ECE 146B/UCSB: Digital Communication
Lab 1: Introduction to Linear Modulation

We will use the labs to gradually develop a reasonably complete matlab simulator for a linearly modulated system. Thus, later labs will build on earlier ones.

Background

Figure 1 shows block diagrams corresponding to a typical DSP-centric realization of a communication transceiver employing linear modulation. In the labs, we model the core components of such a system using the complex baseband representation, as shown in Figure 2. Given the equivalence of passband and complex baseband, we are only skipping modeling of finite precision effects due to digital-to-analog conversion (DAC) and analog-to-digital conversion (ADC). These effects can easily be incorporated into Matlab models such as those we will develop, but are beyond the scope of the current set of labs.

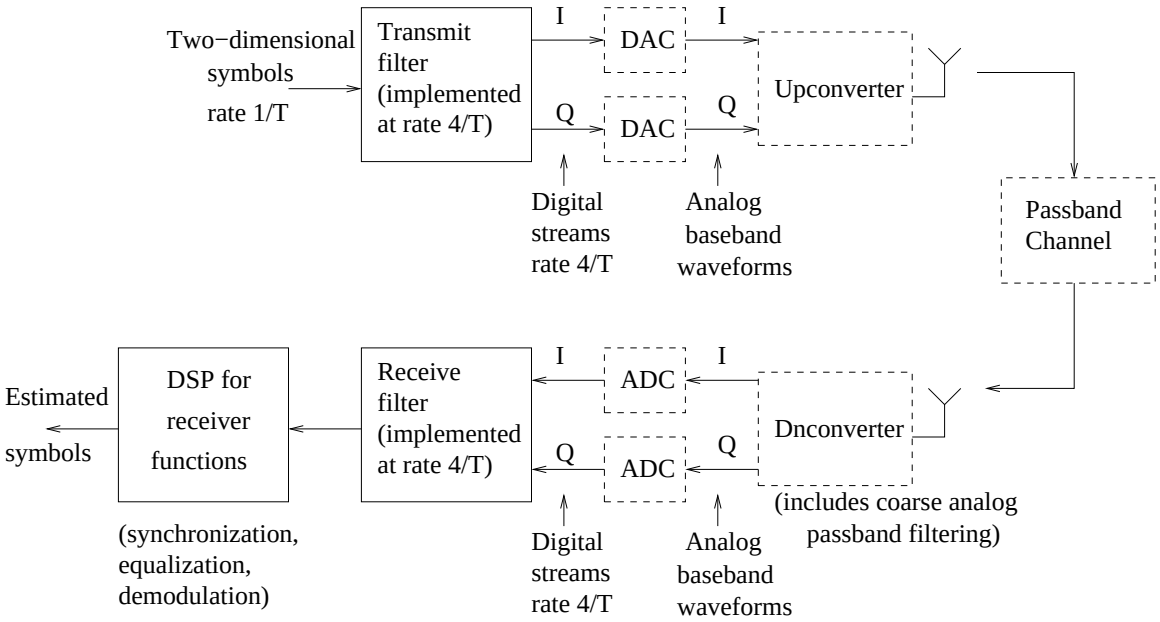

Figure 1: Typical DSP-centric transceiver realization. Our model does not include the blocks shown in dashed lines. Finite precision effects such as DAC and ADC are not considered. The upconversion and downconversion operations are not modeled. The passband channel is modeled as an LTI system in complex baseband.

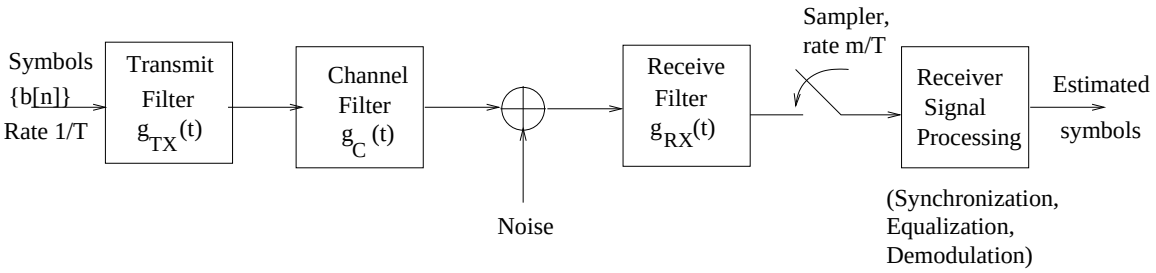

Figure 2: Block diagram of a linearly modulated system, modeled in complex baseband.

A few points worth noting about the model of Figure 2:

Choice of transmit filter: The PSD of the transmitted signal is proportional to $|G_{TX}(f)|^2$ (see Chapter 4). The choice of transmit filter is made based on spectral constraints, as well as considerations such as sensitivity to receiver timing errors and intersymbol interference. Typically, the bandwidth employed is of the order of $\frac{1}{T}$.

Channel model: We typically model the channel as an linear time-invariant (LTI) system. For certain applications, such as wireless communications, the channel may be modeled as slowly time varying.

Noise model: Noise is introduced in a later lab.

Receive filter and sampler: The optimal choice of receive filter is actually a filter matched to the cascade of the transmit filter and the channel. In this case, there is no information loss in sampling the output of the receive filter at the symbol rate $\frac{1}{T}$. Often, however, we use a suboptimal choice of receive filter (e.g., a wideband filter flat over the signal band, or a filter matched to the transmit filter). In this case, it is typically advantageous to sample faster than the symbol rate. In general, we assume that the sampler operates at rate $\frac{m}{T}$, where m is a positive integer. The output of the sampler is then processed, typically using digital signal processing (DSP), to perform receiver functions such as synchronization, equalization and demodulation..

The simulation of a linearly modulated system typically involves the following steps.

Step 1: Generating random symbols to be sent

We restrict attention in this lab to Binary Phase Shift Keying (BPSK). Here, the symbols $\{b_n\}$ in Figure 1 take values ± 1.

Step 2: Implementing the transmit, channel, and receive filters

Since the bandwidth of these filters is of the order of $\frac{1}{T}$, they can be accurately implemented in DSP by using FIR filters operating on samples at a rate which is a suitable multiple of $\frac{1}{T}$. The default choice of sampling rate in the labs is $\frac{4}{T}$, unless specified otherwise. If the filter is specified in continuous time, typically, one simply samples the impulse response at rate $\frac{4}{T}$, taking a large enough filter length to capture most of the energy in the impulse response.

Step 3: Sending the symbols through the filters.

To send symbols at rate $\frac{1}{T}$ through filters implemented at rate $\frac{4}{T}$, it is necessary to zeropad (or up-sample) the symbols before convolving them with the filter impulse response determined in Step 2.

Step 4: Adding noise

Typically, we add white Gaussian noise (model to be specified in a later lab) at the input to

the receive filter.

Step 5: Processing at the receive filter output

If there is no intersymbol interference (ISI), the processing simply consists of sampling at rate $\frac{1}{T}$ to get decision statistics for the symbols of interest. For BPSK, you might simply take the sign of the decision statistic to make your bit decision.

If the ISI is significant, then channel equalization (discussed in later labs) is required prior to making symbol decisions.

Laboratory Assignment

0) Get the matlab code needed (lab1init) to get started from the course home page, and make sure you understand what it is doing. For your information, the transmit filter specified in the code is approximately a square root raised cosine pulse in the frequency domain.

1) Plot the impulse response of the transmit filter versus t/T.

2) Using the DFT, compute the magnitude of the transfer function of the transmit filter versus the normalized frequency fT. (See Example 2.5.4 in Chapter 2 for a discussion of how to use the DFT of the sampled response to estimate the Fourier transform of the original continuous time signal that the samples correspond to). From eyeballing the plot, estimate the normalized bandwidth (i.e., bandwidth as a multiple of $\frac{1}{T}$).

3) Plot the response at the output of the receive filter to a single symbol. Is the transmit filter (approximately) square root Nyquist at rate $1/T$?

4) Generate 100 random bits $\{a[n]\}$ taking values in $\{0, 1\}$, and map them to symbols $\{b[n]\}$ taking values in $\{-1, +1\}$, with 0 mapped to $+1$ and 1 to -1.

5) Send the 100 symbols $\{b[n]\}$ through the system. What is the length of the corresponding output of the transmit filter? What is the length of the corresponding output of the receive filter? Plot separately the input to the receive filter, and the output of the receive filter versus time, with one unit of time on the x-axis equal to the symbol time T.

6) Do the best job you can in recovering the transmitted bits $\{a[n]\}$ by directly sampling the **input** to the receive filter, and add lines in the matlab code for implementing your idea. That is, select a set of 100 samples, and estimate the 100 transmitted bits based on the sign of these samples. (What sampling delay and spacing would you use?). Estimate the probability of error (note: no noise has been added).

7) Do the best job you can in recovering the transmitted bits by directly sampling the **output** of the receive filter, and add lines in the matlab code for implementing your idea. That is, select a set of 100 samples, estimate the 100 transmitted bits based on the sign of these samples. (What sampling delay and spacing would you use?). Estimate the probability of error. Also estimate the probability of error if you chose an incorrect delay, offset from the correct delay by $T/2$.

8) Suppose that the receiver LO used for downconversion is ahead in frequency and phase relative to the incoming wave by $\Delta f = \frac{1}{25T}$ and a phase of $\pi/2$. Modify your complex baseband model to include the effects of the carrier phase and frequency offset. When you now sample at the "correct" delay as determined in 7), do a scatter plot of the complex-valued samples $\{y[n], n = 1, ..., 100\}$ that you obtain. Can you make correct decisions based on taking the sign of the real part of the samples, as in 7)?

9) Now consider a *differentially encoded* system in which we send $\{a[n], n = 1, ..., 99\}$, where

$a[n] \in \{0, 1\}$, by sending the following ± 1 bits: $b[1] = +1$, and for $n = 2, ..., 100$

$$b[n] = \begin{cases} b[n-1], & a[n] = 0, \\ -b[n-1], & a[n] = 1, \end{cases}$$

Devise estimates for the bits $\{a[n]\}$ from the samples $\{y[n]\}$ in 8), and estimate the probability of error.

Hint: What does $y[n]y^*[n-1]$ look like?

Lab Report: Your lab report should answer the preceding questions in order, and should document the reasoning you used and the difficulties you encountered. Comment on whether you get better error probability in 6) or 7), and why?