ECE3311 – Principles of Communication Systems

Project 02

Due: 5:00 PM (EDT), Monday 9 November 2020

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.

For this project, you will need to download the Jupyter Notebook "*Project 2 ECE3311.ipynb*" from the ECE3311 Canvas website and open it in VSCode, which will provide you with all the example source code and outputs of subsequent topics discussed in this project.

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 Python code provided, 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`.

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 Python code. Let us see how this is used in the next several sections.

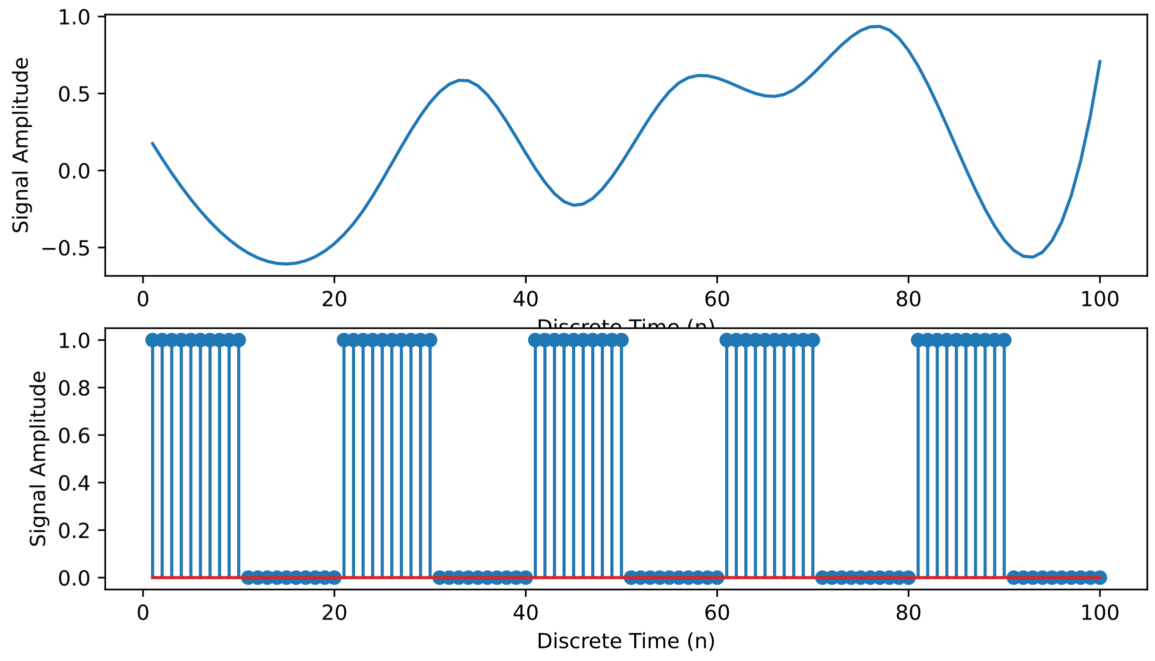


Figure 1: Examples of a randomly generated analog waveform (top) and a periodic stream of rectangular pulses (bottom).

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
```

Question 1 (1 point): Using the sample Python code that generated `analog_wavefm`, implement and plot three "analog" cosine waveforms with each possessing a different frequency of your choosing. Please make sure to choose frequencies where you can clearly observe the periodicity of the cosine waveforms.

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. Leveraging the waveforms from Section 2, we can readily generate both naturally sampled and flat-top sampling PAM waveforms of a random analog baseband signal. The results should appear to be similar to those shown in Figure 2.

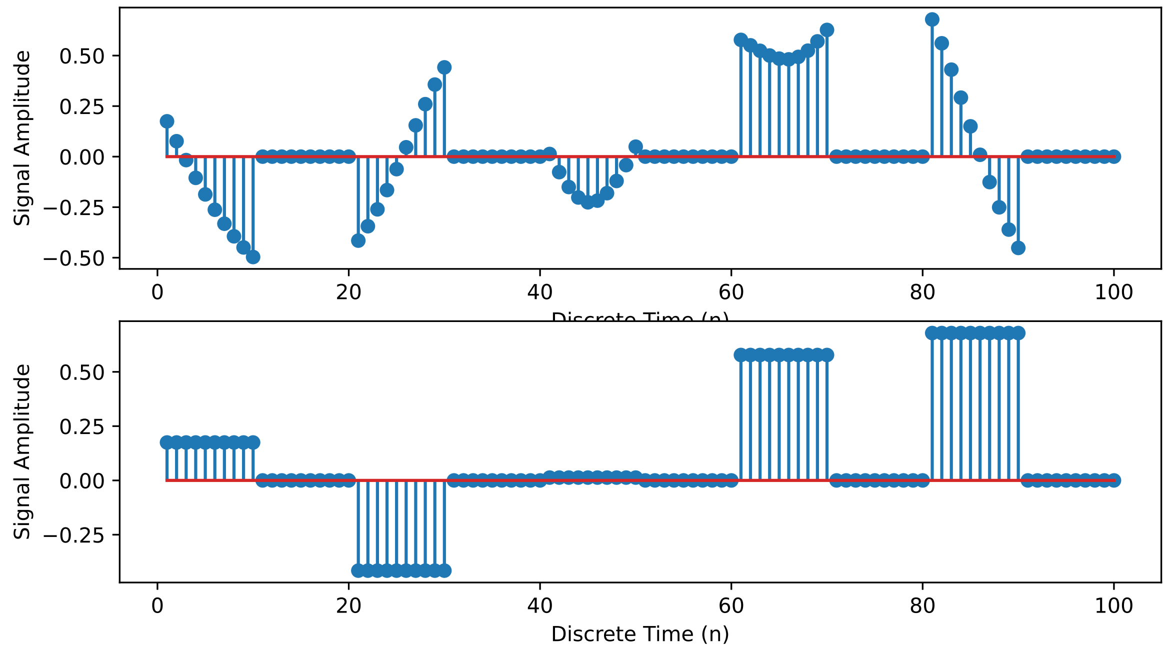


Figure 2: Examples of a naturally sampled PAM waveform (top) and a flat-top sampling PAM waveform (bottom).

Question 2 (2 points): Using your cosine waveforms from Question 1, generate both naturally sampled PAM and flat-top PAM waveforms. Please explain your results.

4 Pulse Coded Modulation

Now that you are comfortable with manipulating analog waveforms in Python, 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. An example of how an analog waveform gets converted into a PCM waveform, as well as the quantization error that results, is shown in Figure 3. In the Jupyter Notebook provided for this project, a custom Python function `quantize` has been defined to perform this operation.

Since the purpose of the PCM waveform is to ultimately convert an analog signal into a string of binary values, there exists a mapping between the binary values and each PCM amplitude value.

Question 3 (4 points): Using `quantize`, please plot the three analog cosine waveforms from Question 1, their resulting PCM waveforms, and their associated quantization error.

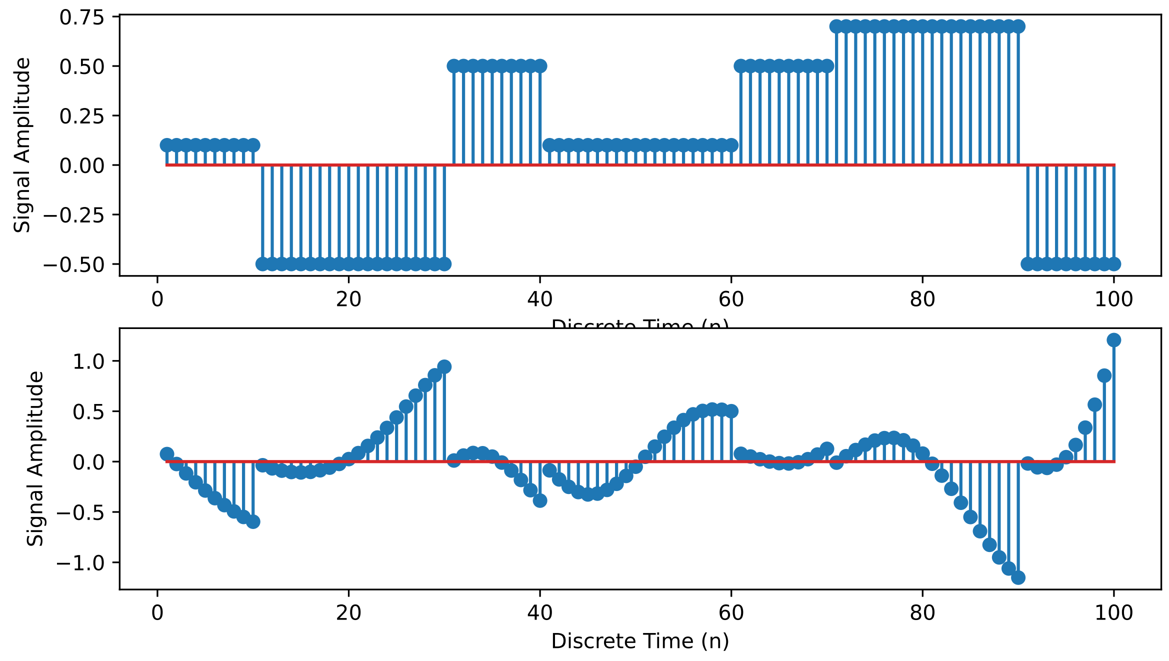


Figure 3: Examples of a quantized analog waveform (top) and the residual quantization error after converting it into a PCM waveform (bottom).

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. The Python code provided in the Jupyter Notebook will generate a unipolar NRZ line coding waveform given a random binary sequence.

Question 4 (6 points): Using the example Python code for unipolar NRZ line coding, please generate the corresponding waveforms for these line coding schemes: polar NRZ, unipolar RZ, bipolar RZ, and Manchester NRZ. Your results should reassemble Figure 4.

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 Python for generating eye diagrams. Sample Python code is provided in the Jupyter Notebook for analyzing a unipolar NRZ waveform.

As you can see in Figure 5(a), the Python code example traces both pulses generated from transmitting ones and zeros into a single plot. Thus, we are overlapping the pulses every pulse period in

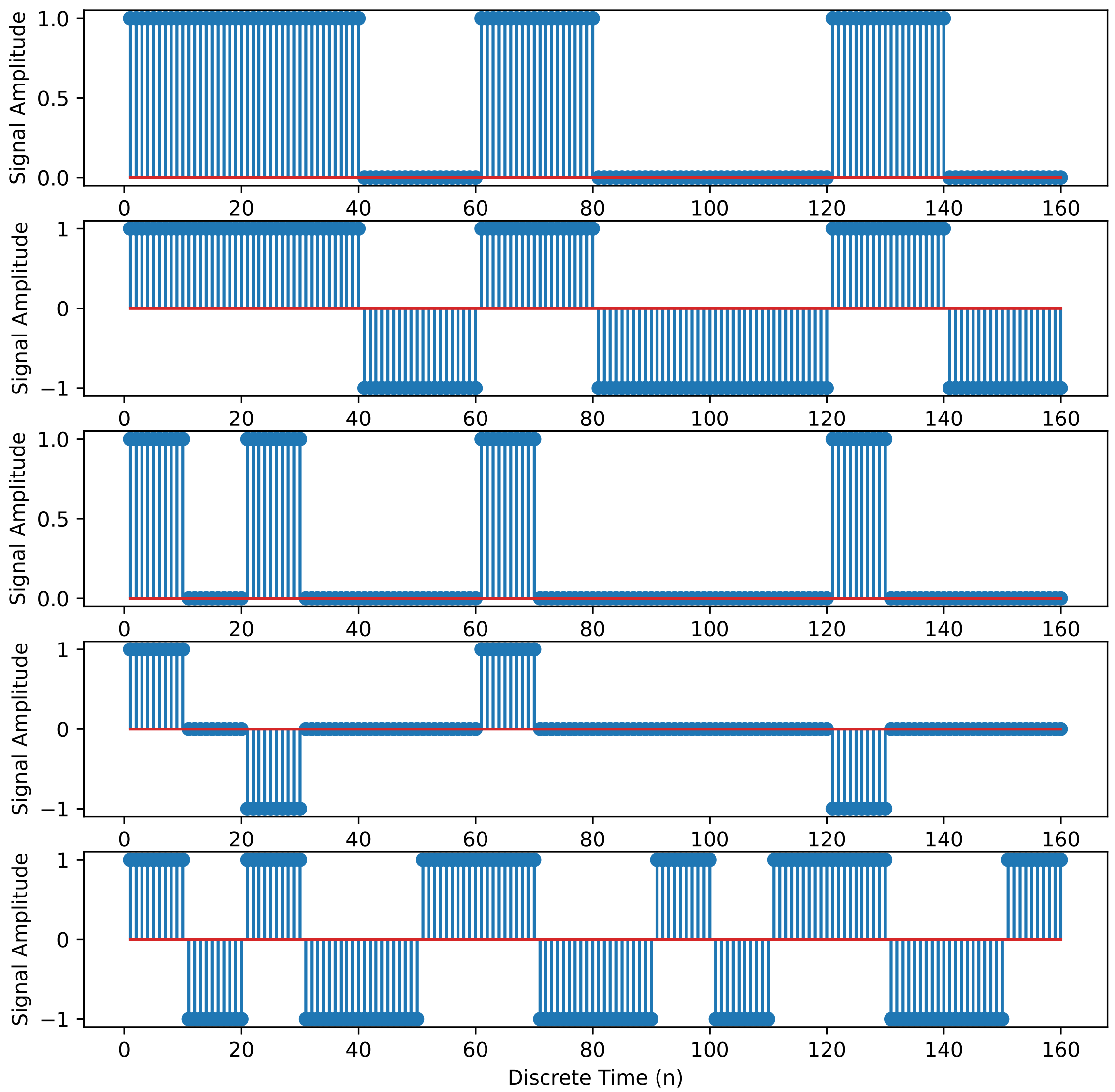
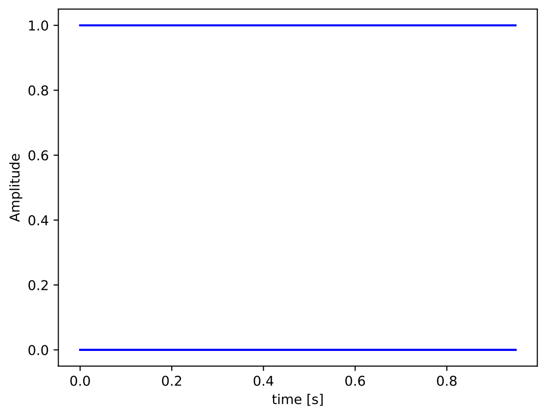
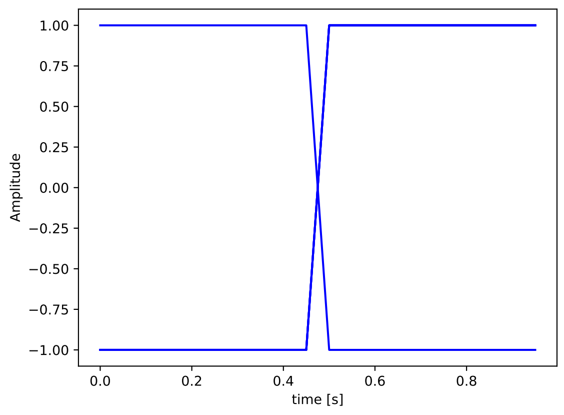


Figure 4: Examples of line coding techniques (from top to bottom): Unipolar NRZ, Polar NRZ, Unipolar RZ, Bipolar RZ, Manchester NRZ.

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).



(a) Unipolar NRZ.



(b) Manchester NRZ.

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

Question 5 (3 points): Please obtain all eye diagrams for unipolar NRZ, polar NRZ, unipolar RZ, bipolar RZ, and Manchester NRZ pulse shapes. Provide an explanation for the appearance of each eye diagram based on each pulse shape.

7 Real-World Application

In order to synthesize the concepts covered in this project, let us do something in the real world. Provided as part of this project are three mystery sound files avaialble in Canvas that contain audio snippets from YouTube (*mystery_1.wav*, *mystery_2.wav*, *mystery_3.wav*). Each audio snippet is approximately 5-10 seconds long in duration. You will need to be careful of the sampling rate used to transform audio file since too few or too many samples per second will result in audio that is going to be completely unintelligible.

Question 6 (4 points): Read the three mystery audio signals into your Python code and generate their corresponding time domain plots. What is the message for each audio file?

Question 7 (12 points): Using PCM based on 10-bit quantization, convert each mystery audio signal into a binary sequence and then convert it back to a reconstructed audio signal. Plot the resulting quantization error for each mystery waveform (original versus reconstructed). Play the reconstructed audio files and describe how they sound.

8 Report Submission

For each project in this course, the report submission to be uploaded to the ECE3311 Canvas website will consist of a single comprehensive Jupyter Notebook (.ipynb) file and nothing else unless it is requested. For this project, only the Jupyter Notebook is to be submitted electronically by the due date. Failure to submit the Jupyter Notebook by the specified due date and time will result in a grade of "0%" for the project.

Several important items to keep in mind when preparing your submission:

- Include the course number, project team number, names of team members, and submission date at the top of the Jupyter Notebook.

- Make sure your source code is thoroughly documented such that it is made clear what exact the program is doing line-by-line. Adequate commenting is worth **5 points** of the total score for this project.

- Responses to all questions indicated in the project handout. Please make sure that the responses are of sufficient detail.