WPI

ECE3311 – Principles of Communication Systems

Project 04

Due: 5:00 PM, Tuesday 5 December 2017

1 Project Objective & Learning Outcomes

The objective of this project is to have you master digital modulation schemes employed in passband communication systems and operating environments. From this project, it is expected that the following learning outcomes are achieved:

- Understand one-dimensional digital modulation schemes such as Amplitude Shift Keying (ASK).
- Obtain an understanding of two-dimensional digital modulation schemes such as Phase Shift Keying (PSK) and Quadrature Amplitude Modulation (QAM).
- Learn about modulation detection using Error Vector Magnitude (EVM) analysis.

2 BER Analysis of 4-ASK, 4-OPSK, and 4-QAM Simulated Transmissions

Unlike the other projects in the course, for this project you will be provided the end-to-end MATLAB source code of a complete communication system. In this case, you are provided with an ASK communication system implementation. For this project, you will be using the following simulation parameters when designing and evaluating your communication systems.

```
% Define simulation parameters
N_bits = 10000; % Total number of bits for entire transmission
f_c = 2; % Carrier frequency
f_s = 10; % Sampling frequency
f_d = 1; % Digital system sampling frequency
beta = 0.5; % Roll-off factor for raised cosine filter
sigma_n = 0.25; % Noise standard deviation
M = 4; % number of codewords, 2^n for n=2 bit codewords
```

As with any communication system, there are three stages: Transmitter, Channel, and Receiver. The sample MATLAB code provided has been defined in much the same way. In the MATLAB code provided for Step 1, transmitter implementations of 4-ASK are defined below. Notice how raised cosine pulse shaping is employed in this example.

The next stage is the channel (Step 2), which we have assumed to be an additive white Gaussian noise (AWGN) channel. Note that we have to modulate the noise to the carrier frequency, and we modulate using both cosine and sine functions. Since we are using cosine and sine modulation, we have to divide the noise energy in half, hence the division of each noise term by the square root of 2.

```
% STEP 1: Modulate 4-ASK waveforms

% Generate random binary data stream
bin_data = round(rand(1,N_bits));

% Assignment of binary sets to amplitude values
bin = reshape(bin_data,2,N_bits/2);
ampl_ask = zeros(1,N_bits/2);
ampl_ask(find( (bin(1,:) == 0)&(bin(2,:) == 0) )) = -3;
ampl_ask(find( (bin(1,:) == 1)&(bin(2,:) == 0) )) = -1;
ampl_ask(find( (bin(1,:) == 0)&(bin(2,:) == 1) )) = 1;
ampl_ask(find( (bin(1,:) == 1)&(bin(2,:) == 1) )) = 3;

% Apply raised cosine pulse shaping filter
rolloff = 0.35;     % Rolloff factor
span = 6;           % Filter span in symbols
sps = 8;            % Samples per symbol
rcos = rcosdesign(rolloff, span, sps);
ampl_ask_rcos = upfirdn(ampl_ask, rcos, sps);

% 4-ASK modulate to carrier frequency f_c
tx_ask_wavefm = ampl_ask_rcos.*...
        cos(2.*pi.*f_c.*(0:(1/f_s):((length(ampl_ask_rcos)-1)*(1/f_s))));
```

```
% STEP 2: Introduce Passband AWGN to transmission (both I and Q components)
noise_ask = (sigma_n/sqrt(2)).*randn(1,length(tx_ask_wavefm)).*...
    cos(2.*pi.*f_c.*(0:(1/f_s):((length(ampl_ask_rcos)-1)*(1/f_s))))+...
    (sigma_n/sqrt(2)).*randn(1,length(tx_ask_wavefm)).*...
    sin(2.*pi.*f_c.*(0:(1/f_s):((length(ampl_ask_rcos)-1)*(1/f_s))));

% Waveforms to detect:
rx_ask_wavefm = 2*(tx_ask_wavefm + noise_ask);
```

At the receiver stage (Step 3), we use coherent detection assuming that we know the carrier frequency at this end of the communication system. Using both cosine and sine functions to demodulate the signal down to baseband and applying a lowpass filter to remove its double frequency term, the key is to find the appropriate time instances to sample the filtered waveforms. Note that since both the raised cosine filters and the lowpass filters introduce something called group delay, we need to start our sampling after we have taking into consideration the delay introduced by these finite impulse response (FIR) filters. Even though the ASK waveforms transmitted did not have a quadrature component, we still use the sine functions in the demodulation process.

Finally, to show how these modulation schemes work, we plot the outputs of the sampling process on something called a scatter plot using MATLAB code provided in Step 4. Examples of the 4-ASK, 4-OPSK, and 4-QAM scatter plots are shown in Figure 2. The scatter plots provide us with imformation about the demodulated inphase and quandrature amplitude information. If things work out with the communication system design, the scatter plots should be concentrated around defined I/Q locations. On the other hand, if there is a problem, our scatter plots will not be neatly confined to these locations. Since there will always be noise present in our system, you will notice that the

scatter plots produce a small cloud of points rather than an exact point. This is due to how the noise displaces the I/Q amplitude values.

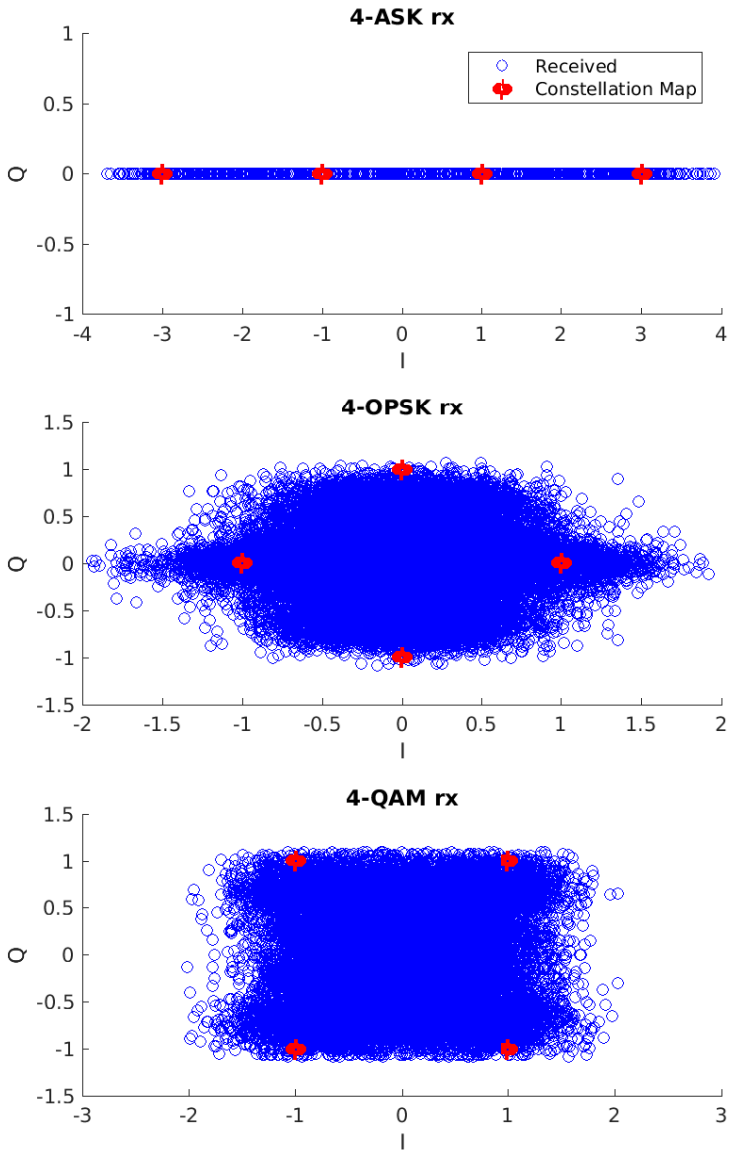

Figure 1: The initial received waveforms for 4-ASK, 4-OPSK, and 4-QAM. The receive chain begins with these waveforms.

```
% STEP 3: Demodulate 4-ASK waveform using coherent detection

% Remove the carrier by multiplying by cos()+j*sin()
no_carrier_ask = rx_ask_wavefm.*cos(2.*pi.*f_c.*(0:(1/f_s):...
            ((length(ampl_ask_rcos)-1)*(1/f_s)))) + ...
            1i*rx_ask_wavefm.*sin(2.*pi.*f_c.*(0:(1/f_s):...
            ((length(ampl_ask_rcos)-1)*(1/f_s))));

% Remove double frequency terms via LPF
filt_coeffs = fir1(61,0.25);
lpfed_ask = conv(filt_coeffs,no_carrier_ask);

% truncate convolution tails
lpfed_ask = lpfed_ask(floor(length(filt_coeffs)/2):1:end-...
    floor(length(filt_coeffs)/2));

% matched filter to remove pulse shaping waveform
downsampled_ask = upfirdn(lpfed_ask, rcos, 1, sps);

% truncate tails from convolution in upfirdn
downsampled_ask = downsampled_ask(7:1:end-6);

%%%%%%%% INSERT EVM ANALYSIS HERE %%%%%%%%

% demodulate
demod_ask = pamdemod(downsampled_ask, M);

% back to binary from decimal
% 4-ask
ask_results = zeros(2,length(demod_ask));
ask_results(1, find( demod_ask == 0 )) = 0;
ask_results(2, find( demod_ask == 0 )) = 0;
ask_results(1, find( demod_ask == 1 )) = 1;
ask_results(2, find( demod_ask == 1 )) = 0;
ask_results(1, find( demod_ask == 2 )) = 0;
ask_results(2, find( demod_ask == 2 )) = 1;
ask_results(1, find( demod_ask == 3 )) = 1;
ask_results(2, find( demod_ask == 3 )) = 1;
correct_bits_ask = sum(sum(ask_results == bin));
BER_ask = (N_bits - correct_bits_ask)/N_bits
```

```
% STEP 4: Generater scatter plots of outputs
% downsampled vs constellation
figure(2)
subplot(3,1,1)
hold on; title('4-ASK downsampled'); xlabel('I'); ylabel('Q');
scatter(real(downsampled_ask), imag(downsampled_ask),'b');
scatter(real(ampl_ask), imag(ampl_ask),'r', 'LineWidth', 5);
legend('pre-demodulation','Constellation Map');
hold off;
subplot(3,1,2)
hold on; title('4-OPSK downsampled'); xlabel('I'); ylabel('Q');
scatter(real(downsampled_psk), imag(downsampled_psk),'b');
scatter(real(ampl_psk), imag(ampl_psk),'r', 'LineWidth', 5);
hold off;
subplot(3,1,3)
hold on; title('4-QAM downsampled'); xlabel('I'); ylabel('Q');
scatter(real(downsampled_qam), imag(downsampled_qam),'b');
scatter(real(ampl_qam), imag(ampl_qam),'r', 'LineWidth', 5);
hold off;
```

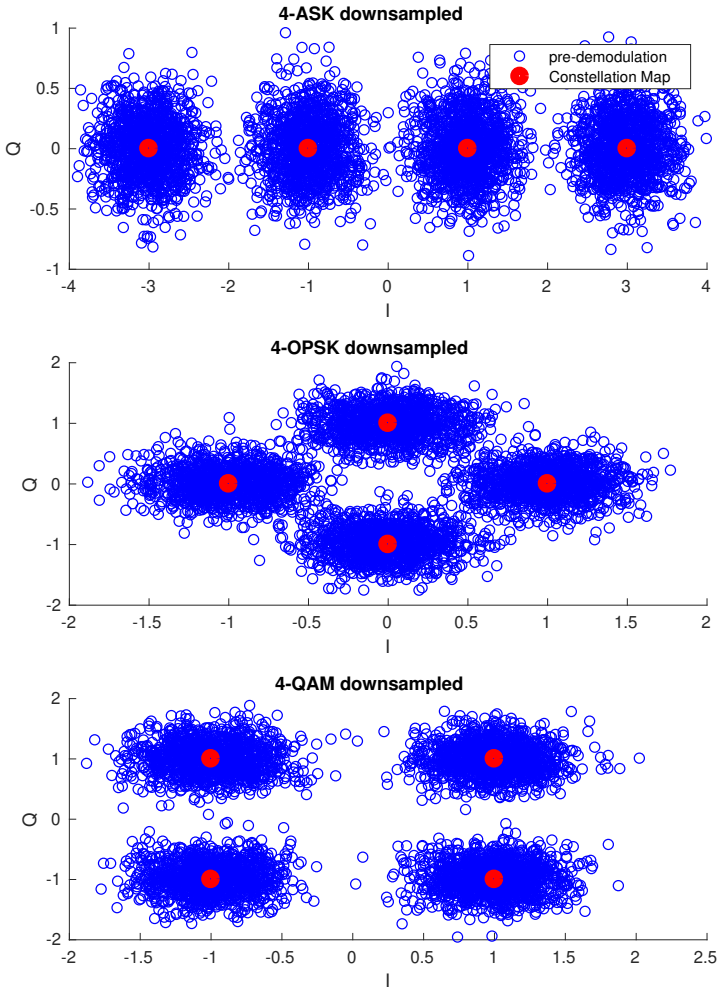

Figure 2: The result of removing the carrier from, lowpass filtering, truncating, matched filtering, truncating, and normalizing Figure 1. To obtain the decimal and then binary values in the data, next they are demodulated using (from top to bottom) *pamdemod*(), *pskdemod*(), and *qamdemod*() as shown for 4-ASK in the provided code snippets.

ACTION (30 Points): Using the code snippets provided, also create 4-OPSK and 4-QAM transmissions, decoding them and showing a near-zero BER. See page 8 for a code snippet showing the required bit-to-decimal mappings to begin. Follow the same process as shown in Step 1 for the two additional modulation schemes.
Additionally, obtain a plot similar to Figure 2.

```
% Binary to Amplitude mappings to begin with when implementing
% 4-OPSK and 4-QAM transmissions. They correspond to MATLAB's
% demodulate funcitons pskdemod() and qamdemod() so do not
% change them or your BER will be very large!

% 4-opsk
ampl_psk = zeros(1,N_bits/2);
ampl_psk(find( (bin(1,:) == 0)&(bin(2,:) == 0) )) = complex(1,0);
ampl_psk(find( (bin(1,:) == 1)&(bin(2,:) == 0) )) = complex(0,1);
ampl_psk(find( (bin(1,:) == 0)&(bin(2,:) == 1) )) = complex(-1,0);
ampl_psk(find( (bin(1,:) == 1)&(bin(2,:) == 1) )) = complex(0,-1);
% 4-qam
ampl_qam = zeros(1,N_bits/2);
ampl_qam(find( (bin(1,:) == 0)&(bin(2,:) == 0) )) = complex(-1,1);
ampl_qam(find( (bin(1,:) == 1)&(bin(2,:) == 0) )) = complex(-1,-1);
ampl_qam(find( (bin(1,:) == 0)&(bin(2,:) == 1) )) = complex(1,1);
ampl_qam(find( (bin(1,:) == 1)&(bin(2,:) == 1) )) = complex(1,-1);
```

3 Error Vector Magnitude (EVM) Modulation Detector

A common practice in communications where transmissions are received without prior knowledge of the transmitted waveform's contents or even structure, is to firstly decide the modulation scheme. Immediately prior to demodulation, the scheme used must be decided. The classical method of determining this is compare each data point to a reference constellation, pictured in Figures 2 and 1 as the red dots. The reference constellation with the lowest Root Mean Square (RMS) euclidean distance to these reference points is chosen as the most likely modulation scheme.

ACTION (20 Points): Using the code snippets provided, inserting EVM analysis where indicated, employ an evm object $comm.EVM()$ in MATLAB and use it to calculate the RMS EVM for the $downsampled_ask$ object with reference to the 4-ASK constellation reference. Show that the RMS EVM of the 4-OPSK and 4-QAM downsampled waveforms are higher, thus indicating the waveform is 4-ASK. Repeat this for a 4-OPSK and 4-QAM constellation reference.

4 Automatic Demodulator and PLL

ACTION (20 Points): Import the file $rx\_mystery\_wavefm.mat$, which is either a 4-ASK, 4-OPSK, or 4-QAM waveform, identical in its transmit-receive chain to the 4-ASK given in code snippets (just like rx_ask_wavefm at the end of Step 2), except for one key difference – a frequency offset due to LO differences. Adjust your receive chain to also make use of a discrete-time PLL to remove this offest, write code to use the EVM analysis from Section 3 to automatically decide its modulation scheme, and utilize your code from Section 2 to automatically demodulate and perform BER analysis on the waveform.

Frequency offset is applied in the from $freq\_offset\_tx\_wavefm = tx\_wavefm. * exp(-1i * 2 * pi * fo/f\_s * [1:1:length(tx\_wavefm)])$. Credit will not be given for removing the frequency offset without a PLL.

5 Final Report Format & Content

Each experiment report should possess the following format:

- A cover page (2 points) that includes the course number, project number, names and WPI ID numbers, submission date.
- A narrative of the process (10 points) taken during this experiment and the experiences encountered by the student. Figures, plots, schematics, diagrams, snippets of source code, and other visuals are highly encouraged.
- Responses to all questions indicated in the project handout. Please make sure that the responses are of sufficient detail.
- A summary (5 points) that contains all lessons learned from this project.
- All source code (5 points) generated (as an appendix).

This single document must be electronically submitted in **PDF format** (no other formats will be accepted) via the ECE3311 CANVAS website by the due date. Failure to submit this report by the specified due date and time will result in a grade of "0%".