Lab 6: Zero-Forcing Equalization

Objectives:

1) To implement a simple zero-forcing equalizer in MATLAB Simulink;
2) To observe the conditions under which zero-forcing equalization is effective;
3) To learn about common pulse shaping filters and eye diagram tools.

Introduction:

In most communications systems, the channel over which signals are sent is not ideal in any sense. Echoes, reflections, and time delays can cause a signal to be split into multiple copies which arrive at the receiver at different time delays. In many wired systems, these imperfections can be modeled by a static impulse response Pc(t). When this impulse response is convolved with an incoming signal, it basically has the effect of blurring adjacent symbols together, which is known as inter-symbol interference, or ISI. We wish to construct a filter on the receiving system that can undo or "equalize" the effects of ISI given the approximate channel response. These filters are known as *equalizers*, of which many types exist for different types of channels.

The zero-forcing equalizer is arguably the simplest of these. The basic idea of the ZF equalizer uses the fact that the channel impulse response Pc(t) can be transformed into a frequency response Pc(f). By constructing another filter on the receiver that has the inverse response $H^{-1}(f)$, we can undo the effects of the original channel, since $Pc(f)Pc^{-1}(f) = 1$. The main problem with this is that we are limited to a finite number of equalizer coefficients in a real system, and to get this inverse perfectly, we would need an infinite number of coefficients. Fortunately, we can still get an approximation to this inverse by truncating the response to a finite number of terms. The other major problem, most prevalent in wireless communications, is that the ZF equalizer amplifies noise in any case where Pc(f) has a significant drop-off. If Pc(f) is close to zero at some frequency, the inverse will have a very large gain at that frequency, and any noise in this region will be amplified excessively. For this reason, it is mostly useful in systems that have a very high SNR.

Preliminary:

Assume the channel impulse response is Pc[n] = [0.02, -0.05, 0.2, 1, 0.3, -0.07, 0.03]. Design a ZF equalizer with (2N+1) taps, where N is chosen to be 2 and 5. (Find the coefficients using the matrix inversion method according to the textbook example.)

Procedure:

Part A – Creating the Basic Model

In baseband digital communications, the ISI channel and ZF equalizer are both modeled as an FIR filter. They are functionally the same but with different filter coefficients. In Simulink, this is implemented by "Discrete FIR Filter" module. The coefficients can be entered in as dialog parameters or as input port from other sources such as a file or a workspace variable.

For the first part of this experiment, we consider NRZ coding with rectangular pulse shaping for a binary bit stream with equally likely ones and zeros. We filter this NRZ waveform by the channel FIR filter, add a white noise, and then filter it again by the ZF equalizer. To demonstrate the ISI effect and the equalizer performance, we use a scope as well as three "Eye Diagram" modules to show the signals. The finished model shall look like the one in Figure 1.

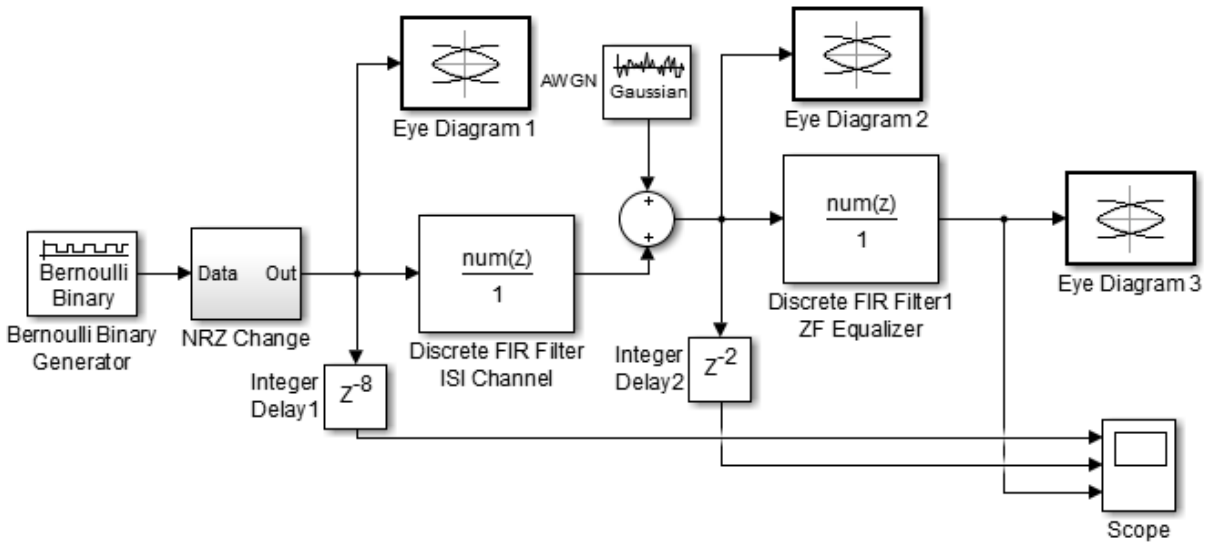

Figure 1 – System setup for ZF equalizer simulation: NRZ pulse

Since the "Eye Diagram" scope requires that each symbol be represented by at least two samples, we have to set the Bernoulli Binary Generator with Sample rate = 0.5. This create two samples per bit and the NRZ model you built in Lab 5 converts the ones and zeros into NRZ waveform. To match the sampling rate of the Discrete FIR Filter with the source sampling rate, the channel coefficients provided in the Preliminary section are to be padded with a zero after each coefficient, as shown in Figure 2.

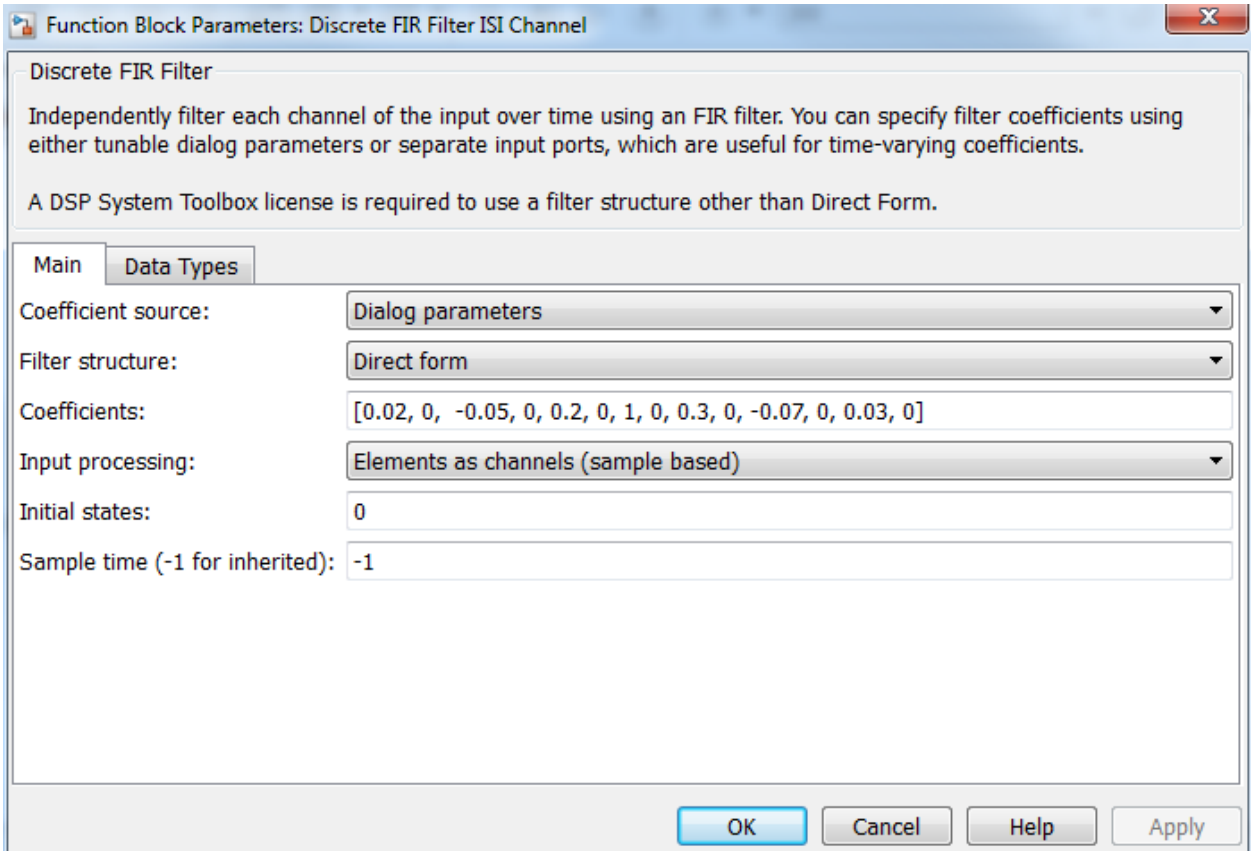

Figure 2 – ISI channel coefficients for source sampling rate=2

Similarly, the coefficients in the ZF equalizer have to be padded with zeros. Use the three coefficients from the textbook first. Also, set the AWGN module as a Gaussian distribution with zero mean, variance =0.01 (that is SNR=20 dB), sample time =0.5 (the same as that in the Bernoulli module). The delay modules are used to line up the source with the filtered outputs because each filter will introduce a delay that equals to floor((L-1)/2) samples, where L is the length of the filter. Since we have 14 coefficients in the channel and 6 coefficients in the ZF equalizer, the delays are set as in Figure 1.

Set the solver to discrete-time and the total simulation time to 100. When finished, run the model to ensure that everything is working properly. You shall see three eye diagram plots and a time domain scope as in Figure 3 and 4, respectively. Figure 4 is zoomed in along x axis so that the delay of each waveform is show clearly. Notice that the equalized output has a slightly wider eye opening in the middle of the eye diagram than the ISI channel output. The time-domain scope also shows some distortion of the waveforms. This is because the ZF equalizer has only three taps, and its ability to combat ISI is limited.

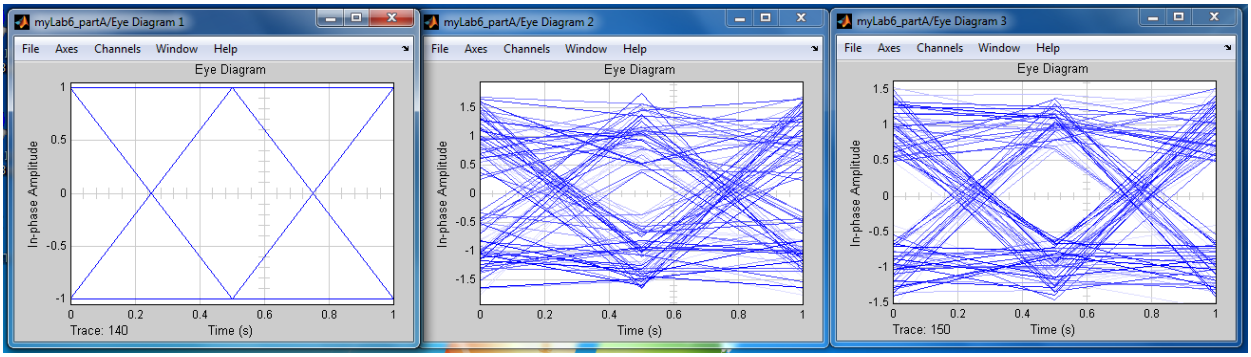

Figure 3 – Simulation results with NRZ waveform on eye diagram scopes. As the ZF equalizer has only three taps, the equalized output has a slightly wider eye opening in the middle than the ISI channel output.

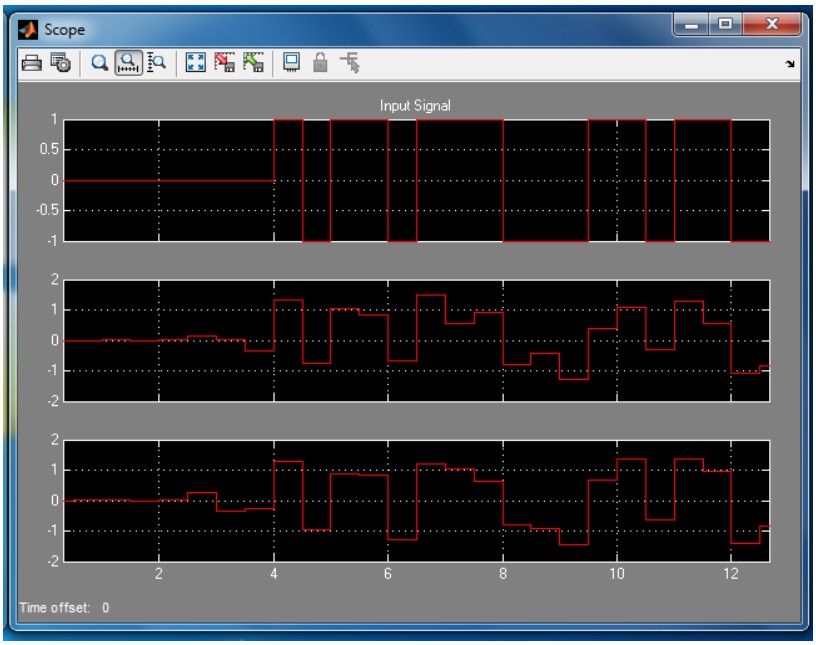

Figure 4 – Zoom-in results with NRZ waveform on time-domain scopes

Once your model is functional, change the simulation time to 1000 and run again. Show all the 1000 traces in your eye diagrams. Plot 20 bits of NRZ waveforms in the scope and save the screen shot. Include the results in your report.

Next, change the channel coefficients into the ones that you calculated for N=2 and N=5 in the Preliminary part and run the model again. Remember to pad a zero after each coefficient and change the delays in the two delay blocks. The results: Your eye diagram 1 and 2 shall look the same as before, but eye diagram 3 would be different for N=1, N=2, and N=5. Compare the ZF equalizer outputs with the three cases using both eye diagrams and time-domain scopes.

Change the variance of the AWGN block to -2 dB -- that is 10 to the power of (-2/10). Run the model with N=2 and N=5 again and compare the results with the case of SNR= 20 dB. Submit your models with your report. Is the ZF equalizer effective on combating AWGN, too?

Part B – Equalizer with pulse shaping filter.

As we learned in class, rectangular pulses for NRZ is never practical because they occupy too much bandwidth and pulse shaping filters are commonly used in practical systems to limit the bandwidth of the transmitted signal. The square root raised cosine (SRRC) filter is one of the ISI free pulse shaping filters for transmitter. When paired with a SRRC matched filter at the receiver, the combined response becomes a raised cosine filter that introduces no ISI.

Matlab Simulink has a library block called Pulse Shaping Filter that interface nicely with the sources, processing blocks, and sinks in the communication toolbox. To simplify the simulation model, we combine the transmit SRRC and receive SRRC together and use the Raised Cosine pulse in the model. The finished model shall look like the one in Figure 5.

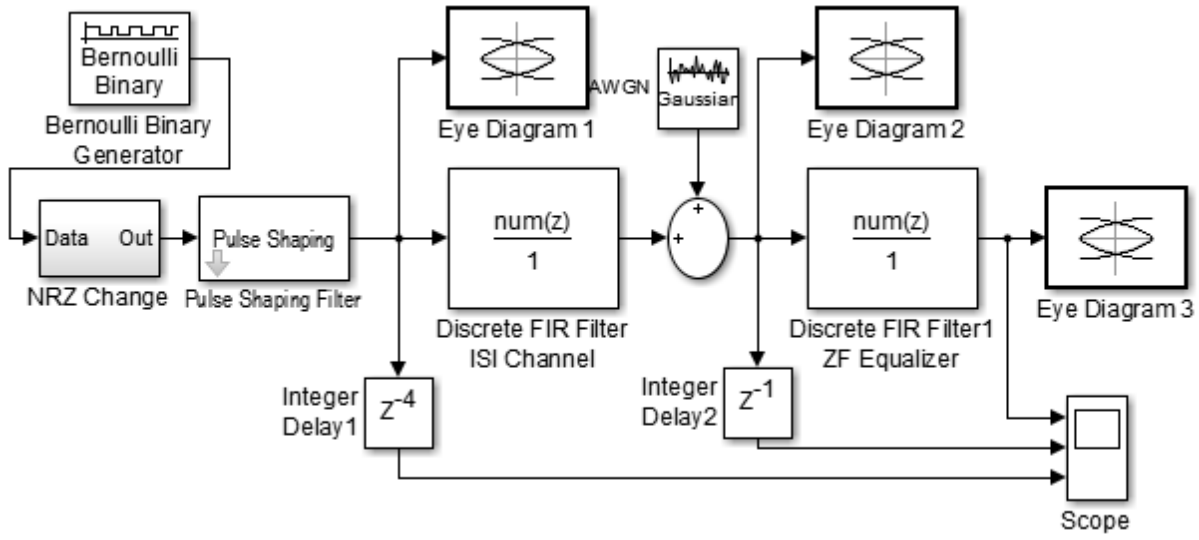

Figure 5 – Equalizer with raised cosine pulse shaping filter

With the proper parameters selected, the pulse shaping filter can "automatically" interpolate a bit sample into a 1D vector that represents the pulse-shaped bit. Therefore, we set the Bernoulli Binary Generator with sample time =1. The bit stream is processed by the Pulse Shaping Filter with frame-based processing and a single rate. The ISI channel filter can inherit the data format without explicitly padding zeros in the coefficient entry. The eye diagrams shall use 2 times the "Samples per symbol" value in the Pulse Shaping Filter to display properly. The main parameter

settings are shown in Figure 6. Again, we choose simulation time 100 to test out the model first. Your results should look like the ones in Figure 7 when the offset is half of the "Sample per symbol" in the pulse shaping filter. Change the simulation time to 1000 after your model is tested and include the results in your report.

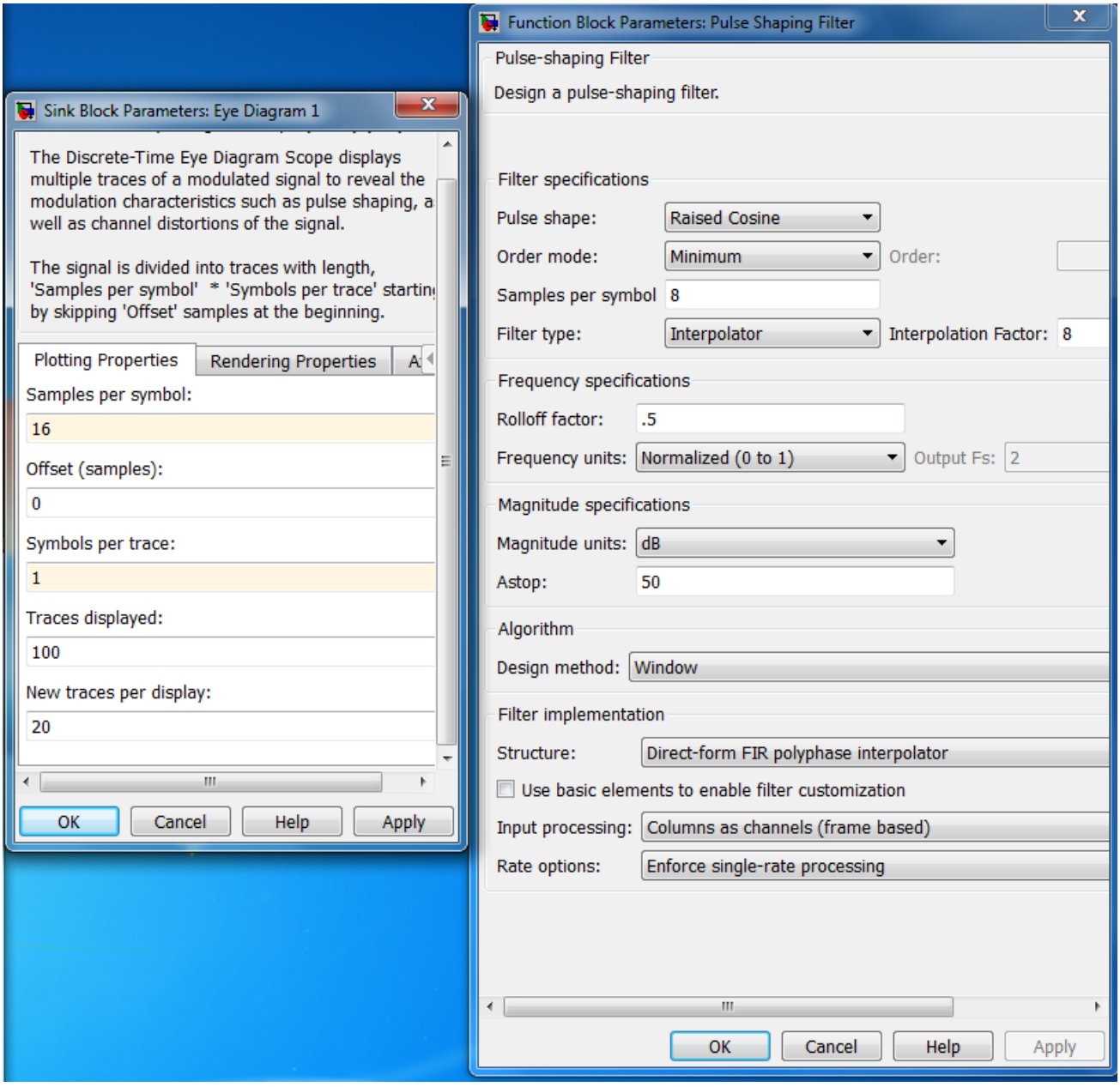

Figure 6 – Parameters for equalizer with pulse shaping filter

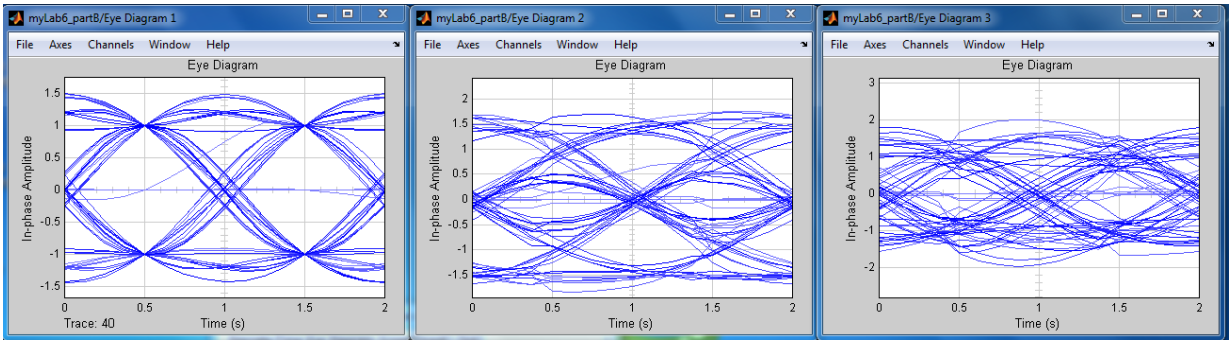

Figure 7 –Eye diagram outputs for equalizer with pulse shaping filter, with offset =4 when samples per symbol =8, each diagram shows two eye openings.

Again, change the ZF equalizer coefficients into those you calculated for N=2 and N=5, respectively. Be sure to change the parameters in the delay blocks for properly aligned time-domain plots. Run the model to compare the results of the ZF equalizer with N=1, N=2, and N=5.

Bonus (20 points): If the ISI channel is set as Pc=[0.227, 0.46, 0.688, 0.46, 0.227] in Part A. re-design the ZF equalizer with N>=5. Run the model and compare the results with that in Part A.

Bonus (20 points): If SRRC filters are required for simulating Tx and Rx individually instead of a combined raised cosine pulse, how is your model different from that in Part B? Submit your model and explain the differences in the model and the results.

Post-Lab Questions:

1. From Part A results, is the ZF equalizer effective on combating ISI and/or AWGN?
2. How are the results different between Part A and Part B when different pulses are used to represent bits?
3. In wireless communications, the channels are no longer completely static, which means that Pc(t) can change with time as the transmitter, receiver, or other objects move around. What are some possible difficulties with designing an equalizer under these conditions?
4. What are the most important things you learned from this lab exercise?