ECEN310 Communication Engineering (XMUT) Lab 4 (worth 8%)

Digital Modulation Techniques: M-PSK Error Rate Performance

In this Lab, we will study the Symbol Error Rate (SER) performance of phase shift keying (PSK) systems. In the first part we will confirm the theoretical results derived in class obtained with ideal assumptions of zero timing offset, perfect coherent demodulation (no phase and frequency offset).

The SER plots will be obtained by means of a Monte Carlo simulation. Essentially, we will generate a sequence of data, simulate a transmission over a (in our case AWGN) channel, and count the number of errors at the receiver. This is a common way of evaluating error performance, especially in complex systems where simple analysis is not possible.

We will start with an M-PSK system operating in an AWGN channel. For this we will be assuming a Nyquist pulse shaping with perfect timing synchronisation as well as coherent demodulations (no phase nor frequency offset).

In this case, we simply have the complex received signal samples given by

$$r_k = s_k + n_k \qquad (1)$$

where s_k are the transmitted symbols (±1 for BPSK, $e^{j\pi(2m+1)/M}$, m = 0,1,…, M-1 for M-PSK), n_k is the complex AWGN noise with zero mean and variance $\sigma_n^2 = N_0/2$ (in other words, $n_k = N(0, N_0/2)$). We will define the Signal-to-Noise Ratio (SNR) as

$$SNR = E_s/N_0 \qquad (2)$$

where E_s is the average symbol energy. Typically when simulating a digital system we set $E_s = 1$ and vary the SNR by adjusting N_0.

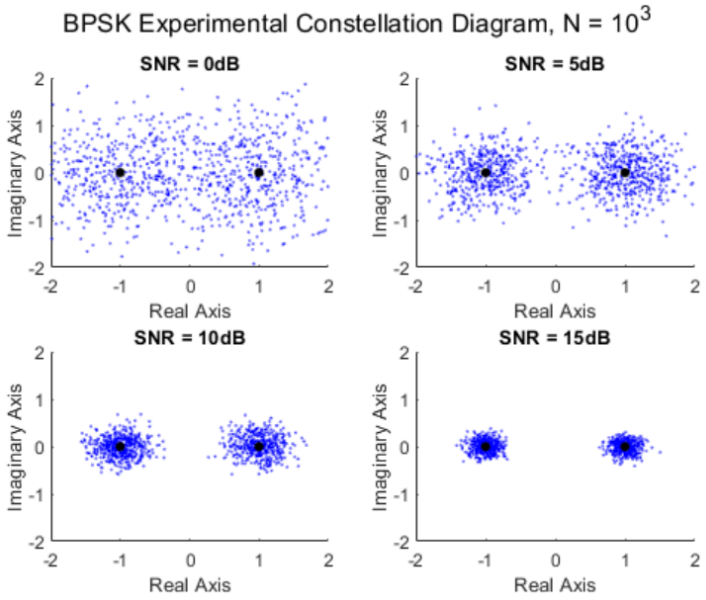
Part A: Transmitter & AWGN

(a) Generate N_s = 1e3 BPSK data symbols s_k -- the simplest way to generate a sequence of symbols drawn randomly from a constellation is to use the *randsrc* command. To do this, define an alphabet, which in the case of BPSK is simply a vector [1 −1]. Corrupt them by AWGN. To do this, generate a vector of N_s Gaussian distributed complex noise samples using the *randn* command. Note that *randn* will return a zero mean, unit variance sequence, so you must scale it to achieve a variance of $N_0/2$.

(b) Plot the transmitted signal constellation and the received symbols for SNR values of 0, 5, 10 and 15 dB (in decibels). Put all four plots on a single Matlab figure (using subplot). Mark the transmitted and received symbols with different markers. Observe the effects of SNR on the received constellation.

(c) Repeat the above for QPSK, 8-PSK and 16-PSK modulations (M=4, M=8 and M=16). Your alphabet will now be a set of M equally spaced points on a unit circle in a complex plane.

For example, with BPSK, you are expected to see/produce this kind of subplots:



Generate the above subplots for QPSK, 8-PSK and 16-PSK modulations (M=4, M=8 and M=16) too.

Part B: Receiver

(d) Let's move on to the receiver, starting with BPSK. Implement a decision device to estimate the received symbols, which for BPSK is simply a matter of comparing it to the threshold of zero. Generate the estimates $\hat{s}_k$ of the data symbols s_k, and count the number of symbol errors. One way to do this is to use the *nnz* command on a difference vector $\hat{s}_k - s_k$. The SER will simply be the number of errors divided by N_s.

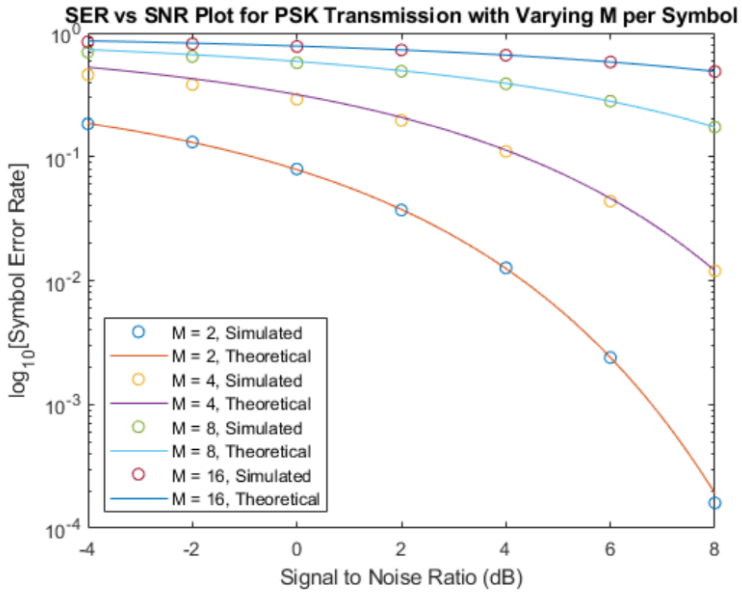
(e) Generate SER for QPSK, 8-PSK and 16-PSK modulations. You will have to modify the decision device, as you are no longer comparing to a threshold but deciding depending on which region the received data point fall in. The easiest way to do this is to find the constellation point closest to the received symbol. To achieve this, use *min* command to find the minimum distance and the associated index of the constellation vector: *[d,i]=min(abs(r(n)-alphabet))* is the estimate. Then the $\hat{s}_k$ will be the *i*th point in the constellation.

(f) Now generate SER curves, that is $\log_{10}$(SER) versus SNR_{dB}. It might be wise to put the code generated so far in a function *getSER(M,Ns,SNRdB)*, that is one which takes in the MPSK order M, number of data symbols N_s and the SNR in dB, and returns the SER. Run the simulation for N_s = 1e4 symbols and SNR

of -4 to 8 (in steps of 2 to speed up the simulation). Plot the SERs (BPSK, QPSK, 8-PSK and 16-PSK) on a logarithmic scale versus the SNR in dB. Use the *semiology* command.

(g) Compare the above with the theoretical results derived in class - plot the theoretical curves using lines and simulated using symbols all on the same figure. Be sure to label it nicely!

For Part B, you are expected to see/produce this kind of plots:



Submit an individual report containing your Matlab codes, plots and brief explanations of your results. Remember to annotate your Matlab codes with appropriate comments to help the reader understand the codes. The report does not need to be formal, but please write it with good English.