ESE 351 Spring 2023 Homework #6 – Matlab Supplement
Assigned Tuesday, March 28
Group assignment (background for Case Study 2)
Due Friday, April 7 – Submit on gradescope
See syllabus and canvas announcements for late submission policy

Complete the tasks below. Include comments in your m-file so that a published pdf of your script output is a self-contained description of your work. See ESE351ExampleScript.m and ESE351ExampleScript.pdf on canvas for a Matlab example with general expectations. Pay close attention to the generation of figures that illustrate your results clearly. When helpful, use subplot and/or plot multiple items in a single graph. Include titles on all figures or subplots and label all axes. Use the **plot**( ) command for plotting CT functions and **stem**( ) for DT functions when appropriate to illustrate your methods and results.

Background - Binary PAM Simulation

Pulse amplitude modulation (PAM) is a method commonly used for modulation in a digital communication system (Figure 1). In this assignment, you will develop a simulation tool for modeling binary PAM communication and analyzing its performance. In this method, a binary message is defined as a sequence of amplitudes, $x_n \in \{-1, +1\}, n$ an integer. Those amplitudes modulate a continuous-time pulse shape, $p(t)$, to create pulses at regular intervals, T_s (the symbol period), producing a continuous-time signal, $y(t)$, to be transmitted across a medium (communication channel). Typically, $y(t)$ would be "up-converted" to a higher frequency for transmission across a range of the available spectrum. Note that the available frequency range would impose a limitation on the bandwidth of the pulse shape (to be explored in a case study to follow this assignment). The signal, $y(t)$, consists of repeated copies of the pulse shape, each with amplitude x_n, expressed as

$$y(t) = \sum_{n=-\infty}^{\infty} x_n p(t - nT_s).$$

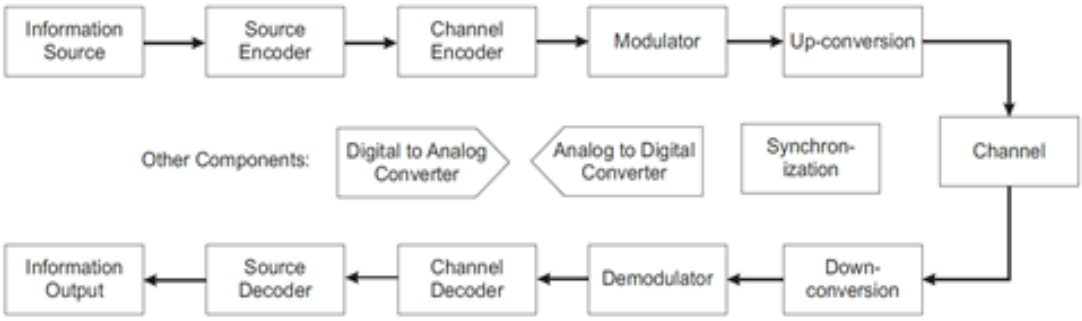

Figure 1: Block diagram of a single-user digital communication system, including (top) transmitter
[From Patwari, Neal, course notes, ESE 471 Communication Systems]

To recover the original message, the received signal, $r(t)$, (after down-conversion in the block diagram) is demodulated to estimate the original message x_n. A simple sign-based receiver could estimate x_n from the sign of the received signal at times corresponding to the peak of the pulse

shape, e.g., $\hat{x}_n = \begin{cases} +1, & r(t)|_{t=nT_s} > 0 \\ -1, & r(t)|_{t=nT_s} < 0 \end{cases}$. Any physical system is corrupted by noise, or random variations in the signal, though, causing errors in the received signal. A typical approach to representing that corruption is to model it as additive noise, $n(t)$, resulting in received signal $r(t) = y(t) + n(t)$. In this case, a better receiver than the simple sign-based one above uses a *matched filter* to match the pulse shape in the received signal, i.e., with $z(t) = r(t) * p(-t)$, let

$$\hat{x}_n = \begin{cases} +1, & z(t)|_{t=nT_s} > 0 \\ -1, & z(t)|_{t=nT_s} < 0 \end{cases}$$

Note that the matched filter uses a convolution to compute a moving, or sliding, correlation between the received signal and the pulse shape.

The relative impact of noise is often interpreted in terms of the signal-to-noise ratio (SNR), calculated as the ratio of the average signal power and the average noise power. For Gaussian noise, the average power is equal to the variance of the random variable (in case you haven't had probability theory yet, see more info in the tasks below).

Overall, this system needs to transmit a message as efficiently as possible, with performance measured in terms of the bit rate (rate of information transfer), the error rate (rate of incorrect estimates), the sensitivity to noise, and the bandwidth required for transmission.

Tasks For this assignment, complete the tasks below to develop a Binary PAM simulation tool and analyze its performance.

1. ***Simulation tool*** – write Matlab code to simulate a Binary PAM communication system with the following requirements:
    a. Use a triangular pulse, i.e., $p(t) = \begin{cases} 1 - \frac{|t|}{T_p}, & |t| < T_p \\ 0, & else \end{cases}$, ($T_p$ is half the pulse width).
        i. Note that this pulse shape is time-limited. Recall that the communication performance depends on the bandwidth of the pulse. Find the pulse's Fourier transform analytically and approximately in Matlab using **fft**(). Comment on the bandwidth required for communication with this shape.
    b. To simulate the system, you will need to approximate the continuous-time signals of the model (approximate as sampled, discrete-time signals). Use a sampling period $dt = \frac{T_p}{50}$. All timing parameters in this assignment are relative, so the value of T_p is arbitrary. For consistency, use $T_p = 0.1$ seconds.
    c. Generate a binary message, x_n, with *N* bits, i.e., $n \in [0,1, \dots, N-1]$. For testing, use any values, but for performance analysis, use a random message, e.g., *bits = 2*((rand(1,N)>0.5)-0.5)*; for values of ± 1 from a uniform random distribution.
    d. Generate transmitted signal, $y(t)$, as defined above. One method is to generate a vector with the necessary length, assign values of x_n at the appropriate locations, then convolve with the pulse shape $p(t)$. (Remember that *conv*() in Matlab

returns an output of different length than the input). Write your code such that the signal can be generated with a bit rate, $f_b = \frac{1}{T_s}$, of either $f_b = \frac{1}{T_p}$ or $f_b = \frac{1}{2T_p}$.

e. Generate the received signal $r(t)$ as the sum of the transmitted signal, $y(t)$, and a noise signal, $n(t)$. Model the noise as i.i.d. (independent, identically-distributed) Gaussian additive noise with zero-mean, variance σ^2. Use $randn()$ to generate samples of Gaussian noise (e.g., *sigma*randn(1,length(y))* will generate a vector the same length as y, with samples from a Gaussian, or normal, distribution with mean 0 and variance σ^2).

f. As an output of your simulation tool, plot the following, or similar:
   i. Pulse shape, $p(t)$, and spectrum, $P(\omega)$
   ii. Noise-free transmitted signal, $y(t)$
   iii. Noisy received signal, $r(t)$
   iv. Sent message, x_n, and decoded message, $\hat{x}_n$ for the receivers above
      1. Sign-based receiver
      2. Matched filter
   v. On your plots or elsewhere, report the bit rate, noise standard deviation σ, SNR, and error rate (for each receiver method). Note: you can calculate SNR here as $\frac{P_y}{P_n} = \frac{\int y^2(t)dt}{\int n^2(t)dt}$, with summations over the signal duration to approximate the integrals.

g. For your report, demonstrate operation of your tool for both bit rates above, with noise parameter $\sigma = 1$. For illustration, use a value of N that allows visualization of the signals and bit values.

2. ***Performance analysis***: use your simulation to analyze the error rate vs. bit rate, noise level σ, and receiver method, then plot error vs. SNR over a range of noise variance (e.g., $\sigma \in [0,1]$), and repeat for each bit rate and each receiver method. (It's fine to generate these results and plots offline, then have your m-file import the data or figure into your published pdf).