Simulation and Analysis of Digital Communications System

Craig Carlson, Brian Hulette, Ruth Stoehr, ECE 5654

May 1, 2013

Abstract

A digital transceiver system is presented that has been optimized for performance in AWGN channels. This paper provides an overview of the algorithms used in both the transmitter and receiver, to include packetization, Viterbi decoding, QPSK/16QAM modulation, and synchronization techniques. A detailed performance analysis shows that the system achieves performance that is consistent with theoretical results in AWGN channels, and is able to operate in the presence of moderate time, frequency, and phase shifts introduced by the channel. The receiver is also able to provide sufficient performance against flat Rayleigh fading channels for QPSK modulation. A few outstanding issues with the system still exist, including more robust amplitude estimation for 16QAM modulation in the presence of Rayleigh fading. Implementation choices that were made while designing the system are discussed throughout the report.

Contents

1 Introduction

The purpose of this report is to describe the implementation of a simulated digital communication system, and to prove the effectiveness of that system with an in-depth performance analysis. In this paper, we discuss our system design in detail as well as show plots that verify the system's efficiency in AWGN and flat Rayleigh fading channels, and in the presence of varying time, frequency, and phase offsets.

Section 1 discusses the system requirements, system design, and the project responsibilities for each team member. Next, Section 2 describes the algorithms and implemenations used for packetization, coding, signal detection, and syncronization. In Section 3, performance and limitations of the system in both AWGN and flat Rayleigh fading channels are discussed. Finally, in Section 4, we conclude with a summary of our results and findings, as well as suggestions for improvement.

1.1 System Requirements

This system must be written in MATLAB and it must incorporate the following major components of a digital communications system.

Digital Transceiver	
Transmitter	Take in a vector of bits of unknown length a priori Encode the bits using the error correction code of choice Convert the encoded bits to symbols at complex baseband Apply pulse shaping Output complex baseband samples
Receiver	Time Synchronization Frequency Synchronization Phase Synchronization Equalization Demodulation (symbol demapping) Decoding

The MyTransmitter function must accept a vector of bits and output a series of complex samples with unit power at a sample rate of 10 Msps with at least 4 samples per symbol. The MyReceiver function must accept complex samples at 10 Msps and output bits.

1.2 System Overview

Both the transmitter and receiver in the digital communication structure are constructed with several different modules that are linked through the data flow. The final system employs QPSK modulation, though it is easily configurable to use 16QAM modulation in cases where bandwidth efficiency is desired. While the details of individual module implementations will be given in Section 2, an overview of the transmitter and receiver systems is presented here.

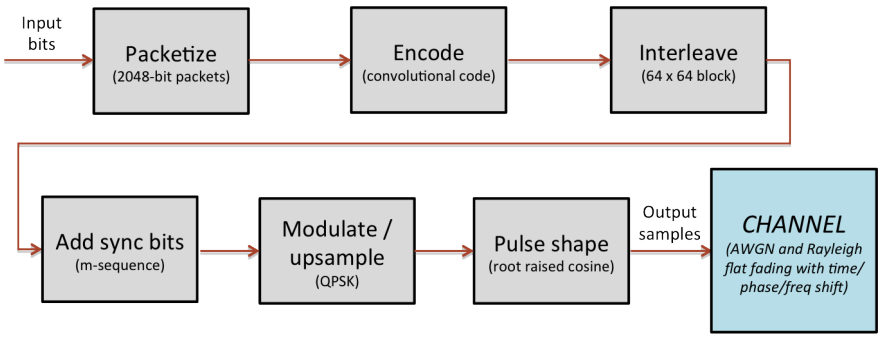


Figure 1: Digital Transmitter Block Diagram

A block diagram of the QPSK transmitter is depicted in Figure 1. When bits are input into the system, they are first packetized into packets of length 2048 bits. The sequence of packets is then convolutionally encoded with a [15, 17] code (constraint length of 4) and interleaved with a 64 x 64 block interleaver. Next, 62 synchronization bits are prepended to each packet, where these bits are generated from a maximum likelihood sequence in order to maximize correlation properties at the receiver. The bits are then QPSK modulated and upsampled by a factor of 4, and pulse shaped with a root raised cosine filter with a roll-off factor of 0.25. The resulting samples are transmitted over the channel at the implied sample rate of 10 Msps.

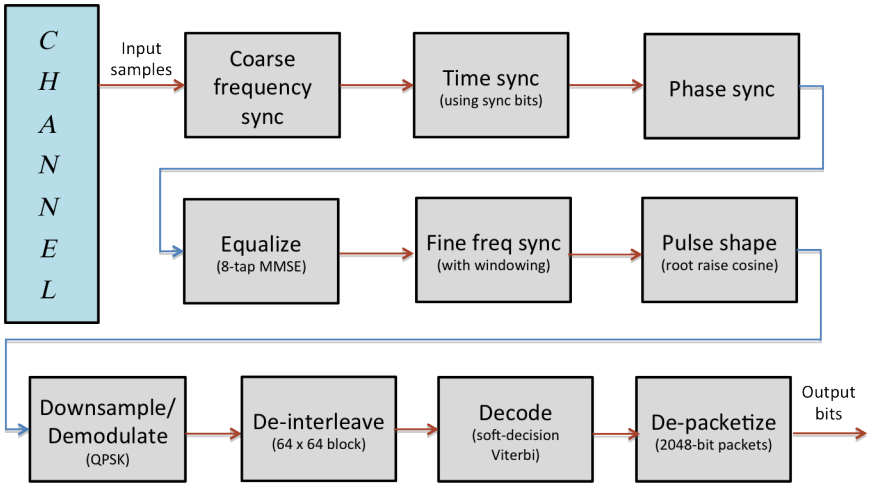


Figure 2: Digital Receiver Block Diagram

The corresponding QPSK receiver is shown in Figure 2. The received samples are first sent through a coarse frequency synchronization block. The sync bits are then used to synchronize the signal in time and phase. At this point, it is assumed that our receiver is coherent in time and phase, and processing continues with equalization (i.e. amplitude estimation). Then the samples are passed to a fine frequency synchronization block which

performs phase synchronization over time, followed by pulse shaping and demodulating. Soft output bits from the demodulator are deinterleaved, then passed to a soft-decision Viterbi decoder, and the bits are finally de-packetized, producing an output stream of bits.

The code has been implemented and optimized for transmission over AWGN and flat Rayleigh channels. The simulated channels used in testing will be described further in Section 3.1.

Note that these diagrams represent the system using QPSK modulation. The code has been written such that a simple configuration parameter can be changed to utilize 16QAM modulation for band-limited scenarios. In that case, the systems would be nearly identical, though the pilot sequences would be 124 bits long.

1.3 Project Responsibilities

Due to the complexity of the system our team broke it down into distinct modules, and different team members were responsible for each module. The table below lists which team member was responsible for each component of the transceiver. The team worked together on the project presentation and paper, while Ruth and Brian performed most of the system integration.

Team member	**Responsibilities**
Craig Carlson	Modulation/demodulation, Equalization
Brian Hulette	Synchronization bits, pulse shaping, time/frequecy/phase synchronization
Ruth Stoehr	Packets, encoding/decoding, interleaving, AWGN/Rayleigh channel models

2 Algorithms and Implementation

This section provides detailed descriptions of the algorithms used in the digital transmitter and receiver system. In particular, implementation details are given for the packet construction, soft-decision Viterbi decoder, demodulation and detection (to include equalization), as well as the various synchronization techniques.

2.1 Packetization

As stated in the overview, the first component in the transmitter is a packetizer, that partitions the incoming bits into packets of 2048 bits. Each packet consists of a 16-bit header, followed by 2032 bits of data and (optional) padding, as shown in 3. The first 13 bits of the header denote the size of the data contained in the packet, while the remaining 3 bits are all parity check bits on the 13 data size bits. These three bits are identical in transmission (and will be identical at reception in the absence of errors).

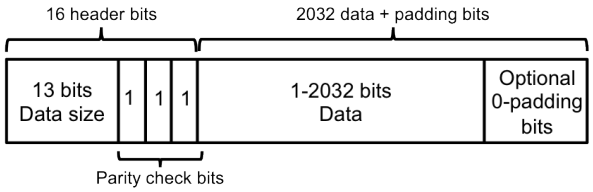


Figure 3: 2048-bit packet structure

After the entire bitstream is encoded, pilot bits are added to the beginning of each packet to facilitate synchronization. These are inserted every $2048/r$ bits in the encoded bitstream (where r is the coding rate), so they correspond to the beginning of each packet in the unencoded bitstream. These bits are decribed in more detail in Section 2.6.1.

It should be noted that the size of 2048 was chosen to minimize overhead, while maximizing performance achieved through synchronization. A shorter packet would allow for better synchronization and equalization, because the pilot bits would be added at smaller intervals, but there would be a loss in bandwidth efficiency due to a larger percentage of overhead bits.

In the depacketization, the header is first read to determine the number of data bits in the packet. If the data size is determined to be greater than 2032 or at least two of the parity check bits are incorrect, then a potential error is detected in the header. In this case, the receiver produces a warning message and assumes the packet has 2032 data bits (which could lead to extra bits in the output if padding is present in this packet). Note that if only one of the parity check bits are incorrect, it is assumed that that bit is in error and the data size of the header is correct. While this packetization structure is not optimal and some header errors will go undetected, it will catch most header errors and prevent a system crash.

2.2 Convolutional coding with Viterbi decoding

In order to optimize channel capacity, convolutional encoding with soft-decision Viterbi decoding is used for error correction. Performance plots are shown in Section 3.2 which demonstrate the coding gain achieved for various constraint lengths, but the final system employs a [15, 17] convolutional code in order to balance performance with computational complexity. A block diagram of the shift register used in encoding is shown in Figure 4.

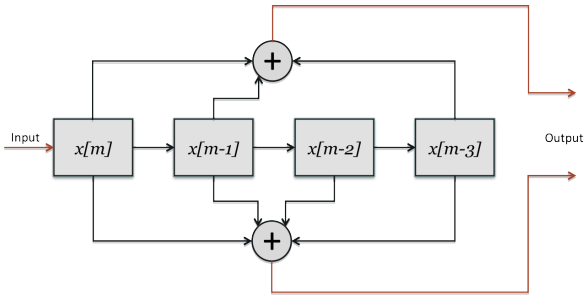


Figure 4: [15,17] convolutional encoder

2.2.1 Viterbi decoder

The well known Viterbi algorithm uses a trellis structure to find the code sequence with the smallest distance to the received sequence of bits [1]. The algorithm takes in the incoming soft decision bits, and outputs a maximum likelihood (ML) sequence of bits through three main processing steps: computing branch metrics, selecting optimal path metrics, and performing traceback. It is worth noting that the current implementation uses truncation and does not

force the shift register to the all-zero state. Traceback then starts at the state with the best path metric and ends in the all zero state.

As the most computationally intensive portion of the code (see Section 3.5 for a runtime analysis), significant effort was made to reduce the complexity of the Viterbi decoder. Several tables were precomputed to allow for efficient computation of the branch and path metric steps. These precomputed tables include a table indicating state transitions for each input bit, a table giving encoder output for each state and input bit, a table indicating the input value required to produce the next state, as well as tables providing decimal to binary vector conversion. One computational drawback of the current system is that it does not operate continuously. Traceback does not begin until the entire sequence is received and path metrics are computed. While this method does optimize performance, it could require modification for use in continuous scenarios.

It should also be noted that for hard-decision decoding (HDD), the hamming distance is used as the branch metric, while for soft-decision decoding (SDD), Euclidean distance is used. Euclidean distance is used since it provides optimal performance in AWGN through the path metric $p(\mathbf{r_j}|\mathbf{c_j})$. Though Rayleigh fading channels and other non-Gaussian channels may follow different probability distributions, we utilize the Euclidean distance metric for its simplicity. Details on the computation of soft output from the demodulator is given in Section 2.3.2.

2.2.2 Interleaving

An interleaver is used to maximize coding gain, especially in flat fading channels. The transceiver utilizes a 64 by 64 bit block interleaver, that operates immediately following encoding and prior to the addition of synchronization bits. Each sequence is then deinterleaved in the receiver before decoding is performed. The use of a size-64 block interleaver provides a balance between performance and read time of the block. In addition, a size of 64 is compatible with a 2048-bit packet (or 4096-bit encoded bit packet), thus ensuring that each packet will enter the interleaver independently.

2.3 Signal Detection

2.3.1 Modulation and Detection

Two modulation/demodulation schemes were implemented: QPSK and 16QAM. These two modulation types were chosen in order to provide a comparison between energy efficiency (QPSK) and bandwidth efficiency (16QAM).

In order to ensure that we are getting the best BER for our system, each of the modulation schemes are encoded using gray coding, where each smmbol differs by only one bit with each adjacent symbol in the constellation diagram. As an example, Figure 5 shows the gray coding used for the 16QAM constellation.

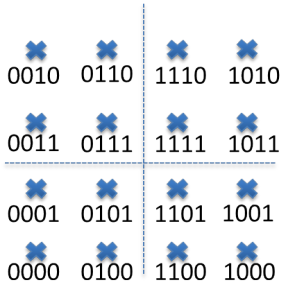
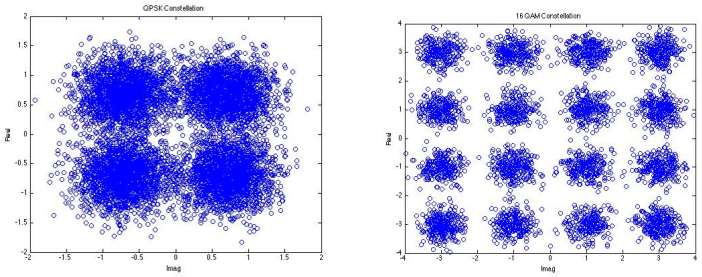


Figure 5: 16-QAM Gray Coding



(a) Received QPSK Symbols (b) Received 16 QAM symbols

Figure 6: The matched filter output for each modulation type at low E_b/N_0

In the transmitter, the modulator maps the bitstream to a sequence of symbols and then upsamples it to four samples per symbol. Coherent detection in the receiver is implemented with a matched filter. Each of the received symbols is match filtered with the pulse shape (described in Section 2.5) and then downsampled at the symbol time. Figure 6 shows the decision statistics at the output of the matched filter for each modulation type.

After downsampling, a complex vector of decision statistics is passed to the demodulator where pre-computed decision boundaries are used to determine the corresponding bits for each symbol.

2.3.2 Soft decision demodulator output

In order to use soft-decision decoding, soft bits must be output by the demodulator. These soft outputs are of the form $(2c_j - 1) + n_j$, such that in the absence of noise, the values input to the decoder will be a sequence of +1s and -1s. In the case of QPSK, these soft bits will simply be the real and imaginary parts of the decision statistics formed from the matched filter.

For 16QAM, the decision statistics are used to determine the soft output bits by using the constellation diagram shown in Figure 5. That is, by denoting the decision statistics output from the matched filter as `stat`, the four soft bits for each sample are computed by

- $\texttt{bit0} = \texttt{real}(\texttt{stat}(\texttt{i}))$
- $\texttt{bit1} = -1 * \texttt{abs}(\texttt{real}(\texttt{stat}(\texttt{i}))) + 2$
- $\texttt{bit2} = \texttt{imag}(\texttt{stat}(\texttt{i}))$
- $\texttt{bit3} = -1 * \texttt{abs}(\texttt{imag}(\texttt{stat}(\texttt{i}))) + 2$

2.4 Equalization

In digital communications, the goal of equalization is to reduce and correct the effects caused by a frequency selective or dispersive communication link between the transmitter and the

receiver [2]. For this system, a finite linear minimum mean square error (MMSE) equalizer was implemented to correct the effects of a Rayleigh fading flat channel, with the intent to extend it in the future to a frequency selective channel. Since we have no a priori knowledge of the channel, the equalizer uses the pilot symbols inserted for synchronization to help determine the optimum equalizer. The equation used to model the equalizer is found in [5] and given by the following :

$$\begin{bmatrix} R_z[0,0] & R_z[1,0] & \dots & R_z[M,0] \\ R_z[0,1] & R_z[1,1] & \dots & R_z[M,1] \\ \vdots & \vdots & \ddots & \vdots \\ R_z[0,M] & R_z[1,M] & \dots & R_z[M,M] \end{bmatrix} \begin{bmatrix} f[0] \\ f[1] \\ \vdots \\ f[M] \end{bmatrix} = \begin{bmatrix} R_{sz}[-u] \\ R_{sz}[-u+1] \\ \vdots \\ R_{sz}[-u+M] \end{bmatrix}$$

where R_z and R_{sz} are given by

$$R_z[i,j] = \frac{1}{n_2-n_1+1} \sum_{n=u+n_1}^{u+n_2} z[n-i]z^*[n-j]$$

$$R_{sz}[-u+j] = \frac{1}{n_2-n_1+1} \sum_{n=u+n_1}^{u+n_2} s_{n-u}[n-i]z^*[n-j].$$

In the above equations, $n_2 - n_1$ is the length of the pilot symbol, u is the offset delay, $z[n]$ corresponds to the received pilot symbol, and $s[n]$ is the known pilot symbol. The time average approximations of the correlation functions for $(i, j = 0, 1, ..., M)$ are calculated, where M is the equalizer order (chosen to be 8 for our implementation). The filter coefficients, $f[0], ..., f[M]$, are calculated and are used to filter the pilot symbols. The equalized pilot symbols are then divided by the known pilot symbols to find the equalized ratio. That ratio, which indicates the mean amplitude offset imposed by the channel, is multiplied by the data packets.

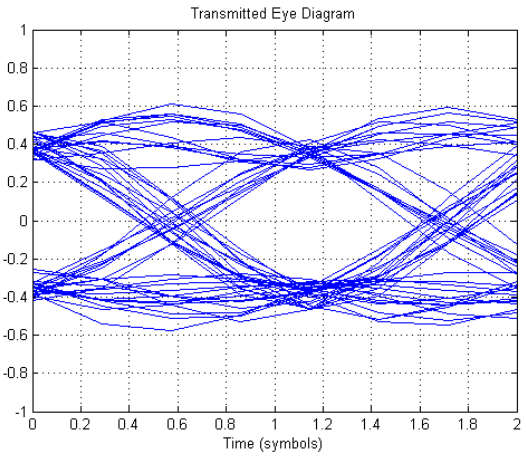
While this was not the optimum equalizer to implement, it provided sufficient amplitude estimation for QPSK in both AWGN and flat Rayleigh fading channels. Unfortunately, while the equalization method that was implemented provided sufficient amplitude estimation for 16QAM in AWGN, our team was unable to achieve adequate performance for 16QAM in Rayleigh fading channels, as discussed further in Section 3.4.

2.5 Pulse Shaping

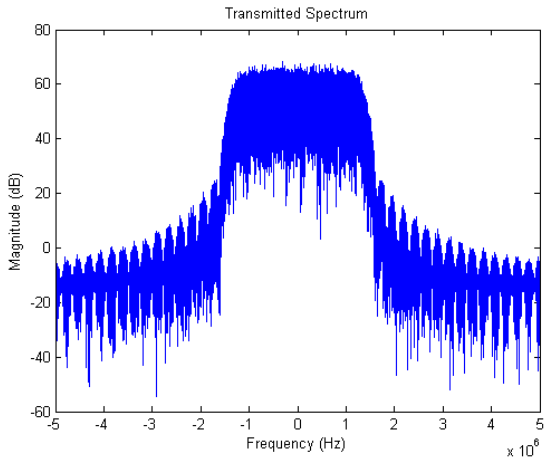
The communication system utilizes a unit-energy root raised cosine pulse shape. This pulse shape is applied in the transmitter after the data sequence has been modulated and upsampled, and is then used in a matched filter in the receiver's demodulator. The pulse shape is designed to be 10 symbol lengths, with a filter rolloff of $\alpha = 0.25$, though the latter parameter is easily configurable and can be changed according to the bandwidth efficiency required.

Figure 7a and 7b show the eye diagram and frequency spectrum (respectively) produced by this pulse shape at the output of the receiver. The eye diagram illustrates that this pulse shape meets the Nyquist zero-ISI criterion, while the frequency spectrum demonstrates the

improvement in bandwidth efficiency.



(a) Transmitter Output Eye Diagram for QPSK Modulation



(b) Transmitter Output Spectrum

Figure 7: Pulse shaping performance

2.6 Synchronization

While synchronization techniques are employed mainly in the receiver, a pilot sequence must be added to the beginning of every packet in the transmitted bitstream. Then in the receiver, several synchronization algorithms are used. First, a coarse frequency estimator corrects any major frequency offsets ($>\sim 100$ Hz). Then, a correlator searches for the pilot sequence in order to identify the beginning of each packet. Once each pilot sequence is found, it is used to correct any phase offset in the following data packet. Before the data packet is passed on to the demodulator, a decision directed phase estimator is used to correct any residual frequency offset.

2.6.1 Pilot Sequence

Many of the synchronization algorithms in the receiver (as well as the equalizer) utilize a pilot sequence prepended to each 2048 bit data packet. These bits are injected into the bitstream at the appropriate locations after the entire stream has been encoded. This entire bitstream is then pulse shaped and modulated.

The pilot sequence is an M-sequence with variable length. It was determined empirically that a 63 bit M-sequence works best for QPSK and a 127 bit sequence provides optimum performance for 16QAM. Each of these sequences is truncated slightly so that their length is a multiple of the number of bits per symbol (62 bits for QPSK, 124 bits for 16QAM). Note that after modulation, both of these bit sequences will be the same length $\left(\frac{62 \text{ bits}}{2 \text{ bits/symbol}} = \frac{124 \text{ bits}}{4 \text{ bits/symbol}} = 31 \text{ symbols}\right)$.

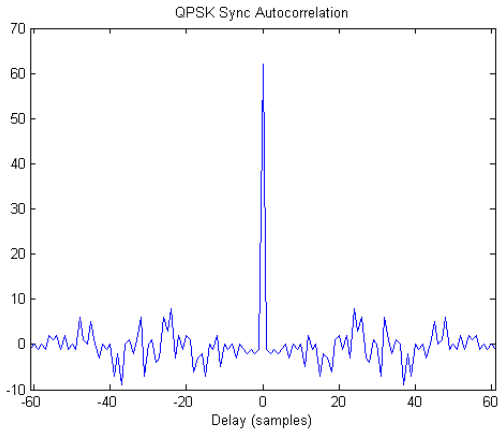


Figure 8: Autocorrelation of 63 bit M-sequence truncated to 62 bits

As can be seen in Figure 8, the autocorrelation of these sequences approximates a delta function - a desirable property which will help the system identify the sequence's location in time.

2.6.2 Coarse Frequency Synchronization

The coarse frequency estimator operates by raising the incoming complex samples to the fourth power and taking a 2^{15} sample FFT of the result (averaged with no overlap). This FFT can be expected to have a peak at approximately 4 times the frequency offset.

In order to understand how this works, consider what happens to each incoming sample when it is raised to the fourth power. For QPSK modulation (for simplicity here, we assume no pulse shaping), each sample will be of the form

$$x[n] = e^{j\left(\frac{\pi i[n]}{2} + \Delta f \frac{n}{f_s}\right)}, \tag{1}$$

where $i[n]$ is the information sequence (with values in the range [0, 3]), f_s is the sampling frequency in Hz, and Δf is some frequency offset in Hz.

Raising this sequence to the fourth power yields:

$$(x[n])^4 = e^{j\left(2\pi i[n] + 4\Delta f \frac{n}{f_s}\right)}. \tag{2}$$

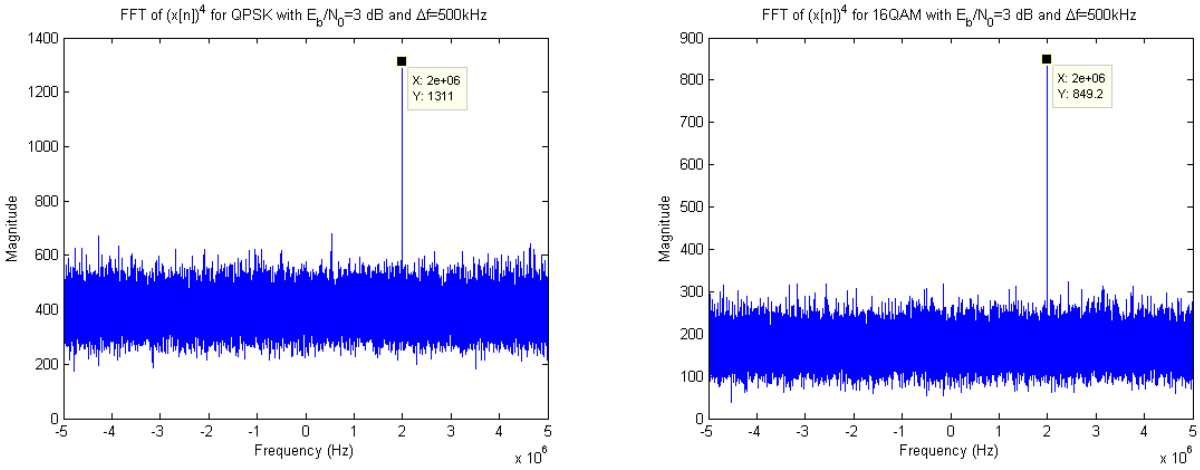
Note that $2\pi i[n]$ is just a sequence of zero phase, so the Δf term is all that remains. Thus it is easy to see that the only remaining component in this signal's frequency domain representation is a delta function at $4\Delta f$. An FFT of this sequence will easily identify that peak.

The accuracy of this estimator is only limited by the size of the FFT. We chose to use a 2^{15} sample FFT which should have a resolution at $f_s = 10$ MHz of

$$\frac{10 \text{ MHz}}{2^{15}} \approx 305.18 \text{ Hz}. \tag{3}$$

Note that this is the resolution that will be used to determine $4\Delta f$, so the resolution for Δf is actually approximately 76.3 Hz.

Even though this derivation is specific to QPSK, the same technique works for 16QAM. As you can see in Figure 9, the fourth power spectrum yields a peak at four times the frequency shift for either modulation type.



(a) QPSK (b) 16QAM

Figure 9: Spectrum of $(x[n])^4$ for $E_b/N_0 = 3$ dB and $\Delta f = 500$ kHz

At low E_b/N_0 it is very probable that this coarse frequency synchronizer will misidentify the frequency shift. If this happens, it can actually **harm** the receiver's performance by correcting a frequency shift that is not there. For this reason, we have added a "false-alarm" check that tries to identify this kind of failure. This check simply computes the standard deviation of the fourth power spectrum, and checks that whatever maximum we have identified is at least six standard deviations away from the mean. If this condition is not met, then the estimate is thrown away and no coarse frequency shift is corrected.

2.6.3 Time Synchronization

After the coarse frequency synchronization, the samples are synchronized in time using the pilot sequences. The receiver modulates and pulse shapes the pilot sequence bits to get an "ideal" sequence of samples for the pilot. This pilot sequence is then correlated with the signal, and the result is searched for peaks. The location of these peaks will correspond to pilot sequence locations.

The synchronizer only searches for the first peak over one packet length in order to avoid any confusion with subsequent peaks. This means that our system is currently only capable of correcting time offsets which are less than one packet length. This algorithm could easily be modified to search over a larger range and only identify the earliest peak, however as this seemed like more of an implementation detail than a signal processing problem, it is not currently used.

To correct for any drift in timing after the first packet, the time synchronizer then searches for all subsequent packets. The search is done over a much smaller window (three pilot

sequence lengths) around the end of the last packet, using the same correlation and peak search technique.

2.6.4 Phase Synchronization

Once all of the pilot sequences and their corresponding packets have been identified in time, the pilot sequences are used for phase synchronization at the packet level. The phase synchronizer measures the average phase offset between the received pilot sequences and the ideal pilot sequence in order to obtain an estimate of the overall phase offset for each packet. The average phase offset is measured using the following equation:

$$\hat{\phi} = arg\left\{\sum_{k} x_k y_k^*\right\}, \tag{4}$$

where $\bar{x}$ is the received pilot sequences, and $\bar{y}$ is the ideal pilot sequence. Once the phase offset is measured, it can be corrected in the entire packet by

$$\bar{x}' = \bar{x}e^{j\hat{\phi}}, \tag{5}$$

where $\bar{x}'$ is the phase corrected packet.

2.6.5 Fine Frequency Synchronization

The final sync step is the fine frequency synchronization, which employs a decision-directed phase estimator. This method not only corrects any residual frequency offsets not corrected by the coarse frequency estimator, it can also correct time-varying phase offsets introduced by fading.

The decision-directed phase estimator operates similarly to our phase estimator, but with two distinct differences: 1) It operates on samples in the data packets rather than the pilot sequences and 2) it works over a rolling window to correct **time-varying** phase errors (i.e. frequency errors).

The estimator first demodulates the data, and then re-modulates it. This re-modulated sequence should represent an "ideal" sequence with no phase offset or AWGN - similar to the ideal pilot sequence used previously for time and phase synchronization.

It then uses Equation 4, where $\bar{x}$ is the received signal and $\bar{y}$ is the re-modulated signal, to compute the average phase offset of that window. This phase offset represents the effect of the residual frequency offset over one window length.

The phase offset is then corrected in the window samples, as well as in every other subsequent sample in the packet. This is done so that when the estimator window slides forward, the next window has already had most of its phase offset corrected, thus yielding a better demodulation and re-modulation.

3 Performance Analysis

3.1 Simulation Setup

Simulations were performed in order to test the performance of the system in both AWGN and flat Rayleigh fading channels, with varying time, frequency, and phase offsets. In addition, these simulations provided a basis for parameter and implementation choices throughout the system. The system was initially optimized for performance in an AWGN channel, where the received samples are modeled as

$$r(t) = y(t) + n(t) = e^{j(2\pi\nu t+\theta)}s(t-\tau) + n(t), \tag{6}$$

where ν, θ, and τ are the frequency, phase, and timing offsets respectively, $s(t)$ are the pulse shaped transmitted samples, and $n(t)$ is a Gaussian random process.

Section 3.4 demonstrates the system performance in a flat fading Rayleigh channel, where the received samples follow the channel model

$$r(t) = \gamma(t)y(t) + n(t) \tag{7}$$

$$\gamma(t) = \frac{1}{\sqrt{N}}\sum_{k=1}^{N} e^{j(2\pi f_k t+\theta_k)} \tag{8}$$

where $f_k = f_d \cos\phi_k$, f_d is the maximum doppler frequency, ϕ_k and θ_k are uniformly distributed random variables, and $y(t)$ is the time, phase, and frequency shifted samples as shown in Equation 6.

3.2 Performance in AWGN Channel

As stated previously, the communication system employs both QPSK and 16QAM modulation schemes. In order to provide an even comparison between the two systems, simulations were run that compare the energy efficiency and performance of the two systems at an equivalent bandwidth efficiency. In other words, a comparison was made between QPSK modulation with 1/2 rate coding and 16QAM with 1/4 rate coding for a data rate of 2.5 Mbps (this data rate ignores packet header and synchronization bits). This data rate was chosen as a basis, since it provides a bandwidth efficiency equivalent to that of uncoded binary modulation. The result of this comparison is shown in Figure 10. As can be seen, the QPSK

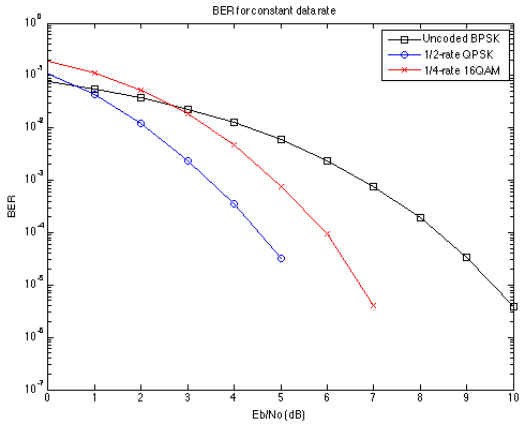
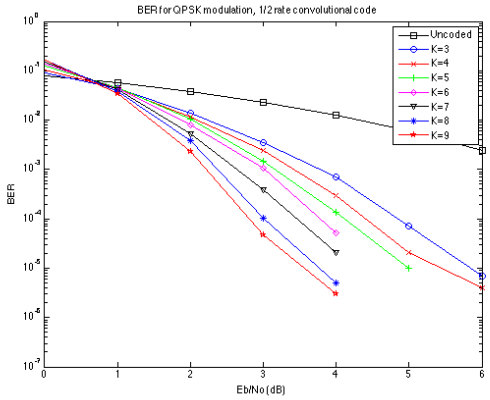


Figure 10: Comparison of modulation techniques with constant data rate



(a) System performance

Constraint Length, K	Generator polynomials	Min free distance, d_{free}	Runtime (seconds)
3	[5, 7]	5	5
4	[15, 17]	6	8
5	[23, 35]	7	13
6	[53, 75]	8	24
7	[133, 171]	10	45
8	[247, 371]	10	88
9	[561, 753]	12	175

(b) Viterbi decoder runtime

Figure 11: Comparison of convolutional code with varying constraint lengths

system has a 1.5 dB gain over the 16QAM system. For this reason, the final transceiver system employs QPSK modulation, but in situations with a more limited spectrum, 16QAM modulation is a viable substitute and easily configurable.

In order to validate the parameter choices for the chosen coding scheme, additional simulations were run to provide a performance analysis. First, soft decision decoding was a clear choice as it provided a 2.5 dB coding gain for QPSK and a 3 dB coding gain for 16QAM over HDD at a target BER of 10^{-4}, with virtually no additional computational complexity.

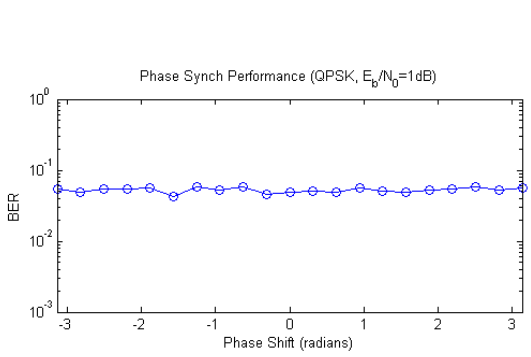
For analyzing constraint lengths, Figure 11a shows the performance of a QPSK system with SDD and optimal generating polynomials as provided by [3] and [4], against an uncoded system. Figure 11b includes the runtime of these same simulations (for 100,000 input bits). Due to the high computational complexity for increasing constraint lengths, the final system employs a constraint length of 4 to balance performance with runtime. For systems with more computational power requiring a high energy efficiency, larger constraint lengths can be used. Though we do not provide the details here for sake of brevity, additional simulations also showed very little gain achieved for coding rates lower than 1/2. For this reason, a 1/2 rate code was chosen in order to maximize bandwidth efficiency.

3.3 Performance against Time, Frequency, and Phase Shifts

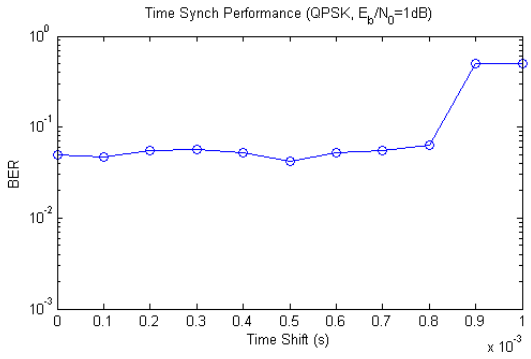
The system's performance against time, frequency and phase shifts in an AWGN channel was evaluated in order to confirm that the synchronization algorithms function as intended.

Figures 12a and 12b show the system's performance against phase and time shifts, respectively. Note that both of these plots were generated with $E_b/N_0 = 1$ dB, and the system was configured for QPSK modulation. As can be seen in Figure 12a, the system can detect and compensate for any phase offset in the range $[-\pi, \pi]$. Figure 12b reveals that the time synchronization performs well for any time shift up to $819.2\mu s$. This cut-off corresponds to the length of one packet. For details on why our system has this limitation see the description

of our time synchronization algorithm in Section 2.6.3.



(a) Phase synchronization performance

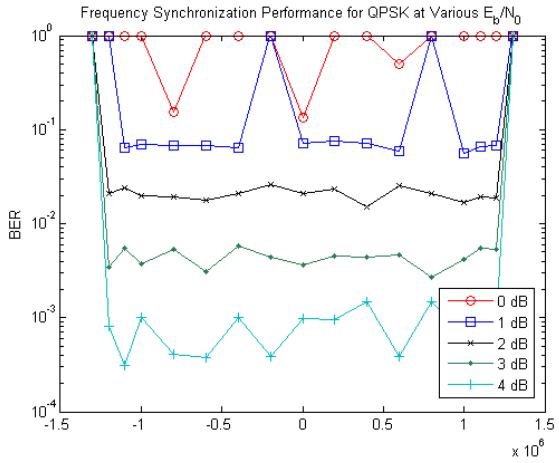


(b) Time synchronization performance

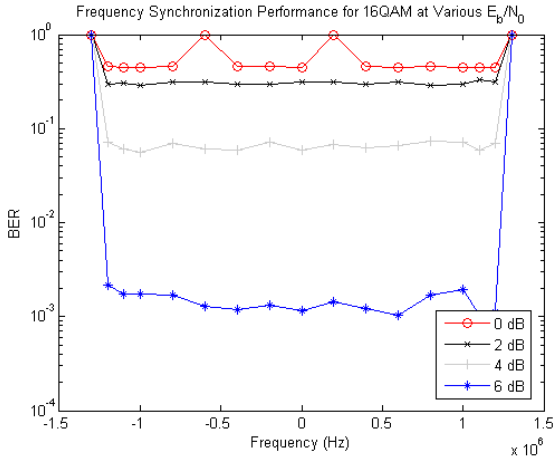
Figure 12: The system performs well against any phase shift, and against time shifts up to about 819.2μs

Our system's coarse frequency estimator is able to correct frequency offsets of up to ± 1.25 MHz for either QPSK or 16QAM modulation at almost any E_b/N_0. This performance is shown in Figure 13. Notice that the BER is essentially constant for QPSK with an E_b/N_0 greater than 1 dB and for 16QAM with E_b/N_0 greater than 0 dB.

One odd feature in these plots is the presence of 100% BERs. This occurs when the coarse frequency estimator fails to correctly estimate the frequency shift. As a result, the system attempts to process a severely frequency shifted signal, and fails to return the correct number of bits. In our testbed, returning the wrong number of bits was recorded as a 100% BER.



(a) QPSK frequency synchronization performance



(b) 16QAM frequency synchronization performance

Figure 13: The system performs well against frequency shifts up to ± 1.25 MHz with moderate to high E_b/N_0

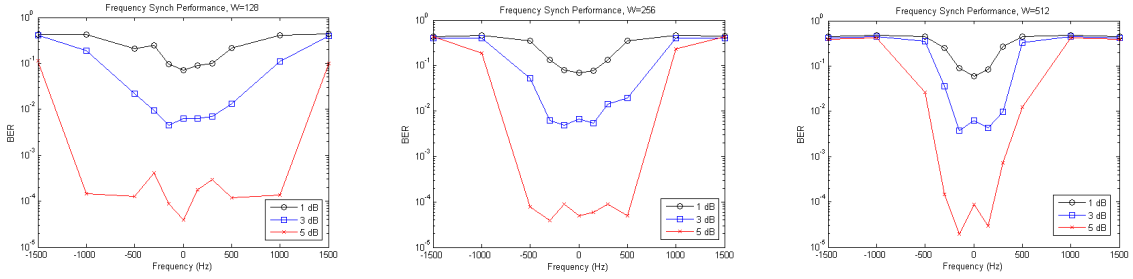
It is important to note that the system can still correct minor frequency offsets at low E_b/N_0 using only fine frequency synchronization, even when the coarse synchronizer fails.

3.3.1 Coarse vs. Fine Frequency Synchronization

In a system that performs both coarse and fine frequency synchronization it is important to decide on a threshold between a fine and a coarse frequency shift. That way the coarse synchronizer can be tuned to adjust any shifts above that threshold, and the fine synchronizer can be tuned to correct shifts below it.

In our case the coarse frequency synchronizer is tuned by adjusting the FFT size, while the fine synchronizer is tuned by adjusting the window size. Note that both of these parameters yield a tradeoff in the form of robustness to noise. Increasing the coarse synchronizer's FFT size lowers its minimum correctable shift, but reduces the number of FFTs that can be performed and averaged. Decreasing the fine synchronizer's window size increases its maximum correctable shift, but it means there will be fewer samples over which to average.

Section 2.6.2 discusses the limitations of the coarse frequency synchronizer for various FFT sizes. The theoretical limitations of the fine frequency synchronizer are more challenging to determine, so we determined the performance empirically. The results can be seen in Figure 14 below.



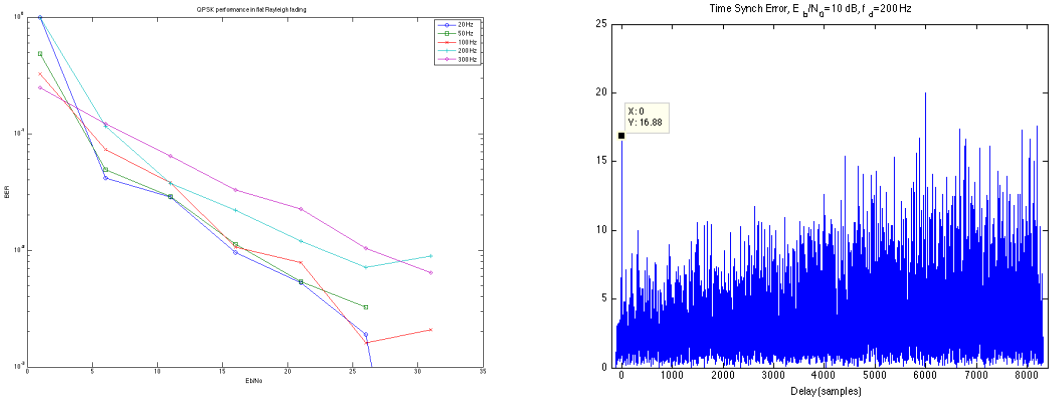
(a) 128 sample window size (b) 256 sample window size (c) 512 sample window size

Figure 14: Decreasing the fine frequency synchronizer window size improves its ability to correct frequency shifts at the cost of robustness to noise

Based on these empirical results and our theoretical calculation of the coarse synchronizer resolution we decided to use a 256 sample window size in the fine synchronizer and a 2^{15} sample FFT in the coarse synchronizer. This creates a threshold between coarse and fine at about 100 Hz.

3.4 Performance against Flat Rayleigh Fading

After verifying performance in an AWGN channel, the system was tested and optimized for a flat fading Rayleigh channel. Figure 15a shows the QPSK performance in the presence of varying max doppler frequencies. As expected, performance is much worse than AWGN



(a) Performance in flat Rayleigh fading

(b) Timing sync error due to fading

Figure 15: Comparison of convolutional code with varying constraint lengths

performance, but consistent with what can be expected in a flat fading scenario. Performance degrades as the max doppler increases due to the fact that the fading varies more rapidly in these situations, and channel estimation with pilot symbols is still only performed once per packet.

Figure 15b shows the results of of the timing synchronization when a deep fade overlaps with the pilot sequence transmission. In these instances, a false timing synchronization can be detected, causing the system to fail. It should be noted that this happened on occassion at all max doppler frequencies, but occurred more frequently as the max doppler increased above 300 Hz.

Unfortunately, while adequate channel estimation was completed for QPSK in flat fading Rayleigh scenarios, this same method proved insufficient to compensate for the accuracy of estimation required for 16QAM in Rayleigh channels. The team initially decided to approach the equalizer with an MMSE equalizer in order to compensate for both flat fading and frequency selective fading. Unfortunately, this proved to be too ambitious of an approach, and while 16QAM modulation is operational in the presence of AWGN, it is left as a place of improvement for Rayleigh fading.

3.5 Runtime Analysis

A runtime analysis was done to ensure that the developed system is using computing resources efficiently. As could be expected, an overall analysis using the MATLAB profiler showed that the soft output Viterbi decoder required the largest amount of computing power, accounting for approximately 3/4 of the total computing resources in the receiver. As noted in section 2.2.1, significant effort was put into optimizing the decoder to prevent unnecessary processing. Additional analysis using the profiler showed that no one function call was taking up a significant amount of processing time. The team recognizes that additional steps could be taken to further optimize this decoder, including continuous decoding, parallelization,

and a native language implementation.

4 Conclusion

The communication system design and implementation achieved the project goals in the presence of an AWGN channel. By using a soft-decision Viterbi decoder, we were able to achieve a coding gain of approximately 4 dB for QPSK modulation at a target BER of 10^{-4}, a result that is consistent with theoretical performance. In addition, the team was able to compensate for any phase shift from 0 to 2π and any time shift corresponding to at least one packet length. Through the use of both a coarse and fine frequency estimation, we were able to achieve synchronization in the presence of frequency offsets up to 1.25 MHz. The system also was shown to be operational using QPSK modulation in the presence of a flat fading Rayleigh channel.

The design and implemenation of this transmitter and receiver system was a learning experience for the entire team. One of the biggest challenges encountered was accurately performing amplitude estimation in the presence of Rayleigh fading. While the team is still working on achieving this, we have successfully implemented many algorithms that were unfamiliar prior to this course, including (among others) Viterbi decoding, MMSE equalization, and both frequency synchronization techniques.

Finally, our team recognizes that several improvements could be made to the system to enhance operability. First, more time would be devoted to amplitude estimation and tracking, in order to get reliable results for 16QAM in the presence of flat Rayleigh fading. This would include more research into equalization techniques that could perhaps be extended to frequency selective channels, as was initially intended. Next, packetization would be improved with a more comprehensive parity check and correction on the packet header. Time synchronization could also be optimized to account for larger timing offsets. Finally, all the code could be optimized for performance and complexity enhancements.

References

[1] A.J. Viterbi, *Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm*, IEEE Transactions on Information Theory, Volume IT-13, pages 260-269, in April, 1967.

[2] Scott C. Douglas, *Equalization in Digital Communications.* Matlab. http://www.mathworks.com/help/dsp/examples/equalization-in-digital-communications.html

[3] K.J. Larsen, *Short convolutional codes with maximal free distance for rates 1/2, 1/3, and 1/4*, IEEE Trans. Inform. Theory, vol. IT-19, pp. 371372, May 1973.

[4] J.P. Odenwalder, *Optimal decoding of convolutional codes*, Ph.D. dissertation, School Eng. Appl. Sci., Univ. California, Los Angeles, 1970.

[5] B.P. Lathi and Zhi Ding. *Modern Digital and Analog Communication Systems*, Oxford University Press, New York, 2009.