ASE 389P-7 Problem Set 7

Posting Date: December 2, 2024

You need not hand in anything. Instead, be prepared to answer any of these problems—or similar problems—on an upcoming take-home exam. You may discuss your solutions with classmates up until the time that the exam becomes available, but do not swap work (including code).

Readings

None.

Problems

1. Write a Matlab function that executes a single update to a first-order carrier-aided code tracking loop. In lecture, we derived the mean of the error signal e_k for a coherent phase detector of the form $e_k = C[I_{e,k} - I_{l,k}]$, where C is a normalization constant given by $C = \frac{T_c}{2N_k}$. We showed that $E_c[e_k] = \Delta t_k$.

   The coherent code phase detector is simple and has low error variance, but it requires the phase tracking loop to be in lock so that the quadrature components of S_k can be safely ignored.

   Consider instead the *dot-product* code phase detector:

   $$e_k = C\left[(I_{e,k} - I_{l,k})I_{p,k} + (Q_{e,k} - Q_{l,k})Q_{p,k}\right]$$

   where the normalization constant C is given by

   $$C = \frac{T_c}{2}\left[\frac{2}{N_k \bar{A}_k}\right]^2$$

   It can be shown that for small code phase errors, the mean value of e_k is approximately equal to Δt_k seconds.

   The dot-product phase detector has slightly higher error variance than the coherent detector but does not require phase lock and so is better suited to the initial stages of tracking immediately following acquisition. Implement your code tracking loop with a dot-product phase detector. To obtain the normalization constant C, estimate $E[|S_k|^2]$ by averaging the past values of $|S_k|^2 = I_k^2 + Q_k^2$ (you could set up a moving-window average over the past 100 values, for example, or you could set up a low-pass filter that effectively does the same thing). You'll find in your lecture notes (and in the interface below) a relationship between $E[|S_k|^2]$ and N_k, $\bar{A}_k$, and σ_{IQ}. Use this relationship to compute the normalization constant C from your estimate of $E[|S_k|^2]$.

   Your code tracking update function should adhere to the following interface:

```
function [vTotal] = updateDll(s)
% updateDll : Perform a single update step of a carrier-aided first-order code
%             tracking loop.
%
%
% INPUTS
%
% s ------------- A structure with the following fields:
%
%    Bn_target -- Target bandwidth of the closed-loop code tracking loop, in
%                 Hz.
%
%    IsqQsqAvg -- The average value of |Sk|^2 = Ik^2 + Qk^2; also equal to
%                 (Nk*Abark/2)^2 + 2*sigmaIQ^2.
%
%    sigmaIQ ---- The standard deviation of the noise in the prompt in-phase
%                 and quadrature accumulations.
%
%    Ip --------- The in-phase prompt accumulation over the interval from
%                 tkm1 to tk.
%
%    Qp --------- The quadrature prompt accumulation over the interval from
%                 tkm1 to tk.
%
%    Ie --------- The in-phase early accumulation over the interval from
%                 tkm1 to tk.
%
%    Qe --------- The quadrature early accumulation over the interval from
%                 tkm1 to tk.
%
%    Il --------- The in-phase late accumulation over the interval from
%                 tkm1 to tk.
%
%    Ql --------- The quadrature late accumulation over the interval from tkm1
%                 to tk.
%
%    vp --------- The aiding signal from the phase tracking loop, in seconds
%                 per second.  This is equal to the Doppler frequency shift
%                 that will be used to drive the receiver's carrier-tracking
%                 numerically controlled oscillator during the time interval
%                 from tk to tkp1, divided by the carrier frequency and
%                 multiplied by -1 (high-side mixing) or 1 (low-side mixing)
%                 depending on whether the RF front-end peforms high- or
%                 low-side mixing.  Thus, vp = sMix*fDk/fc, with sMix = +/- 1.
%
%    Tc --------- Spreading code chip interval, in seconds.
%
% OUTPUTS
%
%    vTotal ------ The code tracking loop's estimate of the code phase rate at
%                  sample time tk, in sec/sec. vTotal is equal to the code
%                  tracking loop's correction term v plus the carrier aiding
%                   term vp.
%
%+------------------------------------------------------------------------------+
% References:
%
%
%+==============================================================================+
```

2. Develop Matlab code that acquires and tracks a single GPS L1 C/A signal. The tracking component of your code will be a mechanization of the "full system" block diagram introduced in lecture. Apply your code to the data in the binary file

   `http://radionavlab.ae.utexas.edu/datastore/gnssSigProcCourse/dfDataHead.bin`

   which includes about 70 seconds of digitized IF data from a GP2015 front end. This is the same data set you used to perform acquisition in Problem Set 5. Signals corresponding to PRNs 14, 16, 20, 22, 29, 30, 31, and 32 are present in the data. Don't forget that the GP2015 is a high-side-mixing front end.

   Track the GPS signal corresponding to PRN 14 over the complete data set. PRN 14's signal is strong: its C/N_0 approximately equals 48 dB-Hz. Make the accumulation interval T_a equal to the C/A code interval (nominally 1 ms). (Because you'll be tracking a strong signal, there will be no need for long coherent integration, and with a code-length accumulation interval you won't have to worry about data bit transitions occurring during the interval.) Set $t_{\text{eml}} = 0.5$ chips and set the bandwidth of the carrier tracking loop to 10 Hz and the bandwidth of the (carrier-aided) code tracking loop to 0.1 Hz. Use a 3rd-order carrier phase tracking loop and a 1st-order carrier-aided code phase tracking loop. Empirically determine a value for σ_{IQ}^2 by using your acquisition routine to calculate and average hundreds of null-hypothesis S_k values (e.g., corresponding to a PRN that is not present in the data).

   Here is a step-by-step procedure for your top-level code:

   (a) Perform a coarse acquisition of the target PRN using your FFT-based acquisition function operating on an accumulation time of $T_a = 1$ ms. The outputs of the acquisition function should be (1) an estimate $\hat{t}_s(\tau_{j_0})$ of the start time of the 0th code interval (you can approximate this as τ_{j_0}, the receiver time of the sample that immediately follows the beginning of the 0th C/A code), and (2) the approximate Doppler frequency $f_{D,0}$ over the 0th code interval.

   (b) Perform a fine acquisition of the target PRN. Because of the short accumulation time used in the coarse acquisition, the coarse acquisition processing will be fast but $f_{D,0}$ may not be accurate enough for the Doppler error to be within the "pull-in" range of your phase tracking loop. Refine your initial estimate of f_D by performing FFT-based acquisition with $T_a = 10$ ms over a reduced test interval for f_D. You'll need to modify your FFT-based acquisition routine to handle the longer T_a.

   (c) Initialize the beat carrier phase estimate $\hat{\theta}(\tau_{j_0})$ to some arbitrary value (e.g., 0).

   (d) Initialize a moving-window average of $|S_k|^2 = I_{p,k}^2 + Q_{p,k}^2$ with the peak $I_{p,0}^2 + Q_{p,0}^2$ value obtained during acquisition. This average can be converted to C/N_0 and is useful for calculating the normalization constant C required in the dot-product code phase detector.

   (e) Call `configureLoopFilter` function to set up the phase tracking loop filter.

   (f) Figure out how to set the initial state $x_{k=0}$ of the phase tracking loop filter so that for a zero initial error signal $e_k = 0, k = 0, 1, 2, ...$, the loop would output $v_k = 2\pi f_D, k = 0, 1, 2, ...$, where f_D is the Doppler estimate (in Hz) produced by the acquisition routine. "Priming the loop filter" like this is necessary for all 2nd- and 3rd-order carrier tracking loops because they have state (memory). Note that solving for the 3rd-order loop's $x_{k=0}$ will require you to think some. Your solution will involve both the A matrix and the C matrix of the state space representation of $D[z]$.

(g) Pass the input data starting at sample $x(j_k)$ to a function that performs correlation over one accumulation interval to produce early, prompt, and late accumulations $S_{e,k}, S_{p,k}, S_{l,k}$. The function should mechanize the correlation "recipe" introduced in Problem Set 5. Other inputs to the function will include: f_{IF}, $\hat{t}_s(\tau_{j_k})$, $v_{\theta,k} = 2\pi f_{D,k}$, $\hat{\theta}(\tau_{j_k})$, t_{eml}, and the target PRN. The function should call a sub-function that generates early, prompt, and late versions of the oversampled C/A code corresponding to the target PRN.

(h) Update the moving-window average of $|S_k|^2 = I_k^2 + Q_k^2$ using the prompt accumulation.

(i) Route the correlation outputs $S_{e,k}, S_{p,k}$, and $S_{l,k}$ to `updatePll` and then to `updateDll`.

(j) Update the beat carrier phase estimate that will apply at time $\tau_{j_{k+1}}$, which is the receiver time of the first sample of the $(k+1)$st accumulation, according to the formula

$$\hat{\theta}(\tau_{j_{k+1}}) = \hat{\theta}(\tau_{j_k}) + v_{\theta,k}(\tau_{j_{k+1}} - \tau_{j_k})$$

where $v_{\theta,k} = 2\pi f_{D,k}$ is the Doppler estimate, in radians per second, produced by the phase tracking loop after operating on $S_{p,k-1}$, the prompt complex accumulation whose final participating sample occurs at time $\tau_{j_k - 1}$.

(k) Operate on the output of `updateDll` to obtain $\hat{t}_s(\tau_{j_{k+1}})$ as described in lecture.

(l) If all data have been exhausted, then quit. If not, set $k \leftarrow (k+1)$ and go to step 2g.

Hints: As a check, compare the Doppler estimates produced by your refined acquisition routine with those given in the GRID screenshot at $\tau = 3$ seconds for the same data:

```
==========   GRID: General Radionavigation Interfusion Device   ===========
 Receiver time:    0 weeks      3.0 seconds        Build ID:       3349
 GPS time:         0 weeks      0.0 seconds
---------------------------------------------------------------------------
CH  TXID    Doppler       BCP            PR      C/N0    Az      El   Status
             (Hz)       (cycles)      (meters)  (dB-Hz) (deg)  (deg)
---------------------------- GPS_L1_CA_PRIMARY ----------------------------
 1    5u    -3664.5           0.0           0.0  38.2     0.0     0.0     5 *
 2   14u    -2208.0           0.0           0.0  49.9     0.0     0.0     5 *
 3   16u     3341.3           0.0           0.0  47.8     0.0     0.0     5 *
 4   20u     1472.3           0.0           0.0  49.6     0.0     0.0     5 *
 5   22u    -3811.3           0.0           0.0  39.0     0.0     0.0     5 *
 6   29u     1602.1           0.0           0.0  44.4     0.0     0.0     5 *
 7   30u    -2775.9           0.0           0.0  48.1     0.0     0.0     5 *
 8   31u     -256.5           0.0           0.0  54.6     0.0     0.0     5 *
 9   32u      396.1           0.0           0.0  52.7     0.0     0.0     5 *
10   --   ---------  -------------  -----------  ----   -----  -----    -
===========================================================================
```

Also compare the Doppler time history for PRN 14 with the one shown in Fig. 1.

Note that the Doppler values reported by GRID follow the RINEX convention: in the absence of receiver and satellite clock errors, approaching satellites will give rise to a positive Doppler. Thus, the Doppler values shown in the screenshot and in Fig. 1 will be opposite in sign compared to the apparent Doppler in `dfDataHead.bin`, which is high-side mixed.

Also, recall that the GP2015 produces digitized data with an intermediate frequency $f_{IF} = 1.405396825396879$ MHz and a sampling rate $N_s = 40e6/7$ samples per second. In the absence of Doppler, there would be $N_s/1000 = 40000/7 \approx 5714$ samples per GPS L1 C/A code.

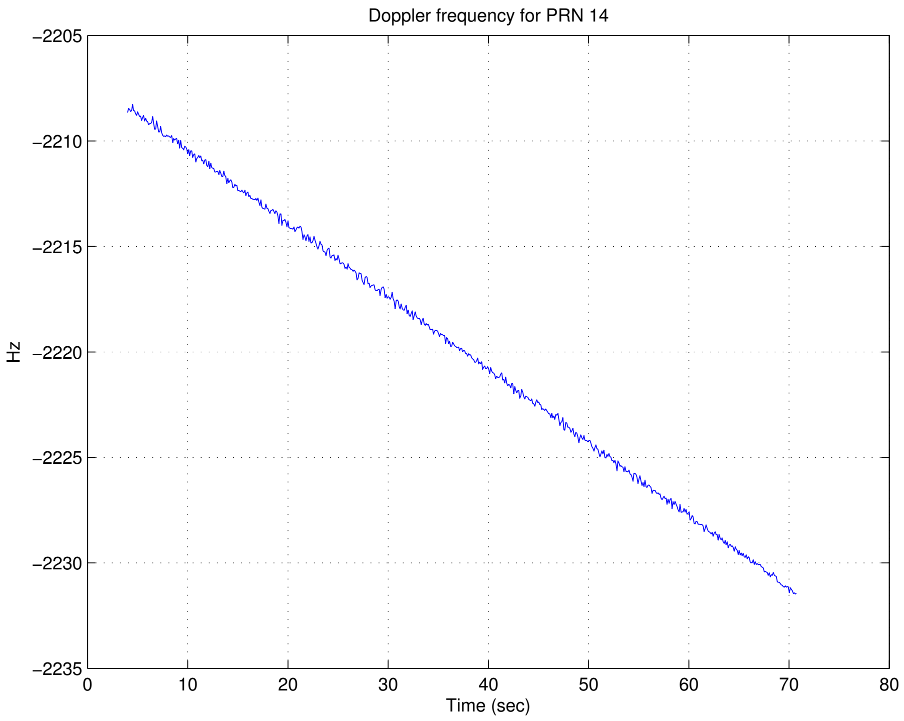

Figure 1: Doppler time history for GPS PRN 14 based on GRID tracking of `dfData.bin`. Note that GRID produced this time history with $T_a = 10$ ms and a PLL bandwidth of $B_{\text{PLL}} = 7.5$ Hz. Your plot will look noisier if you use a shorter T_a or a wider B_{PLL}. This is expected.

If all else fails: First, have a look at the single accumulation peformed in the script `exampleAccumulation.m`. This simple operation is the basis of acquisition and tracking. Make sure your code is performing it correctly.

Next, consider open-loop tracking. One of the challenges of getting signal tracking to work is that both code and carrier tracking have to work—tracking fails if either one has problems. A good approach to diagnosing problems is a divide-and-conquer scheme whereby you implement a simple open-loop strategy for code tracking. This will work well enough to allow you to fix problems with carrier tracking, after which you can turn your focus to getting full feedback code tracking to work. Here is the approach (for PRN 14):

(a) Take $f_D = -2208$ Hz as your initial Doppler value. Note that this is the RINEX-convention Doppler value (see the boxed note above). Since the GP2015-produced data in `dfData.bin` are high-side mixed, acquisition and tracking of PRN 14 will see the *negative* of this Doppler value, or 2208 Hz. Prime your carrier tracking loop filter (step (f) above) so that $v_{\theta,0} = -2\pi f_D = 2\pi \cdot 2208$ rad/sec.

(b) Calculate an approximate code interval for PRN 14. You know that each GPS C/A code is approximately 1 ms. But the code shortens if Doppler is positive (i.e., if the SV is moving toward you or your clock is running slow). Accounting for Doppler, the code interval over the first few seconds of the data should be very close to

$$T_{\text{code}} = \frac{0.001}{1 + \frac{f_D}{f_{L1}}} \quad \text{seconds}$$

The Doppler value used in this equation is the RINEX-convention Doppler value, i.e., $f_D = -2208$ for PRN 14.

(c) Calculate the code start time for the first few seconds by assuming that the code interval is exactly T_{code}. Although this isn't actually the case, your code start times will be close enough that you'll have strong prompt accumulations for a few seconds. This is open-loop code tracking. Eventually, because you're not employing any feedback in the code tracking, your prompt tap will slip down the correlation peak and your $S_{p,k}$ values will diminish. But that won't happen until after a few seconds. Meanwhile, you'll get enough strong $S_{p,k}$ values to get your carrier tracking working. Let $\hat{t}_{s,0}$ be the approximate code start time you got from acquisition. Then the kth code start time (in seconds) can be approximated as

$$\hat{t}_{s,k} = \hat{t}_{s,0} + kT_{\text{code}}$$

(d) Use the open-loop value of $\hat{t}_{s,k}$ you calculated to generate the local code replica you'll need for the kth accumulation.

(e) Feed the $S_{p,k}$ values you get from correlation to your carrier tracking loop. Work with that loop until you see it doing its job of aligning the $S_{p,k}$ to the real axis.

(f) Once carrier tracking is working, close the code tracking feedback loop.

3. Recall that in lecture we defined the pseudorange measurement in terms of the recipe by which it is computed within a GNSS receiver:

$$\rho(t_R) = c\,[t_R - t_S]$$

In this expression, c is the speed of light, t_R is the receiver clock time at receipt of the alignment feature and t_S is the satellite clock time at the time of transmission of the alignment feature. Time values are typically expressed in seconds and c in meters per second, leaving ρ in units of meters. The alignment feature is usually taken to be the beginning of a spreading code sequence (e.g., the beginning of a 1-ms GPS L1 C/A code), or the beginning of some chip within the spreading code.

Repeat Problem 2 for PRNs 16, 20, 29, 30, and 31. Each sample in the data set arrives with a uniform spacing of $T_s = (40e6/7)^{-1} = 175$ ns according to the receiver clock. In fact, the sample train can be thought of as *defining* the receiver clock: the arrival of each sample is the "tick" of the receiver clock.

If you have coded your tracking loops correctly, you will find that the spreading code phase for PRN 14 is aligned such that a particular C/A code happens to begin just after sample 200048937 (assume that the first sample is considered to be sample 0). Let's call the start of this C/A code our "alignment feature." As it turns out, this alignment feature was transmitted from the PRN 14 satellite at the following GPS time: 1490 weeks, 146046.705 seconds. GPS time is counted in weeks and seconds of week since 00:00:00 on January 6, 1980 (a Sunday). This is "zero hour" for GPS. This is a convenient time base—weeks and seconds are much easier to work with than years, months, days, hours, minutes, and seconds.

In the table below, you'll find start times of C/A codes for the PRNs you were asked to track, as measured by the GRID receiver. These start times occur about 35 seconds into the data. The start times are given in terms of *elapsed* receiver samples (the first sample corresponds to 0 elapsed samples, the second to 1 elapsed sample, etc.). They are all within 1 ms of each other according to the receiver clock. You'll also find the time of transmission of the beginning of the corresponding C/A codes, according to the satellite clock. Only the seconds of week are given. Notice that these "time stamps" applied by the satellite are given in integer milliseconds. This is because, according to each satellite's clock, each C/A code starts exactly 1 ms after the last one, with the first C/A code of each week starting at exactly 00:00:00 on Sunday morning.

PRN	tR (samples)	tS (seconds)
14	200048937.4106447696685791	146046.705
16	200047382.089815676212310791	146046.703
20	200049447.80586522817611694	146046.704
29	200046426.1354338526725769	146046.697
30	200046329.94415956735610962	146046.703
31	200050768.35572600364685059	146046.710

From the data in this table, calculate pseudorange measurements for each of the 6 PRNs. You'll need to convert the receiver time stamps from samples to seconds and apply a constant offset to bring the receiver time stamps close to GPS time (expressed here in seconds of week). Your pseudorange values may be different from others in the class because the offset you choose is arbitrary.

The receiver-expressed start times given here are useful as a check of your tracking loops. Be prepared to generate start times for another data set for the final exam.

4. Recall that in lecture we developed a model for pseudorange of the form

$$\rho = \Delta r + c[\delta t_R - \delta t_S] + I_\rho + T + w_\rho$$

Given this model, one can construct a nonlinear least squares problem to solve for the unknown parameter vector $\boldsymbol{x} = [x, y, z, c\delta t_R]^T$, where $[x, y, z]$ is the receiver position in meters in the ECEF reference frame and $c\delta t_R$ is the receiver clock error multiplied by the speed of light. To solve this problem, one needs to know (1) the position of the transmitting SV at the instant of transmission, (2) the SV's clock offset δt_S at the transmitting instant, and (3) I_ρ and T.

The column `tR` in the following table gives the start time of C/A codes for each of 4 PRNs whose signals are being tracked by a receiver, according to the receiver clock. The column `tS` gives the time of transmission of the codes, according to the satellite clock.

```
PRN    tR (seconds)        tS (seconds)
-----------------------------------------
14     429615.504253994    429615.436
21     429615.504514099    429615.446
22     429615.505066282    429615.440
27     429615.505190982    429615.438
```

Apply the recipe presented in lecture to calculate pseudoranges for all 4 PRNs. Express your pseudoranges in meters to three decimal places.

The data in the table below are valid for the pseudoranges you calculated. The SV position `rSvECEF` is given in meters in the ECEF reference frame as $[X, Y, Z]$. The rest of the quantities are given in seconds. I_ρ and T are related to δt_{Iono} and δt_{Tropo} by

$$I_\rho = c\delta t_{\text{Iono}}, \qquad T = c\delta t_{\text{Tropo}}$$

```
Data for PRN 14:
rSvECEF = -15206930.706156    -20832520.6664809     -6469517.7047024
dtIono  =  2.33838930420718e-08
dtTropo =  1.90304034510942e-08
dtS     =  0.000116168688750703

Data for PRN 21:
rSvECEF =  644129.213511344    -22105331.2073303     14614081.7413721
dtIono  =  1.07933295823888e-08
dtTropo =  8.36092156790743e-09
dtS     = -0.000109838783947257


Data for PRN 22:
rSvECEF = -18139647.3560073    -12028727.5758169     15453336.7251709
dtIono  =  1.69238352966293e-08
dtTropo =  1.31971700978366e-08
dtS     =  0.000156846547048633


Data for PRN 27:
rSvECEF =  17478991.6859777    -16545561.3989088     11624502.8670108
dtIono  =  1.71386691737267e-08
dtTropo =  1.61167561114859e-08
dtS     =  0.000217636932174885
```

By combining these data with your four pseudorange measurements, you have all the quantities you need to solve for the vector $\boldsymbol{x}$. Arrange the four pseudorange measurements into a measurement vector as $\boldsymbol{z} = [\rho_1, \rho_2, \rho_3, \rho_4]^T$. Then employ the pseudorange measurement model above, plugging in appropriate values for $\boldsymbol{r}_S$, δt_S, I_ρ, and T, to express $\boldsymbol{z}$ as a nonlinear function of $\boldsymbol{x}$:

$$\boldsymbol{z} = \boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{w}$$

A good estimate of $\boldsymbol{x}$ will minimize the following least-squares cost function:

$$J(\boldsymbol{x}) = ||\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{z}||^2$$

In other words, one can estimate $\boldsymbol{x}$ as

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}} J(\boldsymbol{x})$$

Follow this least-squares approach to estimate the location and δt_R of the receiver at the instant corresponding to the measured pseudoranges. Express your location $[x, y, z]$ in meters in the ECEF reference frame. You may wish to convert these Cartesian coordinates to latitude, longitude, and altitude if you're curious about the location. Express δt_R in seconds to six decimal places.

Hint: The correct receiver location $[x, y, z]$ is one of the following possibilities:

```
(A)  -738641.6553  -5462720.0395  3197898.1150
(B)  -741133.4180  -5456322.9775  3208395.3863
(C)  -740312.5579  -5457063.2879  3207249.7207
(D)  -741991.1305  -5462229.1655  3198021.3786
(E)  1101968.2340  -4583484.2540  4282240.1430
(F)  6378137.0000        0.0           0.0
```