Lab #4 – Bit Error Rate Simulation

In this lab you will empirically simulate the Bit Error Rate (BER) of BPSK in the presence of Additive White Gaussian Noise (AWGN). You will also implement a simple correlation receiver, and compare its performance against theoretical limits.

Section 1: BER Simulation – Baseband Constellation Approach

The theoretical BER formula for BPSK, as derived in class, views BPSK as two points on a one-dimensional vector (also known as the signal constellation). In the first section of this lab, you will perform empirical simulations of the BER and compare the results against theory.

First, predict the BER of BPSK for a range of E_b/N_o from 1dB to 8dB. You will either need to consult a table of the complementary error function, or use the MATLAB function ***erfc***.

Next, perform an empirical simulation of BPSK BER, viewing each symbol as a point in the constellation diagram:

Use ***randsrc*** to generate a pseudorandom sequence of symbols or constellation points. Let each symbol take on a value of either +1 and -1. *What is the average energy per bit (E_b) in this case*? (See *Appendix 1*)

Use ***randn*** to generate Gaussian noise; note that the noise variance will depend on E_b/N_o. Add this noise to the constellation points. (See *Appendix 2*)

Assuming the symbols are equiprobable, the optimal detection threshold at the receiver is 0. Use this to make a decision whether the transmitted symbol is +1 or -1.

Compare the original symbols to the detected symbols and calculate the bit error rate

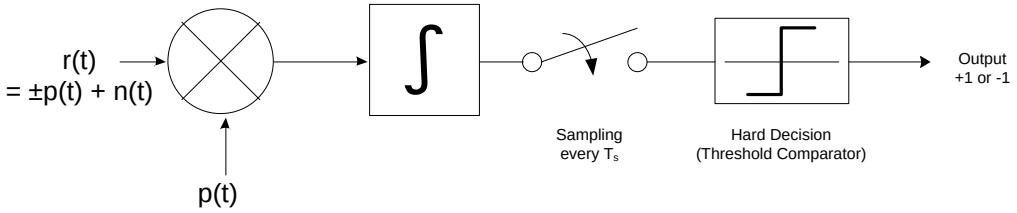
Hint: Use at least a few thousand symbols to ensure the result is statistically valid. You may also want to start with a single value of E_b/N_o to validate your simulation setup is correct.

Plot the theoretical BER curve against the empirical curve. In addition, compare the predicted vs. the measured BER in a table.

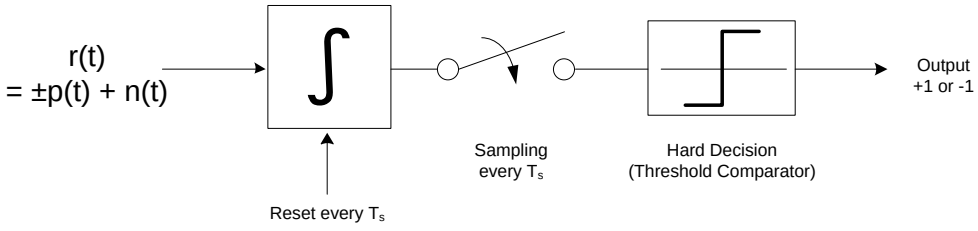
Section 2: Correlation Receiver

The constellation diagram provides a very powerful conceptual framework for BER analysis, and forms the cornerstone for BER analysis in this course. Implicitly assumed in the analysis is that we are taking one sample per symbol, and based on this sample making a *hard decision* in the receiver to determine the actual transmitted symbol.

As discussed in lecture, a matched filter receiver is used to maximize the SNR at the sampling instant. An equivalent form of the matched filter is the correlation receiver. Note that this structure involves integrating the received signal over a period of time prior to sampling:

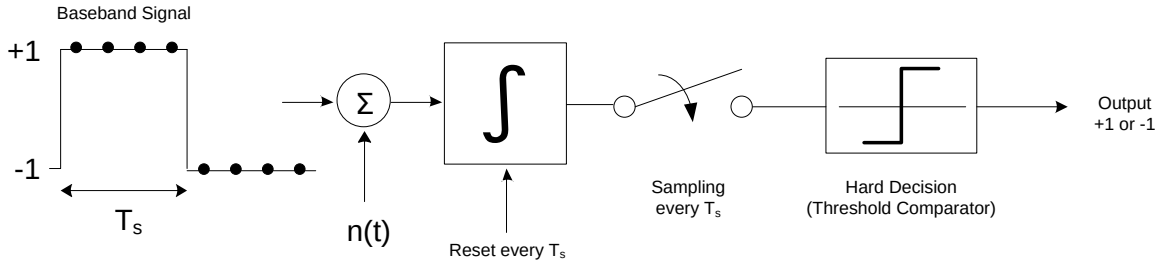


If p(t) is a rectangular pulse, then the receiver simplifies further into an "integrate and dump" structure. Specifically, if p(t) has a magnitude of 1, then the multiplier can be discarded. The received signal is directly integrated during each symbol period T_s, and at the end of each period the integrator output is sampled, a decision is made, and the integrator is reset ("dump" is in reference to an integrator circuit in which the capacitor charge is dumped at the end of every period).



Copy your MATLAB code from section 1 into a new file, and use this file as the starting point for section 2. In this section, we will apply rectangular pulse shaping to the baseband symbols. The baseband signal is then subjected to AWGN, and empirical BER simulation will be performed using an integrate-and-dump receiver.

Use ***randsrc*** to generate the baseband symbols of +1 and -1 for BPSK. Upsample by a factor 4 using ***rectpulse***. Note that we now have 4 samples per symbol.



In this receiver, the signal is integrated over each symbol period. At the end of each symbol period, the sum is then passed into the threshold block to decide whether the received symbol is +1 or -1. For the next symbol period, and result from the previous symbol period is dumped by resetting the integrator back to zero.

Use ***randn*** to generate and add Gaussian noise to the baseband signal, using the same noise variance as Part 1. Implement the integrate-and-dump receiver in Matlab. Assume that the optimal detection threshold is still 0. Use this to make a decision whether the transmitter symbol is +1 or -1. Compare the original symbols to the detected symbols and calculate the bit error rate.

Create a table of the predicted vs. the measured BER for a range of E_b/N_o **from 1dB to 3dB**. How do the results compare to those of part 1? Explain why the results are better or the same as before (hopefully not worse).

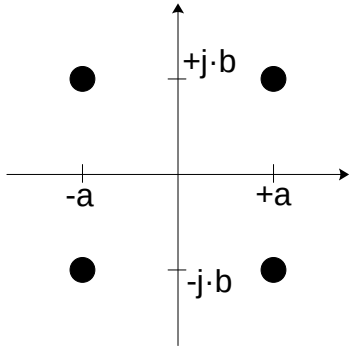
Hand in a lab report by no later than the beginning of the next lab. Include a hard copy of your MATLAB code with your report, and submit a soft copy to ShareIn.

Appendix

1. Calculating the Energy of a Symbol

From the lecture on constellation scaling, we noted that the energy of each point in the constellation is given by the magnitude squared. For example, the energy of the point in the first quadrant is given by:

$E = a^2 + b^2$



For the BPSK constellation shown on the first page, since it is one-dimensional, the energy of one point is given by:

$E = a^2$

Implicitly assumed in the calculations above, and all the BER analysis done in lecture, is that we have one sample per symbol (which by the way is the assumption for part 1 of this lab).

As we delve into the use of a correlation receiver, we are now integrating the received signal over a period of time. As noted in this lab, this means having multiple samples for each symbol. For the BPSK case, if each sample still has a magnitude of "a", the symbol energy is now calculated by using:

$$E = \sum_{i=1}^{Nsamp} |a|^2$$

N_{samp} = number of samples per symbol
a = samples of the baseband symbol

2. Setting the Noise Variance in MATLAB

Since we are assuming an AWGN channel, the noise follows a Gaussian distribution with variance equal to:

$$\sigma^2 = \frac{N_o}{2}$$

Rearranging this equation, we have $N_o = 2\sigma^2$. Hence, we can write the ratio E_b/N_o (scalar number, not in dB) as follows:

$$\frac{E_b}{N_o} = \frac{E_b}{2\sigma^2}$$

Suppose that we wish to set $E_b/N_o = K$. The noise variance required is thus:

$$\sigma^2 = \frac{E_b}{2K}$$

In calculating this variance, you have to know the energy per bit E_b (which you are asked to determine in part 1 of this lab), and you need to find K by converting your desired E_b/N_o from dB to linear units.