## Contents



```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        Homework7_main.m
% Author      : Zach Dischner
% Date        : 10/27/2013
% Description : Matlab script for all calculations required for
%               ASEN 5090 Homework 7
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

## Questions to Answer (First for ease of grading)

```
type('answers.txt')
```

```
1. The main difference between single frequency prefits for joze and onsa is that
while onsa's prefits look mainly randomly distributed, joze has a very obvious
trend. Joze's residules have sharp and steady climbs in prefits in time, and
then the prefit drops back to its previous low, in a manner that looks very much
like a triangle wave. Each satellite's residue is stacked at almost the same
place in time. This is likely due to the fact that the joze clock is much lower
quality than the onsa one. The triangle wave error form is caused by the clock
drifting, then being reset. It is clearly the dominant factor in joze's
residule values.

2. The single frequency prefit residules are different from the ionosphere
free ones because the ionosphere affects each satellite differently, as they are
in different positions in their orbits. The single frequency prefit contains
structure for each PRN plotted, while the ion-free residules have a more normal
distribution.

3. The 4-5 sigma outliers in onsa's ion-free residules are due to low elevation
data points. See the plot of residules vs elevation. The residules grow much
more dispersed for low elevation observations. A good mitigation strategy could
be to raise the elevation mask, so that only higher, more trustworthy observations
would be included.
```

## Setup Work Space

```
clc;clear all;close all
tic;
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 HW7_files
soln_format = '|  %2.0f | %15.3f | %7.3f | %12.3f | %15.3f | %9.3f | %3.2f | %3.2f  \t\n';
% h = waitbar(0,'GO PARTY, ILL STAY HERE WORKING!!!');
wb_tot = 1024+1238;
tot_iters = 0;
```

## Setup Problem

```
%------Define Navigation and Observation File
RINEX_FILES = {'onsa2640.onehour','joze2640.onehour'};
nav_msg  = 'brdc2640.12n';

%------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]

params.options=optimset('Display','off','TolFun',1e-10,'TolX',1e-10);


%------Define Zenith Correction for each site
Tzenith = [2.3858, 2.4086]; %[m]
```

```
%------Read navigation message content
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 latitu
de
%                 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 inclinati
on
%                 col16: Cis, amplitude of the cosine harmonic correction term to the angle of inclinati
on
%                 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)
```

### Calculate Pre-Fit Residuals

```
for file_idx=1:length(RINEX_FILES)
```

```
    rinex_file = RINEX_FILES{file_idx};
    fprintf('Processing Rinex file %s\n', rinex_file)
```

```
Processing Rinex file onsa2640.onehour
```

```
Processing Rinex file joze2640.onehour
```

### Read Observation Files, Sort Data

```
    %------Read a-priori receiver position from header of RINEX obs file
    [ fid, rec_xyz, observables ] = read_rinex_header( rinex_file );

    %------Read Observation file
    obs_data    = read_rinex_obs3(rinex_file);
    cols        = obs_data.col; % Structure of column index descriptions
```

```
%------Make nice column addressing variables (P1_col, P2_col ... etc)
fields = fieldnames(cols);
for kk=1:length(fields)
    eval(cell2mat([fields(kk),'_col = cols.',fields(kk),';']));
end

PRNS = obs_data.data(:,PRN_col);
[GPSSecAryUn,secs_idx] = unique(obs_data.data(:,TOW_col));
GPSWeekAry = obs_data.data(:,WEEK_col);
GPSSecAry = obs_data.data(:,TOW_col);
%     GPS_Secs = obs_data.data(rows,SOW_col);
%     GPS_Weeks = obs_data.data(rows,Week_col);
```

```
Read 1000 lines
ans =
        1238          13


Read 1000 lines
ans =
        1024           8
```

## Fetch/Compute Pseudorange Values

```
%------Allocate
rho_obs = zeros(1, length(PRNS));
rho_model = rho_obs;
el = rho_obs;
res = rho_obs;
ionFree = rho_obs;
sat_prn = rho_obs;
iter = 0;
for sec_idx = secs_idx'
```

## Calculate "Modeled" Pseudorange

```
    %------Setup Range Finding
    GPS_SOW = GPSSecAry(sec_idx);
    GPS_Week = GPSWeekAry(sec_idx);
    params.Secs = GPS_SOW; %(GPS_Secs(1))  % Seconds used to calculate seconds since epoch

    %------Iterate over epoch satellite data
    Nsats = sum(GPSSecAry == GPSSecAry(sec_idx));
    for sat = 1:Nsats
```

```
            data_idx = sec_idx+sat-1;
%             waitbar((data_idx+tot_iters)/wb_tot,h)
            iter = iter + 1;
            PRN = obs_data.data(data_idx,PRN_col);

            %------Calculate Geometric Range
            [R, rel_dt, satXYZ] = getSatGeomRange(rec_xyz', GPS_Week, GPS_SOW, PRN, nav_data, params);
```

```
            %------Check Elevation Angle
            [az, el(iter), r] = ecef2azelrange(satXYZ', rec_xyz);
            if el(iter) < 10
                iter=iter-1;
                tot_iters = tot_iters+1;
                continue
            end
            rel_corr = rel_dt*params.c;

            %------Get clock correction
            sat_clk_t_corr = getSatClockCorrection(GPS_Week, GPS_SOW, PRN, nav_data);

            %------Get Satellite Correction
            sat_corr = sat_clk_t_corr*params.c;

            %------Get Tropospheric Correction
            Tropo_corr = getTropoCorrection(Tzenith(file_idx), el(iter) );
            rho_model(iter) = R - sat_corr + Tropo_corr + rel_corr;
```

## Fetch Observed Pseudoranges

```
            %------Get Observed pseudorange
            if sum(strcmp(observables,'P1'))>0
                %------Retrieve P1 as pseudorange
                P1 = obs_data.data(data_idx,P1_col);
                P2 = obs_data.data(data_idx,P2_col);
                rho_obs(iter) = P1;
                ionFree(iter) = 2.5457*P1-1.5457*P2;
            else
                %------Retrieve C1 as pseudorange
                P2 = obs_data.data(data_idx,P2_col);
                C1 = obs_data.data(data_idx,C1_col);
                rho_obs(iter) = C1;
                ionFree(iter) = 2.5457*C1-1.5457*P2;
            end

            sat_prn(iter) = PRN;

            if iter < secs_idx(2) && file_idx == 1
                if sat == 1
                    fprintf('|_PRN_|___geomRange_____|___rel___|____satClk____|_______P3________|__Prefi
t___|___el___|__Trop___|\n')
                end
                fprintf(1,soln_format,PRN,...
                    R,rel_corr,sat_corr,ionFree(iter),...
                    ionFree(iter)-rho_model(iter),...
                    el(iter), Tropo_corr)
            end
```

```
|_PRN_|___geomRange_____|___rel___|____satClk____|_______P3________|__Prefit___|___el___|__Trop___|
|   8 |    23184871.678 |   0.170 |      658.362 |     23184839.757 |   621.408 | 29.37 | 4.86


|  13 |    20755252.832 |  -0.747 |    53957.609 |     20701918.525 |   621.280 | 59.51 | 2.77
```

```
|   2 |     22097650.863 |  -7.167 |    120949.414 |     21977320.598 |    622.368 | 37.19 | 3.95

|  23 |     23167048.158 |  -4.552 |     48800.721 |     23118868.512 |    619.834 | 24.32 | 5.79

|  30 |     24256720.422 |   1.159 |  -121274.267 |     24378632.870 |    625.139 | 11.58 | 11.88

|   5 |     22594455.897 |   1.699 |  -105918.494 |     22701002.705 |    622.124 | 32.08 | 4.49

|   7 |     20911335.144 |  -2.461 |     37162.821 |     20874793.510 |    620.936 | 61.61 | 2.71

|   4 |     23763019.853 |   5.954 |     92798.164 |     23670857.062 |    622.696 | 20.78 | 6.72

|  16 |     23201845.354 |   4.177 |   -73858.535 |     23276333.148 |    619.396 | 24.81 | 5.69

|  10 |     20302503.937 |   7.640 |   -15696.401 |     20318833.307 |    622.842 | 73.60 | 2.49

|_PRN_|___geomRange_____|___rel___|____satClk____|_______P3________|__Prefit___|___el___|__Trop___|
|   8 |     23165194.363 |   0.132 |       658.363 |     23165163.273 |    622.312 | 29.61 | 4.83
```

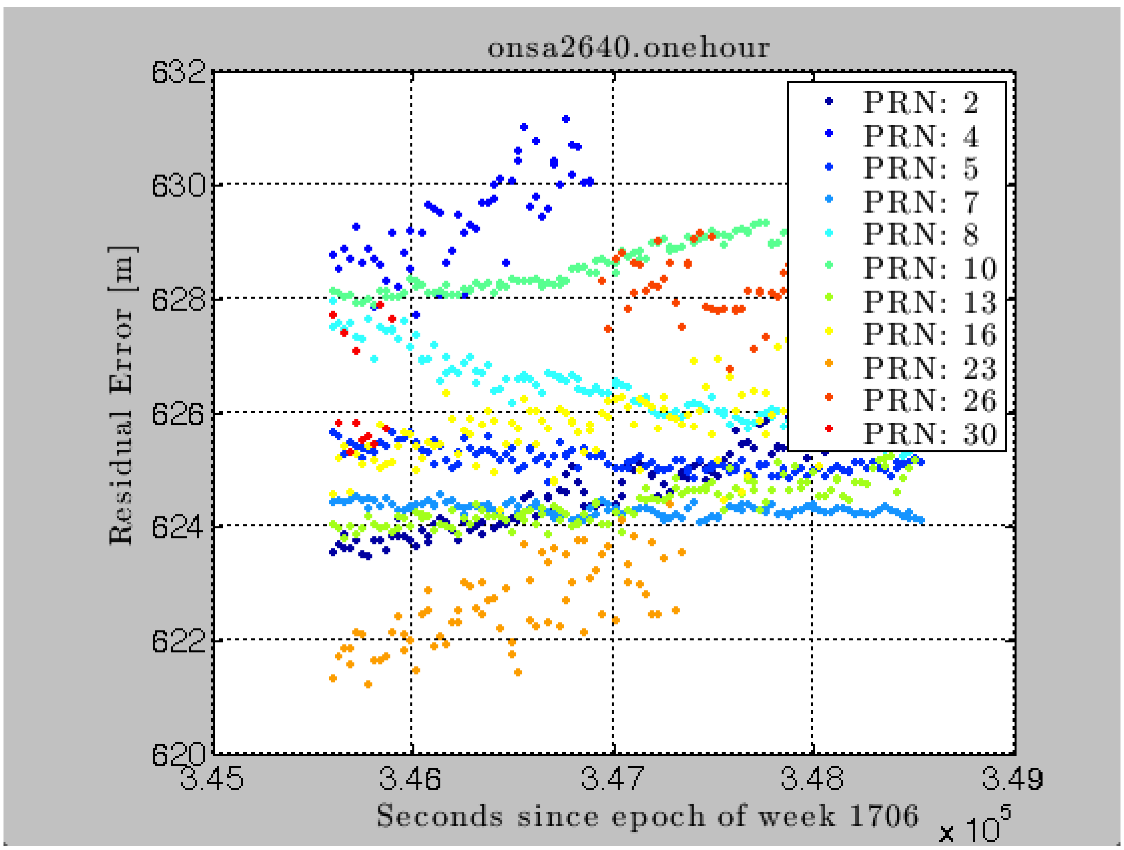
```
        end     % End Satellite Iteration
```

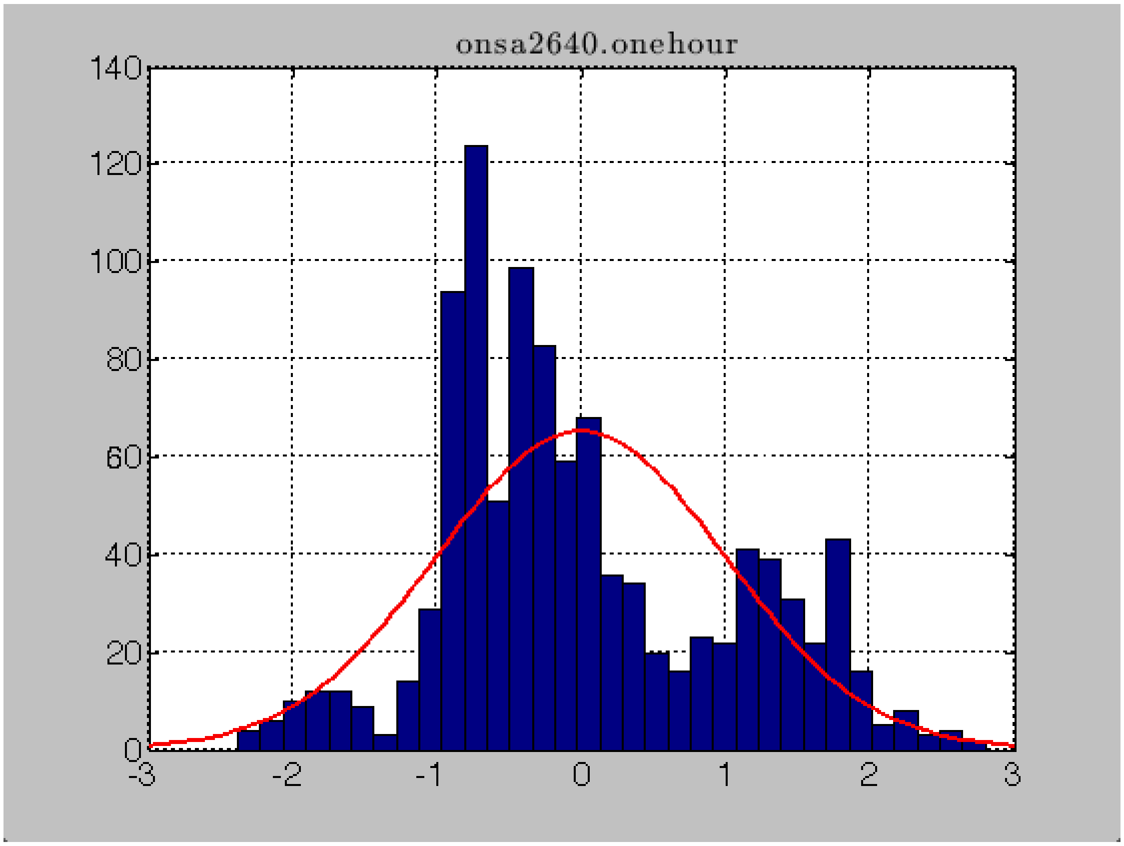
```
    end     % End time iteration

    %------Remove data with '0' observation
    zs = rho_obs == 0;
    rho_obs(zs) = []; rho_model(zs) = [];
```

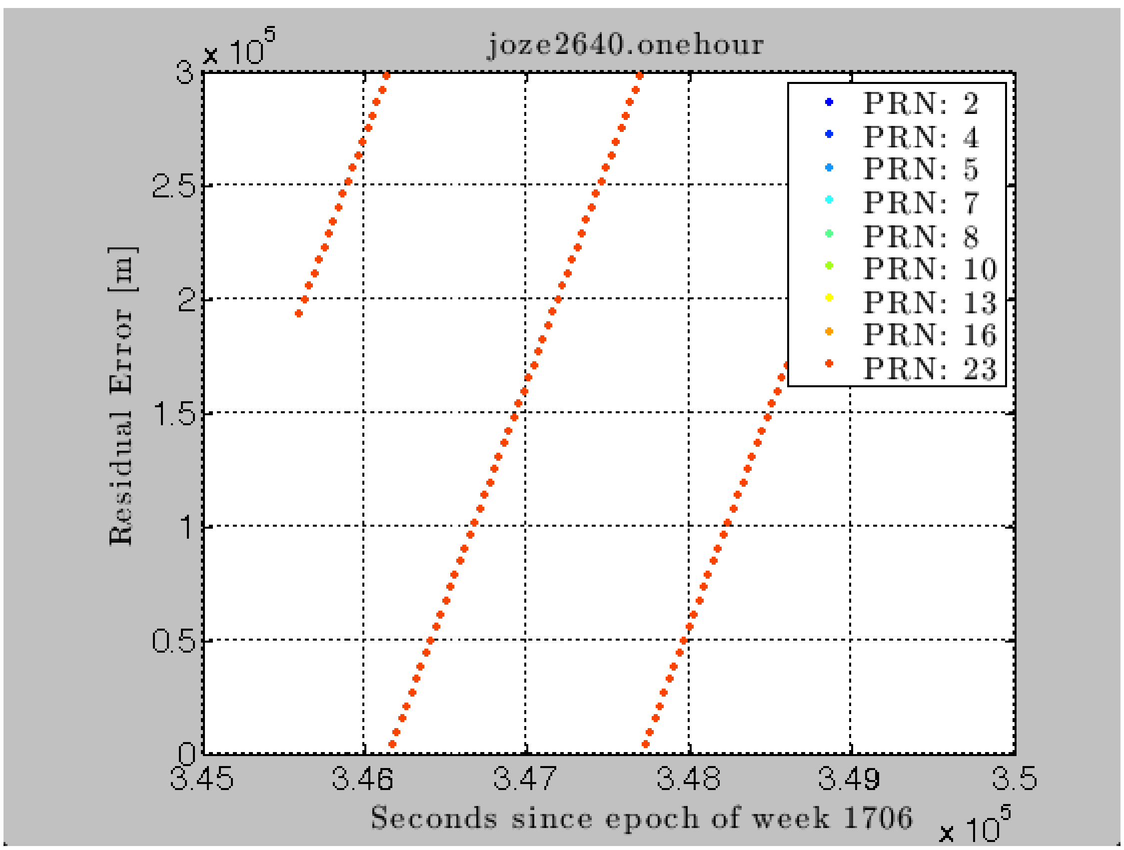
## Plot

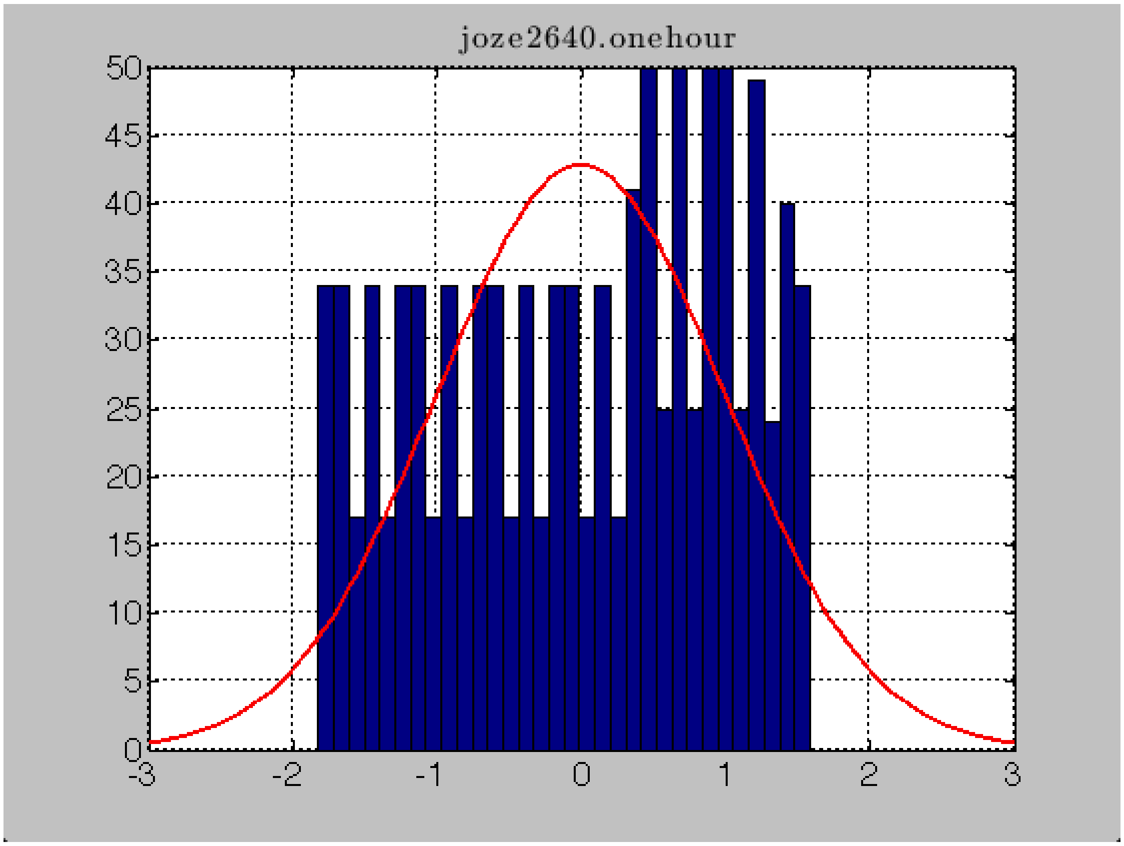
```
    res = rho_obs-rho_model;


    obs{file_idx} = rho_obs; %#ok<*SAGROW>
    model{file_idx} = rho_model;
    elevation{file_idx} = el;
    prefit_res{file_idx} = res()';
    prns = unique(sat_prn);
    prns(prns==0)=[];

    %------plot residuals
```

```
figure
colors = jet(length(prns));
for ii=1:length(prns)
    rows = sat_prn == prns(ii);
    sat_res{file_idx, prns(ii)} = res(rows);
    plot(GPSSecAry(rows),res(rows),'.','color',colors(ii,:),'Markersize',15)
    leg{ii} = ['PRN: ' , num2str(prns(ii))];
    hold on
end
xl = ['Seconds since epoch of week ',num2str(GPS_Week)];
xlabel(xl);ylabel('Residual Error [m]')
title(rinex_file)
legend(leg);

%------Plot Histogram
figure
datan = (res-mean(res))/std(res);
hist(datan)
histfit(datan)
tot_iters = tot_iters + iter;
title(rinex_file)
```




onsa2640.onehour
140
120
100
80
60
40
20
0
-3
-2
-1
0
1
2
3


joze2640.onehour
x 10^5
Residual Error [m]
Seconds since epoch of week 1706
x 10^5
PRN: 2
PRN: 4
PRN: 5
PRN: 7
PRN: 8
PRN: 10
PRN: 13
PRN: 16
PRN: 23
3
2.5
2
1.5
1
0.5
0
3.45
3.46
3.47
3.48
3.49
3.5




### Ion Free plot for Onsa

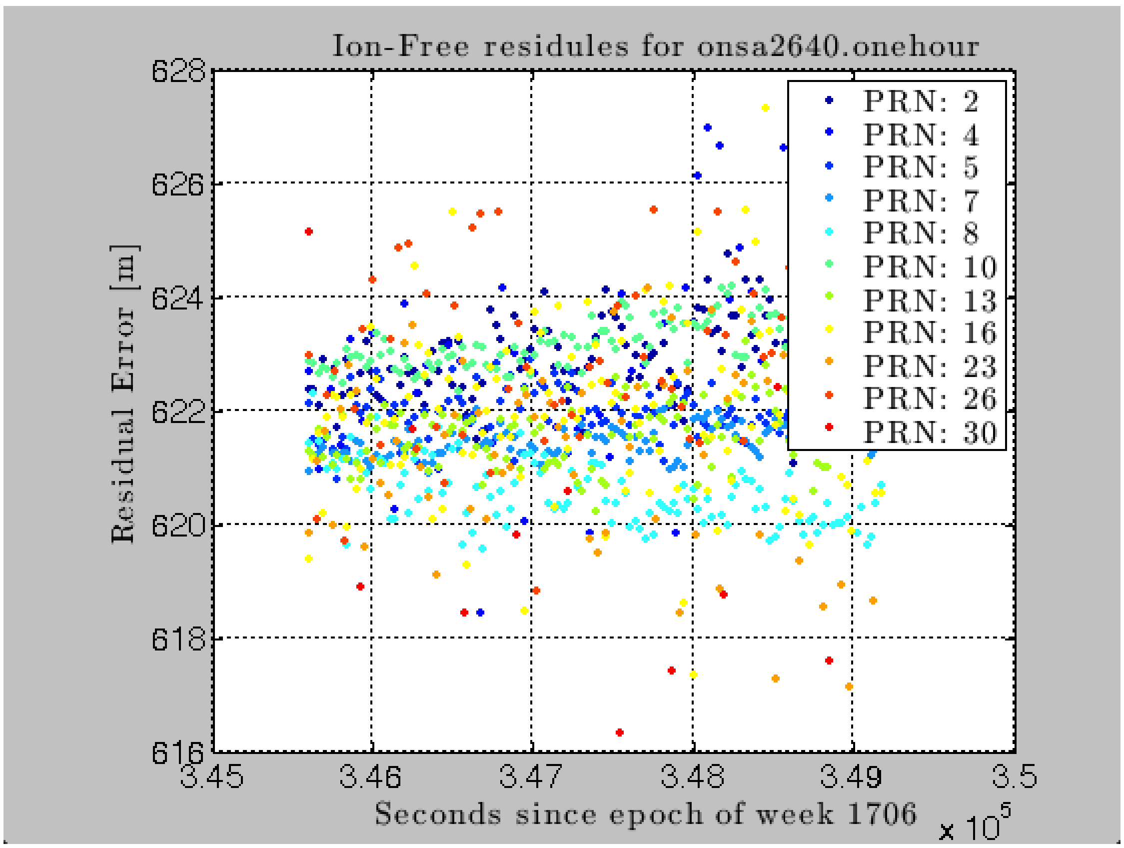
```
    if strcmp(rinex_file,'onsa2640.onehour')
```

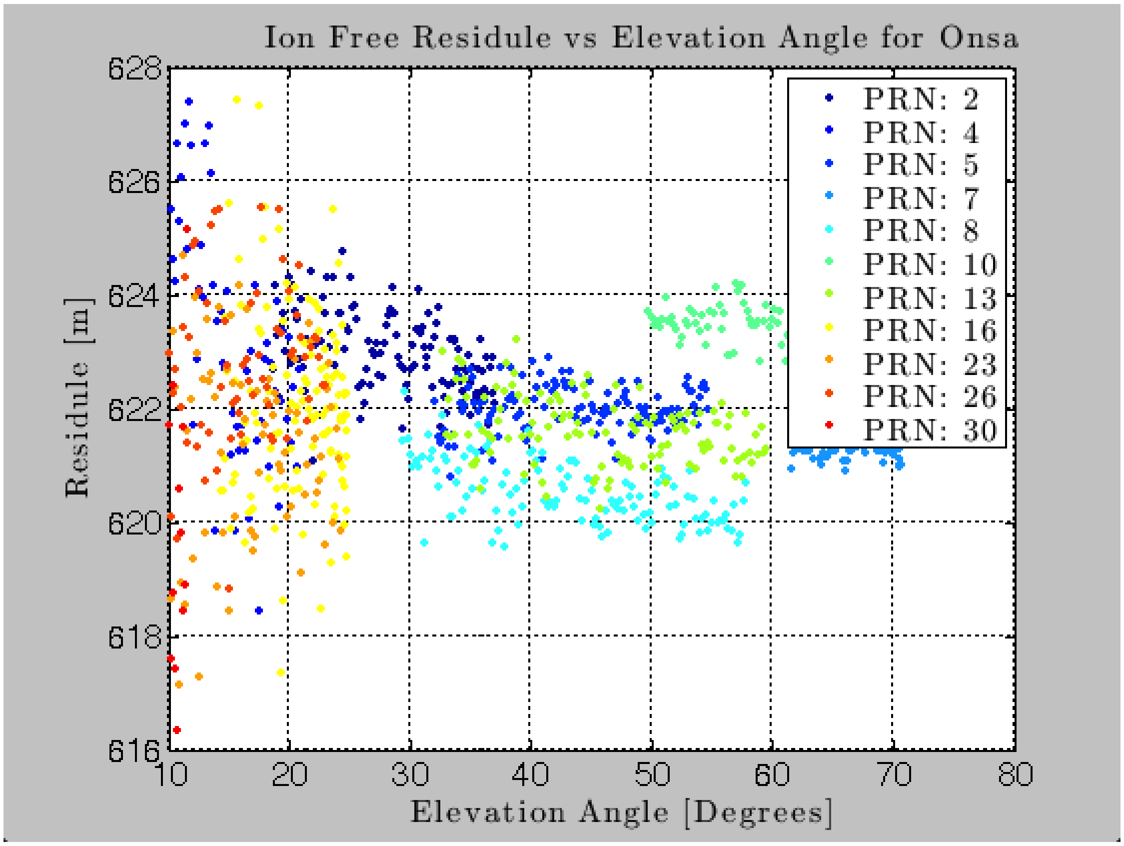
```
        %------Remove data with '0' observation
        ionFree(zs)=[];
```

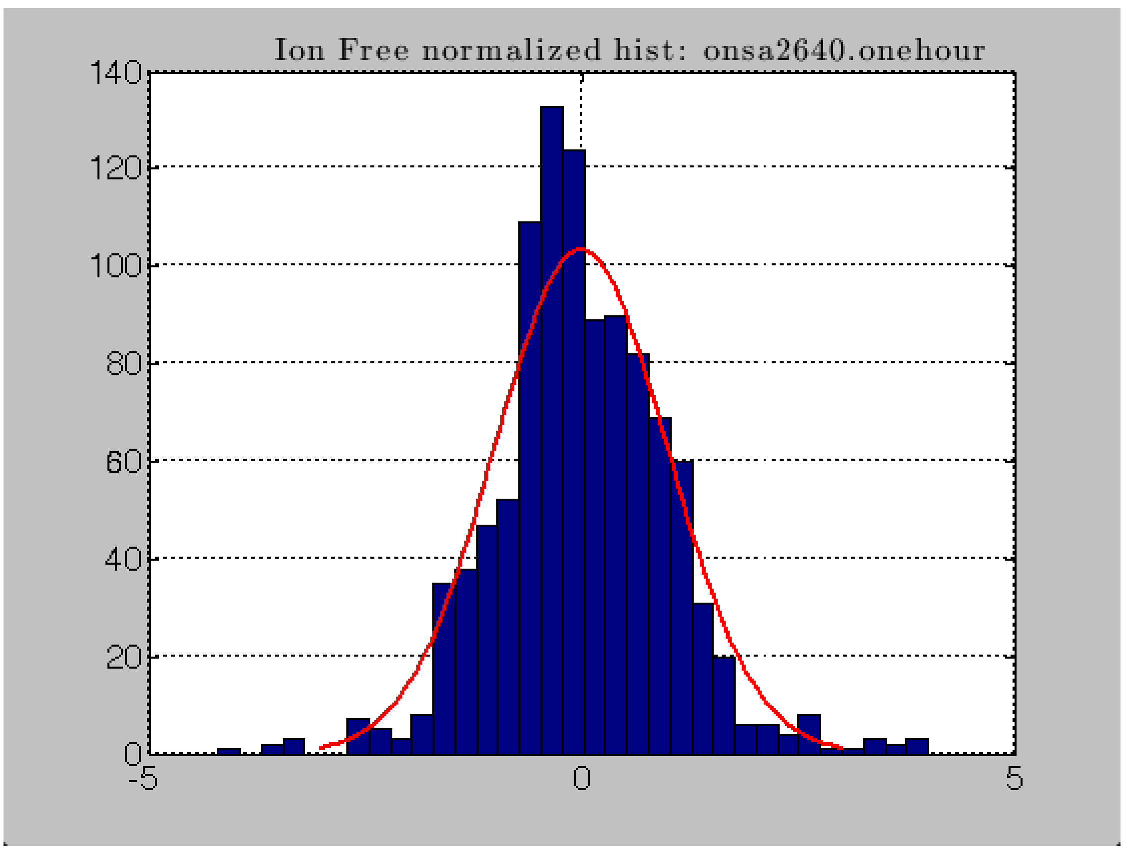
### Plot

```
        res = ionFree-rho_model;
        prns = unique(sat_prn);
        prns(prns==0)=[];

        %------plot residules
        figure
        colors = jet(length(prns));
        for ii=1:length(prns)
            rows = sat_prn == prns(ii);
            sat_res{file_idx, prns(ii)} = res(rows);
            plot(linspace(GPSSecAry(1),GPSSecAry(end),length(res(rows))),res(rows),'.','color',colors(ii
,:),'Markersize',15)
            leg{ii} = ['PRN: ' , num2str(prns(ii))];
            hold on
        end
        xl = ['Seconds since epoch of week ',num2str(GPS_Week)];
        xlabel(xl);ylabel('Residual Error [m]')
        title(['Ion-Free residules for ', rinex_file])
        legend(leg);
```

```
figure
for ii=1:length(prns)
    rows = sat_prn == prns(ii);
    sat_res{file_idx, prns(ii)} = res(rows);
    plot(el(rows),res(rows),'.','color',colors(ii,:),'Markersize',15)
    leg{ii} = ['PRN: ' , num2str(prns(ii))];
    hold on
end
xlabel('Elevation Angle [Degrees]');ylabel('Residule [m]')
title('Ion Free Residule vs Elevation Angle for Onsa')
legend(leg)

%------Plot Histogram
figure
datan = (res-mean(res))/std(res);
hist(datan)
histfit(datan)
title(['Ion Free normalized hist: ', rinex_file])
```





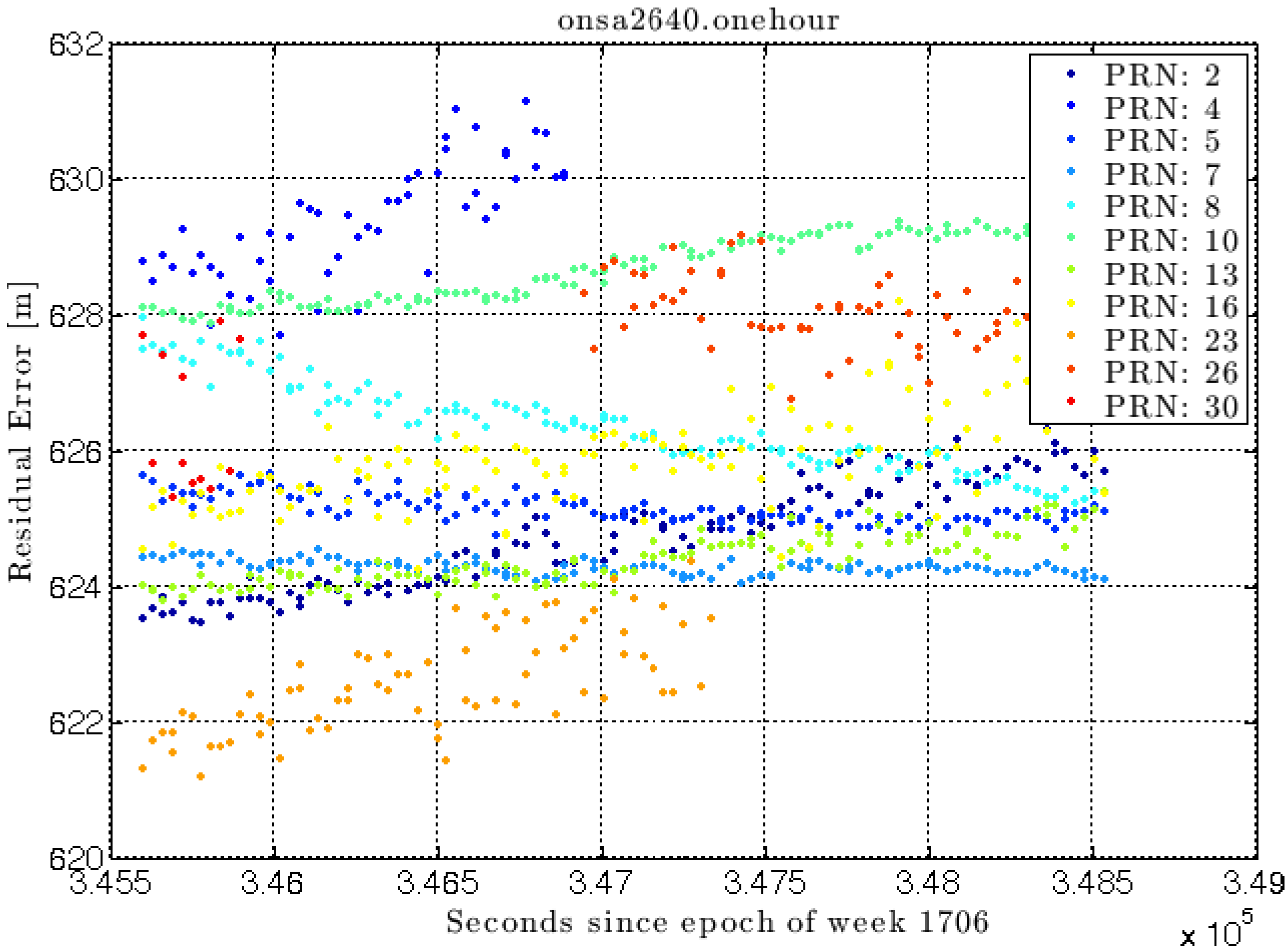



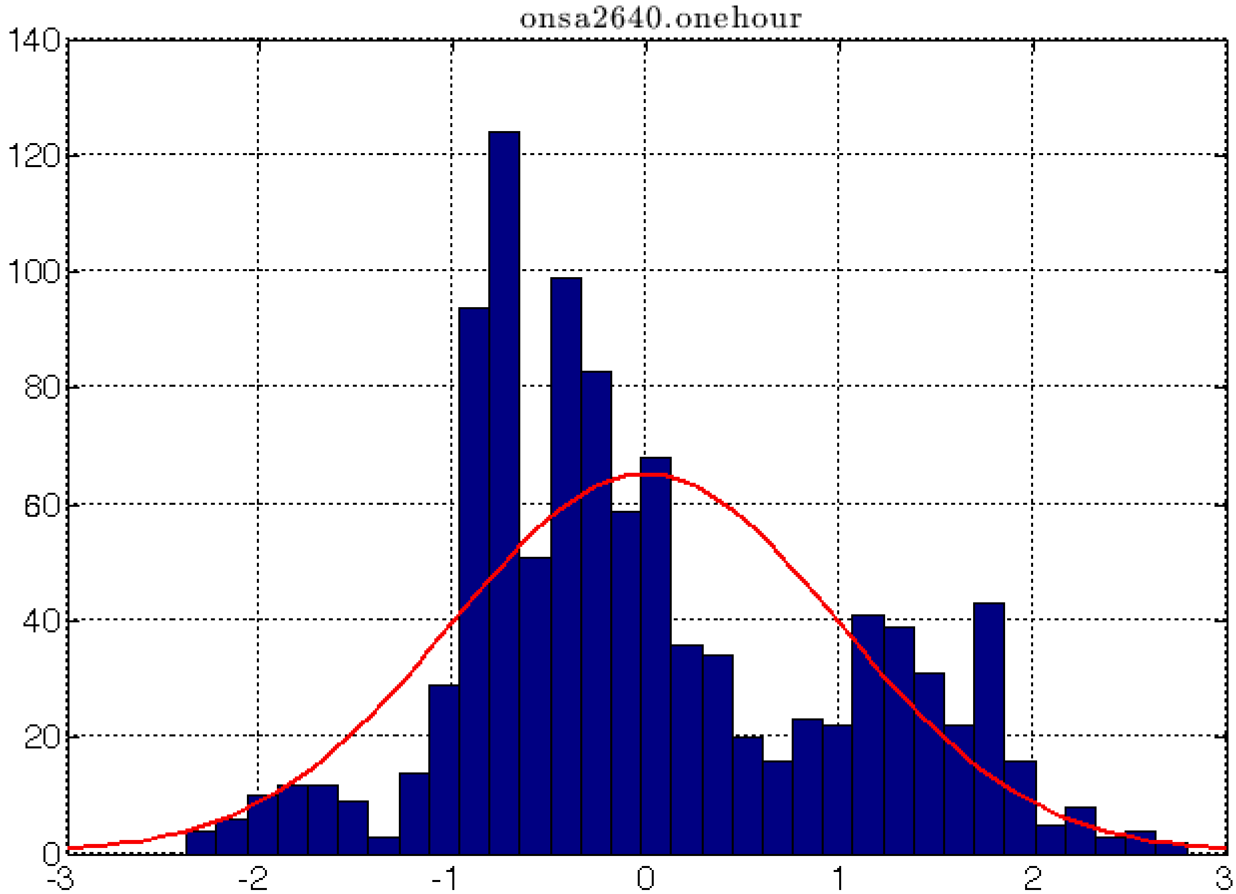

```
    end

    clear leg rows outliers sat_prn
```

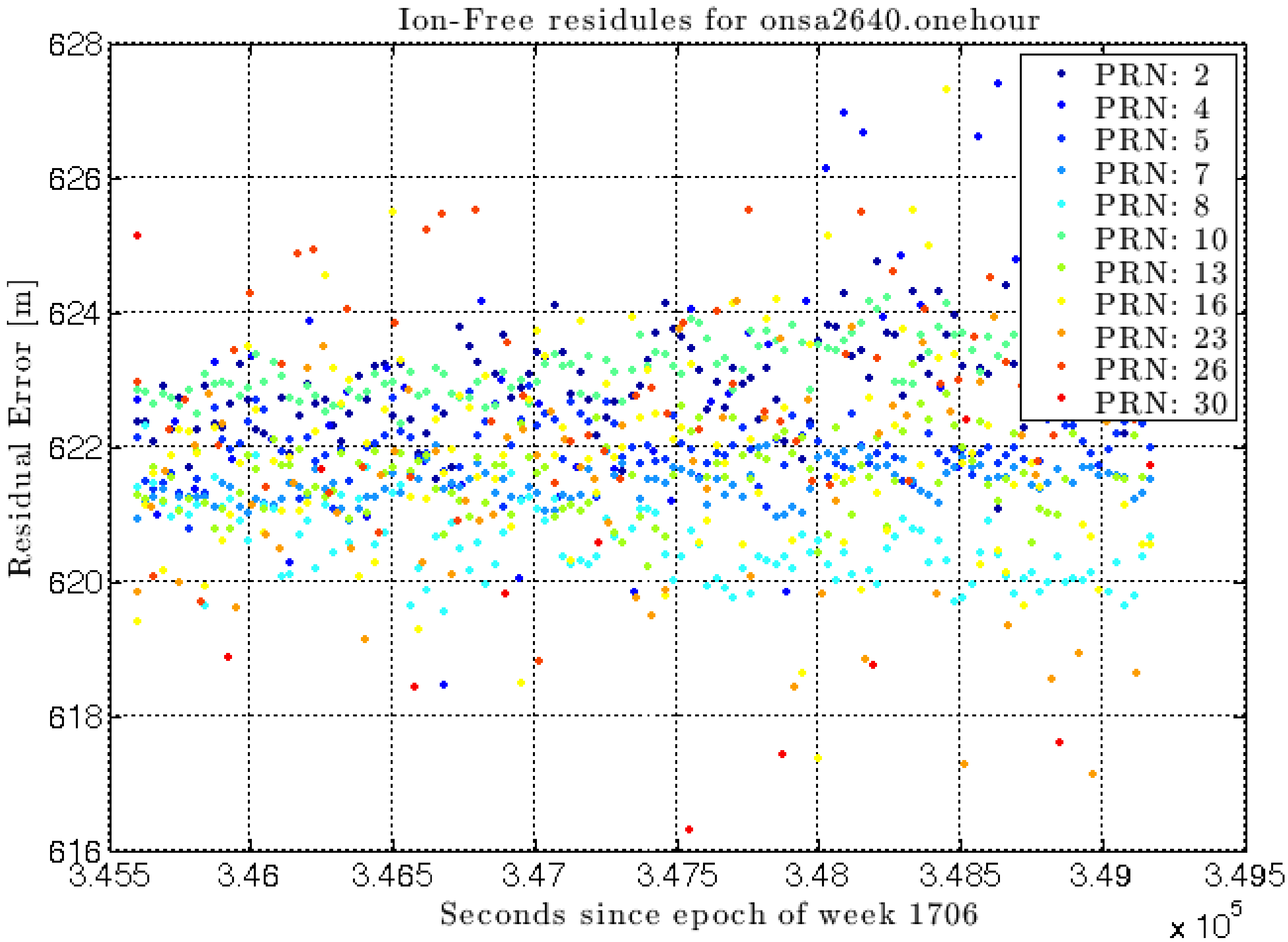
```
end     % End Rinex File Iteration
```

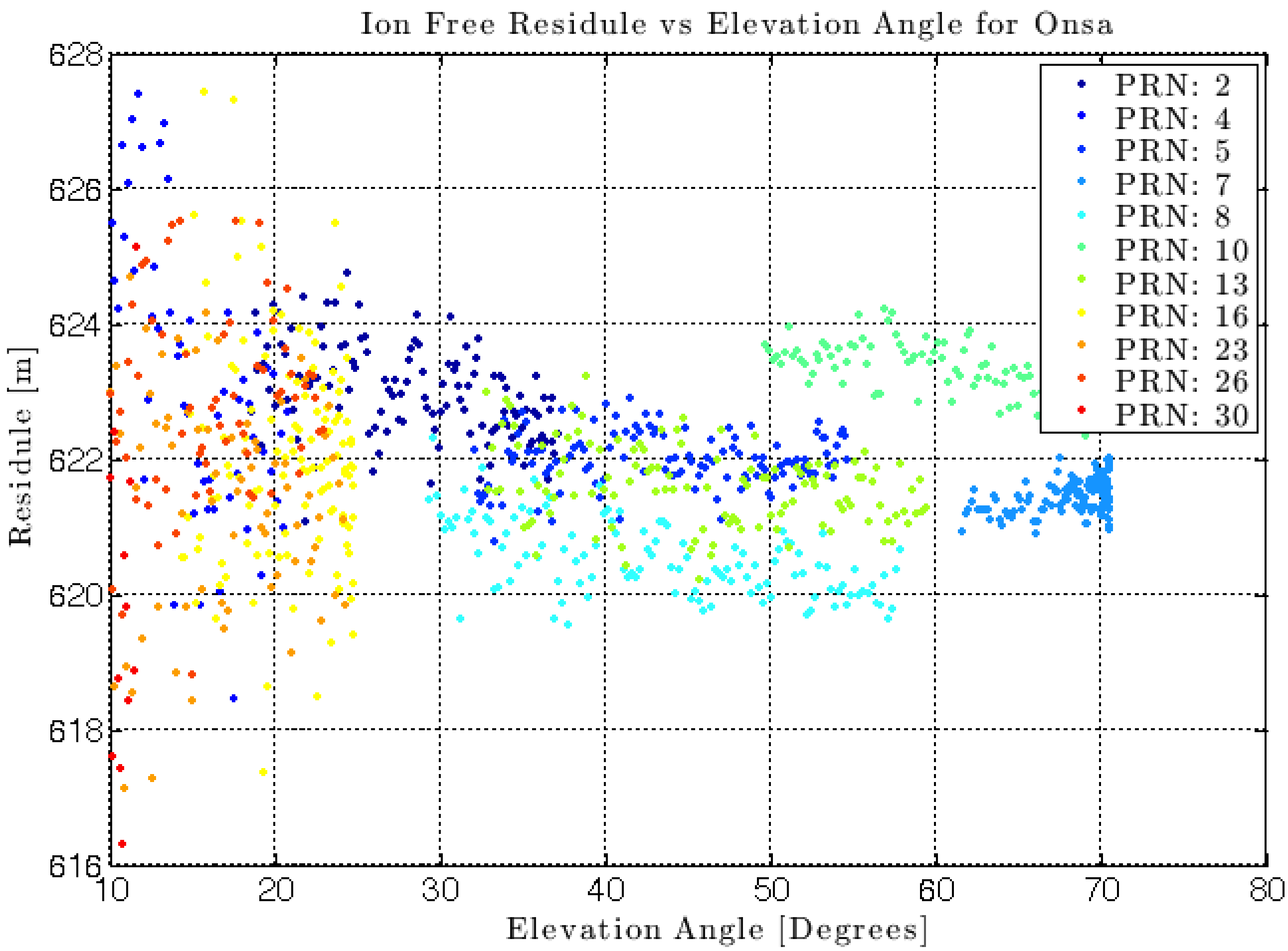
## Clean, Reformat

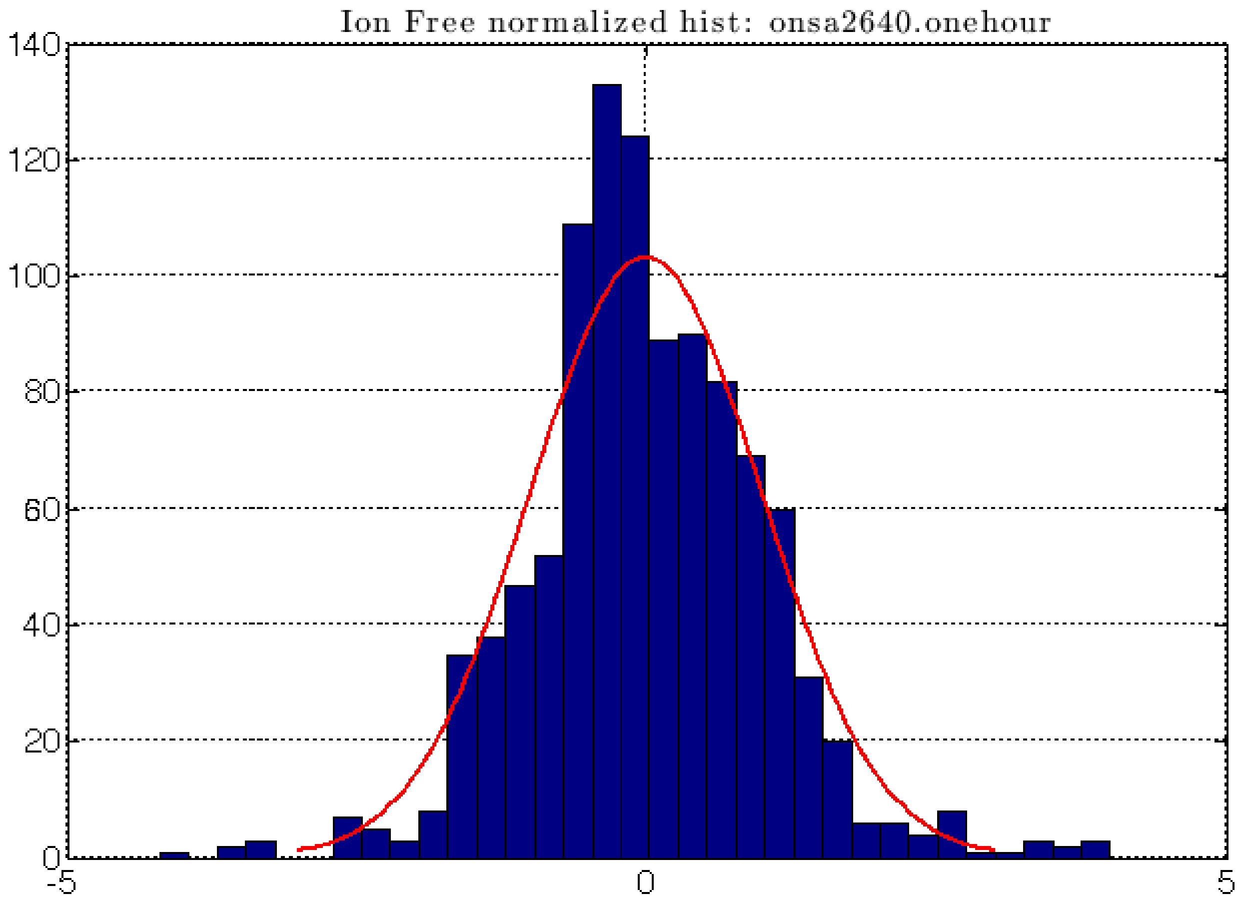
```
figure_awesome
```

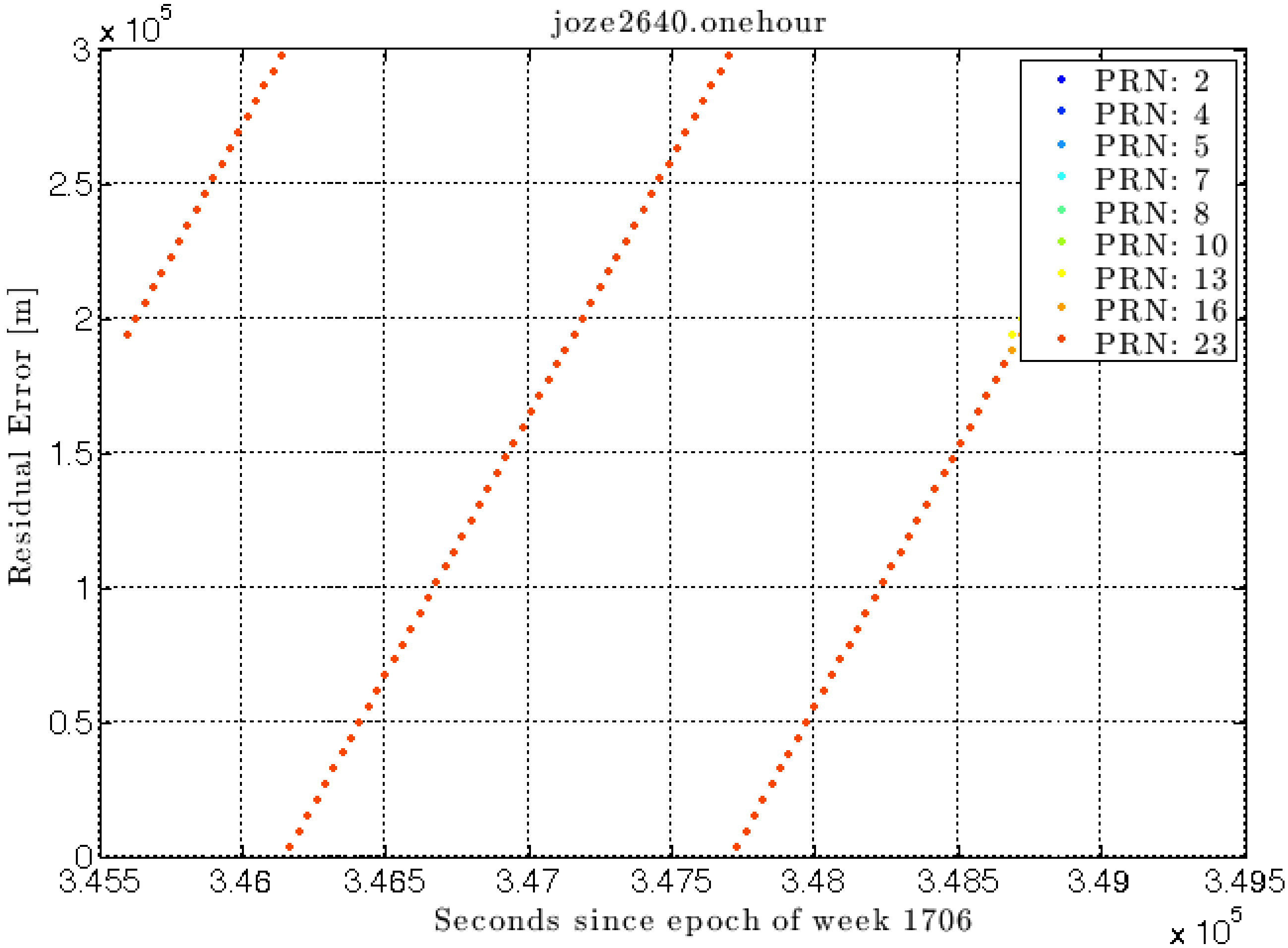


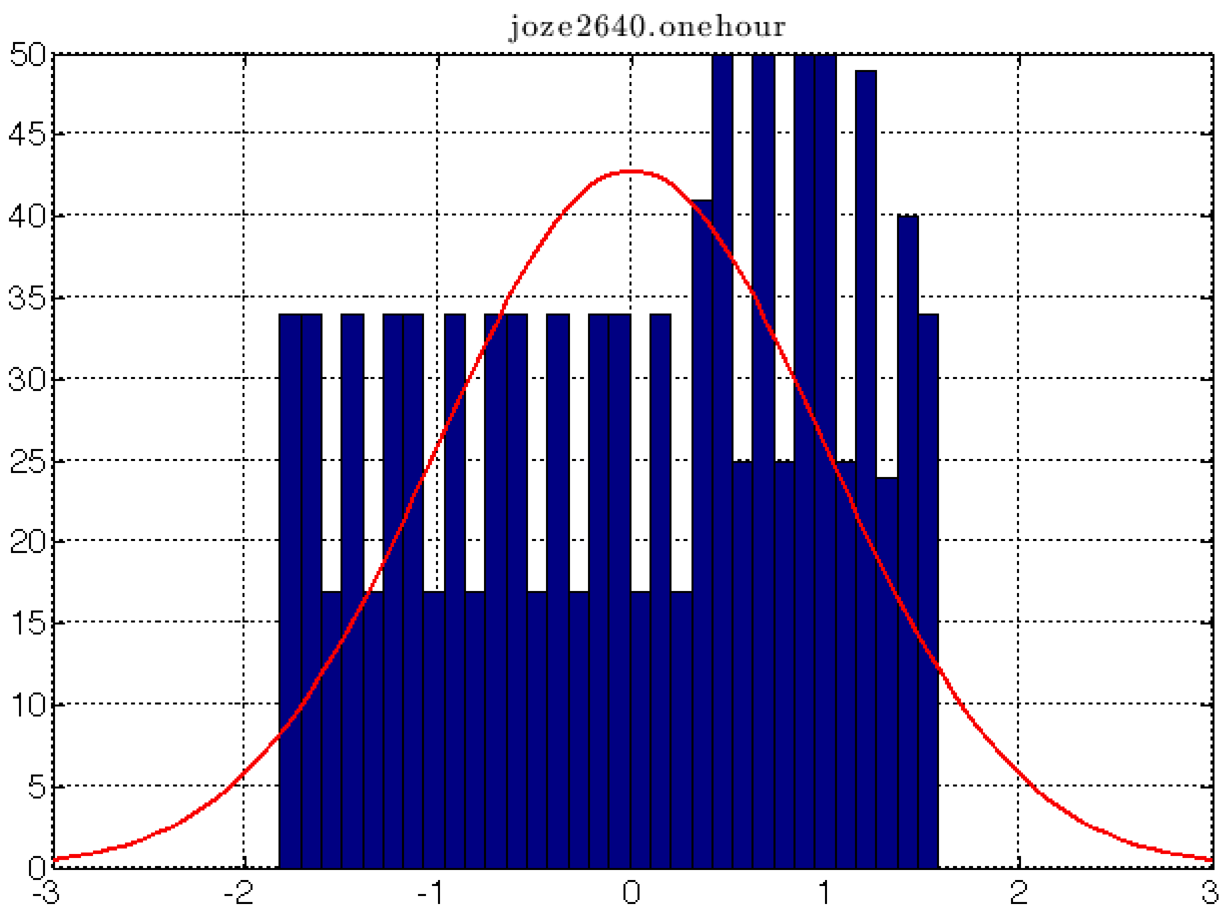

onsa2640.onehour
140
120
100
80
60
40
20
0
-3
-2
-1
0
1
2
3


Ion-Free residules for onsa2640.onehour
Residual Error [m]
Seconds since epoch of week 1706
x 10^5
PRN: 2
PRN: 4
PRN: 5
PRN: 7
PRN: 8
PRN: 10
PRN: 13
PRN: 16
PRN: 23
PRN: 26
PRN: 30
628
626
624
622
620
618
616
3.455
3.46
3.465
3.47
3.475
3.48
3.485
3.49
3.495


Ion Free Residule vs Elevation Angle for Onsa
628
626
624
622
620
618
616
10
20
30
40
50
60
70
80
Residule [m]
Elevation Angle [Degrees]
PRN: 2
PRN: 4
PRN: 5
PRN: 7
PRN: 8
PRN: 10
PRN: 13
PRN: 16
PRN: 23
PRN: 26
PRN: 30


Ion Free normalized hist: onsa2640.onehour
140
120
100
80
60
40
20
0
-5
0
5


joze2640.onehour
x 10^5
3
2.5
2
1.5
1
0.5
0
Residual Error [m]
3.455
3.46
3.465
3.47
3.475
3.48
3.485
3.49
3.495
x 10^5
Seconds since epoch of week 1706
PRN: 2
PRN: 4
PRN: 5
PRN: 7
PRN: 8
PRN: 10
PRN: 13
PRN: 16
PRN: 23




## SUPPORTING FUNCTION - Homework7_main.m

```
type('Homework7_main.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        Homework7_main.m
% Author      : Zach Dischner
% Date        : 10/27/2013
% Description : Matlab script for all calculations required for
%               ASEN 5090 Homework 7
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--..______
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
%% Questions to Answer (First for ease of grading)
type('answers.txt')

%% Setup Work Space
```

```
clc;clear all;close all
tic;
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 HW7_files
soln_format = '|  %2.0f | %15.3f | %7.3f | %12.3f | %15.3f | %9.3f | %3.2f | %3.2f  \t\n';
% h = waitbar(0,'GO PARTY, ILL STAY HERE WORKING!!!');
wb_tot = 1024+1238;
tot_iters = 0;


%% Setup Problem
%------Define Navigation and Observation File
RINEX_FILES = {'onsa2640.onehour','joze2640.onehour'};
nav_msg  = 'brdc2640.12n';

%------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]

params.options=optimset('Display','off','TolFun',1e-10,'TolX',1e-10);


%------Define Zenith Correction for each site
Tzenith = [2.3858, 2.4086]; %[m]


%------Read navigation message content
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 latitu
de
%                 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 inclinati
on
%                 col16: Cis, amplitude of the cosine harmonic correction term to the angle of inclinati
on
%                 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)

%% Calculate  Pre-Fit Residuals
for file_idx=1:length(RINEX_FILES)
    rinex_file = RINEX_FILES{file_idx};
    fprintf('Processing Rinex file %s\n', rinex_file)

    %% Read Observation Files, Sort Data
    %------Read a-priori receiver position from header of RINEX obs file
    [ fid, rec_xyz, observables ] = read_rinex_header( rinex_file );

    %------Read Observation file
    obs_data    = read_rinex_obs3(rinex_file);
    cols        = obs_data.col; % Structure of column index descriptions
    %------Make nice column addressing variables (P1_col, P2_col ... etc)
    fields = fieldnames(cols);
    for kk=1:length(fields)
        eval(cell2mat([fields(kk),'_col = cols.',fields(kk),';']));
    end

    PRNS = obs_data.data(:,PRN_col);
    [GPSSecAryUn,secs_idx] = unique(obs_data.data(:,TOW_col));
    GPSWeekAry = obs_data.data(:,WEEK_col);
    GPSSecAry = obs_data.data(:,TOW_col);
    %     GPS_Secs = obs_data.data(rows,SOW_col);
    %     GPS_Weeks = obs_data.data(rows,Week_col);



    %% Fetch/Compute Pseudorange Values

    %------Allocate
    rho_obs = zeros(1, length(PRNS));
    rho_model = rho_obs;
    el = rho_obs;
    res = rho_obs;
    ionFree = rho_obs;
    sat_prn = rho_obs;
    iter = 0;
    for sec_idx = secs_idx'

        %% Calculate "Modeled" Pseudorange
        %------Setup Range Finding
        GPS_SOW = GPSSecAry(sec_idx);
        GPS_Week = GPSWeekAry(sec_idx);
        params.Secs = GPS_SOW; %(GPS_Secs(1))  % Seconds used to calculate seconds since epoch

        %------Iterate over epoch satellite data
        Nsats = sum(GPSSecAry == GPSSecAry(sec_idx));
        for sat = 1:Nsats
            data_idx = sec_idx+sat-1;
%             waitbar((data_idx+tot_iters)/wb_tot,h)
            iter = iter + 1;
            PRN = obs_data.data(data_idx,PRN_col);
```

```
            %------Calculate Geometric Range
            [R, rel_dt, satXYZ] = getSatGeomRange(rec_xyz', GPS_Week, GPS_SOW, PRN, nav_data, params);

            %------Check Elevation Angle
            [az, el(iter), r] = ecef2azelrange(satXYZ', rec_xyz);
            if el(iter) < 10
                iter=iter-1;
                tot_iters = tot_iters+1;
                continue
            end
            rel_corr = rel_dt*params.c;

            %------Get clock correction
            sat_clk_t_corr = getSatClockCorrection(GPS_Week, GPS_SOW, PRN, nav_data);

            %------Get Satellite Correction
            sat_corr = sat_clk_t_corr*params.c;

            %------Get Tropospheric Correction
            Tropo_corr = getTropoCorrection(Tzenith(file_idx), el(iter) );
            rho_model(iter) = R - sat_corr + Tropo_corr + rel_corr;

            %% Fetch Observed Pseudoranges

            %------Get Observed pseudorange
            if sum(strcmp(observables,'P1'))>0
                %------Retrieve P1 as pseudorange
                P1 = obs_data.data(data_idx,P1_col);
                P2 = obs_data.data(data_idx,P2_col);
                rho_obs(iter) = P1;
                ionFree(iter) = 2.5457*P1-1.5457*P2;
            else
                %------Retrieve C1 as pseudorange
                P2 = obs_data.data(data_idx,P2_col);
                C1 = obs_data.data(data_idx,C1_col);
                rho_obs(iter) = C1;
                ionFree(iter) = 2.5457*C1-1.5457*P2;
            end

            sat_prn(iter) = PRN;

            if iter < secs_idx(2) && file_idx == 1
                if sat == 1
                    fprintf('|_PRN_|___geomRange_____|___rel___|____satClk____|_______P3________|__Prefi
t___|___el___|__Trop___|\n')
                end
                fprintf(1,soln_format,PRN,...
                    R,rel_corr,sat_corr,ionFree(iter),...
                    ionFree(iter)-rho_model(iter),...
                    el(iter), Tropo_corr)
            end

        end     % End Satellite Iteration
    end     % End time iteration

    %------Remove data with '0' observation
    zs = rho_obs == 0;
    rho_obs(zs) = []; rho_model(zs) = [];
```

```
%% Plot
res = rho_obs-rho_model;


obs{file_idx} = rho_obs; %#ok<*SAGROW>
model{file_idx} = rho_model;
elevation{file_idx} = el;
prefit_res{file_idx} = res()';
prns = unique(sat_prn);
prns(prns==0)=[];

%------plot residules
figure
colors = jet(length(prns));
for ii=1:length(prns)
    rows = sat_prn == prns(ii);
    sat_res{file_idx, prns(ii)} = res(rows);
    plot(GPSSecAry(rows),res(rows),'.','color',colors(ii,:),'Markersize',15)
    leg{ii} = ['PRN: ' , num2str(prns(ii))];
    hold on
end
xl = ['Seconds since epoch of week ',num2str(GPS_Week)];
xlabel(xl);ylabel('Residual Error [m]')
title(rinex_file)
legend(leg);

%------Plot Histogram
figure
datan = (res-mean(res))/std(res);
hist(datan)
histfit(datan)
tot_iters = tot_iters + iter;
title(rinex_file)

%% Ion Free plot for Onsa
if strcmp(rinex_file,'onsa2640.onehour')
    %------Remove data with '0' observation
    ionFree(zs)=[];
    %% Plot
    res = ionFree-rho_model;
    prns = unique(sat_prn);
    prns(prns==0)=[];

    %------plot residules
    figure
    colors = jet(length(prns));
    for ii=1:length(prns)
        rows = sat_prn == prns(ii);
        sat_res{file_idx, prns(ii)} = res(rows);
        plot(linspace(GPSSecAry(1),GPSSecAry(end),length(res(rows))),res(rows),'.','color',colors(ii
,:),'Markersize',15)
        leg{ii} = ['PRN: ' , num2str(prns(ii))];
        hold on
    end
    xl = ['Seconds since epoch of week ',num2str(GPS_Week)];
    xlabel(xl);ylabel('Residual Error [m]')
    title(['Ion-Free residules for ', rinex_file])
    legend(leg);
```

```
        figure
        for ii=1:length(prns)
            rows = sat_prn == prns(ii);
            sat_res{file_idx, prns(ii)} = res(rows);
            plot(el(rows),res(rows),'.','color',colors(ii,:),'Markersize',15)
            leg{ii} = ['PRN: ' , num2str(prns(ii))];
            hold on
        end
        xlabel('Elevation Angle [Degrees]');ylabel('Residule [m]')
        title('Ion Free Residule vs Elevation Angle for Onsa')
        legend(leg)

        %------Plot Histogram
        figure
        datan = (res-mean(res))/std(res);
        hist(datan)
        histfit(datan)
        title(['Ion Free normalized hist: ', rinex_file])

    end

    clear leg rows outliers sat_prn

end     % End Rinex File Iteration


%% Clean, Reformat
figure_awesome


%% SUPPORTING FUNCTION - Homework7_main.m
type('Homework7_main.m')

%% SUPPORTING FUNCTION - getSatGeomRange.m
type('getSatGeomRange.m')

%% SUPPORTING FUNCTION - date2GPSTime.m
type('date2GPSTime.m')

%% SUPPORTING FUNCTION - findNearestEphem.m
type('findNearestEphem.m')

%% SUPPORTING FUNCTION - calculateSatellitePosition.m
type('calculateSatellitePosition.m')

%% SUPPORTING FUNCTION - findFirstEpoch.m
type('findFirstEpoch.m')

%% SUPPORTING FUNCTION - getSatClockCorrection.m
type('getSatClockCorrection.m')

%% SUPPORTING FUNCTION - date2GPSTime.m
type('GPSTime2Date.m')

%% SUPPORTING FUNCTION - getTropoCorrection.m
type('getTropoCorrection.m')
```

```
fprintf('\nSim took %3.1f seconds to run\n',toc)
```

## SUPPORTING FUNCTION - getSatGeomRange.m

```
type('getSatGeomRange.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                         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       : R - Geometric Range value, in meters
%                 rel_dt - clock offset due to relativity (in seconds)
%                 rSat - 3d ECI coordinates of satellite [x,y,z]
%
% History       October 11 2013 - First Rev
%               October 24 2013 - Reformatted output to [R,tk]
%                               - Added check for time field in params,
%                                 other that that in the ephemeris data
%               October 30 2013 - Reformatted output to include XYZ sat
%                                 position
%
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [R, rel_dt, rSat] = 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-15;
```

```
max_iters = 200;
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)
        fprintf(2,'YO BRO!! Range Calculation not converging!\n')
        break
    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,params] = calculateSatellitePosition(epochData,params);

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

end
rmfield(params,'E_guess');
```

## 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 = difference
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
%                Oct 31 2013 - Changed all terms to datenums, to account for
%                                 week changeover
```

```
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

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

% weeknums = nav_ephem(:,19);
% secofweeks = nav_ephem(:,17)
% sec_diff = abs(navData(:,17)-GPS_SOW);
% rownums = find( (sec_diff) == min(sec_diff) & ismember(navData(:,1),PRN) & navData(:,19)==GPS_Weeks);
% rownums = find( navData(:,17)<=GPS_SOW & ismember(navData(:,1),PRN) & navData(:,19)==GPS_Weeks);
GPSNUMBOOL = 1;
epoch_time = GPSTime2Date(GPS_Weeks, GPS_SOW, GPSNUMBOOL);
nav_time   = GPSTime2Date(navData(:,19),navData(:,17), GPSNUMBOOL);
datediff = abs(nav_time-epoch_time);
rownums = (datediff==min(datediff) & ismember(navData(:,1),PRN));

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]
%               October 30 2013 - Added params to return, check for a
%                                   better Ek guess (to speed up 'find')
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function [rk,dt_rel, params] = 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
if isfield(params,'E_guess')
    guess=params.E_guess;
else
    guess=0;
end
Ek = fsolve(@(Ek) (Ek)-ecc*sin(Ek)-Mk,0,params.options);
params.E_guess = Ek;

%------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 - getSatClockCorrection.m

```
type('getSatClockCorrection.m')
```

```
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
```

## SUPPORTING FUNCTION - date2GPSTime.m

```
type('GPSTime2Date.m')
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        GPSTime2Date.m
% Author      : Zach Dischner
% Date        : 10/24/2013
% Description : Convert a date type object into [GPS_Weeks, GPS_SOW] time
%
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--.._____
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `""""""""""`
% Inputs        : [GPS_Weeks, GPS_SOW]-weeks and seconds of week
%
% Outputs       : utcDate - Satellite PRN number
%
% TODOS         : Vectorize!
%                 Build in mod options
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
function utcDate  = GPSTime2Date(GPS_Weeks, GPS_SOW, GPSNUM_BOOL)
% gps_week_start = 'January 6 1980 00:00:00';
gps_weeks_start = 723186;   % datenum(gps_week_start) Save time
%------GPS date in numerical form, since Matlab's 'epoch'
GPS_Num = (GPS_Weeks+GPS_SOW/7/24/3600)*7 + gps_weeks_start;
if nargin == 3
    if GPSNUM_BOOL == 1
        utcDate = GPS_Num;
    else
        utcDate = datestr(datevec(GPS_Num));
    end
else
    utcDate = datestr(datevec(GPS_Num));
end
```

## SUPPORTING FUNCTION - getTropoCorrection.m

```
type('getTropoCorrection.m')

fprintf('\nSim took %3.1f seconds to run\n',toc)
```

```
%>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
%                        getTropoCorrection.m
% Author      : Zach Dischner
% Date        : 10/30/2013
% Description : Retrieve tropospheic correction from simple model (in
%               meters)
%
%
%                 __...____________________          ,
%                `(\ [ ===NCC-1700===--|__|) ___..--"_`--.._____
%                  `"""""""""""""""""| |""` [_""_-___________"_/
%                                    | |   /..../`'-._.-'`
```

```
%                                ____| |__/::..'_
%                               |\ ".`"` '_____//\
%                               `"'-.   """""  \\/
%                                    `"""""""""""`
% Inputs        : ZenithCorr - Zenith delay [meters]
%                   el - elevation angle in degrees
%
% Outputs       : TropoDelay - Delay value [meters]
%
% History       October 30 2013 - First Rev
%
% <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

function TropoDelay = getTropoCorrection(ZenithCorr, el)
TropoDelay = abs(ZenithCorr/sind(el));
```

```
Sim took 498.1 seconds to run
```

*Published with MATLAB® R2013b*