Laboratory #2

Spectral Analysis of Digital Baseband Signals

SYSC 4600 Digital Communications

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University

Purpose and Objectives

The purpose of this laboratory experiment is to observe and understand the effects of different line coding techniques on the spectral characteristics of digital baseband signals. It builds on Laboratory #1, which you should review. You are encouraged to refer back to that lab manual while working on this lab.

Prelaboratory Exercise

1. Review lab #1.
2. Read the lab manual.
3. Draw the power spectral density (PSD) (in dB) of polar NRZ signalling, where $s_0(t)$ shown in Figure 1 is used to represent a message bit of 0, and $s_1(t)$ is used for a 1. The PSD of polar NRZ signaling and independent message bits is given by $P(f)=\frac{1}{T}|H_T(f)|^2$, where $H_T(f)$ is the Fourier transform of the transmit filter $h_T(t)$, and $h_T(t)=s_0(t)$.

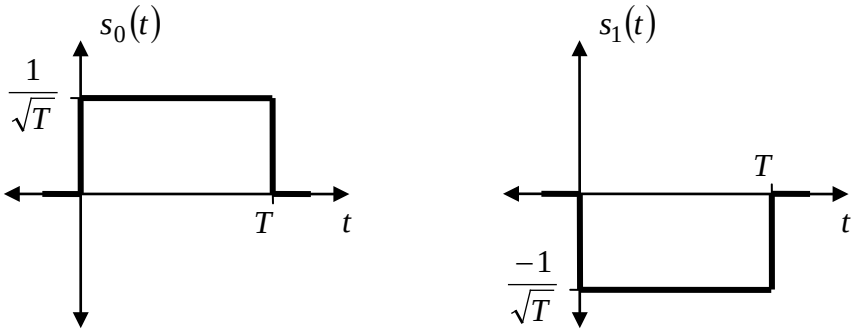

Figure 1. Signals used with polar NRZ.

Laboratory

Preliminaries: Improving Your Simulator

The simulator you developed in lab #1 is a good start, but it does have some drawbacks that should be rectified before commencing this lab. Make the following changes to your simulator:

1. The transmit filter pulse shape should be normalized to have unit energy. The rectangular pulse shape should be as shown in Figure 2.

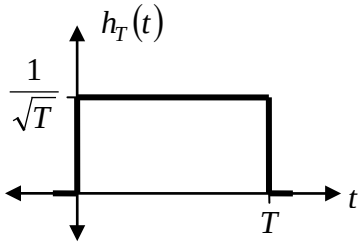

Figure 2. Normalized rectangular pulse shape.

It is a good idea to always use normalized pulse shapes, so that the transmit filter neither amplifies nor attenuates the signal. We want the transmitted energy per symbol to be the same regardless of the pulse shape, to make it easier to study the effects of the pulse shape on system performance without the effects being obscured by potentially different transmitted signal strengths (average energy per bit).

2. The modulator should be modified so that the transmitted bandpass signal, $v_c(t)$, has the same energy as the baseband signal, $v(t)$. This is done by using a scaling factor of $\sqrt{2}$, so that

$$v_c(t) = v(t)\sqrt{2}\cos(2\pi f_c t) \ .$$

A similar change must be made in the demodulator so that

$$r_o(t) = r_c(t)\sqrt{2}\cos(2\pi f_c t) \ .$$

3. To make the discrete-time convolution performed in the detector better mimic the continuous-time convolution that would be performed by an analogue receive filter, you should multiply the result of the discrete-time convolution by T_s. You should also make sure that the impulse response of the receive filter is normalized to unit energy, just like the pulse shape used by the transmit filter.

Here are some other suggestions that will make your simulator easier to maintain and possibly run faster.

1. Because MATLAB .m files are interpreted at run-time, not compiled in advance, loops can run very slowly. It is important to replace loops in your program with vector operations wherever possible. Speed was not an issue in lab #1, but is more important in this lab (you'll notice your program sometimes takes a few seconds to run). To avoid using a loop for the symbol mapper you can use (for polar signaling)

```
SM = [+1 -1];
v = SM(a+1);
```

where `a` is the MATLAB vector containing the message bits. Elements in `a` that have a value of 0 will be mapped to the value of the first element of the `SM` array, and values of 1 will be mapped to the value of the second element of the `SM` array.

2. The decision device can be implemented without a loop by using

```
ah = (r < 0);
```

   where `r` is the MATLAB vector containing the detector output.

3. To generate the receive filter impulse response (matched filter), use

```
hR = fliplr(hT);
```

   where `hT` is the MATLAB vector containing the sampled transmit filter pulse shape.

After making these changes you should run your simulator and verify that no transmission errors occur. For this lab we will use the following parameters: $N_a = 128$ bits, $T = 0.01$ seconds, $\eta = 64$ samples per symbol, and $f_c = 400$ Hz. **Use a randomly-generated message.**

Step 1: Spectrum of Polar Non-Return-to-Zero (NRZ) Signals

We are interested in the PSD of the transmitted baseband signal, $v(t)$, when polar NRZ signalling is used. One method to estimate the spectrum of a signal is to square the magnitude of the discrete Fourier transform of the sampled signal. This can be accomplished with the following MATLAB commands:

```
vt = vt(1:Ns);                  % Truncate to first Ns samples
Vf = fftshift(fft(vt));         % Calculate FFT
PSD = Vf.*conj(Vf) * Ts / Ns;   % Calculate PSD
```

where `vt` is the MATLAB vector containing the samples of $v(t)$ and $N_s = N_a\eta$ is the total number of samples of $v(t)$. The MATLAB vector `PSD` will contain an estimate of the spectrum over the interval $\left[\frac{-1}{2T_s}, \frac{1}{2T_s}\right)$, at N_s points with an increment of $\frac{1}{N_s T_s}$.

Question 1. Plot the spectrum of $v(t)$ (in dB) vs. frequency (in Hz). Make sure your frequency scale is correct. Limit the *x*-axis to the range from -750 to 750 Hz, and limit the *y*-axis to the range from -50 dB to 10 dB. On the same graph, but in a different colour, plot the theoretical PSD that you found in the pre-lab, evaluated at the same frequencies. Your graph should look similar (but not identical) to the one shown in Figure 3.

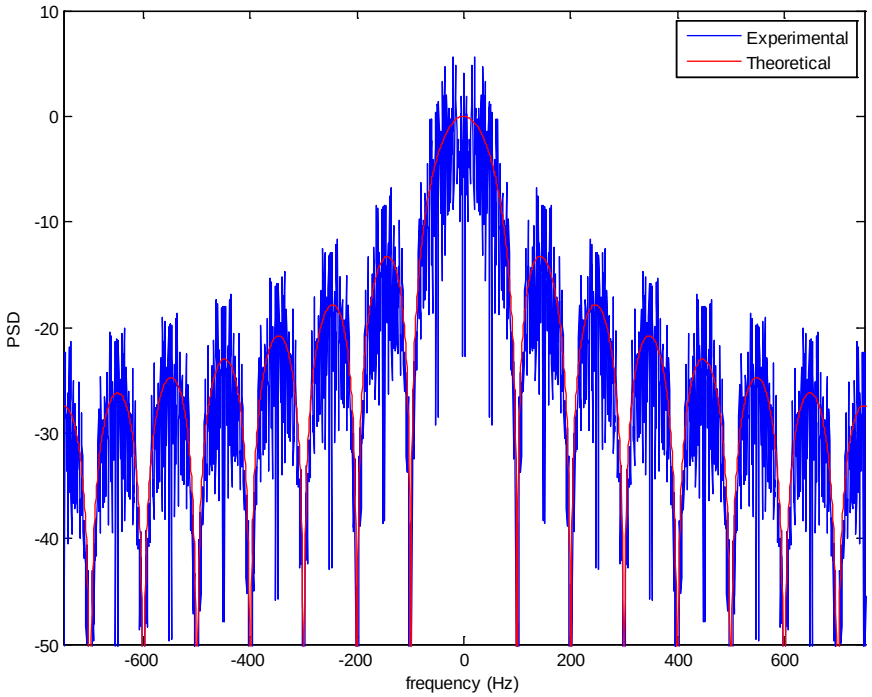

Figure 3. PSD of polar NRZ.

By comparing the theoretical and experimental results, it should be apparent that although the two curves are similar, the experimental results are much more jagged. It is desirable to get a

smoother curve (and hence higher accuracy) for the experimental results. Increasing η doesn't help (higher values of η does however allow us to estimate the spectrum at higher frequencies), and increasing N_a doesn't help either (increasing N_a does allow us to estimate the spectrum with a higher resolution, but this doesn't make the graph smoother). Instead we will use what is known as *Bartlett's method*, which involves taking the average of several different estimates of the spectrum.

Question 2. Rerun your simulator and generate another graph of the spectrum. It should be similar to, but slightly different than, the graph you produced for Question 1. Why is it different?

Modify your simulator by putting nearly everything in a big loop. Instead of just transmitting one message of N_a bits, you will transmit several different randomly selected messages, one after the other. Let N_f be the total number of simulated messages that are transmitted. For each message your program should calculate (but not plot) the spectrum of $v(t)$ for that message. After the loop your program should plot the average of these spectra. Your program should also print the total number of bit errors that occurred (this should be equal to zero).

Question 3. Plot the spectrum of $v(t)$ averaged over N_f = 10 messages. Shown the theoretical spectrum in the same graph. Are the experimental results less jagged than in Question 1? Repeat with N_f = 100 and N_f = 1000. Do the experimental results match the theoretical ones? How and why do they differ?

Keep a copy of the MATLAB array containing your estimated spectrum for polar NRZ with N_f = 1000 – you'll need it in Step 5.

Step 2: Polar vs. Unipolar Non-Return-to-Zero (NRZ) Signals

Unipolar NRZ involves using a rectangular pulse with a positive amplitude to represent a message bit of 1, and sending nothing (0 volts for *T* seconds) to represent a 0. To simulate unipolar NRZ, change your symbol map to

$$v_n = \mathrm{SM}[a_n] = \begin{cases} 0 & \text{if } a_n = 0 \\ \sqrt{2} & \text{if } a_n = 1 \end{cases}.$$

The amplitude of $\sqrt{2}$ is used to ensure that unipolar NRZ has the same average transmitted energy per bit as polar NRZ. Because the symbol map has changed, the decision rule should be changed to

$$\hat{a}_n = \begin{cases} 0 & \text{if } r_n \le \sqrt{2}/2 \\ 1 & \text{if } r_n > \sqrt{2}/2 \end{cases}.$$

Question 4. Plot the spectrum of $v(t)$ (in dB) vs. frequency (in Hz) for unipolar NRZ, using N_f = 1000. Limit the *x*-axis to the range from -750 to 750 Hz, and limit the *y*-axis to the range from -50 dB to 10 dB. On the same graph, plot the theoretical PSD for

polar NRZ that you found in the pre-lab. Comment on the similarities and differences.

Keep a copy of the MATLAB array containing your estimated spectrum for unipolar NRZ.

Step 3: Polar and Unipolar Return-to-Zero (RZ) Signals

Return-to-zero (RZ) signalling involves using a pulse shape the drops back down to 0 volts halfway through the symbol period. The pulse shape is shown in Figure 4.

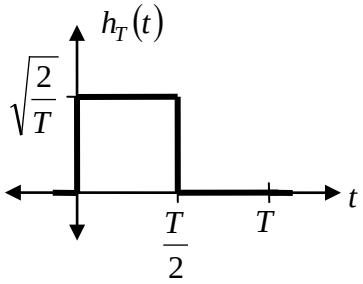

Figure 4. Normalized return-to-zero pulse shape.

Implement this pulse shape and modify your symbol map and decision device to use polar signalling again.

Question 5. Generate a graph showing the experimental spectrum of **polar RZ** and the theoretical spectrum of polar NRZ. Plot the spectra in dB from -50 to 10 dB for frequencies over the range from -750 to 750 Hz. Comment on the similarities and differences.

Keep a copy of the MATLAB array containing your estimated spectrum for polar RZ.

Modify your symbol map and decision device to use unipolar signalling again.

Question 6. Generate a graph showing the experimental spectrum of **unipolar RZ** and the theoretical spectrum of polar NRZ. Plot the spectra in dB from -50 to 10 dB for frequencies over the range from -750 to 750 Hz. Comment on the similarities and differences.

Keep a copy of the MATLAB array containing your estimated spectrum for unipolar RZ.

Step 4: Manchester Coding

Manchester coding uses the pulse shape shown in Figure 5. Similar to polar RZ, this pulse shape also always gives an amplitude transition in every symbol period to facilitate synchronization. It also has an average value of zero (no DC offset), regardless of the transmitted message.

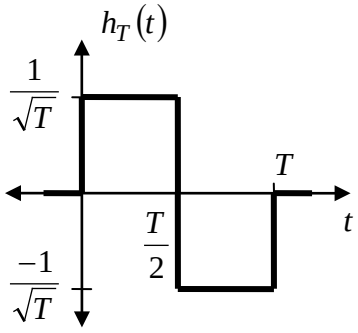

Figure 5. Pulse shape for Manchester coding.

Implement this pulse shape, and modify your symbol map and decision device to use polar signalling again.

Question 7. Generate a graph showing the experimental spectrum of Manchester coding, and the theoretical spectrum of polar NRZ. Plot the spectra in dB from -50 to 10 dB for frequencies over the range from -750 to 750 Hz. Comment on the similarities and differences.

Keep a copy of the MATLAB array containing your estimated spectrum for Manchester coding.

Step 5: Comparison

Question 8. Generate a single graph showing the experimental spectrum of the following line codes:

1. Polar NRZ
2. Unipolar NRZ
3. Polar RZ
4. Unipolar RZ
5. Manchester coding

Use a different colour for each line code. Plot the spectra in dB from -50 to 10 dB for frequencies over the range from -750 to 750 Hz. Comment on the similarities and differences.

Question 9. Unipolar RZ is not used in practice. Why not?

Laboratory Report Instructions

Please email a soft-copy of your .m file to your TA, and include a hard-copy with your report. Please email a soft-copy of your report to the TA in addition to submitting a hard-copy.