WPI

ECE3311 – Principles of Communication Systems

Project 01

Due: 5:00 PM, Tuesday 6 November 2017

1 Project Objective & Learning Outcomes

The objective of this project is to introduce computer simulation tools and techniques for the modeling and analysis of baseband transmission waveforms.

From this project, it is expected that the following learning outcomes are achieved:

- Obtain an understanding of how continuous analog waveforms can be sampled and transformed into discrete and digital signals.
- Master the conversion of analog waveforms into binary strings via the pulse coded modulation (PCM) process.
- Establish competency in constructing various line codes and experiment with them in a computer simulation environment.
- Gain proficiency in using the eye diagram tool as a form of understanding the temporal behavior of pulse shapes used in a transmission.
- Observe and demonstrate the impact of the digitization and recreation of analog waveforms via the conversion of an audio file to a digital format and then converted back to an audio file.

2 Preparations

The first step in this project is to understand some of the basics with respect to representing analog waveforms in a computer simulation environment. Since everything in a computer simulation environment is digital, we need to model digital signals to "appear" as close to analog as possible. One way of achieving this is by representing a specific analog waveform using a substantial amount of discrete samples. In the following MATLAB code, we do exactly this by modeling the signal `analog_wavefm` as a sequence of numerous discrete samples. We also do this with the train of rectangular pulses `impulsetrain_wavefm`.

```
% Generate a discrete version of a random continuous analog
% waveform using a Uniform Random Number Generator and
% an interpolation function to smooth out the result
analog_wavefm = interp((2*rand(1,(L/M))-1),M);

% Generate a rectangular pulse train of samples
impulsetrain_wavefm = reshape(ones(N,1)*rem(1:1:(L/N),2),[1,L]);
```

These two waveforms are going to be very useful for us throughout the rest of this project, especially when we want to model analog waveforms and conduct experiments on them that represent some of the digital operations discussed in the course. For instance, if we look at Figure 1, we can observe some of the "analog" waveforms created by the MATLAB code above. Let us see how this is used in the next several sections.

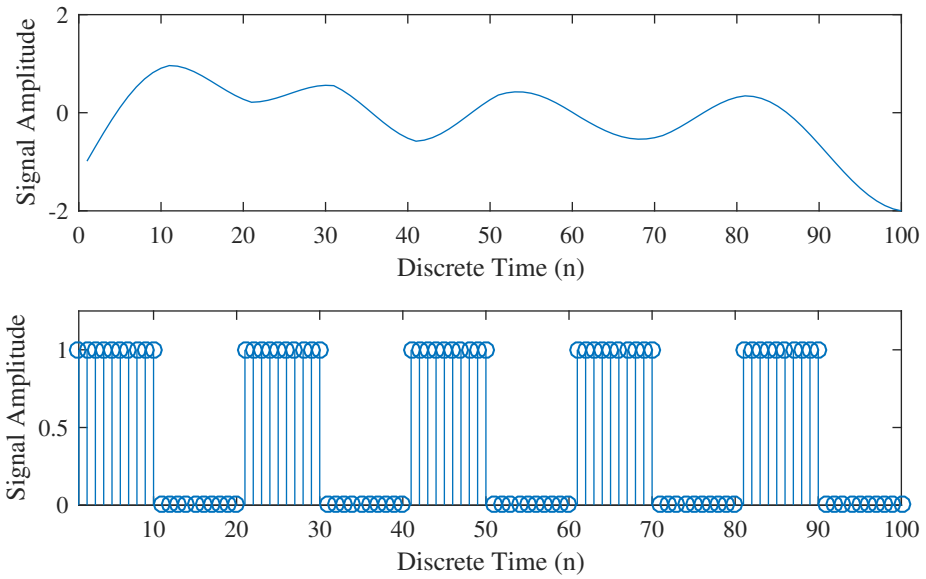


Figure 1: Examples of a randomly generated analog waveform (top) and a periodic stream of rectangular pulses (bottom).

Before continuing forward, note that throughout this project you should be using the following simulation parameters.

```
L = 100;   % Length of the overall transmission
N = 10;    % Pulse duration for rectangular pulse train
M = 10;    % Upsampling factor for generating analog waveform
L_lc = 20;% Line coding pulse duration
```

Note that MATLAB is heavily optimized for matrix operations, and many of the built-in functions are compiled code, which means using these features will signficantly speed up your computer simulations. If you use something like a `for` loop or similar, you will begin to lose the benifits of those accelerated features. Consequently, when in doubt always vectorize.

3 Pulse Amplitude Modulation

The first computer experiment that we will be performing is the generation of pulse amplitude modulation (PAM) waveforms. There are two types of PAM waveforms: naturally sampled PAM and flat-top PAM.

ACTION (5 Points): Using the MATLAB code presented in Section 2, please obtain the naturally sampled PAM waveform and flat-top sampling PAM waveform of a random analog baseband signal. If done correctly, your two stem plots should appear to be similar to those shown in Figure 2

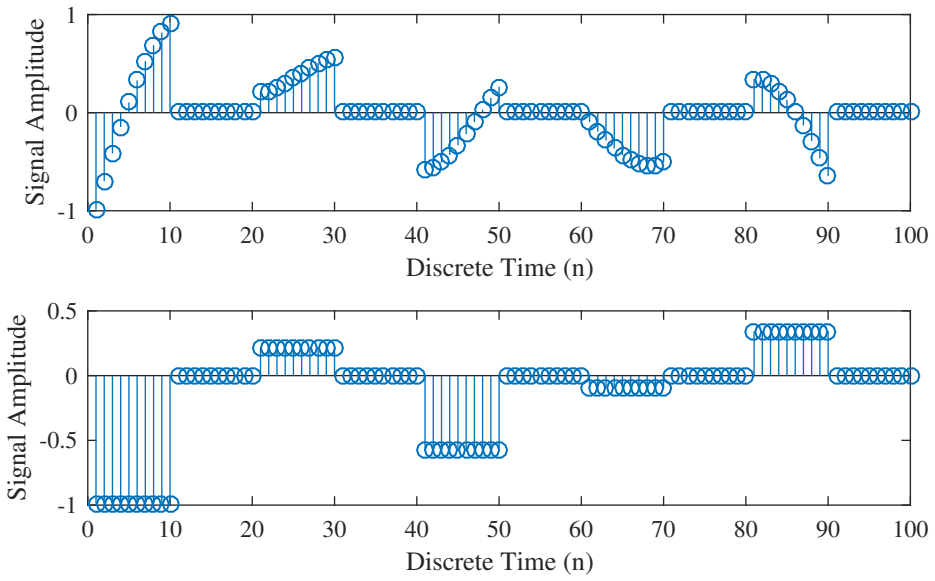


Figure 2: Examples of a naturally sampled PAM waveform (top) and a flat-top sampling PAM waveform (bottom).

4 Pulse Coded Modulation

Now that you are comfortable with manipulating analog waveforms in the MATLAB simulation environment, let us take another analog waveform and convert it into a pulse coded modulation (PCM) waveform. To achieve this, we are going to need to perform an operation known as *quantization*, where we round samples of a waveform to the nearest value defined by a codebook that has a binary codeword representation for that value. In MATLAB, there is a useful function for quantization called `quantiz`. For this section, you will be using the following MATLAB code for quantizing your analog waveform.

```
[ind,quantv] = quantiz(downsample(analog_wavefm,N),...
[-0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8],...
[-0.9 -0.7 -0.5 -0.3 -0.1 0.1 0.3 0.5 0.7 0.9]);
```

ACTION (10 Points): Using this quantizer, please plot the original analog waveform, the PCM version of it, and the associated quantization error such that you produce results that appear similar to Figure 3. Since the purpose of the PCM waveform is to ultimately convert an analog signal into a string of binary values, define the binary values associated with each PCM amplitude value. Please include this mapping in your project report. Print out the binary outputs for the PCM words generated by your computer experiment.

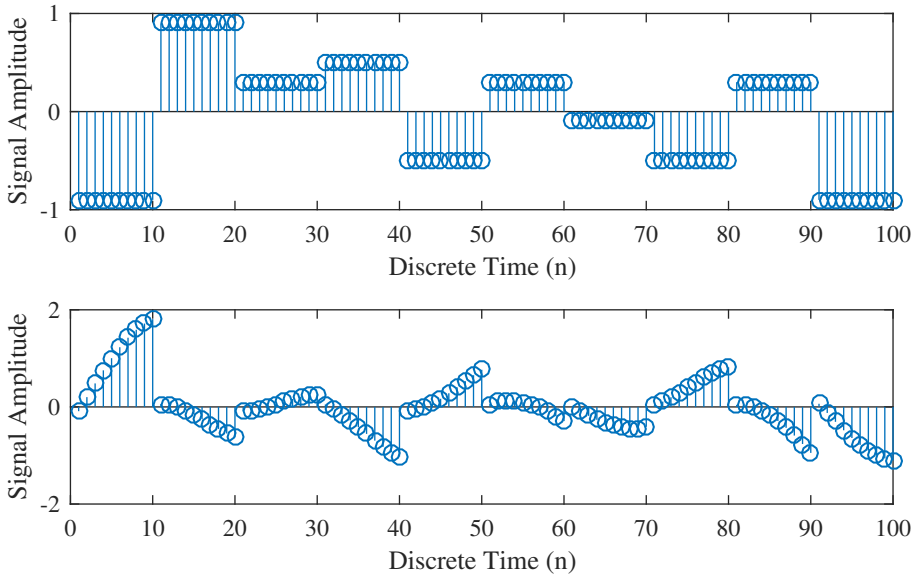


Figure 3: Examples of a quantized analog waveform (top) and the residual quantization error after converting it into a PCM waveform (bottom).

5 Line Coding

Line codes exist everywhere, especially in legacy systems that possess some sort of baseband digital information exchange, *e.g.*, automotive CAN Bus messages. The purpose of this section is to introduce you to these codes and enable you to generate them in a computer simulation environment. As a starting point, the following MATLAB code will generate unipolar non-return to zero (NRZ) waveforms for a random binary sequence.

```
% Generate your own line codes for the binary string `11001100'
bin_str = [1 1 0 1 0 0 1 0];
upnrz1 = ones(1,L_lc);
upnrz0 = zeros(1,L_lc);
upnrz_wavefm = [];
for ind = 1:1:length(bin_str),
    if (bin_str(ind) == 1)
        upnrz_wavefm = [upnrz_wavefm upnrz1];
    else
        upnrz_wavefm = [upnrz_wavefm upnrz0];
    end;
end;
```

As you can see, although a `for` loop was used in the generate of the string of unipolar NRZ pulses, the pulse shapes themselves were vectorized using the convenient `ones` and `zeros` functions.

ACTION (5 Points): Using this code, as well as other similar implementations, examples of line codes using unipolar NRZ, polar NRZ, unipolar RZ, bipolar RZ, and Manchester NRZ pulse shapes we generated in Figure 4. For this section, please, derived these same examples yourself using a randomly generated binary string (do not use the binary string used in this example).

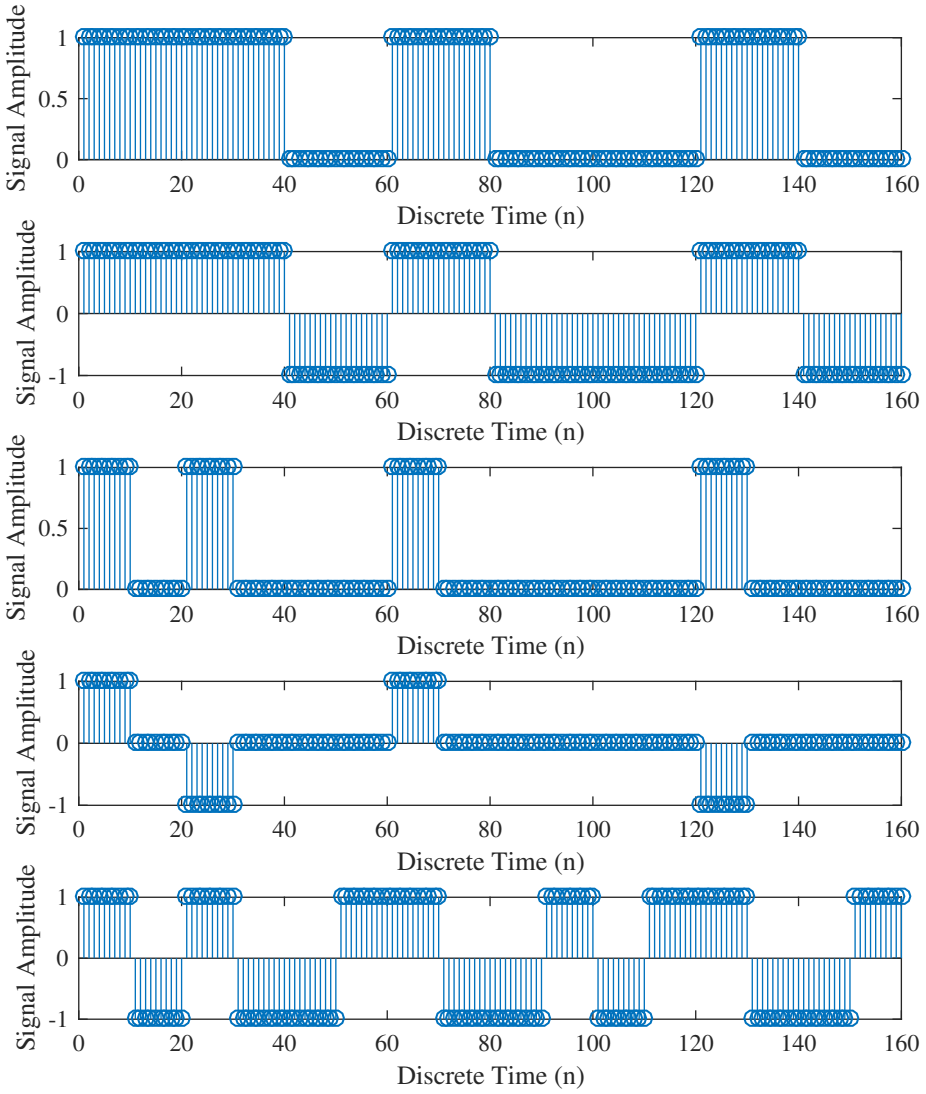


Figure 4: Examples of line coding techniques (from top to bottom): Unipolar NRZ, Polar NRZ, Unipolar RZ, Bipolar RZ, Manchester NRZ.

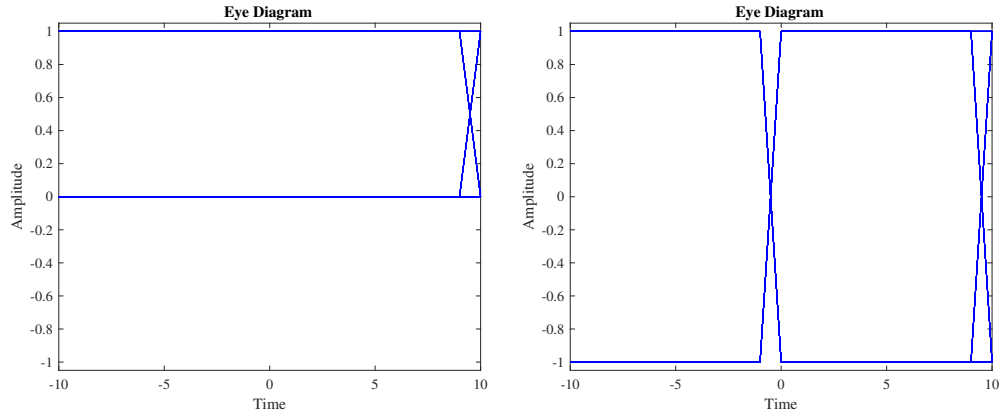
6 Eye Diagrams

A very powerful tool when analyzing a transmission for various forms of distortion is the eye diagram. The eye diagram can tell use how much noise and interference is present within a transmission, as well as if our reception of the transmission is out-of-sync, *e.g.*, timing issues. In this section, we will learn how to use the MATLAB `eyediagram` tool, which is shown in the following example for analyzing a unipolar NRZ waveform.

As you can see, the `eyediagram` tool traces both pulses generated from transmitting ones and zeros into a single plot, which is shown in Figure 5(a). Thus, we are overlapping the pulses every pulse period in order to observe any deviations in the pulse shape characteristics. Also included in this section is the eye diagram for the Manchester NRZ pulse shape, which is shown in Figure 5(b).

```
% Generate unipolar waveform
rand_bin_str = round(rand(1,L));
new_upnrz_wavefm = [];
for ind = 1:1:length(rand_bin_str),
    if (rand_bin_str(ind) == 1)
        new_upnrz_wavefm = [new_upnrz_wavefm upnrz1];
    else
        new_upnrz_wavefm = [new_upnrz_wavefm upnrz0];
    end;
end;

% Visualize
eyediagram(new_upnrz_wavefm,L_lc,L_lc,L_lc/2);
```



(a) Unipolar NRZ. (b) Manchester NRZ.

Figure 5: Examples of an eye diagram for various pulse shapes.

ACTION (10 Points): For this section, please obtain all eye diagrams for unipolar NRZ, polar NRZ, unipolar RZ, bipolar RZ, and Manchester NRZ pulse shapes. Furthermore, please provide at least a paragraph of explanation for each pulse shape regarding why we are observing these eye diagrams.

7 Bringing It All Together

IT IS SUGGESTED TO KEEP YOUR VOLUME LOW IN THIS SECTION, ESPECIALLY IF YOU HAVE HEADPHONES. Some sounds will be purposly loud or metallic. You can always turn the volume up if you can't hear the sound file. Let's look at three real sound files taken from popular Vines (typically 7 second videos exchanged on social media platforms). Three .wav sound files are presented, each looking like Figure 6 upon import to MATLAB. Each file is modulated to a flat-top PAM waveform in the form:

QUESTION (5 Points): What impact would increasing N (downsampling/upsampling rate) have on the quality of the recovered sound file after decoding the binary stream? Identify which one of the mystery sound files has (very high) $N = 32$.

```
[raw_file, Fs] = audioread('./file_name.wav');
L = length(raw_file); % Length of the overall transmission
N = 2; % downsample/upsample rate

% encode the waveforms
impulsetrain = reshape(ones(N,1)*rem(1:1:(L/N),2),[L,1]);

natural_pam = raw_file .* impulsetrain;
flattop_pam = repelem(natural(1:N:end), N);
```

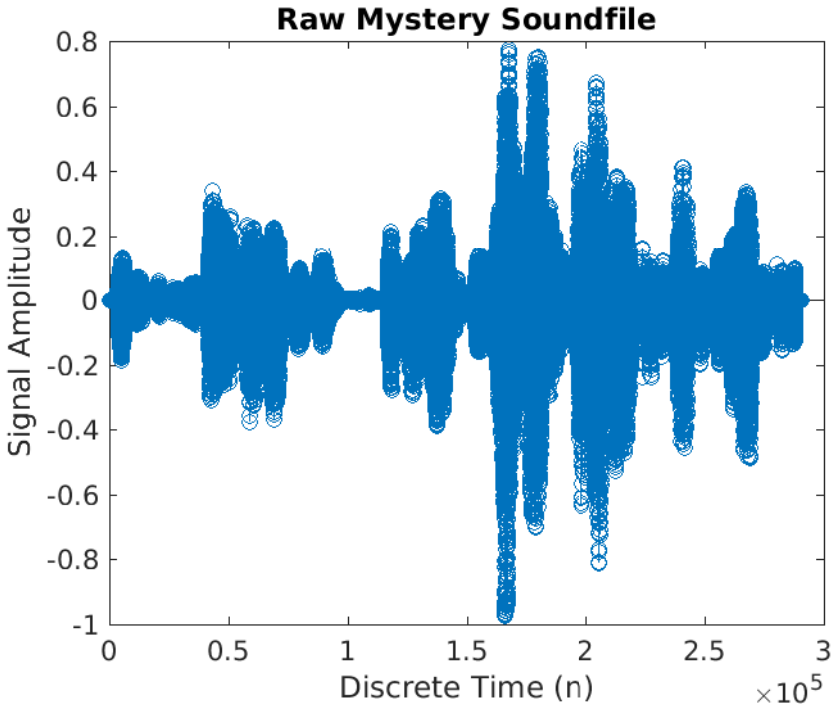


Figure 6: One of the raw sound files brought into MATLAB using audioread. Vmin is about -1 and Vmax is about 0.8.

QUESTION (5 Points): In Section 4, the number of audio levels (10) is insufficient for quality sound. What effect would having a low number of amplitudes have on the sound file? Identify which one of the mystery sound files has (very few) 8 amplitudes. Which sound file has a high number of amplitudes and a low N?

Each sound file is encoded to single binary row vector of codewords of unkown lengths. 2^C evenly spaced amplitdue values from $Vmin$ to $Vmax$ make up the code book, with partitions fill the space inbetween them. Using the quantiz() function, the flattop pam waveform is quantized into a PCM. A dictionary of binary assignments is then created such that for $C = 2$, $|Vmax| = |Vmin| = 1$, $binaryassignments = [00011011]$. In this case, if we were to decode this binary stream, we would map sequential 0s to the amplitude $Vmin$ for N values, or $Vmin/2$ for 0s followed by a 1, or $Vmax/2$ for 1s followed by a 0, or $Vmax$ for double 1s.

```
Vmax = max(raw_file);
Vmin = min(raw_file);
codes = 2^C; % number of codes in the codebook

% 2^C amplitude values ranging from Vmin to Vmax
codebook = linspace(Vmin, Vmax, codes);
partition = linspace(Vmin+abs(codebook(2)-codebook(1)), Vmax, codes-1);
[index,quant_file] = quantiz(flattop,partition,codebook); % Quantize.

% Convert to binary where Vmax is all zero, -Vmax is all 1, and each
% descending value increments the least sig bit
binary_stream = zeros(1, L/N*2);
% remove length-n zero-value segments..they're not in our codebook!
nz_encoded = quant_file(quant_file ~= 0);
binary_assignments = de2bi(linspace(0,codes-1,codes));

% Map amplitudes to binary using the binary_assignments and codebook
for i=1:length(nz_encoded)
    curr_assign = (nz_encoded(i) == codebook);
    binary_stream(1+(i-1)*log2(codes):i*log2(codes)) = ...
        binary_assignments(curr_assign,1:end);
end
```

ACTION (20 Points): For $C = 8$, $N = 2$, and the provided binary streams $\overline{binary\_stream\_1, 2, 3}.m$, form a codebook, partitions, a list of binary assignments, and map the binary to repeated amplitude values.

QUESTION (5 Points): Try playing your recovered sound file using MATLAB's sound(file, Fs) function. What effect does varying the sample rate Fs have on the sound file?

8 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 (10 points) of the process 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%".