Digital Signal Processing: Computing Assignment 2

Silas Waxter

November 21, 2023

1 Introduction

The overarching goal of these computing assignments is to implement a system in which digital information can be transmitted from one location to another via an analog signal. There are multiple communication topologies that can achieve this goal, but this project selects quadrature phase shift keying (QPSK). In this computing assignment, part of a QPSK receiver is implemented. For testing, a large amount of uniformly distributed data is passed to the through the previously implemented transmitter followed by the simulated analog channel; the transmitter manipulates the data into an analog signal capable of traveling through the simulated communication channel. This data is passed to the receiver which is responsible for "sampling" the simulated analog signal and decoding the information back into the symbols. In the final computing assignment, the rest of the receiver will be implemented and the system's overall performance will be evaluated.

This implementation correctly implements all aspects expected in this segment of the computing assignment. Throughout the design and after implementation, multiple verification tests were performed on the system.

2 Design and Calculations

In this segment of the computing assignment projects, very few long-form calculations/derivations were necessary. For this reason, the calculations described in this document are alongside the design aspect in which they are used.

There are three modules that compose this computing assignment: a bandpass filter, a simulated analog to digital converter (ADC), a demodulator, and a down-sampler. This document will describe the system in the order of the data traveling through the system.

The purpose of the bandpass filter is to remove extraneous noise from the signal. Like a real analog signal channel, the simulated analog channel has a noise profile which is imposed on the signal passing through it. To increase the signal to noise ratio, the receiver eliminates extraneous noise that occurs outside of the signal's bandwidth. The signal bandwidth is $\pm\frac{3\pi}{8}$ at 0.44 since the unmodulated signal bandwidth is $\pm\frac{3\pi}{8}$ and the modulation frequency is 0.44. So, the reciever uses an FIR bandpass filter. Like all filters in this assignment, the filter is designed with a custom window-method filter designing function. This method is parametric such that the number of samples, lower cutoff frequency, upper cutoff frequency, and transition width can be specified. The method does not produce the optimal filter, but the implementation is straight forward. The window function, the Kaiser Window, is a parametric function which can be shaped using the filter designing method parameters. The implementation references the SciPi implementation and the Wikipedia articles on the Kaiser Window and FIR window-method design. For specific details see `./fir_window_designer.py::FIRWindowMethod(...)`.

In a real system an ADC would be used to sample the analog signal; however since the analog channel is simulated, the receiver's ADC must adapt accordingly. The analog channel is simulated by up sampling the signal by a factor of 20–i.e. inserting zeros and interpolating. Functionally an ADC can be thought of as a downsampler. An analog signal is really a discrete signal with an infinitely small sampling interval. A real ADC has some defined sampling rate which determines how many samples of the analog signal should be skipped. In this assignment, the simulated ADC will down-sample the signal by a factor of 10. Note that with this factor, the signal will have a symbol rate 8 times the systems input rate.

The next step is to demodulate the signal. The signal is separated into two signals: a signal multiplied by $cos(w_1 n)$ and a signal multiplied by $sin(w_1 n)$. The demodulation frequency, w_1 is equal to the center frequency of the signal, $\frac{3\pi}{8*2}$. When multiplying the signal by a sinusoid, two "images" are created. Only one "image" is desired, so a lowpass filter is constructed using the previously described window-method function such that only the lower "image" remains.

The last step is to adjust the data rate. Thus far, the data rate is 8 times faster than the bit rate. The goal of this step is to match the data rate following the pulse shaping in the transmitter; the data rate at this point is 4 times faster than the bit rate. So, each of the demodulated signals are downsampled by a factor of 2.

3 Experimental Techniques and Results

This section will describe how verification tests were performed.

The up-sampler from the previous computing assignment was modified such that the sampler can down-sample or up-sample by an integer factor. The correctness was verified by examining how a test input signal was modified. See the following from `python factor_sampler.py`.

```
Test Input:
[[-1  1]
 [ 1 -1]
 [ 1  1]
 [ 1 -1]
 [ 1 -1]
 [ 1  1]
 [ 1 -1]
 [ 1  1]
 [ 1 -1]
 [ 1  1]]
Upsampled-by-4 Output:
[[-1.  1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1. -1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1.  1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1. -1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1. -1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1.  1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1. -1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1.  1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]
 [ 1. -1.]
 [ 0.  0.]
```

```
 [ 0.  0.]
 [ 0.  0.]
 [ 1.  1.]
 [ 0.  0.]
 [ 0.  0.]
 [ 0.  0.]]
Test Input:
[[  -1    1]
 [2989 2989]
 [   1    1]
 [2989 2989]
 [   1   -1]
 [2989 2989]
 [   1   -1]
 [2989 2989]
 [   1   -1]
 [2989 2989]]
Downsampled-by-2 Output:
[[-1.  1.]
 [ 1.  1.]
 [ 1. -1.]
 [ 1. -1.]
 [ 1. -1.]]
```

The FIR window-method parametric designing function was implemented while produced specifically for bandpass and lowpass filters. The receiver's usage were test cases for the module.

The bandpass filter at the input was verified by by the plot shown in figure 1.

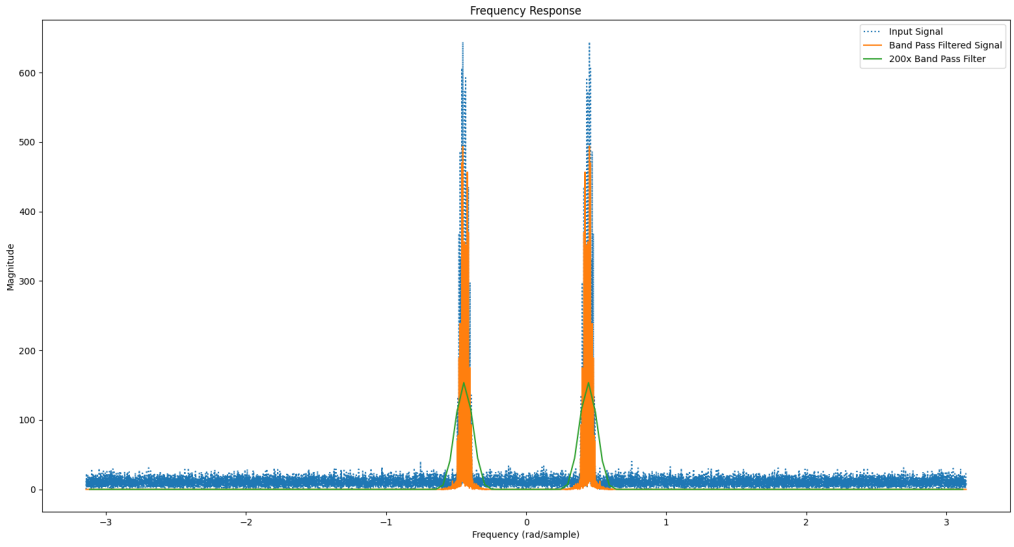

Figure 1: Test Data Input to Receiver, Showing with and without Bandpass Filter

The simulated ADC was verified by plotting. Figure 2 shows the plot.

The demodulation was verified by plotting as shown in figure 3.

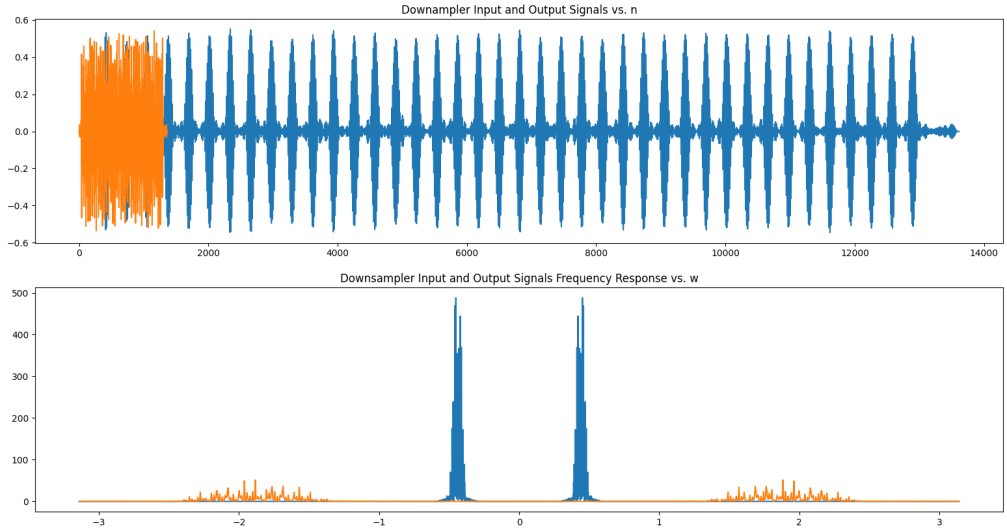

Figure 2: Simulated ADC, Effects on Signal and Frequency

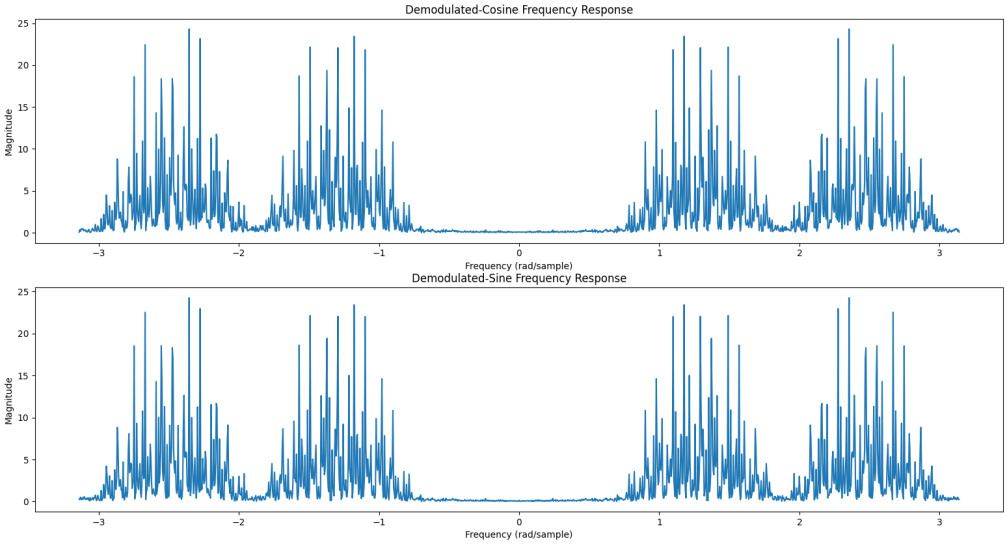

Figure 3: Demodulation Effect on Frequency Response

Correct filtering out of the "image" replica at the higher frequency was verified by plotting the output of the demodulated signal through the lowpass filter. Its shown in figure 4.

Finally, the down-sampler was verified by looking at the receiver's output which is sequential composition of the previous processing actions. Notice that the shape of the frequency response approximately matches the raised root cosine filter used in the transmitter to produce the pulses. The plot is shown in figure 5.

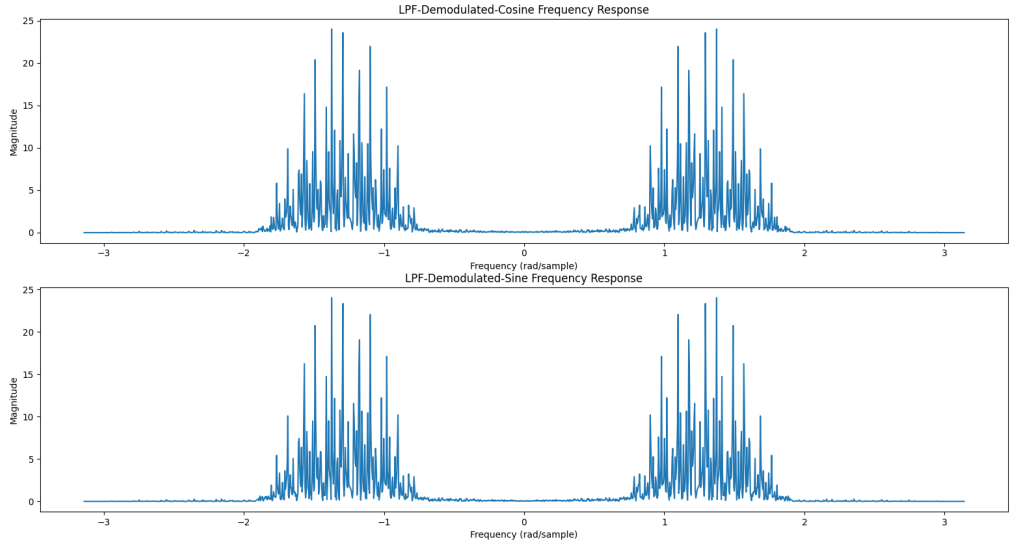

Figure 4: Lowpass Filtered Demodulated Signals Frequency Response

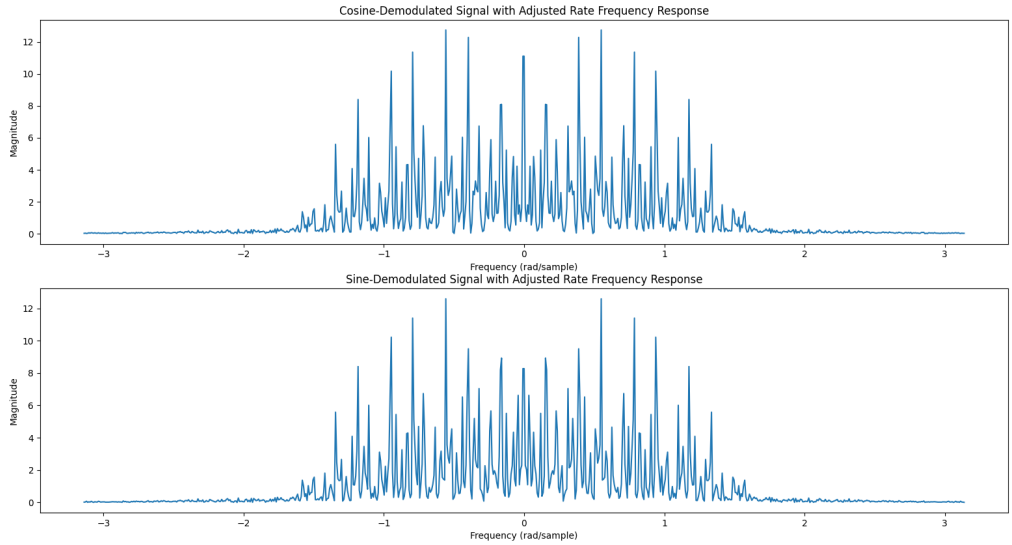

Figure 5: Computing Assignment 2 Output

4 Conclusion

In this computing assignment part of a Quadrature Phase Shift Keying (QPSK) communication topology was implemented. Specifically, the part of the reciever was created. To decode signals produced by the previously implemented transmitter and simulated analog channel.

I learned many things while working on this computing assignment. Implementing something forces you

to understand it well. I feel much more confident about how the window-method works for designing an FIR filter. Essentially, an ideal filter is a rectangular function in the frequency domain and a sinc function in the real domain. Since we want finite length in the real domain, we "truncate" the sinc function in the real domain by multiplying it with a rectangle function itself. The term "truncate" is not the best description since the shape of the "truncation-window" can be varied to apply disciminative scaling to values of the sinc function and modify the frequency response of the filter. Multiplication in the real domain implies convolution in the frequency domain. This is the basis of the window method. Furthermore, after formally learning about the DFT in class, working with the FFT is much more clear and useful.

I have some doubt that I have the exactly correct results. This makes sense; pausing implementation half-way through a black box where one is not intimately familiar with the inter working is likely to produce this situation. However, I'm confident that the results are mostly correct; furthermore, if the results are incorrect, the issues will likely be identified in the next computing assignment. For example, in this computing assignment, I found that my anti-aliasing filter in the transmitter passed 3 "images" instead of 1. This bug took a short amount of time to catch. It will be clear at the end of the next computing assignment if the overall system works or not: do bits flow in and the same bits flow out? The code is not up to par with the standards I usually write. However, the concepts are technically challenging and since no-one is going to read or run the code after this assignment is completed. Its more important for it to work than for it to be perfectly clean and organized.