OL Tracking: Code Usage Instructions v 2.0

Ulvi Acikoz

July 7, 2011

Contents

List of Tables

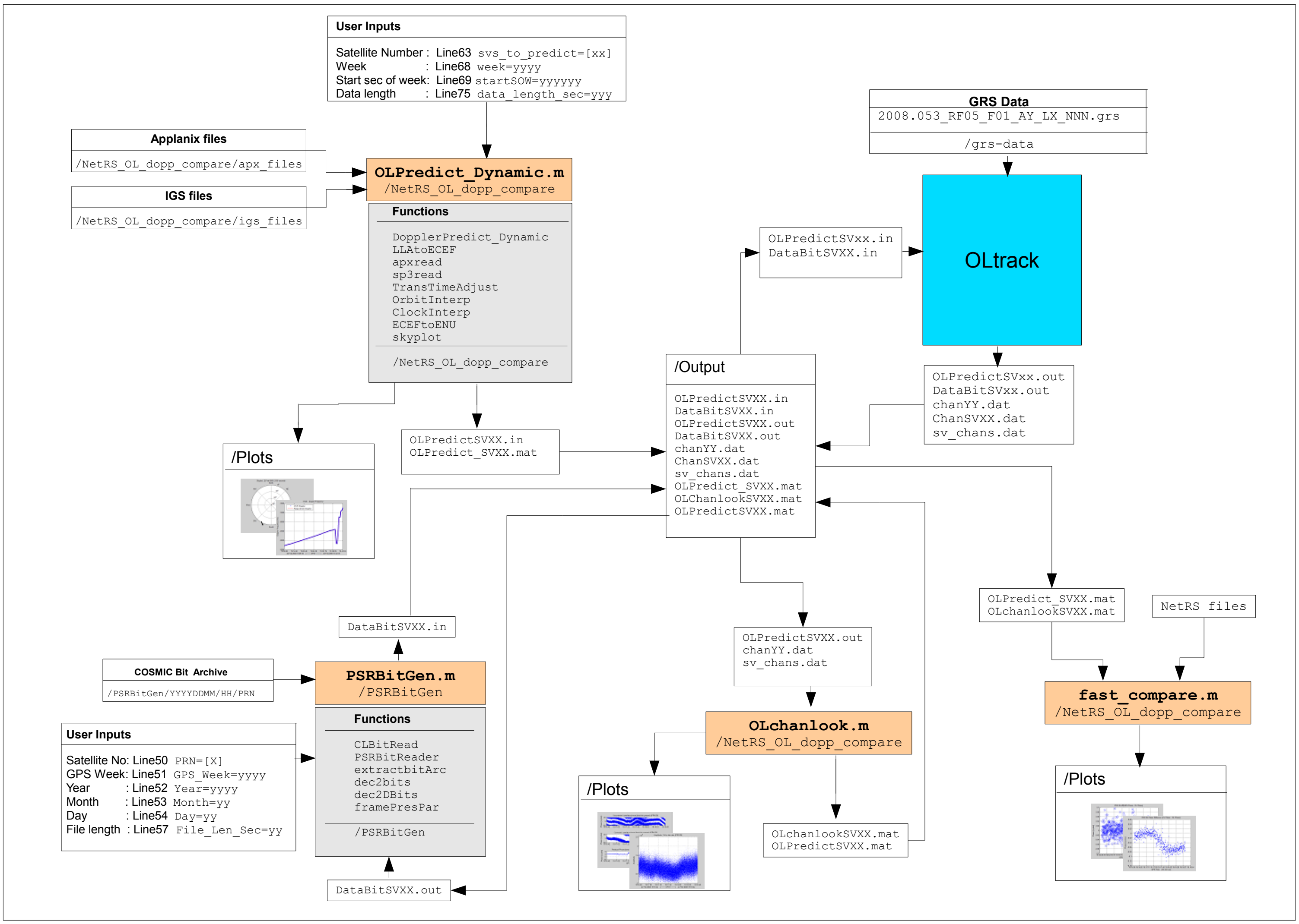

Figure 1. OL Tracking Usage Chart

1 Main Program Names(file path)

i. OLtrack (/ OLtrack)

ii. OLPredict_Dynamic.m (/ NetRS_OL_dopp_compare)

iii. PSRBitGen.m (/ PSRBitGen)

iv. OLchanlook.m (/ NetRS_OL_dopp_compare)

v. fast_compare.m (/ NetRS_OL_dopp_compare)

2 Compiling the OLtrack

The OLtrack requires Linux OS with g++ library to compile. The program includes a "makefile" which holds instructions for all "makes". Also, the compiler options and code paths are written into this file. The makefile is placed in /OLtrack directory. To compile the program, go to this directory and type

```
make
```

If the compiling is successful this massage appears

```
--- Build Complete ---
```

and the executable file *OLtrack* is created into the same directory.

3 Using the OLtrack

OLtrack is designed to track both setting and rising occultations. So, it can be run with two different options,

(1) Forward tracking option (default): `./OLtrack -f`
(2) Backward tracking option: `./OLtrack -b`

The OLtrack accepts the file name and file number of a GRS file. The filename entered by the user is a truncated version of the original that drops all characters after the standard "AX_LY" or "CHX_LY" ending, where X is (0,1,2) for the three different channels and Y is (1,2) for L1 or L2 band.

Example:

```
Filename:  /media/OCCevent/2008.053_RF05_F01_A2_L1
Msec per file:  399001
First file number:  1
Last file number:  5
```

This will cause the OLtrack start with file:

2008.053_RF05_F01_A0_L1_399001_399001.grs

which is the File 1 and the tracking will continue until the end of the File 5. In backward tracking mode tracking starts with the first file and goes backward, so the first file number must greater than or equal to the last file number. If the file is opened successfully this message appears on the screen,

```
File 1:/media/OCCevent/2008.053_RF05_F01_A2_L1_399001_399001.grs
```

and if the file could not find in the given directory it gives the following error message.

```
Unable to Open file!
```

In backward tracking, the program is not able to track bacdwardly more than one satellite at the same time. So, after CL tracking ended the program wants a user input to choose a channel then starts backward OL tracking only for this channel.

The OLtrack outputs files to:

/Output

This is the location which the OLtrack will search for OL input files (OLPredictSVXX.in, DataBitSVXX.in) and put output files (OLPredictSVXX.out, DatabitSVXX.out, chanYY.dat, sv_chans.dat).

4 Using OLPredict.m

The main user inputs will be the

i. SV number

ii. GPS week

iii. MSOW (starting millisecond of the week)

iv. prediction length in seconds

v. name of the applanix position and velocity file

It requires the presence of the IGS orbit file and Applanix (apx) position and velocity file in the same directory, or in an added path. The best course of action is to add the following directories to the path:

i. \NetRS_OL_dopp_compare\apx_files

ii. \NetRS_OL_dopp_compare\igs_files

OLPredict will create OLPredictSVXX.in, which is placed in the output folder for OL tracking. It also creates plots of elevation, predicted doppler frequency, and airplane trajectory (if using less than 1ms spacing, because otherwise there is too much data and we run out of memory). OL-Predict.m will produce a warning if the prediction period is too long, usually between 39 and 45 minutes, because of the discontinuities produced in the velocity measurements. It also saves some data for later use by the *fast_compare.m* program under a file named *OLPredict_SVXX.mat*. This program can handle batch requests.

5 Using PSRBitGen.m

PSRBitGen.m requires the presence of a file from CL tracking to operate. It operates by extracting bits from the COSMIC bit archive (bitArc) starting with the given z-count and ending after a user defined amount of time which is governed by the length of the prediction from *OLPredict.m*. Then, it compares the archive bits with CL bits and changes the sign of the bits, if

needed. Substantial error messaging will alert user to any problems. If a data bit file is not found, it may indicate the user did not add the directory to path, or the bitArc does not contain the data. The bitArc files must be in the same directory and in the format given in Section 9. The program will create *DataBitSVXX.out* place in the output directory for OL tracking. This program can handle batch requests.
User must define:

i. PRN
ii. GPS_Week
iii. Year
iv. Month
v. Day
vi. starting z-count
vii. prediction length

6 Using OLchanlook.m

OLchanlook will extract OL and CL data, creating plots of residual phase and amplitude, as well as reducing the data over 20ms. User must define SV and type of occultation(rising or setting). It also saves some data for later use by the *fast_compare.m* program under a file named *OLchanlookSVXX.mat*. This program can handle batch requests.

7 Using fast_compare.m

This is the final step in comparing the products from OL tracking to NetRS data.The user must have the appropriate NetRS file in the current directory. The user must also define the following:

i. SV
ii. type of occultation

iii. NetRS rinex filename

iv. name of saved file from OLPredict

v. name of saved file from OLchanlook

This will produce comparison plots of the phase from OL tracking, and the phase as recorded by the NetRS.

8 OL tracking flow

i. Run OLtrack to see which SV's are acquired, if some are, let it run long enough to write at least one batch z-count of databits to the data bit output file(DataBitSVXX.out).

ii. Run OLPredict.m

iii. Run PSRBitGen.m

iv. Run OLtrack

v. Run OLchanlook.m

vi. Run fast_compare.m

9 Input-Output File Formats

Table 1: OLPredictSVXX.out Nx11

Column Number	Variables
1	GPS Week
2	GPS millisecond of the week
3	Header milliseconds of the week(same as the (2))
4	Doppler Frequency (Hz)
5	Inphase (V)
6	Quadrature (V)
7	Residual phase (cycles)
8	NCO phase (cycles)
9	Data bit
10	Samples per cycle
11	Edge index

Table 2: chanYY.dat Nx14

Column Number	Variables
1	Doppler frequency (Hz)
2	Code rate (Hz)
3	Carrier phase (cycles)
4	Inphase Early correlator (V)
5	Quadrature Early correlator (V)
6	Inphase Promt correlator (V)
7	Quadrature Promt correlator (V)
8	Inphase Late correlator (V)
9	Quadrature Late correlator (V)
10	Promt inphase avarage
11	Long count(0,1,...,N)
12	Edge index
13	z-count
14	GPS millisecond of the week

Table 3: sv_chans.dat Nx2

Column Number	Variables
1	Channel Number
2	Satellite Number

Table 4: OLPredict_SVXX.mat

Variables	Definition
sp3_filename	IGS filename(ex. igs14675.sp3)
svs_to_predict	Satellites to predict
CarrierPhase	Carrier Phase (cycles)
Range	Distance between the SV and receiver (m)
gps_sow_vec	GPS seconds of the week
GPS_SOW_Range	same as the gps_sow_vec

Table 5: OLPredictSVXX.mat

Variables	Definitions
GPS_Wk	GPS week
GPS_MSOW	GPS milliseconds of the week
Header_MSOW	same as GPS_MSOW
Freq	OL tracking Doppler frequency (Hz)
I	Correlator Inphase
Q	Correlator quadphase
PhiResidual	Residual phase (cycles)
PhiNCO	NCO Phase (cycles)
DataBit	Data bits
SamplesPerCycle	Samples per Cycle
EdgeIndex	Edge index

Table 6: OLchanlookSVXX.mat

Variables	Definitions
Doppler_CL	CL Tracking Doppler frequency (Hz)
Freq	OL tracking Doppler frequency (Hz)
PhiResidual	Residual phase (cycles)
GPS_MSOW	GPS milliseconds of the week
PRN	PRN number
CorrCorrectPhaseR	Reduced corrected phase (cycles)
PhiNCO	NCO Phase (cycles)
GPS_Wk	GPS week
gps_sd_vec	GPS seconds of the day
gps_sd_vecR	Reduced GPS seconds of the day

Table 7: OLPredictSVXX.in (binary file)

Size	Variables
unsigned short	GPS week
unsigned long	GPS seconds of the week
double	Predicted Doppler frequency[1000] (Hz)

Table 8: DataBitsSVXX.in and DataBitsSVXX.out (binary files)

Size	Variables
unsigned long	Z-count
short	Data bits[300]

COSMIC Data Bit Files

Data files are created in the following format:

/<YYYYMMDD>/<HH>/<PRN>cdbs_ <PRN>_ <frameStart>_ <sf-Mask>_ <sfParityMask>_ <CN0>_ <hostNumber>.bin

- **PRN** -Satellite PRN (not SV number)
- **YYYYMMDD** -Year,Month,Day

- **HH** -Hour
- **hostName** -As obtained from gethostname function call(cdbs)
- **hostNumber** - 01-10(cdbs01-cdbs10)
- **frameStart** - GPS time for frame 1 start(time in seconds since GPS epoch)
- **sfMask** - Binary mask of subframes in file(bit0=sf1,bit4=sf5).0=not present,1=present.
- **sfParityMask** - If bit is set, then that subframe passed parity.
- **CN0** - Avarage Carrier to Noise ratio for that frame

The data file is defined by the following structure:

```
char hostName[8];
UINT8 prn;
UINT8 CN0;
UINT8 sfMask;
UINT8 sfParityMask;
UINT32 frameStart;
UINT32 subframe1[10]
UINT32 subframe2[10]
UINT32 subframe3[10]
UINT32 subframe4[10]
UINT32 subframe5[10]
(216 bytes)
```

An example file name:

/20050521/10/02/cdbs_02_0800708070_1f_1f_49_01.bin

This file indicates cdbs01 was tracking PRN02 on 21 March 2005 with a CN0 of 49dBHz during hour 10 (GPS time). All subframes were present and all parity passed.

10 Codes

10.1 Doppler prediction

```
function [DopplerFreq CarrierPhase Range RangeDopp wavelength] = ...
    DopplerPredict_Dynamic(GPS_Wk, GPS_MSOW, Rcvr_Pos, Rcvr_Vel, Trans_Pos, ...
    Trans_Vel, Trans_Clk_Rate, spacing_ms, options)

% This function will return an Nx1 vector of predicted Doppler frequencies,
% where N indicates the total number of data points.
%
% Inputs:
%  Time (referenced exclusively to GPS time):
%   GPS_Wk   : Nx1 vector of GPS week
%   GPS_MSOW : Nx1 vector of GPS millisecond of week
%  Receiver information (drawn from GPS/INS results):
%   Rcvr_Pos : Nx3 array of receiver positions in WGS-84 ECEF XYZ coordinates
%   Rcvr_Vel : Nx3 array of receiver velocities in WGS-84 ECEF XYZ coordinates
%  Transmitter (GPS satellite) information (drawn from IGS orbit data):
%   Trans_Pos : Nx3 array of transmitter positions in WGS-84 ECEF XYZ coordinates
%   Trans_Vel : Nx3 array of transmitter velocities in WGS-84 ECEF XYZ coordinates
%  Transmitter clock information (drawn from IGS orbit data):
%   Trans_Clk_Rate : Nx1 array of transmitter clock rate in seconds/second
%  Options:
%   options : Structure of options (format to be determined)
%
% Outputs:
%  DopplerFreq : Nx1 vector of predicted Doppler frequencies, based on
%  geometric and climatological models.
%
% Brian Ventre, Spring 2006

% Check to see if we have a stationary receiver.  If so, fill out Rcvr_Pos
% and Rcvr_Vel with copies of themselves in order to maintain the matrix
% math below.
if (size(Rcvr_Pos,1) ~= size(Trans_Pos,1))
    if (size(Rcvr_Pos,1) == 1 && size(Rcvr_Vel,1) == 1)
        % Need to stretch Rcvr_Pos and Rcvr_Vel.
        Rcvr_Pos = repmat(Rcvr_Pos,size(Trans_Pos,1),1);
        Rcvr_Vel = repmat(Rcvr_Vel,size(Trans_Pos,1),1);
    else
        error('Transmitter and receiver matrices are different sizes');
    end
end

% Constants
c = 2.99792458e8; % m/s
freqL1 = 1575.42e6; % Hz
freqL2 = 1227.60e6; % Hz
```

```
% Wavelength is a function of the clock rate (because frequency is
affected
% by the transmitting clock's rate).
wavelength = c ./ (freqL1 * (1 + Trans_Clk_Rate)); % units = (m/s)/(cyc/s+
cyc/s*s/s) = m/cyc

% Position vector from receiver to transmitter.
R_rt_m = 0*Trans_Pos;
r_rt_m = 0*Trans_Pos;
for ii = 1:size(Rcvr_Pos,2)
    R_rt_m(:,ii) = Trans_Pos(:,ii) - Rcvr_Pos(:,ii);
end

% Range, distance between receiver and transmitter.
mag_R_rt_m = sqrt(dot(R_rt_m,R_rt_m,2));
% Unit vector pointing from receiver to transmitter.
for ii = 1:size(Rcvr_Pos,2)
    r_rt_m(:,ii) = R_rt_m(:,ii) ./ mag_R_rt_m;
end

% Dot product of each velocity with the vector from receiver to
transmitter
% (line of sight velocity)
Vlos_rcvr_ms = dot(r_rt_m,Rcvr_Vel,2);
Vlos_trans_ms = dot(r_rt_m,Trans_Vel,2);

% Total line of sight velocity along vector from receiver to transmitter
Vlos_total_ms = Vlos_trans_ms - Vlos_rcvr_ms;

% Convert from line of sight velocity to Doppler frequency.
DopplerFreq = -Vlos_total_ms ./ wavelength;

% Calculate carrier phase via division of the magnitude of the position
% vector from receiver to transmitter by wavelength.
CarrierPhase = mag_R_rt_m ./ wavelength; % units = m/(m/cyc) = cyc

% Range and range derived Doppler frequency.
Range = mag_R_rt_m;
diffRange = diff(Range);
derivRange = diffRange/(spacing_ms/1000);
RangeDopp = -derivRange ./ wavelength(2:end);

```

10.2 OL Doppler Prediction

```
% OLPredict.m
% -----
%
% Description:
%
% This is the main file for the Doppler prediction algorithm.
%
% Predicts the Doppler frequency between GPS transmitters and a receiver
by
% interpolating the IGS orbits, determining the positions and velocities
of
```

```
% each, and performs other necessary calculations.
%
% Data Bit support is provided by PSRBitGen.m.
%
% Inputs:
%
%     IGS orbit, various user defined parameters
%
% Outputs:
%
%     doppler freq, data saved for later processing
%
% Assumptions/References:
%
%     January 1, 2000 at 1200, JD = 2,451,545.0
%     January 1, 2000 at 1200, MJD = 51,544.5, diff = 2,400,000.5
%     January 6, 1980 at 0000, gpst = 0, MJD = 44244, serial date = 723186
%
% Dependencies:
%
%     current dir : LLAtoECEF
%     current dir : apxread
%     current dir : sp3read
%     current dir : TransTimeAdjust
%     current dir : OrbitInterp
%     current dir : ClockInterp
%     current dir : ECEFtoENU
%     current dir : DopplerPredict_Dynamic
%     current dir : skyplot
%     toolbox     : map\map\deg2rad.m
%     toolbox     : map\map\rad2deg.m
%
%     Other Files:  Applanix (apx) file (conditional), SP3 file with
%     satellite orbit data.
%
% Modification History:
%
%    Spring 2006   Brian D. Ventre   Original version
%    Summer 2008   Tyler D. Lulich   Update to moving receiver

% Clear memory, command window, close open figures, set formats for
% display, and add relevent file paths.
clear all; close all; clc
format compact; format long g;
addpath(genpath(fullfile(cd,'apx_files')));
addpath(genpath(fullfile(cd,'igs_files')));
%addpath(genpath(fullfile(cd,'nrs_files')));

% Inform the user a job has started.
disp('******************OLPredict starting a job.***********************')

% Which SV?
if (exist('svs_to_predict','var') == 0)

    svs_to_predict = [12];
end
```

```
% Enter the desired starting time.
% format:  gps_start_time = [GPS week, SOW];
week = 1467;
startSOW = 500400;
%PRN 5,12: 500940 starboard
%PRN 15:  515703 starboard
gps_start_time = [week startSOW];

% How long is our data set?
data_length_sec = 44*60;

% What's the spacing [ms]?
% Using 1 ms will write to file for use in PSR.
% Max = 1000.
% spacing_ms = 1000;
spacing_ms = 1;
% 'true' if rcvr was fixed during recording.
stationary_rcvr_flag = 0;

if stationary_rcvr_flag
        Location = 'CIVL';
        fprintf('Static Position: %s\n',Location);
else
    fprintf('Dynamic Receiver Position.\n');
end

% If above false, define the name of the Applanix file containing pos/vel
% data of the receiver.
% apx_filename = sprintf('GPSINS_apx_SFRF_mssc_diff_053.txt');
 apx_filename = sprintf('staraero_2008-053.txt');
% apx_filename = sprintf('topsen_2008-053.txt');

% 'true' if we want to plot elevation, skyplot.
skyplot_flag = 1;

% 'true' if we want to plot doppler, flight path (very high memory usage).
plot_flag = 1;

% if 'true': save important variables for later.
savefile_flag = 1;

% Define orbit product type (ultra-rapid 'igu', rapid 'igr', final 'igs').
% http://igscb.jpl.nasa.gov/components/prods_cb.html
orbit_type = 'igs';

% How many samples should we use to form the interpolation polynomial?
%  See Schenwerk's paper from gps source dot com; 10 is sufficient.
NumSamp = 10;

if stationary_rcvr_flag
    switch (Location)
        case {'CIVL'} % Updated 23.apr.09 tdl
            rcvr_lat_rad   = deg2rad(40.4304200);
            rcvr_lon_rad  = deg2rad(-86.9148483);
            rcvr_alt_m     = 183.7;
            rcvr_vel_ms    = [0, 0, 0];
        case {'AgFarmDirect'} % Updated 14Dec06 tdl
```

```
            rcvr_lat_rad   = deg2rad(40.47517822);
            rcvr_lon_rad  = deg2rad(273.00791630);
            rcvr_alt_m     = 212.7896;
            rcvr_vel_ms    = [0, 0, 0];
        case {'RF05'} % a general location for debugging
            rcvr_lat_rad   = deg2rad(33.1729806100783);
            rcvr_lon_rad  = deg2rad(-83.2011608389817);
            rcvr_alt_m     = 12500;
            rcvr_vel_ms    = [0, 0, 0];
        otherwise
            error('Unknown location (or unprogrammed switch...)');
    end
    [rcvr_pos_m] = LLAtoECEF([rcvr_lat_rad, rcvr_lon_rad, rcvr_alt_m]);
else

    % Read and parse the applanix file.
    [apx_clk rcvr_pos rcvr_vel rcvr_lat rcvr_lon rate] = apxread(startSOW, data_length_sec, apx_filename);
    if spacing_ms/1000 ~= rate
        % Straight line approximation to interpolate the apx data to the sample
        % rate.
        sample_rate = 1000/spacing_ms; % [Hz]

        fprintf('Interpolating Applanix data to %dHz...\n', sample_rate)

        rcvr_pos_m = zeros(length(apx_clk)*sample_rate,1);
        rcvr_vel_ms = zeros(length(apx_clk)*sample_rate,1);
        rcvr_lat_rad = zeros(length(apx_clk)*sample_rate,1);
        rcvr_lon_rad = zeros(length(apx_clk)*sample_rate,1);

        diff_pos = diff(rcvr_pos,1,1);
        diff_vel = diff(rcvr_vel,1,1);
        diff_lat = diff(rcvr_lat);
        diff_lon = diff(rcvr_lon);

        % HACK: Manually create an extra difference value to homogenize matrix size.
        diff_pos(end+1,:) = diff_pos(end,:)+diff_pos(end,:)-diff_pos(end-1,:);
        diff_vel(end+1,:) = diff_vel(end,:)+diff_vel(end,:)-diff_vel(end-1,:);
        diff_lat(end+1,:) = diff_lat(end)+diff_lat(end)-diff_lat(end-1);
        diff_lon(end+1,:) = diff_lon(end)+diff_lon(end)-diff_lon(end-1);

        frac_pos = diff_pos/sample_rate;
        frac_vel = diff_vel/sample_rate;
        frac_lat = diff_lat/sample_rate;
        frac_lon = diff_lon/sample_rate;

        space = 0;
        for lcv = 1:length(rcvr_pos)
            for lcv2 = 1:sample_rate
                for lcv3 = 1:3
                    rcvr_pos_m(lcv2+space,lcv3) = rcvr_pos(lcv,lcv3) + frac_pos(lcv,lcv3)*(lcv2-1);
```

```
                    rcvr_vel_ms(lcv2+space,lcv3) = rcvr_vel(lcv,lcv3) + frac_vel(lcv,lcv3)*(lcv2-1);
                    if lcv3 == 1
                        rcvr_lat_rad(lcv2+space) = rcvr_lat(lcv) + frac_lat(lcv)*(lcv2-1);
                        rcvr_lon_rad(lcv2+space) = rcvr_lon(lcv) + frac_lon(lcv)*(lcv2-1);
                    end
                end
            end
            space = space + sample_rate;
        end
    else
        rcvr_pos_m = rcvr_pos;
        rcvr_vel_ms = rcvr_vel;
        rcvr_lat_rad = rcvr_lat;
        rcvr_lon_rad = rcvr_lon;
    end
end

% Read in the appropriate SP3 file.
gps_start_wk = gps_start_time(1);
gps_start_day = floor(gps_start_time(2)/3600/24);
sp3_filename = sprintf('%s%04d%1d.sp3',orbit_type,gps_start_wk,gps_start_day);
[sv_XYZ_m, sv_clk_s, gps_wk, gps_sec, header_info] = sp3read(sp3_filename);
% Check to be sure we aren't going to run over that single sp3 file.
gps_end_time = gps_start_time+[0 data_length_sec];
if (gps_end_time(2) > 3600*24*7)
    gps_end_time(1) = gps_end_time(1) + 1;
    gps_end_time(2) = gps_end_time(2) - 3600*24*7;
    error('Ending time is into the following GPS week...');
elseif(floor(gps_end_time(2)/3600/24) ~= gps_start_day)
    error('Ending time is into the following GPS day...');
end

% Create a vector of times at some spacing (usually 1 millisecond).
gps_msow_vec = [(gps_start_time(2)*1e3):spacing_ms:(gps_end_time(2)*1e3-1)]';
gps_wk_vec = ones(size(gps_msow_vec))*gps_start_wk;
gps_sow_vec = gps_msow_vec/1000;
GPS_SOW_Range=gps_sow_vec ;
% For skyplot with stationary rcvr, fill vectors of rcvr info for math
% continuity.
if skyplot_flag && stationary_rcvr_flag
    rcvr_pos_m = repmat(rcvr_pos_m,length(gps_msow_vec), 1);
    rcvr_lat_rad = repmat(rcvr_lat_rad,length(gps_msow_vec), 1);
    rcvr_lon_rad = repmat(rcvr_lon_rad,length(gps_msow_vec), 1);
    rcvr_vel_ms = repmat(rcvr_vel_ms,length(gps_msow_vec), 1);
end

% Define MATLAB Serial Date offset (used for nice plots).
SerialDateGPSTstart = 723186;
DaysPerWeek = 7;
gps_start_sd = SerialDateGPSTstart + DaysPerWeek*week;
gps_sd_vec = gps_sow_vec/3600/24 + gps_start_sd;
```

```
% Parameters for nice plot axis labels.
num_divisions = 6;
AxisTime = datenum([gps_sd_vec(1) gps_sd_vec(end)]);
ticksize_date = diff(AxisTime)/num_divisions;
tickmarks = AxisTime(1):ticksize_date:AxisTime(2);

% Check to see if we're going over a specific number of satellites or all
of
% the ones in the sp3 file.
if isempty(svs_to_predict)
    % All SVs in the file.
    svs_to_predict = header_info.sat_id;
end

% Should we make a skyplot?
if skyplot_flag
    lcv = 1;
    color_str = {'k','r','g','b','m','c','y'};
end

% Loop over all of the satellites in the Doppler prediction algorithm.
for lcv2 = 1:length(svs_to_predict)
    % What's the corresponding index in our sp3 file?
    svi = find(header_info.sat_id == svs_to_predict(lcv2));
    if isempty(svi)
        fprintf('SP3 file has no information for SV%02d.\n',
svs_to_predict(lcv2))
        fprintf('    ...continuing to next SV\n');
        continue
    end

    % Determine the corresponding time of transmission for each of the
    % receiver positions.
    [TimeOfTransmission] = TransTimeAdjust(rcvr_pos_m, gps_msow_vec/1000,
...
        sv_XYZ_m(:,:,svi), sv_clk_s(:,:,svi), gps_sec, NumSamp);

    % Interpolate the positions and velocities using some method.
    [SV_Pos_XYZ_m, SV_Vel_ms] =  OrbitInterp(TimeOfTransmission,...
        sv_XYZ_m(:,:,svi), gps_sec, NumSamp);

    % Find the clock rate, too.
    [SV_Clk_Bias_s, SV_Clk_Rate_ss] = ClockInterp(TimeOfTransmission,...
        sv_clk_s(:,:,svi), gps_sec, NumSamp);

    % Update the position and velocity of the SV by the rotation of the
    % ECEF frame during the transit time.
    wE = 7292115.0e-11; % rad/sec
    transit_time = gps_msow_vec/1000 - TimeOfTransmission; % sec
    for lcv3 = 1:length(transit_time)
        trans_matrix = [
            cos(wE*transit_time(lcv3)),  sin(wE*transit_time(lcv3)), 0
            -sin(wE*transit_time(lcv3)), cos(wE*transit_time(lcv3)), 0
            0,                           0,                          1];
        SV_Pos_XYZ_m(lcv3,:) = (trans_matrix * SV_Pos_XYZ_m(lcv3,:)')';
        SV_Vel_ms(lcv3,:)    = (trans_matrix * SV_Vel_ms(lcv3,:)')';
```

```
    end

    % If we have a stationary receiver, then let's make a sky plot, too.
    if skyplot_flag
        % Shift the origin of the ECEF coordinates of the SV to the
        % receiver's current location.
        SV_Pos_XYZ_m_shift = SV_Pos_XYZ_m - rcvr_pos_m;

        % Convert the ECEF satellite position to the ENU frame, then
        % calculate the azimuth and elevation.
        az = zeros(size(gps_msow_vec));
        el = zeros(size(gps_msow_vec));
        for lcv4 = 1:length(az)
            [SV_Pos_E SV_Pos_N SV_Pos_U] = ...
                ECEFtoENU(rcvr_lat_rad(lcv4),rcvr_lon_rad(lcv4),SV_Pos_XYZ_m_shift(lcv4,:));

            % Calculate the elevation and azimuth of this SV and plot it.
            az(lcv4,1) = atan2(SV_Pos_E, SV_Pos_N);
            el(lcv4,1) = atan2(SV_Pos_U, sqrt(SV_Pos_E^2 + SV_Pos_N^2));
        end
        AZ_all(:,lcv) = az;
        EL_all(:,lcv) = el;
        SVs(lcv) = svs_to_predict(lcv2);
        lcv = lcv + 1;

        % Give us a globe view, too.
        figure(1); hold on;
        color_ind = mod(lcv-1,length(color_str))+1;
        plot3(SV_Pos_XYZ_m(:,1),SV_Pos_XYZ_m(:,2),SV_Pos_XYZ_m(:,3),color_str{color_ind})
        plot3(SV_Pos_XYZ_m(end,1),SV_Pos_XYZ_m(end,2),SV_Pos_XYZ_m(end,3),[color_str{color_ind},'*'],'MarkerSize',5)
        figure(2); hold on; grid on;
        plot(gps_sd_vec, rad2deg(el),color_str{color_ind})
        legend_str{lcv2} = sprintf('SV%02d',svs_to_predict(lcv2));
    end

    % Do Doppler/Phase prediction.
    fprintf('Predicting Doppler for SV%02d...\n',svs_to_predict(lcv2))
    [DopplerFreq CarrierPhase Range RangeDopp Wavelength] = ...
        DopplerPredict_Dynamic(gps_wk_vec, gps_msow_vec, rcvr_pos_m, rcvr_vel_ms, ...
        SV_Pos_XYZ_m, SV_Vel_ms, SV_Clk_Rate_ss, spacing_ms, []);

    % How do the two Doppler Freqs compare?
    dopp_diff = DopplerFreq(2:end) - RangeDopp;
    fprintf('SV%02d: Average Doppler Difference %11.7f Hz\n',svs_to_predict(lcv2),mean(dopp_diff))
    if plot_flag
        figure; hold on; grid on; box on;
        plot(gps_sd_vec,DopplerFreq,'.b',gps_sd_vec(1:end-1),RangeDopp,'r')
        v = axis; axis([AxisTime,v(3:4)])
        xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
        ylabel('Doppler Frequency [Hz]')
```

```
        set(gca,'xtick',tickmarks)
        datetick('x','HH:MM:SS','keeplimits','keepticks')
        date = datestr(gps_sd_vec(1)); DMY = date(1:11);
        title(sprintf('SV%02d - Doppler Frequency',svs_to_predict(lcv2)))
        legend('VLOS Doppler','Range-derived Doppler','Location','Best');
        print('-dpdf','-r300',sprintf('../Plots/SV%02d-%s-PredictedDopp',svs_to_predict(lcv2),DMY))
        if spacing_ms ~= 1
            figure; hold on; box on;
            plot3(rcvr_pos_m(:,1),rcvr_pos_m(:,2),rcvr_pos_m(:,3))
            plot3(rcvr_pos_m(end,1),rcvr_pos_m(end,2),rcvr_pos_m(end,3),'r*','MarkerSize',5)
            xlabel('Feet')
            ylabel('Feet')
            zlabel('Feet')
            title('Actual Flight Path, ECEF XYZ')
        end
    end

    % Output the values to our prediction file.  Write a header for every
    % second, followed by 1000 values of Doppler (1 for each millisecond).
    % Binary output
    if (spacing_ms == 1)
        fprintf('Writing SV%02d Doppler prediction to binary file...\n',svs_to_predict(lcv2))
        outfilestr = sprintf('../Output/OLPredictSV%02d.in', svs_to_predict(lcv2));
        output_file = fopen(outfilestr, 'w+b');
        if (output_file == -1)
            error('Unable to open %s!', outfilestr)
        end
        for lcv5 = 1:data_length_sec
            start_ind = (lcv5-1)*1000+1;
            stop_ind = (lcv5)*1000;

            if (floor(gps_msow_vec(start_ind)/1000) ~= gps_msow_vec(start_ind)/1000)
                error('Oops');
            end
            fwrite(output_file, gps_wk_vec(start_ind), 'ushort');
            fwrite(output_file, gps_msow_vec(start_ind)/1000, 'ulong');
            fwrite(output_file, DopplerFreq(start_ind:stop_ind), 'double');
        end
        fclose(output_file);
    end

    % Create a matrix containing the phase predictions for each SV.
    CarrierPhase_mtrx(:,lcv2) = CarrierPhase;

    % Save only the necessary variables for later comparison.
    if savefile_flag
        save(sprintf('../Output/OLPredict_SV%02d',svs_to_predict(lcv2)),...
            'sp3_filename','svs_to_predict','CarrierPhase','Range','gps_sow_vec','GPS_SOW_Range')
    end
```

```
    fprintf('Done with SV%02d...\n',svs_to_predict(lcv2));
end

% Print the skyplot.
if skyplot_flag
    % Finish the 3-D globe view first.
    figure(1)
    [x y z] = ellipsoid(0,0,0,6378137,6378137,6356752.3142,24);
    surf(x,y,z)
    plot3(rcvr_pos_m(:,1),rcvr_pos_m(:,2),rcvr_pos_m(:,3))
    axis equal; colormap pink; grid on
    view(rcvr_pos_m(1,:))

    % Then, the elevation profiles
    figure(2)
    v = axis; axis([AxisTime,v(3:4)])
    xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
    ylabel('Elevation [deg]')
    set(gca,'xtick',tickmarks)
    datetick('x','HH:MM:SS','keeplimits','keepticks')
    date = datestr(gps_sd_vec(1)); DMY = date(1:11);
    title([sprintf('Elevation, %s, %d seconds',DMY,data_length_sec)])
    legend(legend_str,'Location','Best');
    print('-dpdf','-r300',sprintf('../Plots/elevation-profile-%s-SOW%d',DMY,startSOW))

    % Print the sky plot.
    skyplot(AZ_all,EL_all,SVs)
    title([sprintf('Skyplot, %s, %d seconds',DMY,data_length_sec)])
    print('-dpdf','-r300',sprintf('../Plots/skyplot-%s-SOW%06d',DMY,startSOW))
end

disp('******************OLPredict finished a job.******************')
```

10.3 OL Phase

```
% if (exist('do_not_clear_olchanlook','var') == 1)
%     close all
%     clear all
%     clc
% end

close all
clear all
clc

%what type of occultation(setting or rising)?
forward = input('type of tracking(0 for backward)? ');
% Load the channel and PRN numbers.
sv_chans = load('../Output/sv_chans.dat');
fprintf('Available PRNs: ');
fprintf('%02d ',sv_chans(:,2));
fprintf('\n');
```

```
% Which PRN?
if (exist('PRN','var') == 0)
    PRN = 3;
end

% Create a filename for each of the plots.
filename_olchan = sprintf('../Plots/PRN%02dOLchan',PRN);

% Create an PRN specific handle each figure title.
figure_title = sprintf('PRN %02d',PRN);

% How many samples per msec?
NumSamplesPerMs = 10000;

% Find the equivalent closed-loop channel.
index = find(sv_chans(:,2) == PRN);
CLChannel = sv_chans(index,1)

% Let's see if we have the .mat file available.
matfile = 0;
fid = fopen(sprintf('../Output/OLchanlookSV%02d.mat',PRN));
if (fid ~= -1)
    matfile = input('Use .mat file (0 for no)? ');
    if (matfile(1) ~= 0)
        fclose(fid);
        disp('Using presaved .mat file...');
        load(sprintf('../Output/OLchanlookSV%02d.mat',PRN));
    end
end

if (matfile==0)
    if (forward~=0)

        % Extract the Open Loop (OL) data.
        A = load(sprintf('../Output/OLPredictSV%02d.out',PRN));
        len = floor((length(A)-40)/20)*20;
        ms=A(:,2);
        start_ms=ms(1);
        start_ind=mod(start_ms,20);
        start_ind=20-start_ind;
        % Index values
        Wki              = 1;
        MSOWi            = 2;
        HeaderMSOWi      = 3;
        Freqi            = 4;
        Ii               = 5;
        Qi               = 6;
        PhiRi            = 7;
        PhiNCOi          = 8;
        DataBiti         = 9;
        SamplesPerCyclei = 10;
        EdgeIndexi       = 11;

        % Extract individual data vectors
        GPS_Wk          = A(start_ind+12:len+start_ind+11,Wki);
        GPS_MSOW        = A(start_ind+12:len+start_ind+11,MSOWi);
```

```
        Header_MSOW     = A(start_ind+12:len+start_ind+11,HeaderMSOWi);
        Freq            = A(start_ind+12:len+start_ind+11,Freqi);
        I               = A(start_ind+12:len+start_ind+11,Ii);
        Q               = A(start_ind+12:len+start_ind+11,Qi);
        PhiResidual     = A(start_ind+12:len+start_ind+11,PhiRi);
        PhiNCO          = A(start_ind+12:len+start_ind+11,PhiNCOi);
        DataBit         = A(start_ind+12:len+start_ind+11,DataBiti);
        SamplesPerCycle = A(start_ind+12:len+start_ind+11,SamplesPerCyclei);
        EdgeIndex       = A(start_ind+12:len+start_ind+11,EdgeIndexi);
        clear A
        save(sprintf('../Output/OLPredictSV%02d.mat',PRN),'GPS_Wk','GPS_MSOW',...
            'Header_MSOW','Freq','I','Q','PhiResidual','PhiNCO','DataBit',...
            'SamplesPerCycle','EdgeIndex')

        %  For Carrier Phase In Chanxx.dat
        %%%%############################################
        %Extract closed loop (CL) data.
        B = load(sprintf('../Output/chan%02d.dat',CLChannel));
        lenCL = floor(length(B)/20)*20;
        starti = find(GPS_MSOW(1) == B(:,14));
        endi = find(GPS_MSOW== B(end,14));

        DopplerCL       = B(starti:endi,1);
        CarrierPhaseCL  = B(starti:endi,3);
        ICL             = B(starti:endi,6);
        QCL             = B(starti:endi,7);
        EdgeIndexCL     = B(starti:endi,12);
        GPS_MSOWCL      = B(starti:endi,14);

%       GPS_MSOW=GPS_MSOW-440326;
%       GPS_MSOWCL=GPS_MSOWCL-440326;
    %   For Carrier Phase In Chanxx.dat
    %%%%############################################
        clear B
        % break
        % Create the CL complex correlator.
    else

        % Extract the Open Loop (OL) data.
        A = load(sprintf('../Output/OLPredictSV%02d.out',PRN));
        len = floor((length(A)-40)/20)*20;
        ms=A(:,2);
        start_ms=ms(end);
        start_ind=mod(start_ms,20);
        start_ind=20-start_ind;
        % Index values
        Wki              = 1;
        MSOWi            = 2;
        HeaderMSOWi      = 3;
        Freqi            = 4;
        Ii               = 5;
        Qi               = 6;
        PhiRi            = 7;
        PhiNCOi          = 8;
```

```
        DataBiti          = 9;
        SamplesPerCyclei = 10;
        EdgeIndexi       = 11;

        % Extract individual data vectors
        GPS_Wk           = flipud(A(:,Wki));
        GPS_Wk           = GPS_Wk(start_ind+12:len+start_ind+11,1);
        GPS_MSOW         = flipud(A(:,MSOWi));
        GPS_MSOW         = GPS_MSOW(start_ind+12:len+start_ind+11,1);
        Header_MSOW      = flipud(A(:,HeaderMSOWi));
        Header_MSOW      = Header_MSOW(start_ind+12:len+start_ind+11,1);
        Freq             = flipud(A(:,Freqi));
        Freq             = Freq(start_ind+12:len+start_ind+11,1);
        I                = flipud(A(:,Ii));
        I                = I(start_ind+12:len+start_ind+11,1);
        Q                = flipud( A(:,Qi));
        Q                = Q(start_ind+12:len+start_ind+11,1);
        PhiResidual      = flipud(A(:,PhiRi));
        PhiResidual      = PhiResidual(start_ind+12:len+start_ind+11,1);
        PhiNCO           = flipud(A(:,PhiNCOi));
        PhiNCO           = PhiNCO(start_ind+12:len+start_ind+11,1);
        DataBit          = flipud(A(:,DataBiti));
        DataBit          = DataBit(start_ind+12:len+start_ind+11,1);
        SamplesPerCycle = flipud(A(:,SamplesPerCyclei));
        SamplesPerCycle = SamplesPerCycle(start_ind+12:len+start_ind+11,1)
;
        EdgeIndex        = flipud(A(:,EdgeIndexi));
        EdgeIndex        = EdgeIndex(start_ind+12:len+start_ind+11,1);
        clear A
        save(sprintf('../Output/OLPredictSV%02d.mat',PRN),'GPS_Wk','
GPS_MSOW',...
            'Header_MSOW','Freq','I','Q','PhiResidual','PhiNCO','DataBit
',...
            'SamplesPerCycle','EdgeIndex')

        %  For Carrier Phase In Chanxx.dat
        %%%%############################################
        %Extract closed loop (CL) data.
        B = load(sprintf('../Output/chan%02d.dat',CLChannel));
        lenCL = floor(length(B)/20)*20;
        CL_msow=B(:,14);
%         starti = find(GPS_MSOW(1) == B(:,14));
        endi = find(GPS_MSOW(end)== B(:,14));

        DopplerCL        = B(lenCL-endi:endi,1);
        CarrierPhaseCL   = B(lenCL-endi:endi,3);
        ICL              = B(lenCL-endi:endi,6);
        QCL              = B(lenCL-endi:endi,7);
        EdgeIndexCL      = B(lenCL-endi:endi,12);
        GPS_MSOWCL       = B(lenCL-endi:endi,14);

   %      GPS_MSOW=GPS_MSOW-440326;
%      GPS_MSOWCL=GPS_MSOWCL-440326;
    %  For Carrier Phase In Chanxx.dat
    %%%%############################################
        clear B
    end
```

```
    CorrCL = ICL + QCL*1i;
    CorrCLAmp = abs(CorrCL);
    CorrCLPhase = angle(CorrCL);

    % Create the complex correlator along with phase and amplitude.
    Corr = I + Q*1i;
    CorrAmp = abs(Corr);
    CorrPhase = angle(Corr);

    % Calculate the total phase (from residual and NCO).
    TotalPhase = PhiResidual + PhiNCO;

    % Use the data bit to "fix" the correlators.  Use the unwrap
    % function to try to fix the discontinuities
    CorrCorrect = Corr./(DataBit);
    CorrCorrectPhase = unwrap(angle(CorrCorrect));%unwrap(angle(CorrCorrect));
    %CorrCorrectCL = CorrCL./DataBit;
    %CorrCorrectCLPhase = unwrap(angle(CorrCorrectCL));
    % Check CorrCorrectPhase and the OL residual phase.  If they don't
    % match, we have a problem (possibly).  Note that we drop the first
    % sample on the residual phase because of the n+1 delay in the
    % processing.
    PhiResidual=PhiResidual-PhiResidual(2);
    MaxAbsDeviation = max(abs(CorrCorrectPhase(1:end-1)-PhiResidual(2:end)));
    MaxDeviationOK = MaxAbsDeviation <= 1e-2;

    % Reduce data rate by summing or average every 20 data points together
    % (coherent over a data bit because we start on a data bit edge).
    % We sum over the OL, CL, and corrected OL correlators.
    len_r           = floor(length(Corr)/20);
    CorrR           = zeros(len_r,1);
    CorrCorrectR    = zeros(len_r,1);
    CorrCLR         = zeros(len_r,1);
    GPS_MSOWR       = zeros(len_r,1);
    for ii = 1:len_r
        start = (ii-1)*20 + 1;
        stop  = (ii-1)*20 + 20;
        CorrR(ii)           = sum(Corr(start:stop));
        CorrCorrectR(ii)    = sum(CorrCorrect(start:stop));
%         CorrCLR(ii)         = sum(CorrCL(start:stop));
        GPS_MSOWR(ii)       = mean(GPS_MSOW(start:stop));
        GPS_MSOWR(ii)       =GPS_MSOWR(ii)-mod(GPS_MSOWR(ii),20);
    end

    % Extract properties of each correlator.
    CorrAmpR = abs(CorrR);
    CorrPhaseR = angle(CorrR);

    CorrCorrectAmpR = abs(CorrCorrectR);
    CorrCorrectPhaseR = unwrap(angle(CorrCorrectR));
    CorrCLAmpR = abs(CorrCLR);
    CorrCLPhaseR = angle(CorrCLR);
end
```

```
%figure(10)
%subplot(3,1,1);plot(GPS_MSOW(1:end),unwrap(CorrCLPhase(1:end)-CorrCLPhase(1))/2/pi,'.')
%subplot(3,1,2);plot(GPS_MSOW(1:end),unwrap(CorrPhase(1:end)-CorrPhase(1))/2/pi,'b.',GPS_MSOW(1:end),unwrap(CorrPhase(1:end)-CorrPhase(1))/2/pi,'r.')
%subplot(3,1,3);plot(GPS_MSOW(1:end),CorrCorrectPhase/2/pi,'.')

%plot(GPS_MSOW(1:200:end),(CorrCLPhase(1:200:end)-CorrCLPhase(1))/2/pi,'.')

%++++++++++++++++++++++++++++++++++++++++++++++++
% Nice Plots

% Define MATLAB Serial Date offset (used for nice plots).
SerialDateGPSTstart = 723186;
DaysPerWeek = 7;
gps_start_sd = SerialDateGPSTstart + DaysPerWeek*GPS_Wk(1);

% Create a vector of times at some spacing (usually 1 millisecond).
gps_sow_vec = GPS_MSOW/1000;
gps_sd_vec = gps_sow_vec/3600/24 + gps_start_sd;
% For reduced rate.
gps_sow_vecR = GPS_MSOWR/1000;
gps_sd_vecR = gps_sow_vecR/3600/24 + gps_start_sd;

% Parameters for nice plot axis labels.
num_divisions = 6;
AxisTime = datenum([gps_sd_vec(1) gps_sd_vec(end)]);
ticksize_date = diff(AxisTime)/num_divisions;
tickmarks = AxisTime(1):ticksize_date:AxisTime(2);

AxisTimeCL = datenum([gps_sd_vec(1) gps_sd_vec(length(GPS_MSOW))]);
ticksize_dateCL = diff(AxisTimeCL)/8;
tickmarksCL = AxisTimeCL(1):ticksize_dateCL:AxisTimeCL(2);
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
%++++++++++++++++++++++++++++++++++++++++++++++++

% Save the variables so we won't have to do this again!
save(sprintf('../Output/OLchanlookSV%02d.mat',PRN))
savefile = sprintf('../Output/OLChanlookSV%02d_%s.mat',PRN,DMY);
save(savefile,'DopplerCL','Freq','PhiResidual','GPS_MSOW','PRN','GPS_Wk',...
    'gps_sd_vec','gps_sd_vecR','AxisTime','ticksize_date','tickmarks',...
    'AxisTimeCL','ticksize_dateCL','tickmarksCL')

% Check if post-processed phase matches the OL phase calculated using the
% 4-quadrant phase extraction in the PSR.  Inform user of result, print
% figure if necessary.
%------------------------------------------
if (MaxDeviationOK)
    disp('Post-processed phase matches the OL calculated phase');
else
    % Hmm, things don't seem to match, possible error somewhere?
       disp('Post-processed phase does not match the OL calculated phase!');
       figure; hold on; grid on;
```

```
        plot(gps_sd_vec(1:end-1),CorrCorrectPhase(1:end-1)/2/pi,'r.','MarkerSize',2);
        plot(gps_sd_vec(1:end-1),PhiResidual(2:end)/2/pi,'k.','MarkerSize',2);
        v = axis; axis([AxisTime,v(3:4)])
        xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
        ylabel('Phase (cycles)')
        set(gca,'xtick',tickmarks)
        datetick('x','HH:MM:SS','keeplimits','keepticks')
        title(sprintf('PRN%02d - Post-processed phase and OL phase - %s', PRN, DMY))
        legend('Post-processed','Residual with 1^{st} sample dropped','Location','Best')
        print('-dpdf','-r300',[filename_olchan,sprintf('_%s_phase_error',DMY)])
end
%-------------------------------------------

% Phase from Correlators, wrapped and unwrapped, 1KHz.
%-------------------------------------------
figure

subplot(3,1,1); hold on; grid on;
plot(gps_sd_vec,(CorrPhase)/2/pi,'.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
ylabel('Phase (cycles)')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Uncorrected correlator phase (data bits present) (',figure_title,')'])

subplot(3,1,2); hold on; grid on;
plot(gps_sd_vec,angle(CorrCorrect)/2/pi,'.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
ylabel('Phase (cycles)')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Corrected correlator phase (data bits present) (',figure_title,')'])
%xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])


subplot(3,1,3); hold on; grid on;
plot(gps_sd_vec,CorrCorrectPhase/2/pi,'.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
ylabel('Phase (cycles)')
xlabel('GPS Time    [hh:mm:ss]')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Residual Phase (data bits removed and unwrapped) (',figure_title,')'])
print('-dpdf','-r300',[filename_olchan,'_phase_1kHz'])
```

```
%------------------------------------------
% Plot Amplitude, 1kHz
%------------------------------------------
% figure; hold on; grid on;
% subplot(2,1,1); hold on; grid on;
% plot(gps_sd_vec,CorrPhase/2/pi,'.','MarkerSize',2);
% v = axis; axis([AxisTime,v(3:4)])
% ylabel('Phase (cycles)')
% set(gca,'xtick',tickmarks)
% datetick('x','HH:MM:SS','keeplimits','keepticks')
% date = datestr(gps_sd_vec(1)); DMY = date(1:11);
%
%
% subplot(2,1,2); hold on; grid on;
% plot(gps_sd_vec,CorrAmp,'.','MarkerSize',2);
% v = axis; axis([AxisTime,v(3:4)])
% ylabel('Amplitude')
% set(gca,'xtick',tickmarks)
% datetick('x','HH:MM:SS','keeplimits','keepticks')
% title(['Amplitude, 1kHz data rate (', figure_title,')'])
% xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
% print('-dpdf','-r300',[filename_olchan,'_amp_1kHz'])
%------------------------------------------

%------------------------------------------
figure; grid on;
if(forward)
    plot(gps_sd_vec(1:length(GPS_MSOWCL)),DopplerCL,'b.','MarkerSize',2);
else
    plot(gps_sd_vec(end-length(GPS_MSOWCL)+1:end),DopplerCL,'b.','MarkerSize',2);
end
hold on;
plot(gps_sd_vec,Freq,'r.','MarkerSize',2);
%plot(gps_sd_vec(1:length(GPS_MSOWCL)),New_Dopp,'w.','MarkerSize',1);

v = axis; axis([AxisTime,v(3:4)])
%xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
xlabel('GPS Time    [hh:mm:ss]')
ylabel('Doppler frequency (Hz)')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Doppler Frequency (',figure_title,')'])
legend('CL','Model')
%legend('CL','Predicted')
print('-dpdf','-r300',[sprintf(filename_olchan,'_doppler_model&CL')])
%------------------------------------------

%------------------------------------------
%TDL 19 SEP 06
if(forward)
    Doppler_diff = DopplerCL - Freq(1:length(GPS_MSOWCL));
else
```

```
    Doppler_diff = DopplerCL - Freq(end-length(GPS_MSOWCL)+1:end);
end
ave = mean(Doppler_diff);
figure; hold on; grid on;
plot(gps_sd_vec(1:length(GPS_MSOWCL)),Doppler_diff,'k.','MarkerSize',2);
v = axis; axis([AxisTimeCL,v(3:4)])
xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
ylabel('Doppler frequency (Hz)')
set(gca,'xtick',tickmarksCL)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['CL minus Model Doppler (',figure_title,')',sprintf('  [Average:  %1.4f Hz]',ave)])
print('-dpdf','-r300',[filename_olchan,'_doppler_diff'])
%------------------------------------------


%------------------------------------------
figure; hold on; grid on;
plot(gps_sd_vec,unwrap(PhiResidual/2/pi),'k.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
ylabel('Phase (cycles)')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Phase from OL Channel (1kHz data rate) (',figure_title,')'])
print('-dpdf','-r300',[filename_olchan,'_residual_phase_from_PSR'])
%------------------------------------------

%------------------------------------------
TotalDiffCodeCycle = sum(abs(mod(abs(EdgeIndex(1:length(GPS_MSOWCL))-EdgeIndexCL),NumSamplesPerMs)));
MaxDiffCodeCycle = 2;%max(abs(mod(abs(EdgeIndex(1:length(GPS_MSOWCL))-EdgeIndexCL),NumSamplesPerMs)));
if (TotalDiffCodeCycle > len*0.1 || MaxDiffCodeCycle > 1)
    disp('OL and CL code cycle edges appear to have a problem!')
    figure

    subplot(3,1,1); hold on; grid on;
    plot(gps_sd_vec(1:length(GPS_MSOWCL)),EdgeIndex(1:length(GPS_MSOWCL)),'.','MarkerSize',4)
    v = axis; axis([AxisTimeCL,v(3:4)])
    ylabel('OL Edge')
    set(gca,'xtick',tickmarksCL)
    datetick('x','HH:MM:SS','keeplimits','keepticks')
    date = datestr(gps_sd_vec(1)); DMY = date(1:11);
    title(['Difference in code cycle edges (',figure_title,')'])

    subplot(3,1,2); hold on; grid on;
    plot(gps_sd_vec(1:length(GPS_MSOWCL)),EdgeIndexCL,'.','MarkerSize',4)
    v = axis; axis([AxisTimeCL,v(3:4)])
    ylabel('CL Edge')
    set(gca,'xtick',tickmarksCL)
    datetick('x','HH:MM:SS','keeplimits','keepticks')
```

```
    date = datestr(gps_sd_vec(1)); DMY = date(1:11);

    subplot(3,1,3); hold on; grid on;
    plot(gps_sd_vec(1:length(GPS_MSOWCL)),EdgeIndex(1:length(GPS_MSOWCL))-EdgeIndexCL,'b.','MarkerSize',4);
    v = axis; axis([AxisTimeCL,v(3:4)])
    ylabel('Sample difference')
    set(gca,'xtick',tickmarksCL)
    datetick('x','HH:MM:SS','keeplimits','keepticks')
    date = datestr(gps_sd_vec(1)); DMY = date(1:11);


    xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(length(GPS_MSOWCL))])
    print('-dpdf','-r300',[filename_olchan,'_code_edge'])
else
    disp('OL and CL code cycle edges within parameters!')
end
%--------------------------------------------

% phase, 50 Hz data rate
%--------------------------------------------
figure; hold on; grid on;
plot(gps_sd_vecR,CorrCorrectPhaseR/2/pi,'k.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
ylabel('Phase (cycles)')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Corrected Correlator Plot (data bits removed and unwrapped) 50Hz data rate (',figure_title,')'])
print('-dpdf','-r300',[filename_olchan,'_phase_corr'])
%--------------------------------------------

% amplitude, 50 Hz data rate
%--------------------------------------------
figure; hold on; grid on;
plot(gps_sd_vecR,CorrCorrectAmpR,'k.','MarkerSize',2);
v = axis; axis([AxisTime,v(3:4)])
ylabel('Amplitude')
set(gca,'xtick',tickmarks)
datetick('x','HH:MM:SS','keeplimits','keepticks')
date = datestr(gps_sd_vec(1)); DMY = date(1:11);
title(['Amplitude - 50Hz data rate (',figure_title,')'])
xlabel([datestr(gps_sd_vec(1)) '    |-------- GPST --------|    ' datestr(gps_sd_vec(end))])
print('-dpdf','-r300',[filename_olchan,'_raw_corr_50Hz'])
%--------------------------------------------

fprintf('...done PRN%02d\n',PRN)
```

10.4 NetRS-OL Excess Phase Compare

```
% fast_compare.m
```

```
% used to quickly compare doppler freq from OLPredict.m, PSR CL/OL, and
NetRS
close all; clear all; clc; format compact; format long g;

setting = input('Type of occultation(0 for rising)?');

PRN_vec = [3];

%rinex_file = sprintf('pu07275-15-19.10o'); %Ferry flight-2010-FF04
rinex_file = sprintf('PU12200802_53.08o');
[Obs epochs sv_vec apcoords] = readrinex(rinex_file);

%some constants
fl1=1575.42e6;  %L1 frequency
fl2=1227.60e6;  %L2 frequency
c = 2.99792458e8;   %speed of light
lam1=c/fl1; %L1 wavelength
lam2=c/fl2; %L2 wavelength

for lcv = 1:length(PRN_vec)

    SV = PRN_vec(lcv);

    load(sprintf('../Output/OLPredict_SV%02d.mat',SV));
    load(sprintf('../Output/OLChanlookSV%02d.mat',SV));

    OL_sow_vec = GPS_MSOW/1000;     %OL time vector
    CL_sow_vec = GPS_MSOWCL/1000;   %CL time vector
    GPS_sow_range=GPS_SOW_Range;    %Range time vector

    nrs_freq_hz = 5;
    OL_freq_hz = 1000;
    CL_freq_hz = 1000;

    rinexSVi =  find(sv_vec == SV);

    epochs(:,1) = str2double(rinex_file(5:8));
    nrs_sd_vec = datenum(epochs);
    NRS = rinex_file(1:4);

    % Parse the orbit type, GPS_week, and Day of Week from .sp3 file name.
    file_type = (sp3_filename(1:3));
    gps_week = str2double(sp3_filename(4:7));
    gps_dow = str2double(sp3_filename(8:9));

    % Convert the rinex epochs to second of the week (SOW).
    nrs_sow_vec = zeros(length(epochs(:,1)),1);
    for i = 1:length(epochs)
        nrs_sow_vec(i,1)=(epochs(i,6)+60*epochs(i,5)+3600*epochs(i,4))+
gps_dow*24*3600;
    end
    %Common time vectors
    [OL_R_common_sow OL_R_sowi R_OL_sowi] = intersect(GPS_MSOW,GPS_MSOWR);
    [NRS_Range_common_vec NRS_Ri R_NRSi]=intersect(nrs_sow_vec,
GPS_sow_range);
    [OL_Range_common_vec OL_Ri R_OLi]=intersect(OL_R_common_sow/1000,
GPS_sow_range);
```

```
    [CL_Range_common_vec CL_Ri R_CLi]=intersect(CL_sow_vec,GPS_sow_range);
    [NRS_OL_R_common NOi ONi]=intersect(NRS_Range_common_vec,OL_Range_common_vec);
    [NRS_OL_common_sow NRS_OL_sowi OL_NRS_sowi] = intersect(nrs_sow_vec, GPS_MSOWR/1000);

    %Phase
    IntPredDopp = (cumsum(Freq).*1/OL_freq_hz);
    IntCLDopp = (cumsum(DopplerCL).*1/CL_freq_hz);

    CLPhase = IntCLDopp(CL_Ri)-IntCLDopp(CL_Ri(1));
    OLPhase = IntPredDopp(OL_R_sowi) + CorrCorrectPhaseR(R_OL_sowi)/2/pi- IntPredDopp(1) - CorrCorrectPhaseR(1)/2/pi;
    nrsPhaseL1 = (Obs.L1(:,rinexSVi))*lam1;
    nrsPhaseL2 = (Obs.L2(:,rinexSVi))*lam2;
    nrsPhaseLC = fl1^2*nrsPhaseL1/(fl1^2-fl2^2)-fl2^2*nrsPhaseL2/(fl1^2-fl2^2);

    %excess phases
    CL_exc=-CLPhase*lam1-(Range(R_CLi)-Range(R_CLi(1)));

    OLPhase=OLPhase*lam1;
    OL_exc=-(OLPhase(OL_Ri)-OLPhase(OL_Ri(1)))-(Range(R_OLi)-Range(R_OLi(1)));
    NRS_range=nrsPhaseL1(NRS_Ri);
    nrs_start_ind=find(~isnan(NRS_range));
    Range_NRS=Range(R_NRSi);


    NRS_excL1=(nrsPhaseL1(NRS_Ri)-NRS_range(nrs_start_ind(1)))-(Range(R_NRSi)-Range_NRS(nrs_start_ind(1)));
    NRS_excL1=NRS_excL1-NRS_excL1(nrs_start_ind(1));
    deriv_tol=0.5;
    NRS_excL1= fix_jumps (NRS_excL1,NRS_Range_common_vec,deriv_tol);

    NRS_excL2=nrsPhaseL2(NRS_Ri)-Range(R_NRSi);
    NRS_excL2=NRS_excL2-NRS_excL2(1);
    NRS_excL2= fix_jumps (NRS_excL2,NRS_Range_common_vec,deriv_tol);
    NRS_excLC=nrsPhaseLC(NRS_Ri)-Range(R_NRSi);
    NRS_excLC=NRS_excLC-NRS_excLC(1);
    NRS_excLC= fix_jumps (NRS_excLC,NRS_Range_common_vec,deriv_tol);

    % Parameters for nice plots
    SerialDateGPSTstart = 723186;
    DaysPerWeek = 7;
    gps_start_sd = SerialDateGPSTstart + DaysPerWeek*gps_week;

    NRS_OL_SD_VEC = NRS_OL_common_sow/3600/24 + gps_start_sd;
    NRS_R_SD_VEC=NRS_Range_common_vec/3600/24 + gps_start_sd;
    OL_R_SD_VEC=OL_Range_common_vec/3600/24 + gps_start_sd;
    CL_R_SD_VEC=CL_Range_common_vec/3600/24 + gps_start_sd;
    NRS_OL_R_VEC=NRS_OL_R_common/3600/24 + gps_start_sd;
    date = datestr(NRS_R_SD_VEC(1)); DMY = date(1:11);

    num_divisions = 4;
    AxisTime = datenum([NRS_R_SD_VEC(1) NRS_R_SD_VEC(end)]);
    ticksize_date = diff(AxisTime)/num_divisions;
```

```
        tickmarks = AxisTime(1):ticksize_date:AxisTime(2);

        %NetRS, OL and CL excess phase plots
        figure; hold on; grid on;
        plot(NRS_R_SD_VEC,NRS_excL1-min(NRS_excL1))
%       plot(NRS_R_SD_VEC,NRS_excL2,'r')
%       plot(NRS_R_SD_VEC,NRS_excLC,'g')

        [m,io]=min(abs(NRS_R_SD_VEC-OL_R_SD_VEC(1)));
        plot(OL_R_SD_VEC,OL_exc-min(OL_exc),'m')

        [n,ic]=min(abs(NRS_R_SD_VEC-CL_R_SD_VEC(1)));
        plot(CL_R_SD_VEC,CL_exc+NRS_excL1(ic),'k')
        datetick('x',13,'keepticks')
        ylabel('Excess Phase(m)')
        xlabel('GPS Time    [hh:mm:ss]')
        title(sprintf('PRN %02d Excess Phase',PRN))
        if(setting)
            legend('NRS L1','OL','location','NorthWest')
        else
            legend('NRS L1','OL','location','NorthEast')
        end
        print('-dpdf','-r300',sprintf('../Plots/PRN%02d-%s-phase-all-detrend',
SV,DMY))
      %____________________________________________

T_CL  = CL_R_SD_VEC;
T_OL  = OL_R_SD_VEC;
T_NRS = NRS_R_SD_VEC;
OL = OL_exc+NRS_excLC(io);
CL = CL_exc+NRS_excLC(ic);
L1 = NRS_excL1;
L2 = NRS_excL2;
LC = NRS_excLC;

fid =fopen('./T_excphase','w');

for k = 1:(length(NRS_R_SD_VEC))
fprintf(fid,'%12.10f %12.10f %12.10f %12.10f %12.10f %12.10f %12.10f 
%12.10f\n',[T_CL(k) T_OL(k) T_NRS(k) OL(k) CL(k) L1(k) L2(k) LC(k)]);
end
fclose(fid);
%________________________________________________________
        %OL-CL compare

%       [ol_cl_vec oli cli]=intersect(OL_Range_common_vec,
CL_Range_common_vec);
%       OL_CL_SD_VEC=ol_cl_vec/3600/24 + gps_start_sd;
%       OL_exc_c=OL_exc(oli)-OL_exc(oli(1));
%       CL_exc_c=CL_exc(cli)-CL_exc(cli(1));
%
%       figure; hold on; grid on;
%       plot(OL_CL_SD_VEC,OL_exc_c-CL_exc_c,'.','markersize',1)
%       datetick('x',13,'keepticks')
%       ylabel('Phase [m]')
%       xlabel('GPS Time    [hh:mm:ss]')
%       title(sprintf('PRN%02d Phase Difference(CL Phase - OL Phase)',PRN))
```

```
%      print('-dpdf','-r300',sprintf('PRN%02d_CL_OLphase',SV))
%      figure; hold on; grid on;
%      plot(OL_CL_SD_VEC(1:end-1),diff(OL_exc_c-CL_exc_c),'.','markersize',4)
%      datetick('x',13,'keepticks')
%      ylabel('d\Phi/dt [m/s]')
%      xlabel('GPS Time    [hh:mm:ss]')
%      title(sprintf('PRN%02d-diff(CL Phase - OL Phase)',PRN))
%      print('-dpdf','-r300',sprintf('../Plots/PRN%02d_diffCL_OLphase',SV))

    %NetRS - OL Compare
    NRS_OL_c=nrsPhaseL1(NRS_OL_sowi);
    NrsOL_start_ind=find(~isnan(NRS_OL_c));
    NRS_OL_c=NRS_OL_c-NRS_OL_c(NrsOL_start_ind(1));
    OLPhase=circshift(OLPhase,0);
    OLPhase2=-OLPhase(OL_NRS_sowi);
    OL_NRS_c=OLPhase2-OLPhase2(NrsOL_start_ind(1));
    OL_NRS_diff=NRS_OL_c-OL_NRS_c;
    OL_NRS_diff=fix_jumps(flipud(OL_NRS_diff-OL_NRS_diff(NrsOL_start_ind(1))),NRS_OL_common_sow,0.5);

    figure; hold on; grid on;
    plot(NRS_OL_SD_VEC,flipud(OL_NRS_diff*100/lam1),'.','markersize',3)
    datetick('x',13,'keepticks')
    ylabel('Phase [m]')
    xlabel('GPS Time    [hh:mm:ss]')
    title(sprintf('PRN %02d NetRS Phase - OL Phase)',PRN))
    set(get(gcf,'CurrentAxes'));
    print('-dpdf','-r300',sprintf('../Plots/PRN%02d-%s-NetRS-minus-OL-phase',SV,DMY))
    print('-dpng','-r300',sprintf('../Plots/PRN%02d-%s-NetRS-minus-OL-phase',SV,DMY))
    saveas(gcf,sprintf('PRN%02d-%s-NetRS-minus-OL-phase',SV,DMY),'fig')

    figure; hold on; grid on;
    plot(NRS_OL_SD_VEC(1:end-1),diff(NRS_OL_c-OL_NRS_c),'.','markersize',6)
    datetick('x',13,'keepticks')
    ylabel('d\Phi/dt [m/s]')
    xlabel('GPS Time    [hh:mm:ss]')
    title(sprintf('PRN %02d diff(NRS Phase - OL Phase)',PRN))
    print('-dpdf','-r300',sprintf('../Plots/PRN%02d-%s-diff-NRS-minus-OL-phase',SV,DMY))

end
```