# ECE 464 Computing Project 3

**Thomas Snyder**

November 2020

**Abstract**

This report summarizes the implementation of a digital communication system from sequence generation to decoding and sequence recovery. Two prior works have created the transmission of the signal and the bulk of the receiver. The third installment builds on previous work to implement the final stages of the receiver. All remaining parts of the receiver are detailed, and the final simulation is thoroughly tested with multiple signal-to-noise ratios. Experimental performance verification shows that this quadrature phase-shift keying digital communication simulation performs well up to the point where the variance of the zero-mean Gaussian noise that is added in the communication channel is equal to the variance of the transmitted signal. As the signal-to-noise ratio decreases on the Decibel scale, the number of incorrect symbols increases in a cubic manner.

## Contents

## List of Figures

## List of Tables

## 1 Introduction

This project is the third installment of a series of projects that culminate in a complete simulation of a digital communications channel, from sequence generation, pulse shaping, and modulation to demodulation, matched filtering, and quantization (symbol detection) of the recovered noisy sequence. The final portion implements three stages: an equalizer, a matched filter, and a quantizer. Important aspects in this installment is the implementation of optimized algorithms for finite impule response (FIR) filtering and the discrete Fourier transform (DFT). This report will summarize the implementation details and design decisions when creating the three blocks: equalizer, matched filter, and quantizer. Also included is an evaluation of the performance of the demodulation and signal recovery process through varying noise floors added in the communication channel.

## 2 Methods

Implementing each stage in the final installment of the series required implementation of more than just the three stated system blocks; custom optimized algorithms for the convolution and fast Fourier transform (FFT) algorithms were also among this project's deliverables. This section will cover each of the parts, and discuss some omissions from the original project guidelines due to anticipated effects from the FIR filtering in previous iterations of the project. Among the omissions is the first block: the equalizer.

**Equalizer** Implementation of an equalizer was omitted due to the ideal conditions of the simulated channel. In practice, the channel will contain some memory, which acts as a filter, attenuating or amplifying some frequencies within the transmitted signal. It is important to characterize the impulse response of the channel to accurately equalize the signal after reception. Characterization may be performed by transmitting an impulse response function as the input signal and observing the received spectrum. Due to the flat frequency response of the impulse response, any distortion added by the channel will be plain, and an equalizer may be obtained by mirroring the results around the 0 dB point. After equalization, the demodulated signals need to be filtered to recreate the sequence from the pulse-shaped signal.

**Matched Filter** The matched filter is a time-reversed version of the pulse-shaping filter used on the upsampled version of the original sequence. Recall that the pulse-shaping was performed to ensure that each symbol in the sequence spread across four samples instead of one. The results of the matched filter will be large positive and negative pulses when the filter aligns with previously pulse-shaped samples. For this system, the matched filter is given by the truncated and time-reversed square-root-raised cosine filter

$$h_{mf} = \begin{cases} h_{rc}[31-n] & ;\ n \in [0, 31] \\ 0 & ;\ \text{Otherwise.} \end{cases} \tag{1}$$

The matched filter is convolved with both demodulated signals to recreate the original upsampled sequence. This convolution is done using a custom overlap-save fir filtering algorithm.

**Overlap-Save FIR Filtering** Two efficient algorithms exist for implementing FIR filters: overlap-add and overlap-save. In this project, the overlap-save method was utilized for fast filtering when the full signal may not be available at the same instant of time. The overlap-save method works with the principle that the $N$-point DFTs perform $N$-point circular convolution of two signals.

To implement the algorithm, the signal $x[n]$ is divided into blocks of length $N$. Denote the length of the FIR filter as $M$ and the length of the signal that will be sampled $L$. In this implementation, $N = M + L - 1$ because linear convolution of two signals of length $L, M$ produces a final signal of length $N = M + L - 1$. The length of the FIR filter $M$ was constrained to be a power of two due to the constraint imposed by the FFT implementation that all input signals must have length a power of two:

$$h[n] = \begin{cases} \text{FIR coefficients} & ;\ n \in [0, 2^n) \\ 0 & ;\ \text{Otherwise.} \end{cases} \tag{2}$$

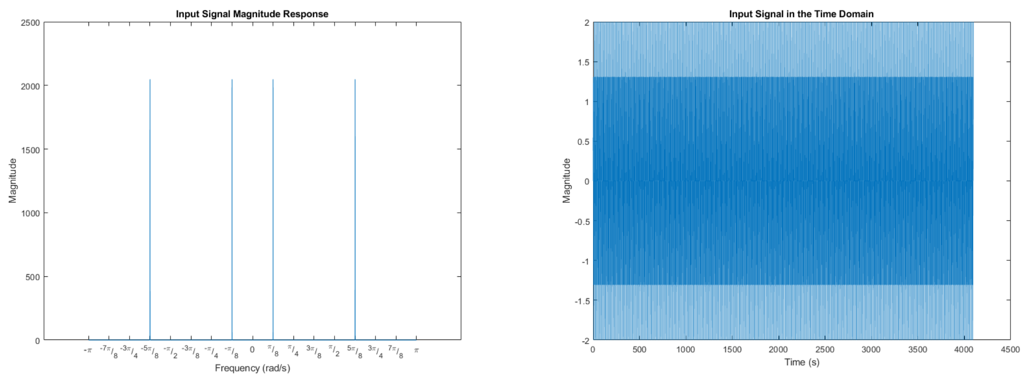

Figure 1: Input Test Signal to the Overlap-Save FIR Filter Algorithm

The DFT of the signal will be an $N$-point DFT; thus, $N$ must also be a power of two. Constraining $M$ to be a power of two means that if $N$ is a two-multiple of $M$ (i.e. $N = 2M, 4M, 8M$ etc.), then $N$ must also be a power of two. With this knowledge, set $N = 2M$. Then $L = M + 1$. So, for every iteration, the $N$-point DFT will be taken of an $N$-length block of $x[n]$. But which samples should be included in each block?

The overlap means that $M$ samples will overlap from the previous block, and the remaining $L+1$ samples will be new samples in the new block:

$$x_i[n] = \begin{cases} x[n] & ;\ n = ((i-1) \times L) - M, ..., (i \times L) - 1 \\ 0 & ;\ \text{Otherwise.} \end{cases} \tag{3}$$

and for the first iteration:

$$x_0[n] = \begin{cases} 0 & ;\ n = ((i-1) \times L) - M, ..., -1 \\ x[n] & ;\ n = 0, ..., L-1 \end{cases} \tag{4}$$

Then, performing the N-point DFT of both the FIR filter and the block of samples from the signal and multiplying them together produces the DFT of the output block of $y_i[n]$:

$$Y_i(k) = X_i(K)H(K). \tag{5}$$

Taking the inverse DFT of the block $Y_i(k)$ and removing the incorrect samples that represent the transient response of the filter produces the final block, which may be appended to the final signal:

$$y[n + L \times i] = y_i[n + M] \tag{6}$$

To test the implementation of the overlap-save algorithm, an input signal of two sinusoids was generated

$$x[n] = \cos\left(\frac{5\pi}{8}n\right) + \cos\left(\frac{\pi}{8}n\right)\ ; \qquad n = 1, 2, 3, 4, ..., 4096. \tag{7}$$

This signal is shown in Figure 1, where it may be observed that there are two reflected impulse functions in the frequency domain at $\pm 5\pi/8$ and $\pm\pi/8$. A low-pass linear phase FIR filter with 512 coefficients was generated with cutoff frequency $\omega_c = 3\pi/8$ such that the output signal should have the sinusoidal signal with frequency $5\pi/8$ completely filtered out. The results of the filtering are shown in Figure 2. These results reflect the expected results. The high frequency impulses are no longer present and the time-domain signal is shifted by 256 samples, which is the expected result for a linear-phase FIR filter. To achieve these results, a custom FFT algorithm implementation was used to perform the $N$-point DFTs.

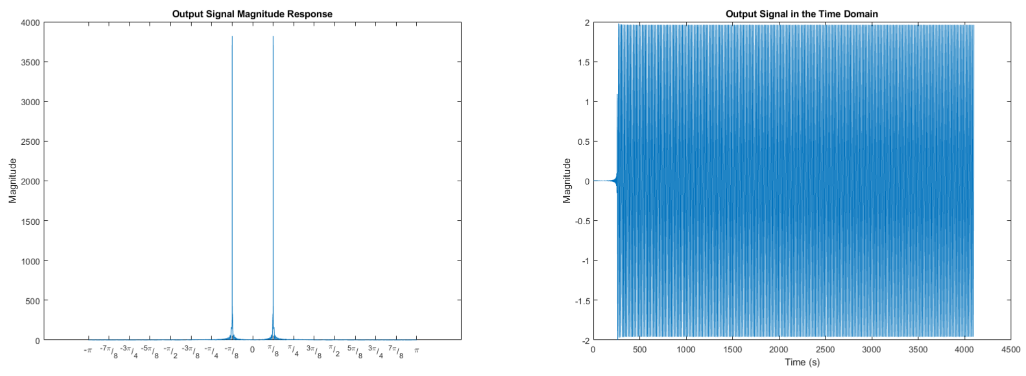

Figure 2: Output Test Signal of the Overlap-Save FIR Filter Algorithm

**Decimation-in-Frequency FFT** Two well-known ways of breaking the DFT problem into lower-complexity problems are decimation-in-time and decimation-in-frequency. This approach uses the decimation in frequency algorithm, which is descriped below.

Let $x[n]$ have a discrete Fourier transform $X(K)$. This was defined

$$X(k) = \sum_{n=0}^{N-1} x[n]e^{-j\frac{2\pi}{N}kn} \qquad [N\text{-point DFT}]. \tag{8}$$

Break $X(k)$ into its even and odd samples, denote $X_e(k), X_o(k)$ respectively. Then, $X_e(k)$ is expressed:

$$\begin{aligned} X_e(k) &= X(2r) \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j\frac{2\pi}{N}2rn} + \sum_{n=\frac{N}{2}}^{N-1} x[n]e^{-j\frac{2\pi}{N}2rn} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j\frac{2\pi}{\frac{N}{2}}rn} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]e^{-j\frac{2\pi}{\frac{N}{2}}r(n+\frac{N}{2})} \\ &= \sum_{n=0}^{N/2-1} (x[n]+x[n+\frac{N}{2}])e^{-j\frac{2\pi}{\frac{N}{2}}rn}e^{-j2\pi r} \\ X(2r) &= \sum_{n=0}^{N/2-1} (x[n]+x[n+\frac{N}{2}])e^{-j\frac{2\pi}{\frac{N}{2}}rn} \end{aligned} \tag{9}$$

where $r = 0, 1, ..., \frac{N}{2} - 1$ and $e^{-j2\pi r} = 1 \; \forall r$. $X_o(k)$ is expressed similarly:

$$\begin{aligned}
X_o(k) &= X(2r+1) \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j\frac{2\pi}{N}(2r+1)n} + \sum_{n=\frac{N}{2}}^{N-1} x[n]e^{-j\frac{2\pi}{N}(2r+1)n} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j\frac{2\pi}{N}(2r+1)n} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]e^{-j\frac{2\pi}{N}(2r+1)(n+\frac{N}{2})} \\
&= \sum_{n=0}^{\frac{N}{2}-1} x[n]e^{-j\frac{2\pi}{N}(2r+1)n} - \sum_{n=0}^{\frac{N}{2}-1} x[n+\frac{N}{2}]e^{-j\frac{2\pi}{N}(2r+1)n)} \\
X(2r+1) &= \sum_{n=0}^{\frac{N}{2}-1} (x[n] - x[n+\frac{N}{2}])e^{-j\frac{2\pi}{N}(2r+1)n}
\end{aligned} \tag{10}$$

where $r = 0, 1, ..., \frac{N}{2} - 1$ and $e^{-j\pi(2r+1)} = -1 \; \forall r$. Using the fact that the 2-point DFT may be computed directly as

$$\begin{aligned}
U(k) &= \sum_{n=0}^{1} u[n]e^{-j\frac{2\pi}{N}kn} \\
U(0) &= u[0] + u[1] \\
U(1) &= u[0] - u[1],
\end{aligned} \tag{11}$$

the $N$-point FFT was implemented as a recursive function, taking the FFT of the even and odd sequences of $X(k)$ until the end condition of $N = 2$ at which point the direct DFT was computed as in Equation 11. The returned values have index that is the bit-reversed order of their time-domain index (i.e. index $12_{10} = 1100_2$ is index $0011_2 = 3_{10}$). This is used to descramble the output into the final sequence, $X(k)$. The tests that confirm the implementation correctness can also be seen in Figure 1 and Figure 2. The input signal with defined frequencies is clearly observed in the plots of the FFT that were computed with the aforementioned FFT algorithm. The inverse FFT that is used in the overlap-save method of convolution makes use of the FFT algorithm described.

**Inverse FFT from Decimation-in-Frequency FFT** The inverse FFT takes advantage of the above decmiation-in-frequency FFT described above. To take advantage of the FFT, note the following property from the definition of the complex conjugate of the inverse DFT:

$$x^*[n] = \frac{1}{N} \sum_{k=0}^{N-1} X^*(k)e^{-j\frac{2\pi}{N}kn}. \tag{12}$$

Note that the value within the summation is the DFT of $X^*(k)$. This means that one may take the following steps to perform the inverse DFT of a signal:

1. Take the complex conjugate of $X(k)$: $X^*(k)$
2. Take the DFT of $X^*(k)$
3. Multiply the result by $\frac{1}{N}$
4. Take the complex conjugate of the scaled signal to produce $x[n]$

The above algorithm is implemented in this work to produce the inverse DFT of a signal. Getting back to the simulation, after convolution, the signals go to delay estimation.

**Delay Estimation** This implementation was also omitted due to the fact that every FIR filter in this work had linear phase and it was known that a shift of $\frac{M}{2}$ samples occured (for an $M$-point FIR filter), which was compensated at every filter. The channel was also assumed to have no delay effects on the signal; thus, all delay was addressed as it was introduced. However, in real-world systems, this may not be enough. To estimate the delay caused by the channel, one may take the original signal $x_1[n]$ and the signal received from the channel $x_2[n]$ and compute the correlation, shifting $x_1[n]$ down a sample each time. When the correlation is at its maximum, number of shifts that have occurred to $x_1[n]$ is the delay introduced by the channel. Once the delay is compensated for, the signals may be subsampled and quantized.

**Subsampling** Due to the fact that the signal is known to have a sample rate of four samples per symbol, every fourth sample was taken. A low-pass filter was omitted because all four samples of the signal represent the original symbol, thus any of the four samples would produce the desired results, so $b[n] = b_4[4n]$. After subsampling, the signals were quantized and their performance was tested.

**Symbol Detection (Quantization)** Symbol detection was trivial. If the sample was greater than zero, it was considered a "1," but if it was less than zero, it was considered a "$-1$."

**Performance Evaluation** To evaluate the performance of the system, a signal of 1000 symbols was generated. The performance was evaluated based on the number of symbol errors per 1000 symbols. Because a symbol was composed of two bits, it was necessary that both bits were correct for a correct symbol. To stress the system, noise was introduced in the channel portion of the simulation. The noise was be measured in terms of the signal-to-noise ratio (SNR). SNR was defined as the ratio of the variance of the signal to the variance of the noise introduced by the channel. Three SNR values were tested: $10, 5,$ and $0\,\text{dB}$. The SNR ratio is given

$$\text{SNR} = 10 \log \frac{\sigma_{\text{input}}}{\sigma_{\text{noise}}} \tag{13}$$

Using Equation 13, the variance of the additive noise of the signal was computed:

$$\begin{aligned} \text{SNR} &= 10 \log \frac{\sigma_{\text{signal}}}{\sigma_{\text{noise}}} \\ 10 \log \sigma_{\text{noise}} &= 10 \log \sigma_{\text{noise}} - \text{SNR} \\ \sigma_{\text{noise}} &= 10^{\log \sigma_{\text{noise}} - \text{SNR}} \end{aligned} \tag{14}$$

Equation 14 was used to compute the variance of the noise added to the signal.

## 3 Results and Discussion

This section discusses the experimental performance evaluation of the system. Seven SNRs were used, each run with a signal of 1,000 samples, 10,000 samples, and 100,000 samples in length. These trials were normalized to represent the number of incorrect samples for 1,000 sample length signal. The rates for the symbol signal as well as the individual bit signals were recorded. Complete results are shown in Table 1. Normalized and averaged results are shown in Table 2.

Figure 3 plots the data displayed in Table 2. It is clear that the number of incorrect samples per 1,000 symbols is not a linear relationship, but it is less than a logarithmic (base 10) relationship. The data shows that as the signal to noise ratio decreases by factors of two, the performance of the system is affected more and more. For ratios where the signal variance is greater than or equal to the noise variance (SNR $\geq 0$), the system performs well, with no issues decoding and recovering the original bit sequence. When the noise variance becomes larger than the signal variance is when performance degradation begins to occur. An interesting note to make is that the number of incorrect symbols is not equal to the sum of the number of incorrect bits in $b_1[n]$ and $b_2[n]$. This is expected due to the fact that for the symbols to be correct, either the sample in $b_1[n]$ or $b_2[n]$ or both may be corrupt. This means that sometimes both samples will be corrupt and that a single symbol may account for an incorrect bit in both $b_1[n]$ and $b_2[n]$. This occurrence followed a trend of decreasing at around $-2\,\text{dB}$ SNR and then greatly increasing in probability as the variance of the

| SNR (dB) | Number of Incorrect Samples | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Per 1,000 Samples | | | Per 10,000 Samples | | | Per 100,000 Samples | | |
| | Symbols | $b_1[n]$ | $b_2[n]$ | Symbols | $b_1[n]$ | $b_2[n]$ | Symbols | $b_1[n]$ | $b_2[n]$ |
| -10 | 396 | 283 | 255 | 3861 | 2588 | 2643 | 38957 | 26237 | 26338 |
| -5 | 55 | 37 | 20 | 564 | 288 | 290 | 5847 | 2988 | 2993 |
| -3 | 8 | 5 | 3 | 49 | 18 | 31 | 649 | 295 | 354 |
| -2 | 1 | 0 | 1 | 7 | 1 | 6 | 99 | 44 | 55 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Table 1: Signal to Noise versus Incorrect Samples

| SNR (dB) | Symbols Average Per 1,000 | $b_1[n]$ Average Per 1,000 | $b_2[n]$ Average Per 1,000 |
|---|---|---|---|
| -10 | 390.56 | 268.06 | 260.89 |
| -5 | 56.62 | 31.627 | 26.31 |
| -3 | 6.46 | 3.25 | 3.21 |
| -2 | 0.89667 | 0.48 | 0.717 |
| 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 |
| 10 | 0 | 0 | 0 |

Table 2: Signal to Noise versus Incorrect Samples Average Results

noise grew larger than the variance of the signal. $b_1[n]$ and $b_2[n]$ had similar performance overall, which is also expected due to their generation as zero-mean Gaussian noise in each trial.

## 4 Conclusion

This paper summarized the final installment of a simulated digital communication channel using quadrature phase-shift keying (QPSK). The objective of the project was to implement knowledge from digital signal processing and previous signals and systems to realize the applications of the technologies. This third part of the project was most challenging in the sense that every algorithm needed a custom implementation. This included the overlap-save FIR filter implementation and the fast Fourier transform implementation. Additionally, the matched filter was a new concept for decoding a demodulated signal that was pulse-shaped with a similar, but time-reversed filter. Ultimately, the implementation of each of the stages of the transmitter and receiver was highly successful.

In testing, the system performed well in test conditions where the variance of the noise was not larger than the variance of the signal (SNR $> 0\,\text{dB}$). Once the variance of the noise became larger than that of the signal, system performance quickly degraded in a cubic manner. This is to be expected because the larger the noise variance, the more that the signal will degrade, making the inherent ability to recover the original signal difficult. Additionally, it was noticed that the ratio between the number of incorrect bits in $b_1[n]$ or $b_2[n]$ to the number of incorrect signals in the final symbol sequence decreased drastically as the SNR decreased. This observation is due to the fact that as more bits in $b_1[n]$ and $b_2[n]$ were incorrect, the likelihood that both signals would have an incorrect bit at the same time step increased; thus, the number of symbols is less likely to be twice the number of incorrect bits in $b_1[n]$ or $b_2[n]$ as the SNR decreases. These results are promising. The QPSK transmission and reception scheme is proven to work in simulation and it is exciting to think of the applications in the real-world, such as wireless communication. Some improvement may still need to occur.

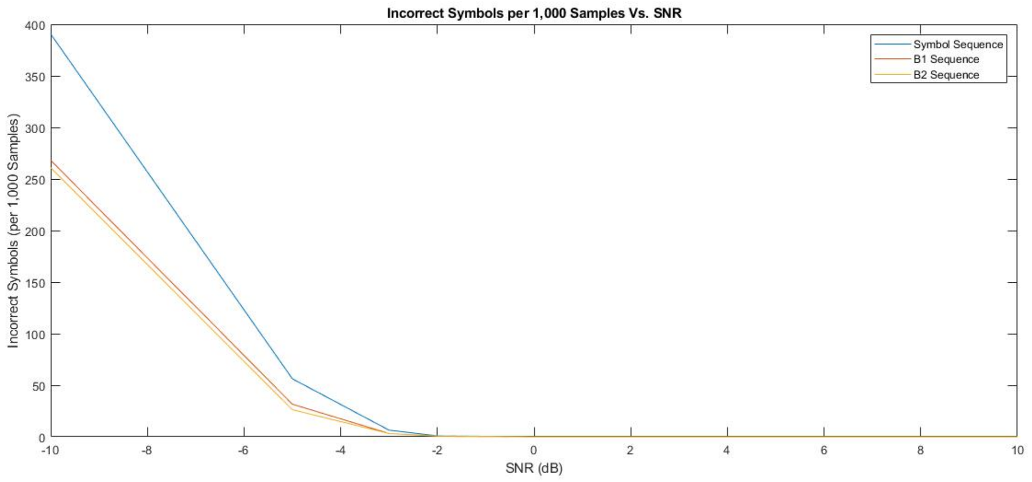

Figure 3: Average Incorrect Samples per 1,000 Sample Signal Length

The FFT algorithm, and by extension, the overlap-save FIR filtering algorithm, both require the length of the filter or the signal on which it operates to be some power of two. While this strict constraint maintains a computationally efficient algorithm, it is somewhat inconvenient when taking the Fourier transform of a signal whose length should be kept constant at some value that is not a power of two. Thus, a future work will be to implement an FFT algorithm that does not require that the lenght of the FFT be a power of two. Additionally, to model real-world effects on the signal in the channel and to exercise the process of creating an equalizer, some memory representative of a typical channel medium will be added in the channel block of the simulation.