 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       : [x,y,z]-3d ECI coordinates of satellite
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [xk,yk,zk] = 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,...
    Toe,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-Toe;  %(Have to worry about it being less than -302400 like sheet says?)

%------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*Toe;

%------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);
```

Published with MATLAB® R2013b