ASE 372N Laboratory Final Project

Posting Date: Oct. 29, 2018
Due Date: Dec. 13, 2018

Reading

Misra & Enge, Ch. 6; lecture notes.

Objectives

In this project you'll perform a navigation solution based on pseudorange data that was downloaded from a CORS receiver. Your solution will implement the least-squares techniques we discussed in lecture. You will account for ionospheric, tropospheric, earth rotation, and relativistic effects.

Teams

There are no teams for this final project. The work turned in must be your own.

Modeled Pseudorange Calculation

Recall that in lecture we derived the following model for the pseudorange measurement:

$$\rho_{1/2} = R + c(\delta t_u - \delta t_s) + T + I_{\rho 1/2} + M_{\rho 1/2} + \varepsilon_{\rho 1/2} \tag{1}$$

where

- $\rho_{1/2}$ pseudorange measured on L1/2 frequency based on code at the user's signal receive time t_R
- R is the geometrical range from satellite s to user u based upon where the user is at t_R and the satellite at $(t_R - \delta t_u(t_R) - t_{\text{OWLT}})$ with t_{OWLT} being the one way light time defined as $R = \sqrt{(x_s - x_u)^2 + (y_s - y_u)^2 + (z_s - z_u)^2}$
- δt_u is the user/receiver clock error (bias) at t_R
- δt_s is the satellite clock error at $(t_R - \delta t_u(t_R) - t_{\text{OWLT}})$
- T is the tropospheric delay
- $I_{\rho 1/2}$ is the ionospheric delay in code measurement on L1/2
- $M_{\rho 1/2}$ is the multipath delay in code measurement on L1/2
- $\varepsilon_{\rho 1/2}$ other delay/errors in code measurement on L1/2 at t_R

For signal transmission through a vacuum, the one way light time t_{OWLT} is related to R by

$$c \cdot t_{\mathrm{OWLT}} = R + T + I_{\rho 1/2} + M_{\rho 1/2} = ||\underline{r}_u(t_R) - \underline{r}_s(t_R - \delta t_R(t_R) - t_{\mathrm{OWLT}})|| + T + I_{\rho 1/2} + M_{\rho 1/2} \tag{2}$$

We call this an *implicit* definition for t_{OWLT} because t_{OWLT} shows up on both sides of the equality.

At any particular t_R, a GNSS receiver will produce multiple pseudorange measurements—one for each unique signal tracked. Each measurement can be modeled by (1). In preparation for least-squares estimation, we can cast (1) into following standard-form model for the ith pseudorange measurement:

$$z_i' = h_i'(\underline{x}) + w_i' \tag{3}$$

where $z_i' = \rho$, $h_i'(\underline{x}) = R + c(\delta t_u - \delta t_s) + T + I_\rho + M_\rho$, $w_i' = \varepsilon_\rho$, and $\underline{x} = \left[\underline{r}_u^T, c \cdot \delta t_u\right]^T$ is the 4-by-1 unknown parameter vector that we wish to estimate. The primes in (3) indicate that we have not yet performed normalization to make the measurement noise have unity variance—we'll do this later. Suppose we have n_z different pseudorange measurements modeled as in (3), all taken at user's receiver time t_R but corresponding to different satellites. We can stack these together to obtain a vector measurement equation of the form

$$\underline{z}' = \underline{h}'(\underline{x}) + \underline{w}' \tag{4}$$

Recall from lecture that we can approximate $\underline{h}'(\underline{x})$ by a first-order Taylor series approximation

$$\underline{h}'(\underline{x}) \approx \underline{h}'(\bar{\underline{x}}) + \underline{\underline{H}}'(\underline{x} - \bar{\underline{x}}) \tag{5}$$

where $\underline{\underline{H}}'$ is the n_z-by-4 Jacobian matrix for n_z independent pseudorange measurements and $\bar{\underline{x}} = \left[\bar{\underline{r}}_R^T, c\ \bar{\delta} t_u\right]^T$ is an approximate value (a guess) for $\underline{x}$. Equation (5) is a better approximation the closer $\bar{\underline{x}}$ is to $\underline{x}$. Recall from lecture that the ith row in $\underline{\underline{H}}'$ is the 1-by-4 Jacobian

$$\left.\frac{\partial h_i'}{\partial \underline{x}}\right|_{\underline{x}=\bar{\underline{x}}} = \left[\frac{\partial h_i'}{\partial \underline{r}_u}, \frac{\partial h_i'}{\partial c\delta t_R}\right]\Bigg|_{\underline{x}=\bar{\underline{x}}} \tag{6}$$

with

$$\left.\frac{\partial h_i'}{\partial \underline{r}_u}\right|_{\underline{x}=\bar{\underline{x}}} \approx \frac{\bar{\underline{r}}_u^T(t_R) - \underline{r}_s^T(t_R - \bar{\delta} t_u - \bar{t}_{\mathrm{OWLT}})}{||\bar{\underline{r}}_u(t_R) - \underline{r}_S(t_R - \bar{\delta} t_u - \bar{t}_{\mathrm{OWLT}})||} \tag{7}$$

which is just the transpose of the 3-by-1 unit vector pointing from the satellite (at the transmission event) to the receiver (at the receipt event), and

$$\left.\frac{\partial h_i'}{\partial c\delta t_u}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}} \approx 1 \tag{8}$$

In (7), $\bar{t}_{\mathrm{OWLT}}$ is also an approximation because it depends on $\bar{\underline{r}}_u$ and $\bar{\delta} t_u$ through (2).

Your first analysis task in the Analysis Tasks section below will be to derive (7) and (8) by taking partial derivatives (this was actually done in lecture).

Our first step toward a full navigation solution will be to write a Matlab function named `satpr` which, given $\bar{\underline{r}}_u$ and $c \cdot \bar{\delta} t_u$, generates modeled pseudorange $\bar{\rho}_i = h_i'(\bar{\underline{x}})$ and its corresponding 1-by-4 Jacobian H_i' for the ith SV. You can generate $\bar{\rho}_i$ by application of (1) and (2) above.

Your `satpr` function will need to account for each of the effects discussed previously. You will not be required to develop functions for modeling the ionospheric and neutral atmospheric delay (troposphere). Rather, you will invoke the functions `getIonoDelay` and `getTropoDelay`, which are found on the course website, within your `satpr` implementation. The `getTropoDelay` function will need a set of model coefficients stored in `NMFcoeffs.mat`, also found on the course website.

The `getIonoDelay` and `getTropoDelay` functions will require a conversion from ECEF coordinates to latitude, longitude, and altitude for which you can use the `ecef2lla` function you wrote in homework 1.

An essential step in your `satpr` function will be to calculate $\bar{t}_{\text{OWLT}}$ from $\underline{\bar{r}}_u$ and $\bar{\delta t}_u$. If (2) were a simple explicit expression with $\bar{t}_{\text{OWLT}}$ only on one side, this would be easy. But $\bar{t}_{\text{OWLT}}$ depends on $\underline{r}_S(t_R - \bar{\delta t}_u(t_R) - \bar{t}_{\text{OWLT}})$, which in turn depends on $\bar{t}_{\text{OWLT}}$, so solving for $\bar{t}_{\text{OWLT}}$ is a bit more tricky. We'll take an iterative approach, as follows:

Given t_R, $\underline{\bar{r}}_u(t_R)$, and $\bar{\delta t}_u(t_R)$, find $\bar{t}_{\text{OWLT}}$:

1. Start out with a rough guess of $\bar{t}_{\text{OWLT}}$, say $\bar{t}_{\text{OWLT}} = 0.075$ seconds. That's about how long a signal takes to reach the earth from a satellite in GPS orbit.
2. Calculate a guess of the true time of the transmission event: $\bar{t} = t_R - \bar{\delta t}_u(t_R) - \bar{t}_{\text{OWLT}}$.
3. Plug $\bar{t}$ into your `satloc` function to get the satellite position $\underline{r}_S(\bar{t})$.
4. Obtain a better guess for $\bar{t}_{\text{OWLT}}$ by applying (2):
$$\bar{t}_{\text{OWLT}} = \tfrac{1}{c}||\underline{\bar{r}}_u(t_R) - \underline{r}_S(\bar{t})||$$
5. Repeat steps 2-4 until $\underline{r}_S(\bar{t})$ changes by less than a centimeter from one iteration to the next. No more than 10 iterations should be required.

On the last iteration of this procedure, $\bar{t}$ will be a very good guess of t, the true time of the transmission event. To complete your calculation of $\bar{\rho}$, you'll need to evaluate the satellite clock error δ_S at $\bar{t}$. Recall that the GPS control segment broadcasts a model for calculating $\delta_S(t)$.

Relativistic Effects and Group Delay Differential

The satellite clock delay $\delta t_S(t)$ can be modeled in simplified form as

$$\delta t_S(t) = a_{f_0} + a_{f_1}(t - t_c) + a_{f_2}(t - t_c)^2 \tag{9}$$

where t represents true time (equivalent to GPS system time for our purposes), the a_{f_i}, $i = 0, 1, 2$, are polynomial clock model coefficients, and t_c is the time of clock epoch. In this homework we'll augment the model for δt_S to include the group delay differntial t_{GD} and a residual relativistic correction δt_{rel}:

$$\delta t_S(t) = a_{f_0} + a_{f_1}(t - t_c) + a_{f_2}(t - t_c)^2 + \delta t_{\text{rel}} - t_{GD} \tag{10}$$

You can get t_{GD} directly from `satdata.TGD`. Calculation of δt_{rel} was discussed in lecture.

Here, t_c is the time of clock epoch, which is stored in the `wc` and `tc` fields of `satdata`. Having calculated δ_S at the transmission event, you now have all the ingredients necessary to generate a modeled pseudorange:

$$\bar{\rho} = h'(\underline{\bar{x}}) = ||\underline{\bar{r}}_u(t_R) - \underline{r}_S(\bar{t})|| + c\left[\bar{\delta t}_u(t_R) - \delta t_S(\bar{t})\right] \tag{11}$$

where $\bar{t} = t_R - \bar{\delta t}_u(t_R) - \bar{t}_{\text{OWLT}}$ comes from the last iteration of the procedure for finding $\bar{t}_{\text{OWLT}}$.

With this preparation, proceed to writing a `satpr` function that conforms to the following interface:

```
function [rhoBar, Hrho] = ...
```

```
      satpr(gpsWeekRx, gpsSecRx, cdtRx, rRxEcef, sd, ionodata, tiFlags)
% satpr : Calculate the modeled pseudorange between the satellite specified in
%         sd and the receiver located at rRxEcef at the given time of signal
%         reception.
%
%
% INPUTS
%
% gpsWeekRx -- GPS week of signal reception event in receiver time.
%
% gpsSecRx --- GPS seconds of week of signal reception event in receiver
%              time.
%
% cdtRx ------ Receiver clock error scaled by the speed of light: cdtRx =
%              c*dtRx. True time t is related to receiver time tRx by t = tRx
%              - dtRx.
%
% rRxEcef ---- 3-by-1 ECEF coordinates of the receiver antenna's phase center,
%              in meters.
%
% sd --------- Ephemeris structure array for a single SV.  Let ii be the
%              numerical identifier (PRN identifier) for the SV whose location
%              is sought.  Then sd = satdata(ii) has the following fields:
%
%       SVID - satellite number
%       health - satellite health flag (0 = healthy; otherwise unhealthy)
%       we - week of ephemeris epoch (GPS week, unambiguous)
%       te - time of ephemeris epoch (GPS seconds of week)
%       wc - week of clock epoch (GPS week)
%       tc - time of clock epoch (GPS seconds of week)
%       e - eccentricity (unitless)
%       sqrta - sqrt of orbit semi-major axis (m^1/2)
%       omega0 - argument of perigee (rad.)
%       M0 - mean anomaly at epoch (rad.)
%       L0 - longitude of ascending node at beginning of week (rad.)
%       i0 - inclination angle at epoch (rad.)
%       dOdt - longitude rate (rad / sec.)
%       dn - mean motion difference (rad / sec.)
%       didt - inclination rate (rad / sec.)
%       Cuc - cosine correction to argument of perigee (rad.)
%       Cus - sine correction to argument of perigee (rad.)
%       Crc - cosine correction to orbital radius (m)
%       Crs - sine correction to orbital radius (m)
%       Cic - cosine correction to inclination (rad.)
%       Cis - sine correction to inclination (rad.)
%       af0 - 0th order satellite clock correction (s)
%       af1 - 1st order satellite clock correction (s / s)
%       af2 - 2nd order satellite clock correction (s / s^2)
%       TGD - group delay time for the satellite (s)
%
% ionodata -- Ionospheric data structure array with the following fields:
%
%       alpha0, alpha1, alpha2, alpha3 - power series expansion coefficients
%           for amplitude of ionospheric TEC
%
%       beta0, beta1, beta2, beta3 - power series expansion coefficients for
```

```
%            period of ionospheric plasma density cycle
%
% tiFlags --- 2-by-1 vector of flags that indicate whether to enable
%             tropospheric delay correction (tiFlag(1)) or ionospheric
%             delay correction (tiFlags(2)).
%
%
% OUTPUTS
%
% rhoBar ---- Modeled pseudorange between the receiver at time of the signal
%             reception event (the time specified by gpsWeekRx and gpsSecRx)
%             and the satellite at time of the signal transmission event, in
%             meters.
%
% Hrho ------ 1 x 4 Jacobian matrix containing partials of pseudorange with
%             respect to rRxEcef and c*dtRx
%
%+------------------------------------------------------------------------------+
% References:
%
%
% Author:
%+==============================================================================+
```

Earth Rotation During Signal Propagation Interval

In this homework we will calculate t_{OWLT} taking into account the rotation of the earth during the interval of time between the signal transmission and reception events. You'll do this by transforming both events into the same reference frame, i.e. the ECEF frame at the receipt event, as discussed in lecture.

To test `satpr` and for further analysis, modify the `viewsats` script that you wrote in the previous lab as follows. After retrieving navigation data and plotting the SVs above the elevation mask angle at a specified time and location, call `satpr` for each SV above the elevation mask angle. Form a complete $\underline{\underline{H}}'$ matrix by stacking the individual 4-by-1 Jacobian row vectors that `satpr` calculates for each visible SV.

Verify your `satpr` function by running your modified `viewsats` script with the following inputs, assuming that `gpsWeek` and `gpsSec` refer to receiver time. You'll find the `getAntLoc.m` function on Canvas.

Pseudorange Model Verification

Verify the accuracy of your `satpr` function by running your `viewsats.m` script with the following inputs, assuming that `gpsWeek` and `gpsSec` refer to receiver time.

```
gpsWeek = 1712;
gpsSec = 143800;
elMask = 15;
rRx = getAntLoc('WRW0');
cdtRx = 0;
```

You should find that the visible SVs are $[2, 4, 8, 9, 17, 20, 24, 28]$. The corresponding modeled pseudorange vector

$\bar{\boldsymbol{\rho}} = \boldsymbol{h}'(\bar{\boldsymbol{x}})$ should be within a centimeter or so of

```
rhoBarVec =
23563949.9389694
20489373.1818216
23420865.0801816
22397464.3850917
21107992.5368601
23961491.4049471
22821307.9488751
21242700.8789888
```

and the H' matrix should be nearly equivalent to

```
Hprime =
0.57487      0.65411      0.49159            1
0.22441      0.97208     -0.068582           1
-0.11912      0.77048      0.62624            1
0.68363      0.043497    -0.72853            1
-0.040289     0.47493     -0.8791             1
-0.88517      0.19064     -0.42441            1
0.13052      0.87905      0.45852            1
-0.49349      0.76644     -0.41114            1
```

To help you diagnose problems you might have in your pseudorange modeling code, compare your intermediate values for the calculation of the modeled pseudorange corresponding to SV ID (PRN) 2 with the following values. Your values should not differ from the ones below by more than a centimeter or so.

1. If you've retrieved the right ephemeris data, your `satdata` structure for SV ID (PRN) 2 should be equal to

   ```
   satdata(2) =
   SVID: 2
   health: 0
   we: 1712
   te: 144000
   wc: 1712
   tc: 143999.999996647
   e: 0.011546980706
   sqrta: 5153.7800827
   omega0: -2.7277551532
   M0: 2.33821189593
   L0: 2.27738442636
   i0: 0.938581804728
   dOdt: -8.17462622028e-009
   dn: 5.09949812885e-009
   didt: -6.03953728526e-010
   Cuc: -2.14762985706e-006
   Cus: 8.45082104206e-006
   ```

```
Crc: 207.46875
Crs: -40.28125
Cic: -2.77534127235e-007
Cis: -3.91155481338e-008
af0: 0.000407380517572
af1: 1.13686837722e-012
af2: 0
TGD: -1.76951289177e-008
```

2. The receiver position $\underline{r}_u$ expressed in latitude (rad), longitude (rad), and altitude (meters) should be

```
lat =  0.528616927215839
lon = -1.70581017303855
alt =  164.932940838858
```

3. The ionospheric delay from the broadcast ionospheric model should be $I_\rho = 10.6903434747677$ meters.
4. The tropospheric delay from `getTropoDelay` should be $T = 6.44739205828992$ meters.
5. The total time of flight multiplied by the speed of light should be $c \cdot t_{\mathrm{OWLT}} = 23686078.9649843$ meters.
6. The satellite position at the time of transmission expressed in the ECEF frame of the receipt epoch should be $r_S = [-14358424.8331188, -20955469.9934282, -8445935.18315543]^T$ meters.
7. The initial guess of receiver clock error should be $c \cdot \delta t_u = 0$ meters.
8. The relativistic SV clock correction should be $c \cdot \delta t_{rel} = -5.8173636919967$ meters.
9. The full SV clock correction (including the relativistic correction) should be $c \cdot \delta t_s = 122129.026014883$ meters.

Navigation Solution

Now that we have a function for estimating $\bar{\rho}_i = h_i'(\underline{\bar{x}})$ and generating the 1-by-4 Jacobian H_i' for the ith SV, we can proceed to the full least-squares solution of (4). Implement the steps outlined in the lectures on least squares estimation in a function called `performNavigationSolution` having the following interface:

```
function [solutionMat,tRVecSolution,tTrueVecSolution,...
          residualVec,badSvIdVec,thetaNominalMat] = ...
      performNavigationSolution(tRVec,svIdVec,obsValidMat,prMat,fDMat,thetaMat,...
                                iiStart,iiStop,rRxAppx, ...
                                satdata,ionodata,svIdAllow,lambda,tiFlags)
% performNavigationSolution : Perform the navigation solution using nonlinear
%                             least squares techniques.
%
%
% INPUTS
%
% tRVec --------------- Structure of two Nt-by-1 vectors that jointly define
%                      the unique receiver time instants corresponding to all
%                      received observables; tRVec.w contains the GPS week
%                      and tRVec.s contains the GPS seconds of week.
%
% svIdVec ------------- Nsv-by-1 vector of unique SVIDs corresponding to
```

```
%                       SVs that were tracked at some point during the
%                       measurement interval spanned by tRVec.
%
% obsValidMat --------- Nt-by-Nsv matrix whose (ii,jj)th element indicates
%                       whether (1) or not (0) observables were valid for SVID
%                       svIdVec(jj) at time tRVec.s(ii).
%
% prMat --------------- Nt-by-Nsv matrix whose (ii,jj)th element is the
%                       measured pseudorange value for SVID svIdVec(jj) at
%                       time tRVec.s(ii), in meters.
%
% fDMat --------------- Nt-by-Nsv matrix whose (ii,jj)th element is the
%                       measured Doppler value for SVID svIdVec(jj) at time
%                       tRVec.s(ii), in Hz.
%
% thetaMat ------------ Nt-by-Nsv matrix whose (ii,jj)th element is the
%                       measured beat carrier phase value for SVID svIdVec(jj)
%                       at time tRVec.s(ii), in cycles.
%
% iiStart,iiStop ------ Start and stop indices into tRVec between which a
%                       navigation solution will be computed.  These indices
%                       are used to isolate a specific section of data.  The
%                       number of solutions Ns will nominally be Ns = iiStop -
%                       iiStart + 1 but will be less if for some epochs a
%                       solution could not be computed (e.g., too few SVs).
%
% rRxAppx ------------- 3-by-1 approximate location of receiver antenna, in
%                       ECEF meters.
%
% satdata, ionodata --- SV and ionospheric navigation data parameters.
%                       See getephem.m for details.
%
% svIdAllow ----------- Vector of SVIDs for SVs that are allowed to
%                       participate in the navigation solution.
%
% lambda -------------- Nominal wavelength of GNSS signal, in meters.
%
% tiFlags ------------- 2-by-1 vector of flags that indicate whether to enable
%                       tropospheric delay correction (tiFlag(1)) or
%                       ionospheric delay correction (tiFlags(2)).
%
%
% OUTPUTS
%
% solutionMat --------- Ns-by-4 matrix of navigation solutions whose rows
%                       are formatted as [xEcef,yEcef,zEcef,c*deltR].
%
% tRVecSolution ------- Structure of two Ns-by-1 vectors that jointly define
%                       the unique receiver time instants corresponding to
%                       navigation solutions in solutionMat; tRVecSolution.w
%                       contains the GPS week and tRVecSolution.s contains the
%                       GPS seconds of week.
%
% tTrueVecSolution ---- Structure of two Ns-by-1 vectors that jointly define
%                       the unique receiver time instants corresponding to
%                       navigation solutions in solutionMat.  This vector is
```

```
%                       an estimate of 'true' GPS time for each
%                       solution. tTrueVecSolution.w contains the GPS week and
%                       tTrueVecSolution.s contains the GPS seconds of week.
%
% residualVec --------- Ns-by-1 vector of maximum least-squares residuals
%                       magnitudes from each solution epoch, in meters.
%
% badSvIdVec ---------- Ns-by-1 vector of SVIDs corresponding to the SV with
%                       the worst residual at each solution epoch.
%
% thetaNominalMat ----- Ns-by-Nsv vector of nominal beat carrier phase values,
%                       in the same arrangement as those in thetaMat, based
%                       solely on the SV-to-rRxAppx range evaluated at
%                       receiver time, in cycles.
%
%+------------------------------------------------------------------------------+
% References:
%
%
% Author:
%+==============================================================================+
```

At each measurement epoch (each unique receiver time t_R at which pseudorange measurements are available), you'll take the following steps within this function:

1. Start with a guess for $\underline{x} = \left[\underline{r}_u^T, c \cdot \delta t_u\right]^T$. Call this $\bar{\underline{x}}$. Usually it's sufficient to let your guess for $\underline{r}_u$ be anywhere on the correct continent and your guess for $c \cdot \delta t_u$ be zero.

2. Stack all pseudorange measurements that are valid at epoch t_R into the vector $\underline{z}'$. You'll draw these measurements from the appropriate row of `prMat`. Keep track of which SV corresponds to each element of $\underline{z}'$. *Be sure to exclude pseudoranges corresponding to entries marked 0 in* `obsValidMat`.

3. For the ith element (ith pseudorange measurement) in $\underline{z}'$, $i = 1, \ldots, n_z$, find the corresponding predicted pseudorange $\bar{\rho}_i$ and the 1-by-4 Jacobian H_i' by making a call to `satpr`. Stack the $\bar{\rho}_i$, $i = 1, 2, \ldots, n_z$ into the n_z-by-1 vector $\bar{\underline{z}}'$ and the Jacobians H_i', $i = 1, 2, \ldots, n_z$ into the n_z-by-n_x matrix H', where $n_x = 4$ is the dimension of the parameter vector $\underline{x}$. Your pseudorange measurements can now be approximately modeled as
$$\underline{z}' = \bar{\underline{z}}' + \underline{\underline{H}}'(\underline{x} - \bar{\underline{x}}) + \underline{w}'$$
which can be rearranged as
$$\check{\underline{z}}' = \underline{\underline{H}}'\underline{x} + \underline{w}'$$
where $\check{\underline{z}}' \triangleq \underline{z}' - \bar{\underline{z}}' + \underline{\underline{H}}'\bar{\underline{x}}$. Note that the rearranged equation is in the standard form that is the starting point for linear least squares.

4. Assume that all the pseudorange measurements are independent and identically distributed (i.i.d.), with $w_\rho \sim \mathcal{N}(0, \sigma_\rho^2)$, so that $\underline{w}'$ can be modeled as $\underline{w}' \sim \mathcal{N}(\underline{0}, \underline{\underline{P}}_{\underline{w}'})$, with $\underline{\underline{P}}_{\underline{w}'} = \sigma_\rho^2 \underline{\underline{I}}$. (Note that this makes for an especially simple Cholesky-factorized weighting matrix $\underline{\underline{R}}_a^{-T}$, where $\underline{\underline{R}}_a^T \underline{\underline{R}}_a = \underline{\underline{P}}_{\underline{w}'}$.) For now, set $\sigma_\rho = 2$ meters.

5. Normalize the measurement equation to obtain
$$\underline{z} = \underline{\underline{H}}\underline{x} + \underline{w}$$
where $\underline{z} = \underline{\underline{R}}_a^{-T}\check{\underline{z}}'$, $\underline{\underline{H}} = \underline{\underline{R}}_a^{-T}\underline{\underline{H}}'$, and $\underline{w} = \underline{\underline{R}}_a^{-T}\underline{w}'$.

6. Proceed as discussed in lecture to solve for the $\underline{x}$ that would minimize the cost function $J(\underline{x}) = \|\underline{\underline{H}}\underline{x} - \underline{z}\|^2$. This involves QR factorization of $\underline{\underline{H}}$. Call the solution you obtain $\bar{\underline{x}}_{\text{new}}$. If $\|\bar{\underline{x}}_{\text{new}} - \bar{\underline{x}}\|$ is very small (say, less than 10^{-4} meters), then stop, declaring $\bar{\underline{x}}_{\text{new}}$ to be your solution at time t_R; if not, set $\bar{\underline{x}} = \bar{\underline{x}}_{\text{new}}$ and go to step 3.

Note that, within this function, you will call `satpr` for each allowed SV for which you have data at each epoch between `tRxVec(iiStart)` and `tRxVec(iiStop)`, so your new `satpr` function will see plenty of use. For this lab, you'll only perform a pseudorange-based navigation solution so you won't need to use the inputs `fDMat`, `thetaMat`, `ionodata`, and `lambda`, but keep them in the interface as placeholders. More sophisticated solutions that you may implement may make use of these quantities. Also, just return an empty matrix `[]` for `thetaNominalMat`. All of the other return arguments should be properly populated. The jth element of the vector `residualVec` equals the element with the maximum magnitude (absolute value) in the vector $\Delta z' \triangleq \underline{z}' - \hat{\underline{z}}'$ on the last iteration of the jth solution epoch. The SV corresponding to this value is stored in `badSvIdVec(j)`.

Top-level Script

It will be convenient to have a top-level script called `topsol` that performs the following steps. You can either write your own function or use the example `topsol.m` on Canvas. Even if you write your own `topsol` function, make sure your `performNavigationSolution` works correctly within the example `topsol` function, since this is how your code will be evaluated by the TA. Note that example `topsol.m` is slightly more sophisticated than the simple procedure below; it divides the input data into small processing chunks and refreshes the ephemeris and ionospheric data, and performs elevation masking, before processing each chunk.

1. Set an initial guess for the receiver position.
2. Draw in a `channel.mat` file and format the data as described in the interface to `performNavigationSolution`. Extract only the data corresponding to GPS L1 C/A signals and having valid pseudorange measurements. This data formatting task is best implemented as a separate function. You can either write this yourself or use the function `prepareTimeHistory.m` posted on the course website. Note that the vector `tRVec` in `performNavigationSolution` should be made up of offset receiver time (ORT) measurement time stamps.
3. Pull the first ORT from `tRVec`. This first ORT is very close to the true GPS time of the first valid pseudorange measurement.
4. Plot the satellites at the first ORT by a call to `satmap` assuming some elevation mask angle `elMaskDeg`.
5. Set up other input arguments necessary for a call to `performNavigationSolution` such as `iiStart`, `iiStop`, and `lambda`. If your initial guess for the receiver position is good to, say, 100 km or so, then set `svIdAllow` equal to the list of SVs returned by your call to `satmap`. Otherwise, if your initial guess for the receiver position is way off, then `satmap` can't be trusted to provide you a reliable list of SVs above your mask angle. In this case, just set `svIdAllow = [1:32];`.
6. Exclude from `svIdAllow` any unhealthy SVs, as indicated by the `satdata(i).health` flag (0 = healthy; otherwise unhealthy).
7. Call `performNavigationSolution`.
8. Solve for the average receiver antenna position in ECEF coordinates assuming a stationary antenna.
9. Plot the horizontal offset of each individual solution from the average solution as a single dot with East along the x axis and North along the y axis, in meters. The processing for this type of horizontal displacement plot was described in Homework 1.

10. Plot as a function of time (or solution index) the vertical displacement from the mean, in meters.

11. Plot as a function of time the worst-case residual at each solution epoch as contained in the `residualVec` output of `performNavigationSolution`.

12. Plot as a function of time the SV identification number for the SV with the largest residual at each solution epoch as contained in the `badSvIdVec` output of `performNavigationSolution`.

Data Collection

You'll be provided with the data already in a `channel.mat` format. My guess is that it will take a very long time to proces the entire data set. Limit the data to the first 4 hours of data for the anlysis tasks below.

Analysis Tasks

1. By taking partial derivatives, show that (7) and (8) are true. What is neglected in the approximation in (8)?

2. Run your `topsol` script on the data set you collected, assuming an elevation mask angle of 0 degrees. Set the initial guess of $c\delta t_u$ to 0 and the initial guess of $\underline{r}_u$ to

   ```
   rRxAppx = getAntLoc('Fiction_Island');
   ```

   which is the approximate ECEF location of somewhere far away. Include all plots generated by `topsol` in your report.

   Pro tip: Work at first with a short set of data—perhaps just a single epoch of data—until your solver begins functioning correctly. You can also set epochStride to $N > 1$ during debugging to process only every Nth epoch, which will speed up your processing.

3. The nonlinear least squares solution is surprisingly forgiving as regards your initial choice of `rRxAppx`. Investigate what happens to successive position estimates as you iterate within the nonlinear least squares solver (i.e., as you successively set $\underline{\hat{x}}_{new} = \underline{x}^*$ after each iteration of your solver). Where does Fiction Island lie on a map? Print out the position component of first 6 iterations of your navigation solution at some solution epoch and include these in your report. Then plot the position components of the first 6 iterations in a 3-d plot in Matlab, showing how your solution moves with each iteration from Fiction Island to the correct location. Experiment with other values of `rRxAppx`. See if you can find a value of `rRxAppx` that causes the least squares solver to fail. Draw some conclusions about the "attractor region" of the nonlinear solver.

   Note that because you don't have even an approximate guess for the true receiver location for this mystery data set, you'll need to disable elevation masking on your first run by setting `elMaskDeg = -90`. Having obtained an approximate position guess from your first run, you can perform a second run with `elMaskDeg` ≥ 0 if you'd like to eliminate some low-elevation signals.

4. Calculate HDOP and VDOP for a single epoch of data within the 10-minute interval. These values should be representative of the HDOP and VDOP values over the whole interval, since the SV geometry doesn't change much over 10 minutes. Calculate the standard deviation in the offset-from-mean of your solutions in the East, North, and Vertical directions. Compare these standard deviations to HDOP$\cdot\sigma_\rho$ and VDOP$\cdot\sigma_\rho$, with $\sigma_\rho = 2$ meters. What value of σ_ρ best fits the data?

5. Run your `topsol` script again on the data set you collected, but this time impose an elevation mask angle high enough that you only see 4 SVs. Again calculate HDOP and VDOP for a single epoch of data within the 10-minute interval. Compare the DOP quantities from these runs against the corresponding quantities from the previous run with elevation mask angle of 0 degrees.

6. Isolate the pseudorange measurements for a particular measurement epoch. Using these, solve for $\underline{x}^*$. Now add 1000 meters to every one of the pseudorange values and solve again for $\underline{x}^*$. What is the difference in the two values of $\underline{x}^*$? Explain.

7. Perform a full navigation solution on the first 100 seconds of data from your data set. Examine the average size, as an absolute value, expressed in meters, of (1) the modeled ionospheric delay, (2) the modeled neutral atmospheric delay, (3) the relativistic correction δt_{rel}, and (4) the change in modeled pseudorange with and without accounting for earth rotation during signal transit. Average over all pseudoranges at each solution epoch and over all solution epochs in the first 100 seconds of data. Make a bar plot showing the average size of each effect. Rank these four effects in terms of size, and therefore, importance.

8. Choose one of the L2C-capable SV that was visible throughout your data set. Extract the beat carrier phase and the pseudorange data corresponding to this SV for both the GPS L1 C/A signal (`GenericType` = 0) and whichever GPS L2C variant is being tracked, whether CM (`GenericType` = 1), CL (`GenericType` = 2), or CLM (`GenericType` = 3). From these data, form the pseudorange-based and carrier-phase-based delay estimates $\hat{I}_\rho$ and $\hat{I}_\phi$ at the L1 frequency as we discussed in lecture. Negate $\hat{I}_\phi$ so that it trends in the same direction as $\hat{I}_\rho$. Then take the mean of the difference between $\hat{I}_\rho$ and the negated $\hat{I}_\phi$ to estimate the constant bias in the negated $\hat{I}_\phi$. Add this bias to the negated $\hat{I}_\phi$ and plot the rough $\hat{I}_\rho$ and the smooth (negated and unbiased) $\hat{I}_\phi$ together on the same graph. The smooth line should trace through the mean of the rough line. Approximate the RMS noise on the smooth and the rough plots. In terms of RMS noise (units of meters), how much more precise is the carrier-phase-based delay estimate than the pseudorange-based delay estimate?

   Now repeat these steps for another L2C-capable SV that was visible throughout your data set.

9. Your ionospheric delay plots from Task 8 may be significantly biased from the true delay. You may even end up with a *negative* delay estimate, which as you know is impossible for a group delay (it would violate special relativity). Whatever bias is present in your delay plots is due to (1) a slight misalignment of the L1 C/A code and the L2C code at the transmitter (transmitter differential code bias, $\delta\rho_S$) and (2) an excess delay in the receiver's L1 signal path vs. its L2 signal path, or a sample train misalignment between the L1 and L2 portionas of the receiver's RF front end (collectively called receiver inter-frequency code bias, $\delta\rho_R$). Both biases are expressed in meters. If you knew the exact values of these biases, you could eliminate them by calculating $\hat{I}_\rho$ as

$$\hat{I}_\rho = \gamma_1 \left(\rho_{L2} - \rho_{L1} - \delta\rho_S - \delta\rho_R\right)$$

   One way to estimate these biases is to compare the raw value of $\hat{I}_\rho$ (without attempting to eliminate the biases) with a model-based estimate of I_ρ. Pretend that the broadcast model embodied in `getIonoDelay` is perfectly accurate over the 4-hour interval of your data set. Plot the ionospheric delay predicted by this model for your chosen SVs on a copy of your plots from Task 8. Do the trends in this estimate match those of your dual-frequency-based estimates? Assuming the `getIonoDelay` model's predictions are perfectly accurate, estimate a value for the sum $\delta\rho_S + \delta\rho_R$, one for each of the two L2C-capable satellites.

   Hint: You can get the ionospheric delay predicted by `getIonoDelay` for a chosen SV over the interval by selectively saving the `getIonoDelay` outputs during `topsol` processing. But processing CORS data for the whole data set interval takes a long time when the epoch spacing is small. You can shorten your wait by skipping every N_{stride} epochs in the data output by `prepareTimeHistory.m`. In the example `topsol` found on the course website, this is done by setting `epochStride` to N_{stride}. The modeled ionosphere will still look smooth even if you only process epochs spaced by, say, 30 seconds.

Wrap Up

Prepare a formal laboratory report for this lab. As an appendix to this report, attach a hard copy of your Matlab `satpr` and `performNavigationSolution` functions and your `topsol` script. Also send these three files *and all necessary supporting functions* to the TA by email in a zipped archive (using either Windows zip, 7-zip, or gzip).