ASE 389P-7 Problem Set 2

Posting Date: September 8, 2022

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.

1. In lecture, we showed that the triangular signal $\Lambda(t)$ can be thought of as the convolution of two rectangular pulses: $\Lambda(t) = \Pi(t) * \Pi(t)$. We then showed that the Fourier transform of the rectangular pulse $\Pi(t)$ is the $\text{sinc}(f)$ function; that is,

$$\mathcal{F}[\Pi(t)] = \frac{\sin \pi f}{\pi f} = \text{sinc}(f)$$

We then concluded that the power spectral density of a random binary sequence with chip interval T_c is given by

$$\begin{aligned} S_X(f) &= \mathcal{F}[\bar{R}_X(\tau)] \\ &= \mathcal{F}[\Lambda(\tau/T_c)] \\ &= \frac{1}{T_c}\mathcal{F}[\Pi(\tau/T_c) * \Pi(\tau/T_c)] \\ &= \frac{1}{T_c}\mathcal{F}[\Pi(\tau/T_c)] \cdot \mathcal{F}[\Pi(\tau/T_c)] && \text{(by convolution theorem)} \\ &= \frac{1}{T_c}[T_c\,\text{sinc}(fT_c)] \cdot [T_c\,\text{sinc}(fT_c)] && \text{(by time} - \text{scaling property)} \\ &= T_c\,\text{sinc}^2(fT_c) \end{aligned}$$

We noted in lecture that over 90% of the power in $S_X(f)$ lies in the interval $-1/T_c < f < 1/Tc$, which delimits the main lobe. By numerical integration, determine the percentage of power in the interval $-2/T_c < f < 2/Tc$, which contains the main lobe and the first side lobes. Express the percentage to three decimal places (e.g., 90.282%).

Hints: Without loss of generality and to make things easy, we can set $T_c = 1$. Also, simple Euler numerical integration will be adequate if your integration steps are small enough. If you perform the integration in Matlab and you form the $\text{sinc}(f)$ function by explicitly calculating $\frac{\sin \pi f}{\pi f}$, you'll note that Matlab returns a `NaN` (not-a-number) for the value of $\text{sinc}^2(0)$. Therefore, you'll have to replace this `NaN` with unity. Or maybe you'd rather just use Matlab's built-in `sinc` function, which kindly handles this special case for you.

2. The null-to-null bandwidth B_1 of the main lobe of the power spectrum $S_X(f) = T_c\,\text{sinc}^2(fT_c)$ of a binary random process with chip interval T_c is $2/T_c$. So, for example, the GPS L_1 C/A code, for which $T_c \approx 1$ us, has $B_1 \approx 2$ MHz. Because of the rounded shape of the $\text{sinc}^2(f)$ function,

the spectrum within B_1 is not filled uniformly with power—in fact, there is very little power at the edges of the B_1 interval where, as it turns out, power matters most for accurate code phase (i.e., pseudorange) measurements. Let's consider a hypothetical radionavigation system with a spreading waveform that makes more efficient use of the spectrum within B_1. Let the (equivalent baseband) power spectrum of the hypothetical system's spreading waveform be given by

$$S_X(f) = \frac{1}{W}\Pi(f/W)$$

This waveform has a two-sided bandwidth of W Hz.

(a) Find the autocorrelation function $R_X(\tau)$ of the spreading waveform by taking the inverse Fourier transform of $S_X(f)$.

(b) How wide is the main peak in the autocorrelation function $R_X(\tau)$ (from first left to first right zero-crossing)?

(c) Compare this width to the width of the peak of the autocorrelation function $\bar{R}_X(\tau)$ for a random binary sequence with a null-to-null bandwidth $B_1 = W$.

(d) Which of the two autocorrelation functions would lend itself to a more accurate determination of τ_{peak}, the location of the autocorrelation peak, based on a single pair of autocorrelation function samples spaced by $\Delta\tau = 0.2/W$ seconds in a high-C/N_0 scenario? Does you answer change for a low C/N_0 scenario? Assume a receiver with a wide front-end bandwidth of 10 W Hz so that there is not any significant rounding of the triangular autocorrelation function's peak. Explain.

(e) Why do you suppose the random binary sequence (actually a pseudorandom approximation of it) was historically used for GPS?

3. Suppose you buy a fancy low-noise amplifier and cryogenic cooling system for your GNSS receiver so that your receiver temperature drops to $T_R = 50$ K. Assuming an antenna temperature of 100 K, calculate the corresponding thermal noise density N_o in dBW/Hz.

   Now consider how this reduction in noise density gets offset by the interference effects of other GNSS signals due to "multiple access." Suppose that the desired signal is one of $M = 10$ received signals, each with received power $P_s = -155$ dBW. What will be the equivalent noise + interference density $N_{0,eq}$? Assume a chip interval equivalent to that of the GPS L1 C/A signal, or $T_c = 1023^{-1}$ ms.

4. If $C(t)$ is a random binary ($\pm$ 1) spreading code with rectangular pulses, then, as we showed in lecture, the power spectral density of $C(t)$ is given by

$$S_C(f) = T_C \,\text{sinc}^2(fT_C)$$

   where T_C is the chipping interval.

   Suppose instead that $C(t)$ has psd

$$S_C(f) = T_C\Pi(fT_C)$$

   where $\Pi(f)$ is the rect function introduced in lecture. Imagine a constellation of GNSS satellites broadcasting signals having this new $S_C(f)$. Assume the signals have no data bit modulation, so that the signal's baseband representation is of the form

$$r(t) = \sqrt{P}C(t)\exp[j\theta(t)]$$

For this case, calculate $I_0 \triangleq S_I(0)$ for a single multiple-access interference signal with power P_I, where I_0 and $S_I(f)$, the power spectrum of $I(t)$, were defined in lecture. Now consider M multiple-access signals (including the desired signal). If each of the M received signals has power P_I, at what value of M does I_0 exceed N_0? Express your answer in terms of N_0, T_C and P_I.

5. In this problem, you'll recreate the power spectra shown in lecture for the signals taken with the Stanford 46-meter diameter dish and with a patch antenna. To do this, download the scripts `bigDishPSD.m` and `gridPSD.m` from Canvas and download the data files `dfDataHead.bin` and

   `prn31_22apr03_01hrs40min00sec_gmt_fl1_46_08mhz_250msec.mat`

   from the server at

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

   Also download the files `convertBitPackedData.mexw32` and `binloadSamples.m` from Canvas. Put these files in the same directory as the other Matlab scripts. The Matlab function `binloadSamples.m` converts data from the binary file `dfDataHead.bin` to a Matlab vector. `binloadSamples.m` depends on the MEX file `convertBitPackedData.mexw32` to perform the conversion. If your platform is not 32-bit Windows, you will have to recompile this MEX file from the source code `convertBitPackedData.c` using the Matlab `mex` command.

   Change the path to the data in the Matlab scripts `bigDishPSD.m` and `gridPSD.m` to the appropriate path for your platform. Then run the scripts. This should produce for you the same power spectra displayed in lecture.

   Read the Matlab documentation for the `pwelch` function. Experiment with the `nfft` argument to observe the tradeoff between good frequency resolution (large `nfft`) and low variance of the estimate within each frequency bin (small `nfft`). In the `gridPSD.m` script you can also experiment with taking in more data. The file `dfDataHead.bin` includes about 70 seconds worth of data.

   Show plots for both the "big dish" data the patch antenna data. Show plots that exhibit the tradeoff mentioned above for large and small `nfft`. Describe the effect that taking in more or less data for a fixed `nfft` has on the estimated power spectrum, and show plots that reveal this effect, using the patch antenna data. As you probe the low end of this range (toward short data segments), you'll need to take in a very short segment (`Tfull` < 0.1) s to see a significant effect.

6. In this problem we'll further analyze the data taken with the Stanford 46-meter diameter dish (the same data used in Problem 5). The gain of the Stanford dish is so great (about 50 dB at the GPS L1 frequency) that you can actually see the transitions induced by the GPS L1 C/A code and by the GPS L1 P(Y) code. Recall that in lecture the GPS L1 signal structure was presented as

$$S_{L1}(t) = \sqrt{2P_{C1}}D(t)C(t)\cos(2\pi f_{L1}t + \theta_{L1}) + \sqrt{2P_{Y1}}D(t)Y(t)\sin(2\pi f_{L1}t + \theta_{L1})$$

   You can see from this that the GPS C/A code (spreading code $C(t)$, chipping at 1.023 MHz) is in phase quadrature with the GPS Y code (spreading code $Y(t)$, chipping at 10.23 MHz).

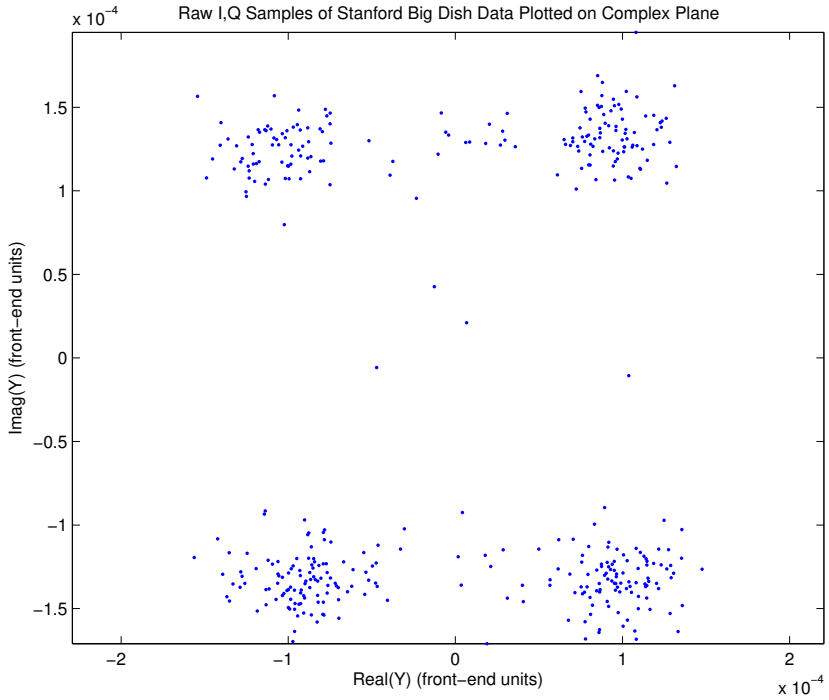

Figure 1: Individual samples of the Stanford "Big Dish" data plotted in the complex plane.

Load the "Big Dish" data into your workspace in Matlab (you can use the bigDishPSD.m script to do this). The sampling rate for these data is $f_s = 46.08$ MHz. The data are loaded into the Matlab complex variable `Y`. Each element in `Y` represents a complex sample of the GPS signal $S_{L1}(t)$ after mixing to baseband.

The researchers who recorded the data mixed the incoming signal to baseband with a fairly accurate estimate of the signal's instantaneous center frequency (including Doppler). Hence, the baseband samples have only a small residual frequency, which gives rise to a slow rotation in the phase. If you only look at, say, 400 samples, the phase doesn't change much and you can see a pattern emerge when you plot individual samples in the complex plane. (If you look at much more than 400 samples, the slow phase rotation turns this pattern into a donut shape.)

Generate a plot similar to Fig. 1 by finding a window of 400 samples of data within `Y` whose real and imaginary components form a square that is aligned with the plot axes. Having identified such a window, look at the real and imaginary components of `Y` over this window in the time domain. Identify for your plot what signal component from $S_{L1}(t)$ happens to be in `real(Y)` and what component happens to be in `imag(Y)` over this window. Measure the smallest chip interval for both codes. Estimate from your data the ratio of the C/A code amplitude to the $P(Y)$ code amplitude.

7. Verify by "Matlab experiment" that the power spectral density of a random binary sequence is given by $S_X(f) = T_c \mathrm{sinc}^2(fT_c)$. Generate a long ± 1-valued sequence in Matlab (say, $N = 2^{14}$ elements or so). Then re-sample this sequence with a sampling interval that is a small fraction of your assumed chip interval T_c. This process is called *oversampling*. To ensure good auto- and cross-correlation properties, the sampling rate should not be an integer multiple of the chipping

rate. See if you can explain why this is so. Then use the `pwelch` function to estimate the power spectrum of the oversampled sequence. Plot the result of the `pwelch` function on a dB scale. Zoom in to see just the main lobe and the first few sidelobes. What is the expected height of the main lobe in dBW/Hz? Does this match your plot? What is the effect of using the option `[...] = PWELCH(...,'twosided')`? Experiment with different values for (1) the original binary sequence length, (2) the oversampling interval, (3) the `pwelch NFFT` parameter. Describe the trends you see along each of these experimental axes.

Turn in all the code you used to respond to this question. If you wish, you may use the Matlab function defined in `oversampleSpreadingCode.m` (found on Canvas) for oversampling.

8. Write a Matlab function that computes an LFSR-generated pseudo-random sequence given as input the LFSR size, connections, and initial state. Your function should adhere to the following interface specification:

```
%[lfsrSeq] = generateLfsrSequence(n,ciVec,a0Vec)
%
% Generate a 1/0-valued linear feedback shift register (LFSR) sequence.
%
% INPUTS
%
% n ------ Number of stages in the linear feedback shift register.
%
% ciVec -- Nc-by-1 vector whose elements give the indices of the Nc nonzero
%          connection elements.  For example, if the characteristic polynomial
%          of an LFSR is f(D) = 1 + D^2 + D^3, then ciVec = [2,3]' or [3,2]'.
%
% a0Vec -- n-by-1 1/0-valued initial state of the LFSR, where a0Vec = [a(-1),
%          a(-2), ..., a(-n)]'.  In defining the initial LFSR state, a
%          Fibonacci LFSR implementation is assumed.
%
% OUTPUTS
%
% lfsrSeq -- m-by-1 vector whose elements are the 1/0-valued LFSR sequence
%            corresponding to n, ciVec, and a0Vec, where m = 2^n - 1.  If the
%            sequence is a maximal-length sequence, then there is no
%            repetition in the m sequence elements.
%
```

Turn in a paper copy of your Matlab function.

Extra credit: In lecture we introduced the so-called Fibonacci implementation of the LFSR. The Galois implementation produces identical results, but executes faster on a computer, which is important for real-time software-defined radios. Learn about the Galois implementation online or at the library and implement `generateLfsrSequence.m` by this implementation instead of the Fibonacci implementation.

9. In this problem, you will experiment with the properties of pseudorandom binary sequences generated by various means. Download the Matlab script `correlationExperiments.m` from Canvas, along with the supporting functions, `oversampleSpreadingCode.m`, `ccorr.m` and `pi2str.m`. Your own `generateLfsrSequence.m` function supplies the final required supporting function.

   The top-level script `correlationExperiments.m` generates pseudorandom binary sequences by one of three methods: (1) via the Matlab `rand` function, (2) from a mapping of digits of the transcendental number π, and (3) from m-sequences. You can use this script to experiment with the properties of sequences generated in these different ways.

   Read through the script and run it a few times with `codeType = 'rand'` to get a feel for what it does. Feel free to adapt the script—or to write an entirely new script—to answer the questions below. As part of your answer to each question, provide illustrating plots.

   (a) In lecture it was claimed that the variance of the cross-correlation

$$R_{ab}(k) = \sum_{n=1}^{N} a'_n b'_{n+k}$$

between two random ± 1-valued sequences of length N at any particular value of k is equal to N. Verify this by evaluating $R_{ab}(k = 0)$ for 10000 different random sequence pairs for some large N (e.g., $N = 1024$) and calculating the sample variance of your collected values.

(b) The autocorrelation

$$R_a(k) = \sum_{n=1}^{N} a'_n a'_{n+k}$$

of ± 1-valued m-sequences of length N is $(N+1)\delta(k) - 1$, where $\delta(k)$ is the discrete time impulse function, which equals 1 when $k = 0$ and 0 otherwise. It was also claimed in lecture that the maximum crosscorrelation value is lower-bounded as follows, where M is the number of m-sequences considered:

$$\max_k R_{ab}(k) \geq N\sqrt{\frac{M-1}{MN-1}} \approx \sqrt{N}$$

Verify these two claims for 6 different m-sequences (different connection vectors) generated by an LFSR with $n = 10$ stages (sequence period $N = 2^n - 1 = 1023$). On the last page of this problem set you'll find the connection indices for maximal length sequences for $n = 10$. You'll find the Matlab function `ccorr.m`, which performs circular correlation, convenient for calculating the auto- and cross-correlation functions of periodic sequences.

(c) To generate a continuous-time spreading signal $X(t)$ from a ± 1-valued pseudorandom sequence, we multiply each element a'_i in the sequence by the support function $p(t)$, appropriately shifted, and sum the result. Thus,

$$X(t) = \sum_{i=-\infty}^{\infty} a'_i p(t - iT_c)$$

where T_c is the chip interval. In computer code, we represent $X(t)$ by a series of discrete samples whose sampling interval is small compared to T_c. The operation of converting the sequence $\{a'_i\}$ (which can already be thought of as a roughly-sampled version of $X(t)$) to an unambiguous representation of $X(t)$ is called *oversampling*. Read through the Matlab function `oversampleSpreadingCode.m`, available on Canvas, to understand what this operation entails. Examine the auto- and cross-correlation functions $R_X(\tau_i)$ and $R_{X_1,X_2}(\tau_i)$ corresponding to an independent pair of oversampled spreading codes derived from two pseudorandom binary sequences of period $N = 1023$ generated by each of the following methods:

i. Via the Matlab `rand` function.

ii. A mapping of digits of the transcendental number π.

iii. A 10-stage maximal length LFSR.

You'll find that the script `correlationExperiments.m` is already set up to do this. For each pseudorandom source, find the ratio of the maximum of $R_{X_1}(\tau_i)$ to the maximum of $R_{X_1,X_2}(\tau_i)$ over all values of τ_i within one period. Which of the three pseudorandom sources appears to produce codes with the best autocorrelation properties? Which produces codes with the best crosscorrelation properties?

10. In problem 7 you verified by "Matlab experiment" that the power spectral density of a random binary sequence is with unity power $P_X = 1 = 0$ dB is given by $S_X(f) = T_c \operatorname{sinc}^2(fT_c)$. For a chip interval $T_c \approx 1$us, this leads to a density function whose peak value is $10 \log_{10}(T_c) = -60$ dBW/Hz. However, for a *periodic* binary sequence with period T_p, the power spectrum does not follow a continuous $T_c \operatorname{sinc}^2(fT_c)$ curve. Instead, power in the frequency domain is concentrated at intervals of $1/T_p$ Hz. Explain why this is so. Examine the power spectra of two different oversampled m-sequences of period $N = 1023$. Include 20 periods of the sequence in your oversampled representation to ensure fine frequency resolution. Do you see power concentrated at discrete frequencies? What is the variation in the power content of the first 30 power-containing frequencies? In your implementation, how many discrete Fourier transform frequency bins of width Δf (`delf` in `correlationExperiments.m`) separate the power-containing bins? Call this N_{sep}. If all the power that would have been spread out over N_{sep} bins is now concentrated in a single bin, how far above the standard $T_c \operatorname{sinc}^2(fT_c)$ curve do you expect the discrete power-containing frequency components to be on average?

11. **Extra credit:** Is is possible to generate an aperiodic deterministic sequence? If it is possible, then give an example algorithm for how the sequence would be generated. If not, then explain why.

Connection indices for maximal length sequences for $n = 10$:

```
10 stages, 2 taps:  (1 set)

[10, 7]


10 stages, 4 taps:  (10 sets)

[10, 9, 8, 5]
[10, 9, 7, 6]
[10, 9, 7, 3]
[10, 9, 6, 1]
[10, 9, 5, 2]
[10, 9, 4, 2]
[10, 8, 7, 5]
[10, 8, 7, 2]
[10, 8, 5, 4]
[10, 8, 4, 3]


10 stages, 6 taps:  (14 sets)

[10, 9, 8, 7, 5, 4]
[10, 9, 8, 7, 4, 1]
[10, 9, 8, 7, 3, 2]
[10, 9, 8, 6, 5, 1]
[10, 9, 8, 6, 4, 3]
[10, 9, 8, 6, 4, 2]
[10, 9, 8, 6, 3, 2]
[10, 9, 8, 6, 2, 1]
[10, 9, 8, 5, 4, 3]
[10, 9, 8, 4, 3, 2]
[10, 9, 7, 6, 4, 1]
[10, 9, 7, 5, 4, 2]
[10, 9, 6, 5, 4, 3]
[10, 8, 7, 6, 5, 2]


10 stages, 8 taps:  (5 sets)

[10, 9, 8, 7, 6, 5, 4, 3]
[10, 9, 8, 7, 6, 5, 4, 1]
[10, 9, 8, 7, 6, 4, 3, 1]
[10, 9, 8, 6, 5, 4, 3, 2]
[10, 9, 7, 6, 5, 4, 3, 2]
```