

Using a butterworth filter, 4th order design, show the filter spectra on the same graph as the random bit waveform for a bandwidth of 1/16 the total sequence width. Scale the bit sequence spectra so its maximum is unity.

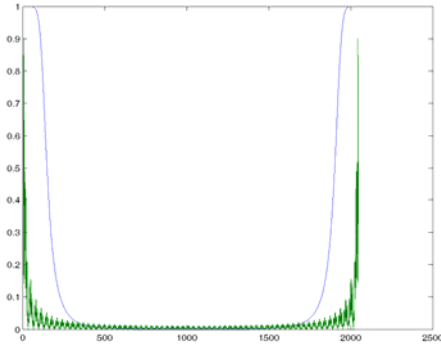


Figure 2.1: Spectra of butterworth filter and random bit waveform.

3. Filtered Bit Sequence

3a. Filter the bit sequence and show spectra.

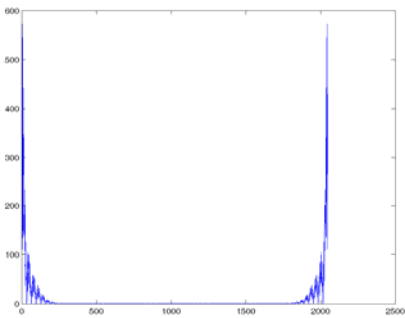


Figure 3.1: Spectra of filtered bit waveform

3b. Show the time domain filtered waveform

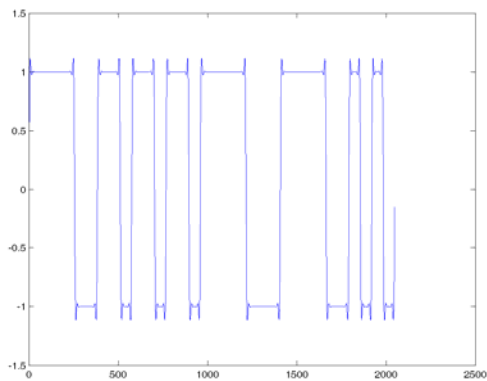


Figure 3.2: Bit waveform after filtering.

4. Recovered Bit Sequence

4a. Assume you know where the center bit location is. Sample these locations and reconstruct the original bit sequence.

4b. Compare with the original.

4c. What cutoff frequency would just start to create errors in your recovered sequence. Show filtered time domain signal, overlaid with the filter impulse response and the original bit sequence.

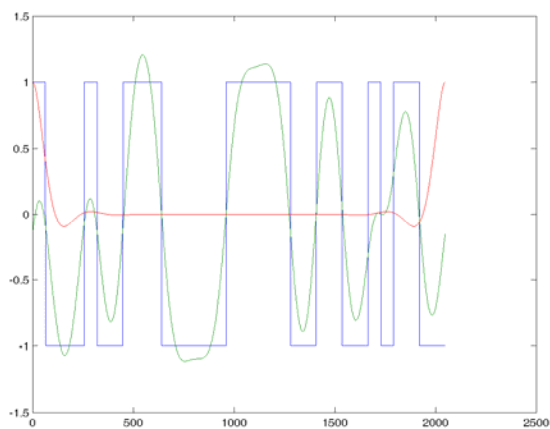


Figure 4.3: Demonstration of ISI for $f_c=N/256$;

APPENDIX: Sample Code

How to generate a Square Wave?

```
% spectra of bit waveform and square wave
bsquare=zeros(1,Nbit); % initialize to all 0 bit values
bsquare(1:2:Nbit)=1; % make every other bit a 1
bsquare=2*bsquare-1; % turn into a bipolar signal
u=ones(1,Nsample); % generate samples per bit
squarewave=kron(bsquare,u); % expand to continuous time approximation
```

How to store a graph as an image file?

```
figure(1); % open active figure window
plot(bwave); % plot waveform
print -dtiff figurefile.tif % store figure window to tiff format
```