ECE 463 Lab 3: Pulse Shaping and Matched Filtering

1. Introduction

The objective of this lab session is to introduce the basic concepts of pulse shaping, matched filter and pulse alignment. In digital communication, the message in digital must be converted into an analog signal to be transmitted. This conversion is done by the pulse shaping filter, which changes each symbol into a suitable analog pulse. Designing the pulse shape is important because the spectrum of the pulse dictates the spectrum of the whole transmission. However, to confine the spectrum, we have to smooth the pulse with slow transition. This requires the pulse extending beyond a symbol time, which introduces inter-symbol interference (ISI). Thus, there is a trade-off between the bandwidth and the ISI of the pulse shape.

After transmission, the matched filter helps to recapture the symbols from the received pulses. The matched filter is aimed at reducing the sensitivity of noise by maximizing the signal-to-noise ratio (SNR) and minimizing the ISI at the receiver.

In a more realistic channel model, the receiver never knows the precise arrival time of the pulse. Therefore, the receiver must know the exact symbol timing in order to correctly demodulate the transmitted symbols from the transmitter.

1.1. Contents

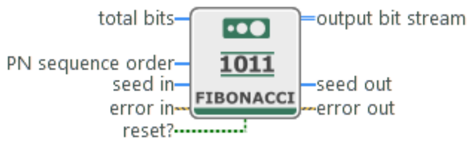
1.2. Report

Submit the answers, figures and the discussions on all the questions. The report is due as a hard copy at the beginning of the next lab.

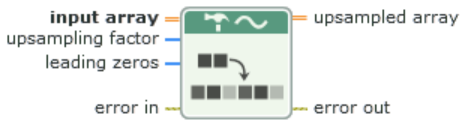
2. Key Components

The following components will be used in this lab session.

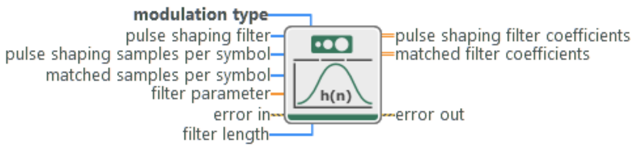
- **MT Generate Bits (Fibonacci, PN Order) (Analysis→Communications→Digital)**: Generates Fibonacci pseudonoise (PN) bit sequences. Use this block to generate the message bits to transmit.



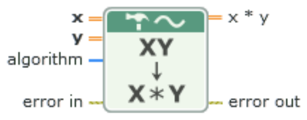
- **Upsample (Analysis→Signal Processing→Conditioning)**: Inserts zeros in a sequence according to a specific upsampling factor.



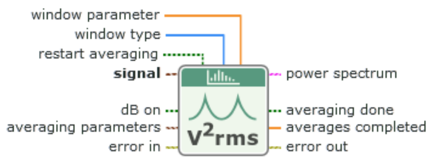
- **MT Generate Filter Coefficients (Analysis→Communications→Digital)**: Calculates filter coefficients for pulse-shaping and matched filters applied by the digital modulation nodes and demodulation nodes.


modulation type
pulse shaping filter
pulse shaping samples per symbol
matched samples per symbol
filter parameter
error in
filter length
h(n)
pulse shaping filter coefficients
matched filter coefficients
error out

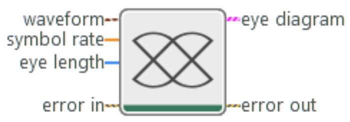
- **Convolution (Analysis→Signal Processing→Operation)**: Computes the convolution of two sequences.



- **FFT Power Spectrum and PSD (Analysis→Signal Processing→Measurement)**: Computes the averaged auto power spectrum of a time-domain signal.



- **MT Format Eye Diagram (Analysis→ Communications→Digital)**: Specifies a real-valued waveform, divides it into segments, and displays those segments as plots on a waveform graph. This node determines the segment length based on the symbol rate and eye length parameters.



3. Pulse Shaping

Start by loading "tx_template.gvi" **which you saved from lab 2.** In this section, we will build a transmitter that performs pulse shaping and upsampling to create the transmit waveform. Binary phase-shift-keying (BPSK) is used for modulation scheme. BPSK is the simplest form of phase-shift-keying (PSK). The data is conveyed in the phase of the carrier signal (i.e. "1" for 0 degree and "0" for 180 degree in BPSK). More details on modulation/demodulation will be covered in Lab 4 and Lab 5.

Set the default parameter values in the panel:

- Carrier frequency: 1G
- IQ rate: 400k
- Gain: 0
- Active antenna: TX1

1. Create two Integer-type controls: **Message length**, and **Symbol rate**. Set the message length to 1000 and the symbol rate to 10k. The IQ rate and the symbol rate will determine the upsampling factor. We will upsample the symbols before performing pulse shaping.
   - Message length: 1000
   - Symbol rate: 10k

2. *(Create data)* Add **MT Generate Bits** to generate a Fibonacci pseudonoise (PN) bit sequence. The result bit sequence is a sequence of binary digits 0 and 1. Notice that the default value of "PN sequence order" is 7. Use the default value. Connect the "Message length" control to "total bits" port.

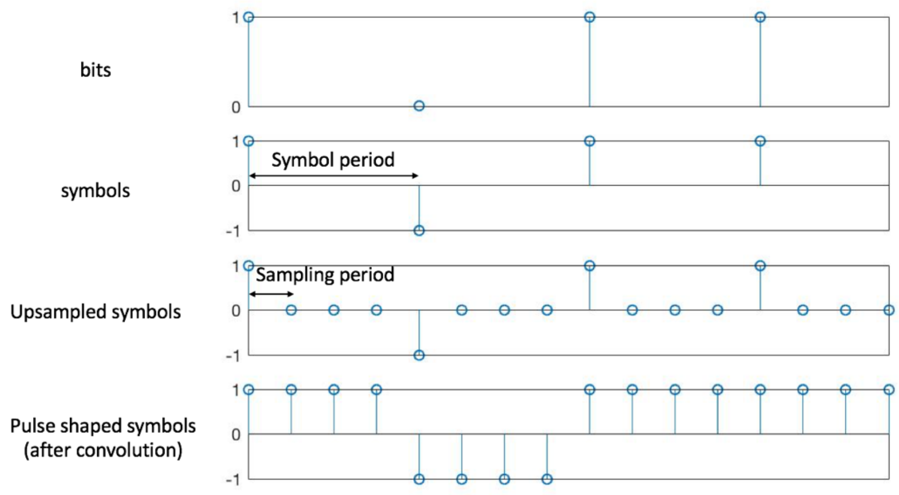
3. *(Bits to Symbols: BPSK)* Use **Multiply** and **Decrement** or **Case** to map the binary 0 and 1 into the symbols -1 and 1.

4. Create a DBL-type indicator and name it as **Upsampling Factor**. Use the appropriate math blocks to calculate the upsampling factor (Upsampling factor is defined by sampling rate/symbol rate).

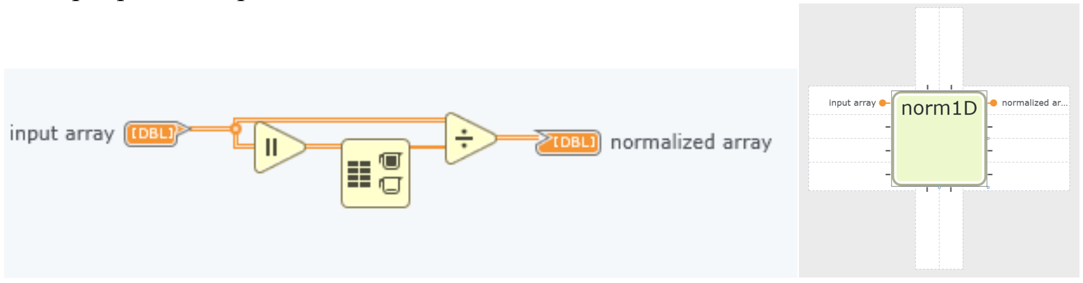
5. *(Upsampling*) Add **Upsample**. Use the symbols and the upsampling factor in the previous steps, and generate the upsampled symbols.

6. *(Pulse Shaping)* Add **MT Generate Filter Coefficients**. Create "pulse shaping filter" terminal by right clicking the "pulse shaping filter" port on the component choosing Create Control. Similarly, create "modulation type" terminal by right clicking and choosing Create Constant. Set it to "PSK". Wire the upsampling factor to "pulse shaping samples per symbol" port. Use "pulse shaping filter coefficients" for the output. Notice that the "filter length" is 8 and the filter parameter (roll-off factor for raised cosine or root raised cosine) is 0.5 by default. The filter length means the desired length, in symbols, of the pulse shaping filter. Use the default values for now.

7. *(Convolution)* Use **Convolution** to convolve two sequences: the upsampled symbols and the pulse shaping filter coefficient. The below figure summarizes the process you've done so far.



8. *(subVI: norm1D)* We will create a subVI that normalizes the filtered symbols by the maximum value of its absolute value. This will ensure the filtered symbols are in between -1 and 1. Create a VI in the project and name it as "norm1D.gvi". Use **Absolute Value** and **Array Max and Min**. Click "Edit Icon" and add the input/output ports. Keep this subVI and we will use it for the future labs.



9. Go back to your transmitter diagram and drag-drop "norm1D" subVI. Use this subVI to normalize the output of the convolution block. Then, wire the normalized samples to **niUSRP Write Tx Data**.

10. *(Power Spectrum and Eye-diagram)* We will measure the power spectrum and the eye-diagram of the outgoing transmitting waveform. Use **Build Waveform** and construct the time-domain waveform from the Tx data. Calculate the sampling period (dt) from the IQ rate (use **Reciprocal** block) and wire to **Build Waveform**. Add **FFT Power Spectrum and PSD** and set the function configuration to "Power Spectrum" and "Continuous". Check "dB On" box in the terminals. Create an indicator by right-clicking the "power spectrum" port of the block. Add **MT Format Eye Diagram** and set the function configuration to "WDT" to accept the waveform data type. Set the "eye length" terminal to 2. Wire the symbol rate to the corresponding terminal. Create an indicator by right-clicking the "eye diagram" port of the block.

11. Save the VI as "BPSKTx.gvi".

3.1. Questions

3.1.1. The PN sequence order for PN bit sequence was 7. What is the period of the PN bit sequence?

3.1.2. Run the code with different pulse shaping filters: None (rectangular) and Root Raised Cosine. Save the spectrum and the eye-diagram of each filter. Compare the bandwidth, the sidelobes and the eye-opening area for each of the filters. (Use the cursors to report quantitative values.)

3.1.3. Set the pulse shape to Root Raised Cosine and change the filter length to 2. Compare with filter length 8 and discuss their spectrum and eye-diagrams.

3.1.4. Choose a pulse shape, i.e. Root Raised Cosine. Change the symbol rate to 100k. Compare their spectrum and eye-diagrams with the symbol rate 10k.

4. Match Filtering

Start by loading “rx_template.gvi”. In this section, we will build a receiver that performs matched filtering. The receiver built in this lab does not perform the complete receiver capability such as decimating, frame synchronization, symbol mapping, and etc. We will resume the rest in the next lab session.

1. Set the default parameter values in the panel:
   - Carrier frequency: 1G
   - IQ rate: 1M
   - Gain: 0
   - Active antenna: RX2

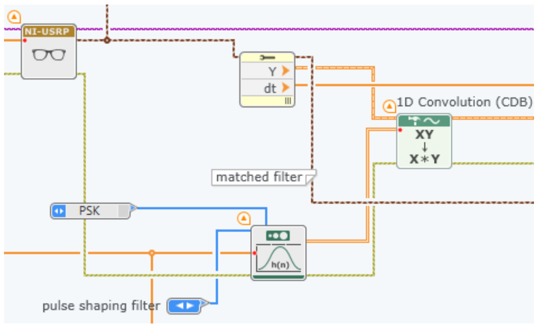
2. Create two Integer-type controls: **Message length**, and **Symbol rate**. Set the message length to 1000 and the symbol rate to 10k. The IQ rate and the symbol rate will determine the downsampling factor. We will upsample the symbols before performing pulse shaping.
   - Message length: 1000 (one frame)
   - Symbol rate: 10k

3. *(Decimating Factor)* The receiver will decimate the samples to obtain the symbols. Calculate the decimating factor similar to the upsampling factor. Note that the two factors do not have to be the same.

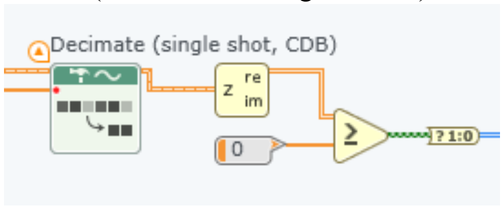
4. We will capture 1 frame of data. Calculate the number of samples to be captured in the receiver (Hint: Use the decimating factor and the message length) and wire to “number of samples” of **niUSRP Fetch Rx Data**. Remove the while loop around the block for a single-shot acquisition. The receiver will stop once it acquires a frame of data. Make sure the **niUSRP Fetch Rx Data** block uses **CDB WDT** type in the function configuration.

5. *(Matched Filtering)* Find **MT Generate Filter Coefficients**. Create “pulse shaping filter” terminal by clicking Create Constant in the Terminals section of the component. Change the constant to Control. Create “modulation type” terminal by clocking Create Constant. Set to “PSK”. Wire the decimating factor found in the previous step to “**matched** samples per symbol”. Use “**matched** filter coefficients” for the output.

6. *(Convolution)* Use **Convolution** to convolve two sequences: the Rx samples and the pulse shaping filter coefficient. Remember the **niUSRP Fetch Rx Data** block outputs **CDB WDT** type and the convolution block takes **CDB array** type, so we need to extract “Y” (CDB Array) from the Rx data. Use Waveform Properties and set it to “Set All to Read”. Wire “Y” to one input port of the convolution and the matched filter coefficient to the other port.



7. (Downsampling & BPSK demodulation) Find **Decimate** block and wire the matched filtered samples to the input array port. Connect the decimating factor. Use **Complex to Real and Imaginary** block to take the real value of the symbols. Recover the bits by returning “0” if the symbol is less than 0 and “1” if the symbol is greater or equal to 0 (use **Greater or Equal?** and **Boolean to Integer** blocks). Display the recovered bits using an array indicator (name it as “unaligned bits”).



8. (Rx Eye-diagram) Take the real values of the matched filtered samples. Similar to Tx, use **Waveform Properties** and **MT Format Eye Diagram** to display the eye-diagram (name it as “unaligned eye”).

9. Run the receiver code while the transmitter is running. Compare the waveform before and after the matched filter. Measure the eye-diagram of the matched filtered waveform.

10. Save the VI as “BPSKRx.gvi”.

4.1. Questions

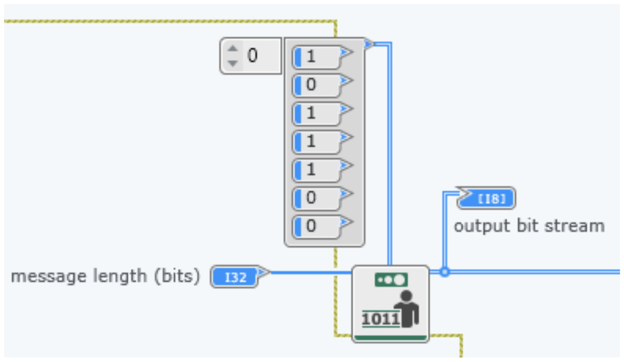
4.1.1. Change the filter parameter of a Root Raised Cosine (both Tx and Rx) to 0.2 and 1. Does the amount of ISI increase or decrease with increasing the filter parameter? Does the bandwidth of received signal increase or decrease with increasing the filter parameter? (Connect a power spectrum block to the Rx samples to measure the spectrum)

4.1.2. Run the receiver a couple of times and observe the eye diagram. Can you see the eye diagrams of each run are aligned? If not, explain why. What problem do you see if the pulse is not aligned?

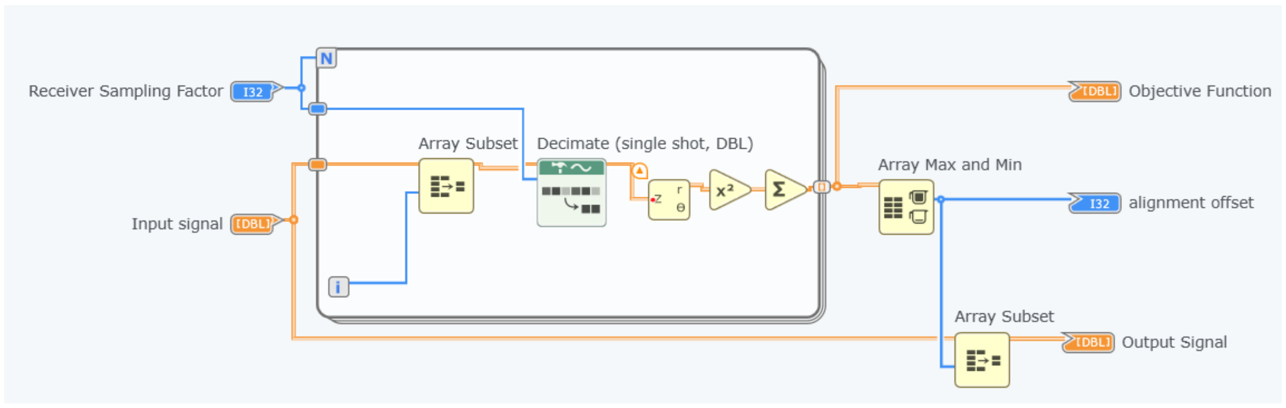
5. Pulse Alignment

1. Duplicate "BPSKTx.gvi" and rename it as "BPSKTx_shortPN.gvi".

2. Modify the **MT Generate Bits** and change its function configuration to "User Defined". Create an array constant for "user base bit pattern". Input the bit sequence "1011100". Create an indicator to see the output bit stream.



3. Implement the following subVI and rename as "PulseAlign.gvi". Click "Edit icon" and create all the inputs and the outputs.



4. Use "PulseAlign.gvi" to align the matched filtered samples and display the eye-diagram (name it as "aligned eye").

5. Demodulate the aligned samples to recover the bits (name it as "aligned bits").

5.1. Questions

5.1.1. Explain how “PulseAlign.gvi” works.

5.1.2. Compare “unaligned eye” and “aligned eye”.

5.1.3. Compare “unaligned bits” and “aligned bits”. Can you detect the transmitted bits “1011100”?

5.1.4. Give a slight frequency offset in Tx, i.e. 1kHz offset by setting carrier frequency as 1.000001G. What can you see in the eye-diagram and the recovered bits? Does the alignment still work?

5.1.5. Save all the VIs and they Project since they will be used in subsequent labs.