Simulation of Quadrature Amplitude Demodulation in a

Digital Telemetry System

by

Heather A. Campbell

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 7, 1996

Author_____

Department of Electrical Engineering and Computer Science
February 7, 1996

Certified by_____

Dr. Lloyd Clark
Thesis Supervisor

Certified by_____

Dr. George Verghese
Thesis Co-Supervisor

Accepted by_____

F. R. Morgenthaler
Chairman, Department Committee on Graduate Theses

Simulation of Quadrature Amplitude Demodulation in a Digital Telemetry System

by

Heather A. Campbell

Submitted to the
Department of Electrical Engineering and Computer Science

February 7, 1996

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The realization of a new wireline acquisition front end has made it possible for Schlumberger to redesign its uphole telemetry receiver. In order to achieve data rates of 500 kbits/second over standard oil well logging cables, the Digital Telemetry System uses Quadrature Amplitude Modulation (QAM) to transmit its measurement data. The demodulator involves timing recovery, filtering, cable equalization, and symbol decoding. The purpose of this thesis is to simulate the demodulator, thereby documenting the demodulation process and creating a design tool that can be used to design future QAM telemetry systems.

Thesis Supervisor: Dr. Lloyd Clark

Title: Senior Engineer, Schlumberger Austin Product Center

Thesis Co-Supervisor: Dr. George Verghese

Title: Professor, MIT Department of Electrical Engineering and Computer Science

Not the quarry, but the chase.
Not the laurel, but the race.

--Inscribed on wall of MIT Johnson Athletic Center

Simulation of Quadrature Amplitude Demodulation in Schlumberger's Digital Telemetry System

PART I -- Telemetry

PART II -- Demodulation

Appendices

PART I -- Telemetry

A brief overview of thesis goals, Schlumberger's telemetry projects, and the operation of the current telemetry system will be helpful before we delve into demodulation and simulation details in future sections.

PART I -- Telemetry

1. Thesis Overview

1.1 Opportunity

Schlumberger's new Wireline Acquisition Front End (WAFE) has created several telemetry opportunities. There are three projects on the horizon at Schlumberger Austin Product Center:

1. **Redesigning the uphole Digital Telemetry Module (DTM)**

Due to the introduction of a VME bus on the WAFE and 5 years of technological advancement, it should be possible to replace the six custom boards (including 9 Digital Signal Processors (DSPs)) currently needed to receive uplink data and send downlink instructions in Schlumberger's Digital Telemetry System (DTS) with one board and a minimal number of DSPs.

Redesigning the uphole receiver will be beneficial in three ways: the process will bring the demodulation knowledge in-house, the new design will provide an opportunity to make improvements in performance and functionality, and the new implementation will involve fewer components, and therefore be smaller, less expensive, and more robust.

This project is currently underway. It would benefit from a more thorough understanding of the telemetry process, rationale behind the DTS telemetry parameters, and a platform for experimenting with parameter trade-offs.

2. **Replacing Modules with Built-In Telemetry**

In addition to the most recent DTS module, Schlumberger wireline systems must continue to support the modules of older telemetry systems. Current technology should make it possible to combine the functionality of all three tool modules into one or two VME cards, and thereby build the modules into the WAFE itself. [Booker 95, p. 1]

This project is currently in feasibility. It would benefit from a platform for experimenting with parameter trade-offs and for verifying that proposed operation meets specifications.

3. Developing a new telemetry system for Monocable

There is no modern telemetry system that can use monocable as a channel; DTS only works on heptacable. The ability to make wireline measurements with monocable would provide a competitive advantage to Schlumberger. Powerful DSPs should now make it possible to provide a high bit rate even on this poor channel. [Booker 95, p. 5]

This project is also in feasibility. It would benefit from a platform for experimenting with parameter trade-offs, for verifying that proposed operation meets specifications, and for testing the telemetry system on a wide variety of cables.

The purpose of this thesis is to simulate the telemetry demodulation process. Simulating the telemetry process is useful for several reasons:

- A correctly-working simulation will document and verify understanding of the process.
- A simulation with adjustable parameters will allow the parameters to be optimized after investigating trade-offs between performance and implementation cost.
- A simulation will allow testing of a wide variety of cables (including worst case) that aren't available in the lab testing environment.

As we can see from the list of projects above, such a simulation will be of value to Schlumberger.

1.2 Goal

This thesis met its three goals, as summarized in Section 11:

1. To document an understanding of DTS demodulation.

2. To create a simulation of the demodulation process.

3. To demonstrate the use of the simulation as a design tool.

1.3 Approach

The thesis approach evolved as the goal of the thesis became more focused, but remained essentially the same as that suggested in the original thesis proposal [Campbell 95]:

1. **Understand purpose and operation of DTS.**

   Before simulating a component of the Digital Telemetry System, it is advisable to understand the system as a whole. I gained a background in telemetry by reading [Bingham 88], [Gardner 94], [Gardner 95], [Lucky 68], and [Mayhugh 92]. I learned about the specific DTS application by reading [Clark 95], [Kelly 91], and [DTS 91].

2. **Understand theory and implementation techniques of QAM.**

   Quadrature Amplitude Modulation (QAM) is a well-understood modulation technique with an extensive literature. I was familiar with the theory of quadrature amplitude modulation and demodulation from introductory signal processing classes. I learned about the practical implementation of QAM by reading [Samueli 94], [Mayhugh 90], and [Bingham 88], timing recovery from [PCSI 88-1] and [Bingham 88], low pass filtering from [Samueli 94], [PCSI 88-1], and [Bingham 88], and adaptive equalization from [Johnson 95], [TI 90], [Verghese 94], [PCSI 88-1], and [Bingham 88].

3. **Understand current DTS implementation.**

   Because the implementation of the current uphole receiver was contracted out to Pacific Communication Sciences, Inc. (PCSI), no one in Schlumberger had a complete understanding of how the demodulator works. By reading [Mayhugh 90], [Montgomery 93], [Montgomery 9x], and [Kelly 91], as well as the business plans, status reports, and assembly source code written by PCSI ([PCSI 8x], [PCSI 88-1], [PCSI 88-2], and [PCSI 89]), I was able to understand and document the current implementation of the uphole receiver.

4. **Choose simulation platform.**

   I first attempted to simulate the telemetry system using i-Logix's Statemate software [i-Logix 91], but found that the signal processing environment was better in Matlab [Burrus 94]. I worked through several high-level simulation designs before arriving at the one I found most useful.

5. **Simulate system behavior.**

   The simulation was developed in Matlab. Individual simulation steps were written as Matlab functions. The functions were combined together in scripts to form specialized simulation runs.

6. **Verify simulation.**

   The simulation results were compared to those of the original DTS specification in [Kelly 91] and [PCSI 88-1]. Theoretical optimal-tap checks and common sense checks also verified the simulation to be working correctly.

7. **Collect Simulation Results.**

   Several specialized simulation runs were designed and implemented to gain insight into the effect of cable material, cable length, number of equalizer weights, and size of convergence parameter on telemetry performance. I also developed a prototype monocable simulation.

8. **Complete Documentation.**

   Finally, the simulation process was documented in a thesis write-up.

2. The Digital Telemetry System

2.1 Telemetry Overview

Schlumberger makes its money by lowering tools downhole, making measurements, and then using these measurements to decide if there is oil in the well, how much oil there is, and how best to get it out. Wireline measurement data is transmitted from tools downhole to MAXIS software uphole via the cable that suspends the tools. In addition, commands from the uphole software are sent to the downhole tools along the cable. The process by which information is sent along the cable is called telemetry.

2.1.1 Environment

A logging cable is not an ideal information channel. Although the frequency response of a given cable is fairly stable, the electrical characteristics of different cables vary widely with length, mode, temperature, and cable type.

A long cable is 30 000 ft, though many wells are less than 12 000 ft deep. Longer cables are more attenuative and have a more non-linear phase, and therefore are poorer data channels. Most telemetry operates on heptacable. Heptacable's seven conductors allow many possible data paths -- the two modes used for telemetry are called T5 and T7. [Clark 95, p. 15] T5 is a slightly worse channel than T7. The temperature of a cable increases as the cable travels down a well. Hot cable is a worse channel than colder cable. Finally, cables can be made out of several different materials. The frequency characteristics of a variety of cables of different materials and lengths are shown in Figure 2.1-1. For reference, the worst cable the current telemetry system can handle is 30 kft of 250°F 7-46NT. [PCSI 88-1, p. 18]

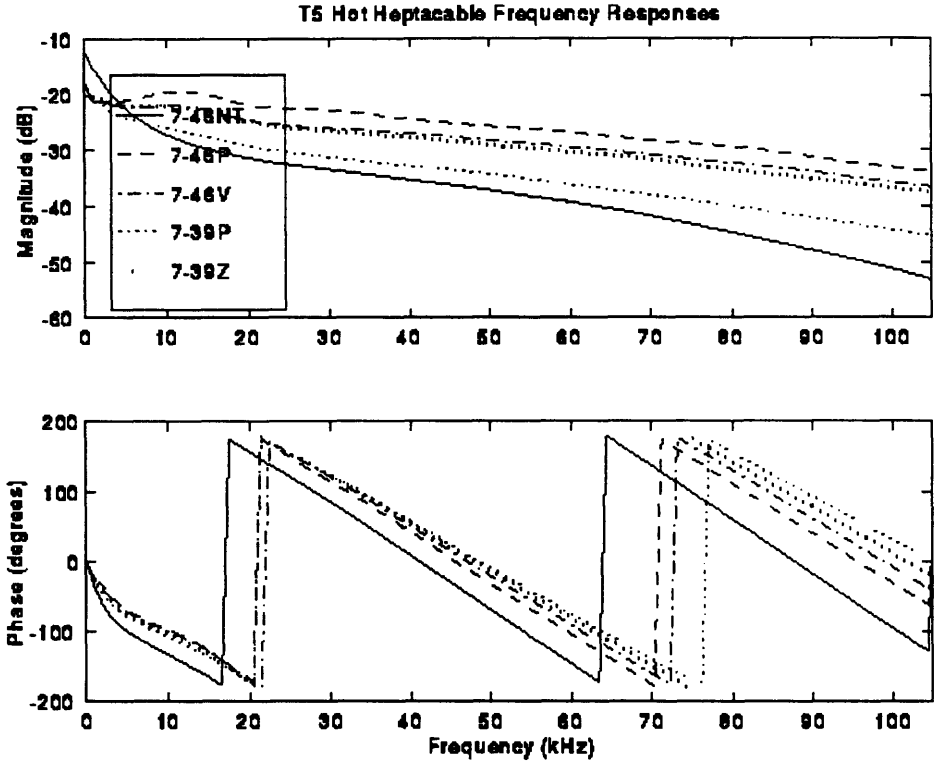


Figure 2.1-1: Frequency Characteristics of Various Heptacables

Wireline telemetry requires a transmitter and receiver at the bottom of the cable as well as at the top of the well. Because of the hostile environment in an oil well, it is always better to put complexity uphole than down.

2.1.2 History

The earliest types of telemetry processed analog signals with analog circuitry. One tool at a time would transmit its information over heptacable. The introduction of a standard cable configuration in the 1970's permitted tools to be connected into a tool string. The first modern telemetry system, Telemetry-B (Tel-B), was specified for 5000 ft to 50000 ft of 7-46NT cable. With the use of biphase encoding uplink, bipolar pulse encoding downlink, and half-duplex operation over the T5 cable mode, Tel-B had a raw uplink rate of 10 kbits/second and a raw downlink rate of 2.5 kbits/second. [Clark 95, p. 6]

A new Cable Telemetry System, CTS, was developed in the early 1980's. CTS used a fixed equalizer to compensate for the low pass characteristics of the cable spectrum. The equalizer was specified for 12000 ft to 28000 ft of 7-46NT cable. CTS could also transmit on coax cable, but not on monocable. By sacrificing cable length range, using biphase encoding in both directions, constraining data packet formats, implementing CRC error detection, transmitting a sync word, using the differential T5 mode to improve noise immunity, and transmitting in half-duplex mode, CTS achieved a raw uplink rate of 100 kbits/second and a raw downlink rate of 100 kbits/second. [Clark 95, p. 12]

2.2 Digital Telemetry System

In the late 1980's, it became clear that data rates higher than the 100 kbits/second provided by CTS were needed. The 500 kbit/second uplink data rate of the MAXIS 500 was accomplished through four major innovations:

1. Encoding uplink data using Quadrature Amplitude Modulation (QAM), instead of biphase encoding.
2. Transmitting on multiple channels, T5 and T7.
3. Equalizing the channel adaptively.
4. Implementing more robust error handling.

The Digital Telemetry System, DTS, went commercial in 1990. A brief overview follows; complete specifications are found in Appendix 1 and [DTS 91, pp. 2-1 - 2-39].

DTS modulates symbols at 70 kHz on to a 52.5 kHz carrier. Each symbol may be set to represent either 3, 4, 5, or 6 bits. With the default 5 bits per symbol, DTS can transmit:

(70 ksymbols/second) * (5 bits/symbol) = 350 kbits/second

of raw data over a logging cable which only has a 70 kHz bandwidth. DTS operates two QAM uplink channels simultaneously to double the raw data rate to 700 kbits/s. After subtracting protocol overhead, the default effective data rate is 500 kbits/second.

2.2.1 DTS Organization

The following explanation of DTS is excerpted from [DTS 91, pp. 2-1, 2-2]:

> The term "Digital Telemetry System" refers to the hardware and the software elements required to move data between a downhole tool and its MAXIS application software. All Schlumberger telemetry systems consist of three physical parts: the interface inside each tool, the cartridge at the bottom of the cable, and the module which remains uphole.
>
> Each downhole tool includes an Interface Package (IP), providing the tool with a digital port consisting of a byte-wide parallel bus, read and write strobes, and handshake lines. A Fast Tool Bus (FTB) connects each tool to its neighbors in the tool string. From the tool's perspective, messages written to this port are deposited in a buffer in MAXIS memory to which its application code is pointed. Downlink messages from tool application software are made available to the tool through this same port.
>
> A Digital Telemetry Cartridge (DTC) must always be the top tool in a tool string. In addition to an IP, the DTC has a controller and a downhole modem. The DTC modulates uplink data from the tools, and demodulates downlink data destined for the tools.
>
> The Digital Telemetry Module (DTM) must be connected to the DTC via heptacable. In order to demodulate uplink data and modulate downlink data, the DTM contains an uphole modem and a protocol processor. A GUI-bus (or Ethernet in later implementations) connects the DTM to the MAXIS system software.

A schematic of DTS elements is shown in Figure 2.2-1.

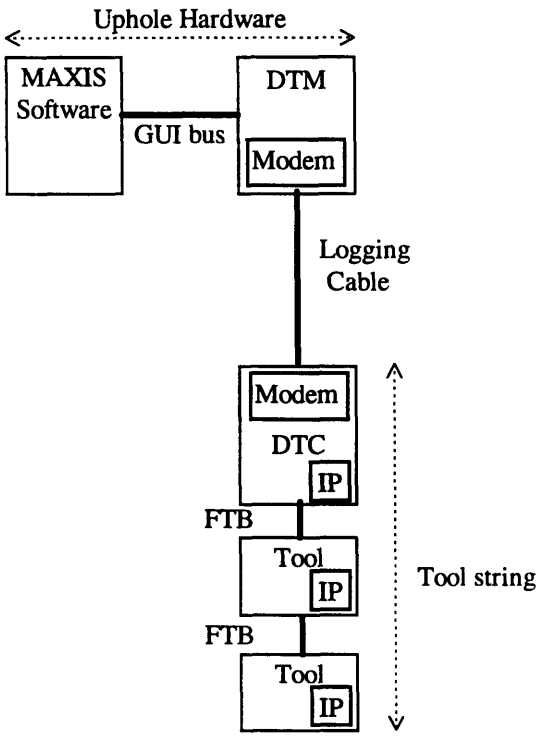


Figure 2.2-1: DTS Elements [DTS 91, p. 2-2]

2.2.2 DTS Frames

In order to facilitate both uplink and downlink data, DTS divides time into frames. At a programmable frequency between 6 and 30 Hz, the cable channel is allocated for a variable length of downward data, then a variable length of upward data. [Clark 95, p. 19] This half-duplex form of time compression multiplexing (TCM, as it is known in telephony literature) gives the impression of a full-duplex (continual uplink and downlink) channel. Operating half-duplex eliminates cross-talk problems while allowing us to use the 'sweet spot' of the transmission channel in either direction of travel. This in turn eliminates the needs for sharp filters, very long length adaptive equalizers, or digital echo cancellers. [PCSI 8x, p. 6]

2.2.3 DTS Downlink Format

The variable period allocated for downlink communication is divided into 16 bit words. The first word must be a sync pattern, the second a status word, and the last word is for CRC error detection. The status word contains a count of the number of data words found in the middle of the transmission. [Clark 95, p. 23]

Downlink data is transmitted over the T5 cable mode. This is a mode defined on heptacable such that signals are sent on the 2nd and 5th cables, with respect to the 3rd and 6th cables in the bundle. [Clark 95, p. 15] All downlink data is modulated using biphase encoding at a rate of 70 kbits/second. [Clark 95, p. 23] An example of biphase encoding is shown in Figure 2.2-2.

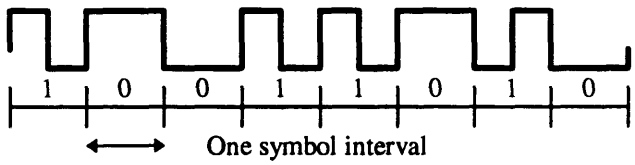


Figure 2.2-2: Biphase Encoding of DTS downlink data [Mayhugh 90, p. 4]

Downlink data rates are less than those of the uplink channel for two reasons:

1. Only a small amount of information needs to be transmitted down to the tool.
2. The hostile environment in which the DTC must operate limits its demodulation capabilities.

2.2.4 DTS Uplink Format

Uplink data is transmitted over both T5 and T7 cable modes. In T7 mode, signals are sent on the center, or 7th, conductor, and return on all other conductors including the armor. In order to achieve high data rates, uplink data is transmitted using Quadrature Amplitude Modulation (QAM).

Figure 2.2-3 illustrates the uplink data flow. The following uplink flow description is excerpted from [DTS 91, p. 2-8, 2-9]:

Both uplink and downlink tool data are structured as messages. Uplink messages are free-form, and may be any length up to 64 kwords. The tool writes data words to the IP, and indicates message boundaries by pulsing the IP's write line with an "End of Message" address on the Tool Interface. The IP divides each message into data blocks (or concatenates multiple messages into a single block), then adds a 16-bit sync word and some status words to the beginning of the block, and a 16-bit CRC (Cyclic Redundancy Check) to the end to form a "packet," which is transmitted up the Fast Tool Bus. Each IP in the tool string repeats packets from IPs below it.

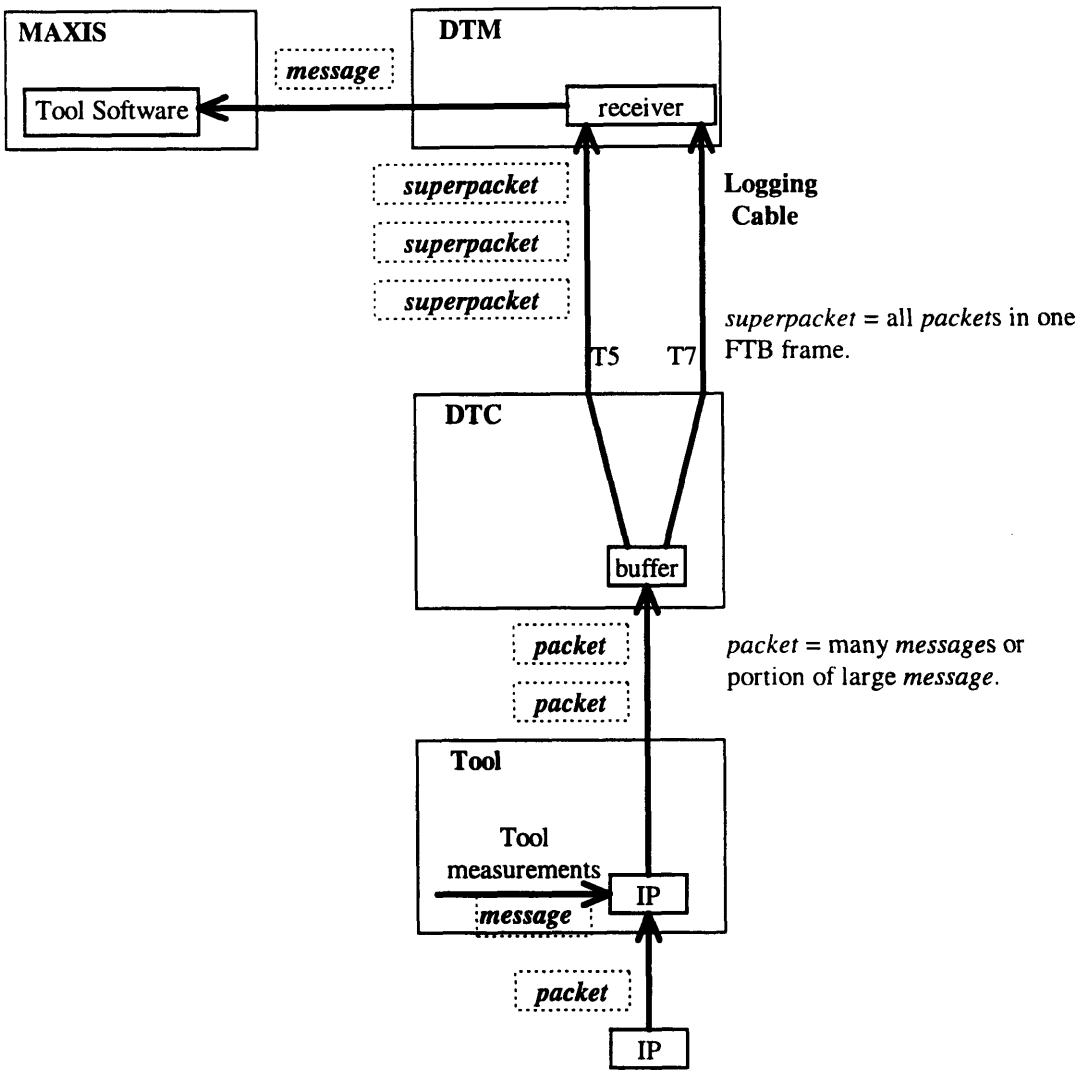


Figure 2.2-3: Uplink Data Flow [DTS 91, p. 2-8]

The DTC discards any uplink packets received in error (as indicated by the CRC), and requests that the IP re-transmit the same data on the next FTB frame. The DTC strips the sync and CRC words from each error-free uplink packet, then concatenates all of the packet cores received during one FTB frame into a single data block. An FTB frame consists of a Frame Start command, a period of uplink transmission, and then a period of downlink transmission. A 32-bit sync word, a 16-bit CRC word, and some header words are added to this data block to

form an uplink superpacket. Each uplink superpacket is stored in one of 32 one-kword buffers in the DTC. The size of these DTC uplink buffers limits the size of the transmission windows assigned to the IPs; i.e., the amount of uplink data transmitted each FTB frame must be less than one kword.

The DTC removes superpackets from its uplink buffers one at a time, splits them into two serial data streams (if both T5 and T7 uplink channels are active), and feeds the two streams to the Downhole Cable Modem. After the data travels up the cable, the Uphole Cable Modem in the DTM receives both T5 and T7 and sends the two data streams to the DTM's mode combiner. The mode combiner converts the two serial data streams into a single stream, converts the serial data to parallel, and presents it to the protocol processor. The protocol processor moves each uplink word into one of 62 one-kword superpacket buffers.

Uplink superpackets are processed by the DTM in the order they were originally formed by the DTC, not in the order of arrival. The DTC places a sequence number in the header of each uplink superpacket for this purpose. Uplink superpackets may not be received in the correct order because the DTM discards any uplink superpacket with a CRC error, and requests that the DTC repeat it.

The DTM reads superpackets from the uplink buffers one word at a time. The microprocessor transfers each data word of an FTB packet (starting after the packet header) from the superpacket buffer to one of 16 one-kword message buffers (It is possible to have up to 16 IPs in a tool string). After the last word of an uplink message has been written to that IP's message buffer, the processor places two words at the beginning of the message indicating the message length and the source IP, plus 5 words of depth and time stamps. It then moves the completed message to a 42927 word output buffer, where it can be read by MAXIS. This output buffer will overflow in 800 ms at 500 kbits/second if the average message length is 10

words, or sooner with shorter messages (because of the 7-word header added to each uplink message).

Many uplink superpackets may be transmitted before the DTC switches the Downhole Cable Modem from transmit to receive. The DTM will switch when the Uphole Cable Modem indicates loss of uplink carrier.

2.2.5 Training Sequences

Training sequences are needed to appropriately set DTS parameters. All training sequences contain three stages:

1. ALT

   Alternates between two symbols. Used to get rough timing lock and set Automatic Gain Control (AGC).

2. PN

   Alternates between two symbols according to a pseudo-random sequence. Used to refine timing lock and train equalizer.

3. SCM1

   Continuously sends one symbol. Used to transition from training to steady state.

A long training sequence of 576 ms provides the initial training required for the QAM receiver. Long training starts from initial settings; no prior knowledge of any settings is assumed. This training must be performed after a reset, selftest, or after the protocol board detects faulty operation or a severe degradation in the performance of the receiver. Long training accomplishes the following tasks:

- Estimates the frequency of the transmitter clock.
- Acquires the proper AGC setting.

- Acquires the optimum sampling phase.
- Trains the adaptive equalizer, starting with the initial "zero" tap setting. [PCSI 88-1, p. 31]

A short training sequence of 10 ms provides the fine tuning for the QAM receiver at the beginning of each normal uplink data transmission. The receiver initializes with the settings from the last receive session. Short training accomplishes the following tasks:

- Re-tunes the AGC setting.
- Re-acquires the optimum sampling phase.
- Trains the adaptive equalizer, starting with the tap setting from previous run. [PCSI 88-1, p. 32]

Medium training was added to maintain frequency lock between the uphole and downhole modems for tool strings with very low data rates. Medium training replaces short training every 8th uplink transmission.

2.2.6 Scope Trace

The photo in Figure 2.2-4 was made with a normally operating DTC connected to 32500 feet of 7-46NT logging cable, on the T5 channel.

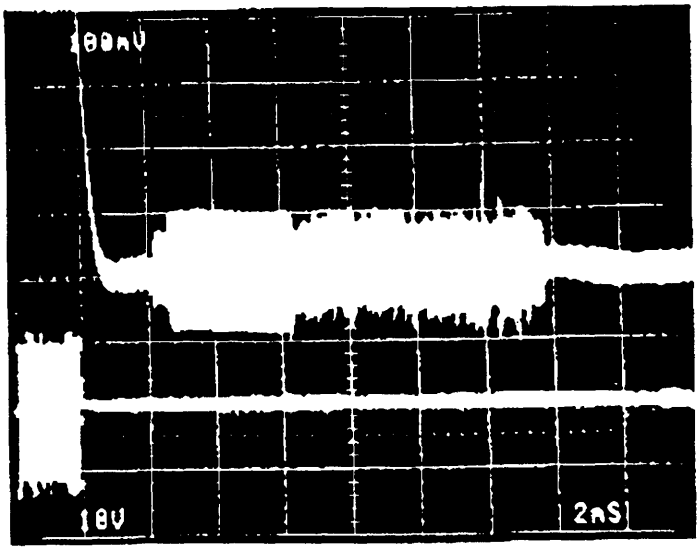


Figure 2.2-4: Scope Trace: Uplink on top, Downlink on bottom [DTS 91, p. 9-6]

The amplitude scale for the top uplink trace is 100 mV per division, while the amplitude scale for the bottom downlink trace is 10 V per division. With this long cable, the downlink signal is clearly much stronger than the uplink signal. In fact, the uplink signal is not visible with the 10 V scale of the lower trace. On the upper trace, the downlink signal is also present in the first division, but its amplitude is too large to be visible in this photo. The time scale is 2 ms per division; the trigger is on the upper trace.

Transmission on the T7 channel will have the same shape, but the downlink signal will be much smaller - perhaps 200 mV peak-to-peak. T7 is not used for downlink transmissions; its small downlink signal is due to coupling between T5 and T7.

Various phases of short training are visible. The uplink signal actually starts just after the second division. The first 0.457 ms of the sequence is called the A-Tone. Next, the Alternate Point Sequence is sent for 3.657 ms, followed by 5.714 ms of the Pseudo-random Number Sequence. The Sync word of the first uplink superpacket is sent shortly after the 9.83 ms training sequence ends. Very little

uplink data is being transmitted in this trace, so data transmission occupies only the last millisecond of the uplink signal.

PART II -- Demodulation

DTS achieves its high uplink data rates through Quadrature Amplitude Modulation, or QAM. A practical quadrature amplitude demodulation implementation requires several signal processing steps. We shall focus on those which will be necessary for our simulation:

- Timing Recovery
- Low pass Filtering
- Adaptive Equalization

Although not simulated, the actual DTS demodulation implementation contains several other important steps:

- Adaptive Gain Control
- Frequency Tracking
- Mode Recombination
- Data Un-Scrambling

These steps were not modeled in the simulation because it was felt that the benefits of inclusion wouldn't be worth the additional effort and complexity.

PART II -- Demodulation

3. Quadrature Amplitude Modulation

3.1 Purpose of QAM

Telemetry systems include modulation for two reasons:

1. Modulation transmits data in the "sweet spot" of the channel, thereby avoiding DC-coupling and high-frequency attenuation and non-linear group delay.
2. Modulation can achieve a higher bit rate than the bandwidth of the cable would otherwise allow. [Mayhugh 90, p. 1]

As excerpted from the original DTS Modem Proposal [PCSI 88-1, p. 7-8]:

> For bandlimited channels where signal-to-noise ratio is relatively high and the emphasis is on cost-effective and very efficient designs, the telephone line modem industry has amply proven that QAM is the modulation of choice. These conclusions have been derived over a period of 20 years, by thousands of workers and hundreds of organizations including such bodies as CCITT, AT&T, the Codex division of Motorola, and IBM. Literally every accepted modulation technique for the telephone channel which conveys in excess of 1 bit/second per Hz (i.e., 2400 bits/second and above) uses a QAM technique.
>
> A straight forward analysis of the logging cable channel shows that it is characterized by severely limited bandwidth (strong amplitude cutoff with accompanying delay distortion) and relatively high signal-to-noise ratios (better than 30 dB under reasonable worst case assumptions). This is completely analogous to the telephone line problem. The unfortunate fact of severe downlink to uplink coupling between cable modes forces the move towards time-compression modulation and half-duplex operation, but leaves intact the selection of the QAM schemes.

3.2 Theory Behind QAM

3.2.1 Transmit in Sweet Spot

One purpose of modulation is to send data through the "sweet spot" of the cable. DTS sends symbols at 70 kHz; if we imagine alternating between two symbols at 70 kHz, we see that the main frequency component of this signal is at 35 kHz. Assuming that the baseband signal is practically bandlimited at 35 kHz, we obtain the baseband frequency spectrum shown in Figure 3.2-1.

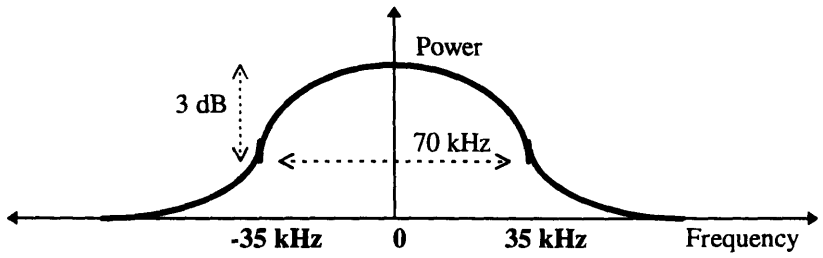


Figure 3.2-1: Frequency Spectrum of Baseband Signal

We do not want to transmit the signal at baseband. As can be seen in the above diagram, sending our baseband signal would mean that the transmitted signal has a DC component. To successfully decode a long string of a given symbol (represented and transmitted as a long time period of a given voltage level), the DC voltage level must remain fairly constant. This DC-coupling is usually impractical when dealing with a lossy transmission channel. [Mayhugh 90, p. 1]

We would like to move our 70 kHz band to a different location in the frequency spectrum, and thereby eliminate the need for DC coupling. This process is called modulation, and is accomplished by multiplying the data with a carrier sine wave. After modulating with a 52.5 kHz carrier (the choice of symbol and carrier frequencies is discussed in Sections 3.4.1 and 3.4.2), the new transmitted data spectrum is presented in Figure 3.2-2.

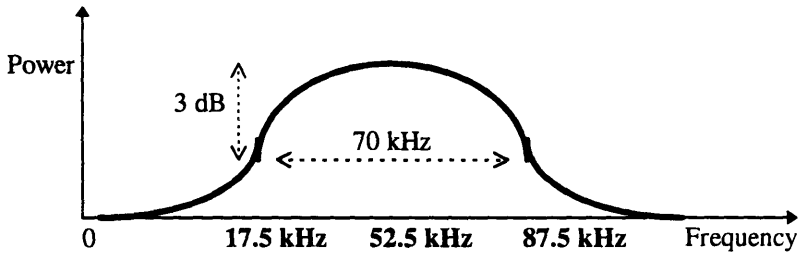


Figure 3.2-2: Frequency Spectrum of Modulated Signal [Mayhugh 90, p. 6]

3.2.2 Achieve High Bit Rate

QAM increases the bit rate in two ways: first, level encoding increases the effective bit rate for a given symbol rate by encoding multiple bits per symbol, and second, QAM doubles the effective symbol rate by allowing a single channel to carry two streams of information.

Until now, we have referred ambiguously to the 'symbols' that we would be transmitting, and not specified how these symbols relate to the bit stream we actually want to send. By choosing the symbols to represent a string of bits rather than the bits themselves, we can increase the bit rate (what we care about) without increasing the symbol rate (limited by the cable response). This increased bandwidth comes at the expense of signal-to-noise ratio. [Mayhugh 90, p. 5]

As mentioned, normal amplitude modulation increases the effective bit rate of a channel by encoding multiple bits per transmitted symbol. Instead of encoding one bit per symbol and having a symbol which takes one of two values, an amplitude modulated signal might encode 2 bits per symbol and have a symbol which takes one of four values, as shown in Figure 3.2-3.

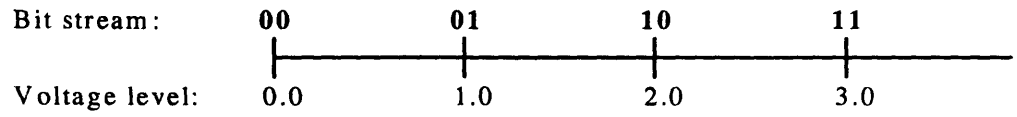


Figure 3.2-3: One-Dimensional 2 bit/symbol Signal Space

This level-encoding increases the bit rate by a factor of 2 without increasing the symbol rate. Alternatively, we can increase the bit rate by a factor of 3 by encoding 3 bits per symbol, as displayed in Figure 3.2-4.

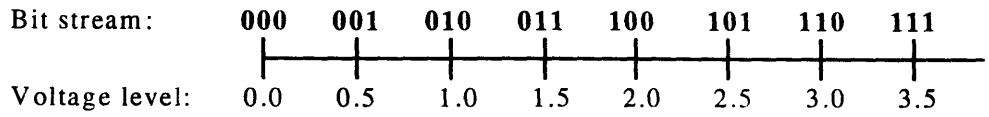


Figure 3.2-4: One-Dimensional 3 bit/symbol Signal Space

The increased channel capacity is gained by trading off signal-to-noise ratio (SNR); assuming that the voltage range is fixed, transmitted symbols are two times closer together in Figure 3.2-4 than they were in Figure 3.2-3. This means that noise which would have been harmless in the 2 bit/symbol signal space (for example, a +0.3V spike) could cause a decision error in the 3 bit/symbol system. Luckily, a typical logging cable channel has nearly an order of magnitude of excess SNR available. [Mayhugh 90, p. 4]

Quadrature Amplitude Modulation (QAM) extends the simple one-dimensional level encoding scheme to two dimensions. A two-dimensional signal space allows the number of symbols to be increased without linearly stacking additional voltage levels, as seen by comparing Figure 3.2-4 to Figure 3.2-5. [Mayhugh 90, p. 5] More symbols allow more bits to be encoded per symbol, so a higher bit rate can be achieved for the same symbol rate and SNR hit.

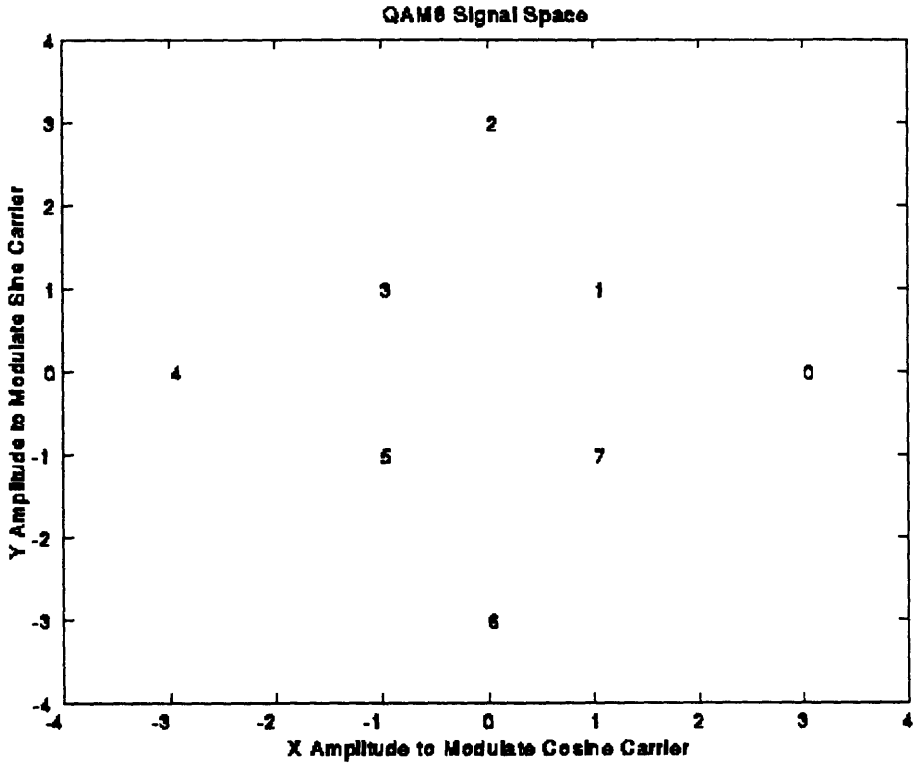


Figure 3.2-5: Two-Dimensional 3 bit/symbol Signal Space

Admittedly, it is not immediately clear how a symbol with two coordinates can be transmitted through a single channel without halving the symbol rate. The trick lies in taking advantage of negative frequencies and the imaginary frequency domain. As seen in Figure 3.2-2, an amplitude-modulated frequency spectrum contains redundant information: the negative frequencies carry the same spectrum as the positive frequencies. This redundancy is eliminated in QAM through the superposition of two orthogonal carriers. One coordinate is amplitude modulated onto each carrier -- the superposition allows the transmission of both coordinates at once, and the orthogonality allows the carriers to be separated at the receiver. [Samueli 94, p. 73]

Sine waves at the same frequency but quadrature phase are orthogonal to one another. We therefore choose one carrier to be a cosine at 52.5 kHz, and the other carrier to be a sine wave at 52.5 kHz. The cosine carrier is multiplied by the X-coordinate data while the sine carrier is multiplied by the Y-

coordinate data. These two waveforms are added together, then sent down the cable. At the receiver, the incoming signal is multiplied by the cosine carrier and low pass filtered to recover the X-coordinate data, and also multiplied by the sine carrier and low pass filtered to recover the Y-coordinate data. A flowchart of this process is shown in Figure 3.2-6. You need not blindly believe that this works -- we shall demonstrate the process in both the frequency and time domains in Section 3.3.

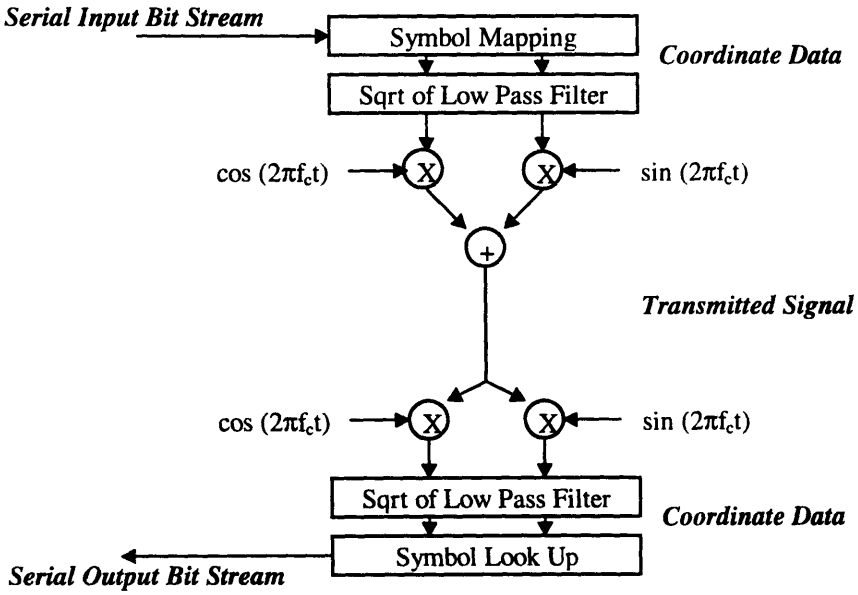


Figure 3.2-6: QAM Flow Diagram [Samueli 94, p. 74]

3.2.3 Bit Error Rates

As mentioned in Section 3.2.2, the addition of noise can cause a symbol to be incorrectly identified as a neighboring symbol. This is called a decision, or slicing, error. A telemetry system has many sources of noise, all in series with one another. Random white noise from the oil well environment during transmission, voltage spike noise from the power source, quantization noise, and unequalized cable response noise all contribute to potential slicing errors. Because the noise sources combine in series, any noise source which is 6 dB smaller (resulting in a SNR 4 times smaller) than another can be considered negligible. In this study we neglect to consider quantization noise and power spike noise: quantization

noise because it is intentionally made small by properly choosing the analog-to-digital converter, and power spike noise because it is too difficult to model. We will look at random white noise in Section 3.4.4, and unequalized cable noise in Section 6.3.3.

The possibility of a slicing error will always exist. Because the noise is random, there will always be a possibility (however small) that a moment of noise will be larger than half the distance between constellation points, and therefore cause an error. We can only quantify how often, on average, we are willing to accept slicing errors. The standard metric is Bit Error Rate (BER). [Wolaver 95, p. 89]

BER decreases with SNR because fewer errors occur when the noise variance decreases. As we would expect by comparing Figure 3.2-3 and Figure 3.2-4, the BER also decreases when fewer bits are encoded per symbol (assuming constant power). Basic probability allows us to find an equation for BER as a function of SNR and number of bits per symbol. If all sources of noise result in a SNR larger than a 2 or 3 dB implementation margin above what we calculate as required for the given bits/symbol, we can believe that on average we will make slicing errors below the specified BER. [PCSI 88-1, p. 18]

To derive the theoretical BER equation, we assume white Gaussian noise of mean μ and variance σ^2, with the probability density function as described by Equation 3.2-1. [Bingham 88, p. 71]

$$p(\mu) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-\mu^2}{2\sigma^2}}$$

Equation 3.2-1: Random Noise Distribution

We will derive the one-dimensional signal space equation before generalizing to two dimensions. If the noise is added at the input of the detector, and if the distance from any symbol to its nearest threshold (halfway to the next symbol) is normalized to unity, the probability of any particular symbol being inaccurately detected as one of its more positive neighbors is given in Equation 3.2-2. [Bingham 88, p. 72]

$$P(\mu > 1) = \frac{1}{\sigma\sqrt{2\pi}} \int_{+1}^{\infty} e^{\frac{-x^2}{2\sigma^2}} dx$$

Equation 3.2-2: Probability of Inaccurate Detection

Since each symbol of an L-level one-dimensional amplitude modulation system (with symbols at ± 1, ± 3, ..., ± (L-1)) has a probability of 1/L, and, except for the two outside levels, each symbol can err in two directions, the total probability of symbol error is given in Equation 3.2-3. [Bingham 88, p. 72]

$$P_E = \frac{2(L-1)}{L} \frac{1}{\sigma\sqrt{2\pi}} \int_{+1}^{\infty} e^{\frac{-x^2}{2\sigma^2}} dx$$

Equation 3.2-3: Total Probability of Symbol Error

It will be useful to rewrite this probability of error in terms of the function Q, as done in Equation 3.2-4. [Bingham 88, p. 72]

$$P_E = 2\left(1 - \frac{1}{L}\right) Q\left(\frac{1}{\sigma}\right)$$

$$Q(v) = \frac{1}{\sqrt{2\pi}} \int_{v}^{\infty} e^{\frac{-x^2}{2}} dx$$

$$\text{where} \quad v = \sqrt{\frac{3}{L^2 - 1} \frac{Signal\ Power}{Noise\ Power}}$$

$$= \sqrt{\frac{3}{L^2 - 1} SNR}$$

$$= \frac{1}{\sigma} \qquad \text{in this case}$$

Equation 3.2-4: Total Probability of Symbol Error in terms of Q

Since P_E is the symbol error rate, and since each symbol conveys $\log_2 L$ bits, the one-dimensional BER is simply the function given in Equation 3.2-5. The multiplier of $Q(1/\sigma)$ can be reasonably ignored for high SNR. [Bingham 88, p. 73]

$$BER = \frac{2(1-\frac{1}{L})}{\log_2 L} Q\left(\sqrt{\frac{3}{L^2-1} SNR}\right)$$

Equation 3.2-5: Bit Error Rate

BER vs. SNR is plotted in Figure 3.2-7 for the basic case of L=2, corresponding to binary signals in the one-dimensional case.

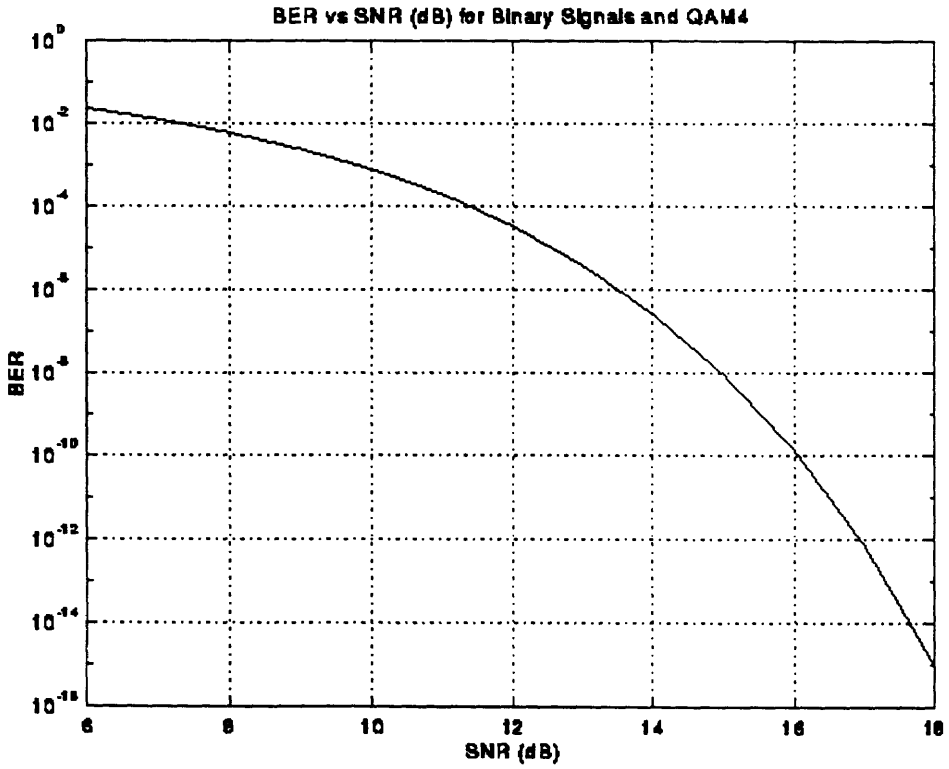


Figure 3.2-7: BER vs. SNR for L=2

For square two-dimensional constellations, the probability of error is twice the probability of that for a one-dimensional signal space because each symbol can now err to 4 nearest neighbors instead of just 2. Luckily, this extra factor of two is canceled in the bit error rate because each symbol now conveys 2 log $_2$ L bits instead of the previous log $_2$ L bits. [Bingham 88, p. 85] Assuming two-dimensional Gray encoding (in which each nearest neighbor symbol error causes one bit error), the square-constellation QAM BER is the same as the one-dimensional BER, given in Equation 3.2-5. The most important result is that the L=2 curve of Figure 3.2-7 is also the QAM4 curve.

For general two-dimensional constellations, the error rate in conditions of high SNR is mainly determined by the minimum distance between points and the number of points thus separated. The error rate for a general constellation can be approximated by Equation 3.2-6. [Bingham 88, p. 85]

$$P_E = K_1 \, Q\left(\sqrt{K_2 \, SNR}\right)$$

Equation 3.2-6: Approximation for Probability of Symbol Error

Using this approximation to derive a BER equation results in the ability to use the one-dimensional L=2 curve, if allowances are made for the SNR loss given in the following table [Bingham 88, p. 88]:

Constellation	Bits Per Symbol	L^2	K_1	K_2	SNR loss (dB)
8-pt integer star	3	8	0.33	0.60	4.4
4x4	4	16	0.75	0.45	7.0
32-pt cross	5	32	0.65	0.32	10.0
8x8	6	64	0.58	0.22	13.2

For example, a BER of 1E-7 requires an SNR of 14 dB at 2 bits per symbol = QAM4, reading off of Figure 3.2-7. QAM32 has much more closely spaced constellation points for the same average power, so requires (14 dB) + (10.0 dB) = 24 dB SNR to achieve a BER of 1E-7.

3.3 Mechanics of QAM

The purpose of QAM is to efficiently transmit bit stream information through a chosen frequency band. To demonstrate the mechanics of the QAM process, we shall modulate and demodulate an example bit stream in both the frequency and time domains.

We shall use the bit stream presented in Data 3.3-1 and graphed in Figure 3.3-1.

Data 3.3-1: Example Bit Stream: 00000 10001 00111 11000 11100 11110

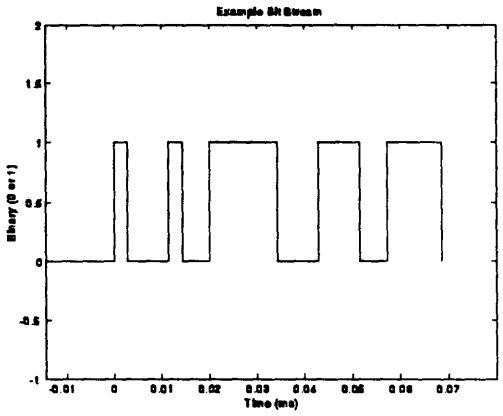


Figure 3.3-1: Example Bit Stream

For reference, the examples assume a symbol rate of 70 kHz and a 52.5 kHz carrier.

3.3.1 QAM Signal Spaces

The first step in quadrature amplitude modulation is to map the bit stream into a symbol stream. Because it is symbols rather than bits that will be transmitted, the more bits encoded per symbol, the

higher the resulting bit rate. Encoding N bits per symbol requires 2^N symbols. In a two-dimensional signal space, 2^N symbols requires at least $\sqrt{2^N}$ distinct voltage levels in each dimension. As the number of symbols increase, the spacing between symbols decreases (for a given average power or voltage range), and so a better SNR is needed to accurately distinguish between the constellation points.

This example will use a signal space containing 32 symbols; the literature calls this QAM32. Thirty-two symbols = 2^5, so five bits will be encoded per symbol. This particular signal space is not rectangular: the expected locations of the constellation points are rotated by 45° to minimize the maximum transmitted power (from the corner constellation points).

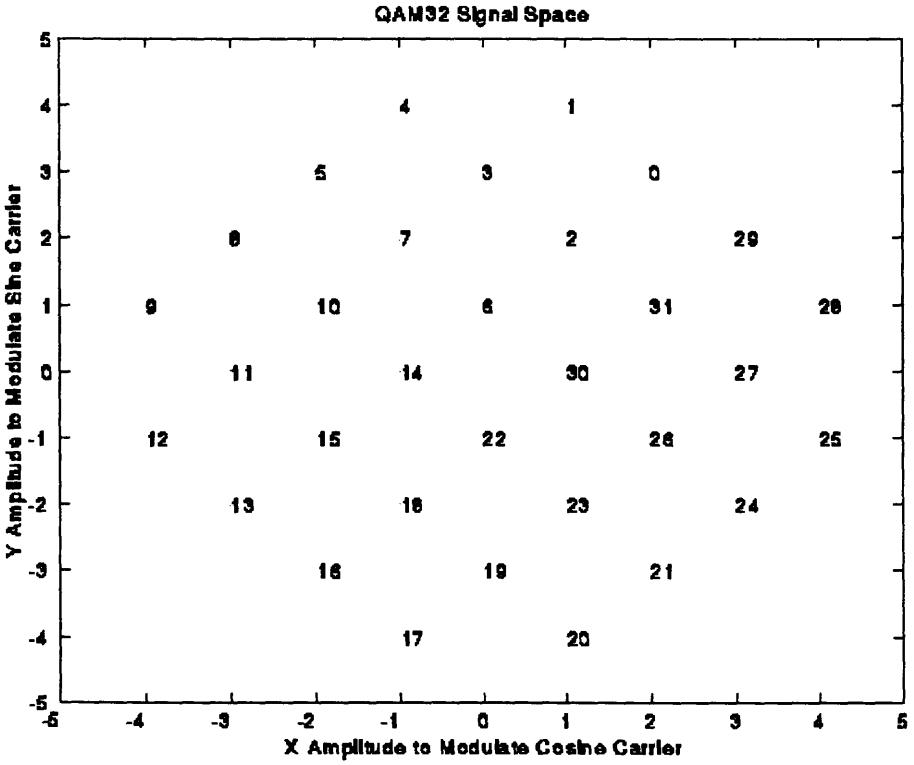


Figure 3.3-2: QAM32 Signal Space

The bit stream of Data 3.3-1 is translated into a QAM32 symbol stream simply by partitioning the bit stream every five bits and interpreting each five bit sequence as a binary number. The resulting symbol stream is presented in Data 3.3-2.

Data 3.3-2: Example Symbol Stream: 0 17 7 22 26 28

In QAM, symbol points are transmitted by transmitting their X and Y coordinates. The X and Y coordinate streams are generated by looking up each symbol in the chosen signal space. The coordinate streams generated from the symbol stream in Data 3.3-2 and the signal space of Figure 3.3-2 are shown in Data 3.3-3 and Data 3.3-4.

Data 3.3-3: Example X-Coordinate Stream: 2 −1 −1 0 2 4

Data 3.3-4: Example Y-Coordinate Stream : 3 −4 2 −1 −1 1

The X-coordinate stream (X data) will be sent on a cosine carrier while the Y-coordinate stream (Y data) is sent on a sine carrier. This modulation and the ensuing demodulation is done by multiplying the amplitude waveforms (the coordinate streams) by the carriers in the time domain, or by convolving in the frequency domain. We shall look at the mechanics of QAM in each domain.

3.3.2 Mechanics of QAM in the Frequency Domain

To look at the mechanics of QAM in the frequency domain we shall temporarily leave our example bit stream, and instead assume convenient frequency domain representations for the X and Y data. Because the coordinate data is purely real, its Fourier transform will be conjugate symmetric (with an even real part and odd imaginary part). To improve the clarity of our pictures, we will pretend that the coordinate data is bandlimited --- a poor approximation unless the data is low pass filtered, as we shall see when we try to demodulate in the time domain. To ease the gain calculations, the frequency peak will be

normalized to unity. Chosen under these constraints, the frequency representation we shall take for the X data is shown in Figure 3.3-3, and the frequency representation for the Y data is shown in Figure 3.3-4.

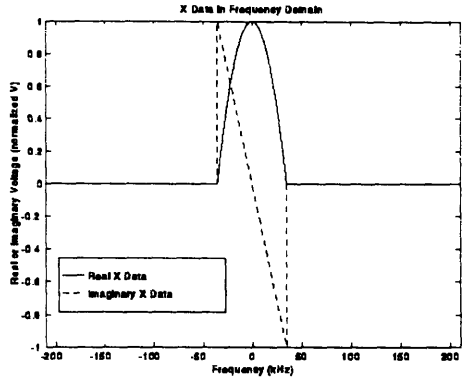


Figure 3.3-3: X Data in Frequency Domain

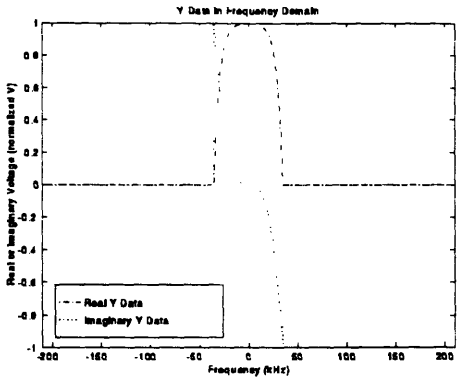


Figure 3.3-4: Y Data in Frequency Domain

The coordinate data is used as an amplitude waveform to modulate the transmitted carriers. Multiplying data by a sine wave in time is equivalent to convolving the frequency representation of the data with the frequency representation of a sine wave. The frequency representations of our cosine and sine carriers are shown in Figure 3.3-5.

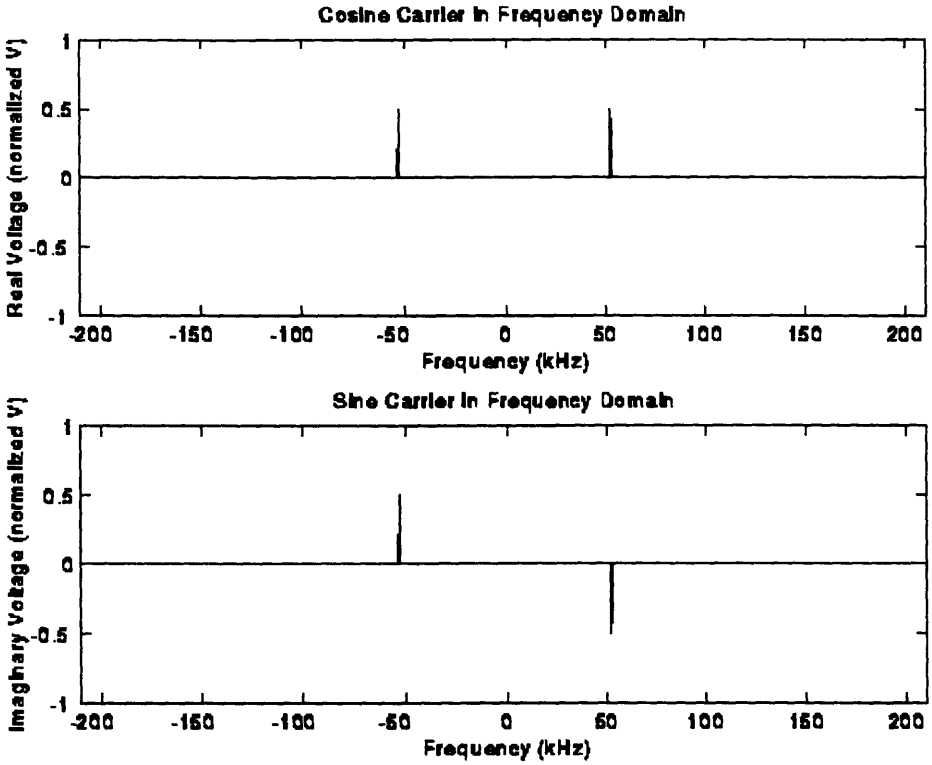


Figure 3.3-5: Cosine and Sine Carriers in Frequency Domain

Convolving with a scaled and shifted impulse simply scales and shifts the input. The result of convolving the X data of Figure 3.3-3 with the real cosine impulses at the top of Figure 3.3-5 is shown in Figure 3.3-6. The result of convolving the Y data of Figure 3.3-4 with the imaginary sine impulses at the bottom of Figure 3.3-5 is shown in Figure 3.3-7.

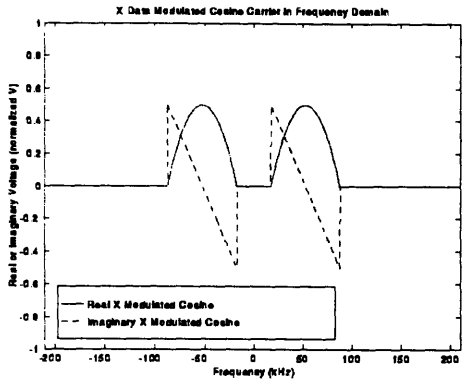


Figure 3.3-6: X Modulated Cosine Carrier in Frequency Domain

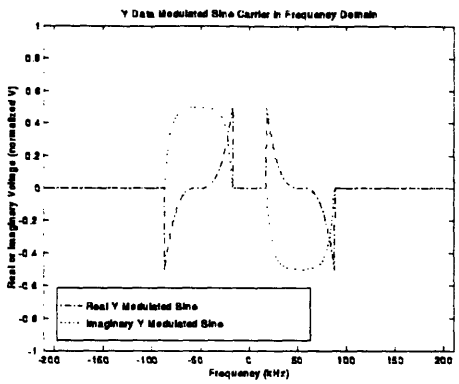


Figure 3.3-7: Y Modulated Sine Carrier in Frequency Domain

The QAM signal that is actually transmitted is a sum of the modulated spectra in Figure 3.3-6 and Figure 3.3-7. As seen in Figure 3.3-8, the transmitted signal appears to be a jumbled mess.

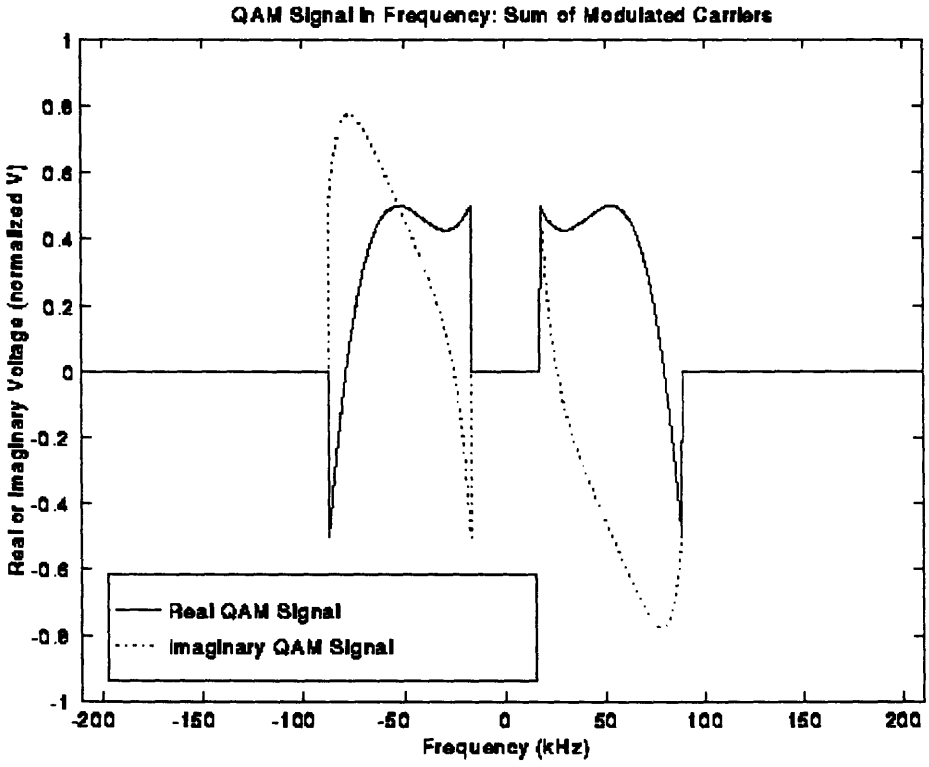


Figure 3.3-8: QAM Signal in Frequency: Sum of Modulated Carriers

Demodulating the transmitted signal is accomplished by using the sum of the modulated carriers to re-modulate the carriers. The X data is recovered by multiplying the time-domain QAM signal with a time-domain cosine wave, while the Y data is recovered by multiplying the time-domain QAM signal with a time-domain sine wave, as illustrated in Figure 3.2-6. This means that convolving the QAM signal in Figure 3.3-8 with the real cosine impulses at the top of Figure 3.3-5 should recover the frequency spectrum of Figure 3.3-3. Sure enough, a low pass filtering of Figure 3.3-9 returns the X data spectrum exactly. Similarly, convolving the QAM signal in Figure 3.3-8 with the imaginary sine impulses at the bottom of Figure 3.3-5 results in Figure 3.3-10: a low pass filter returns the Y data frequency spectrum of Figure 3.3-4. Mission accomplished.

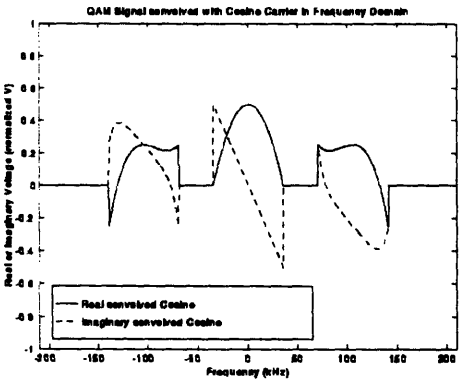


Figure 3.3-9: QAM Signal convolved with Cosine Carrier in Frequency Domain

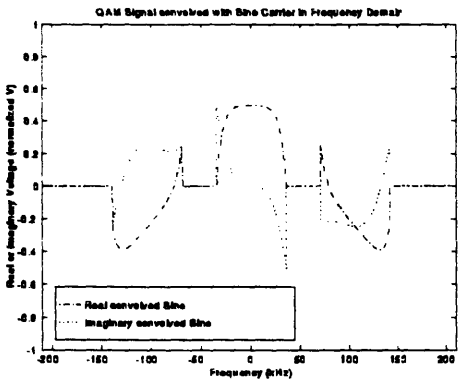


Figure 3.3-10: QAM Signal convolved with Sine Carrier in Frequency Domain

3.3.3 Mechanics of QAM in the Time Domain

It will be instructive to go through the quadrature amplitude modulation and demodulation process again in the time domain: we will not only convince ourselves that it works, but gain a feeling for what we would actually see on an oscilloscope.

Returning to the bit stream example introduced in Data 3.3-1, we plot the modulating X-coordinate amplitude stream, Data 3.3-3, in Figure 3.3-11 and the modulating Y-coordinate amplitude

stream, Data 3.3-4, in Figure 3.3-12. The carrier cosine and sine waves are superimposed on the modulating waveforms. The carriers are scaled to the height of the largest coordinate to ensure that the carriers are not over-modulated, though the carrier scaling is divided away in all figures except the first two.

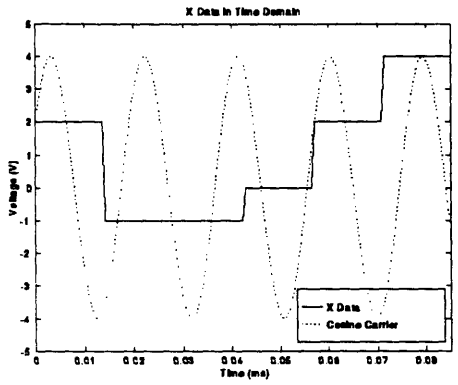


Figure 3.3-11: X Data in Time Domain

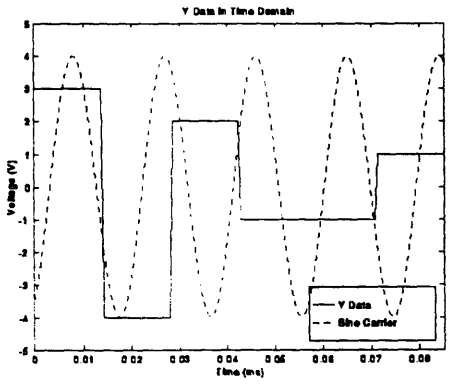


Figure 3.3-12: Y Data in Time Domain

The results of multiplying the coordinate data with the carriers plotted in Figure 3.3-11 and Figure 3.3-12 are shown in Figure 3.3-13.

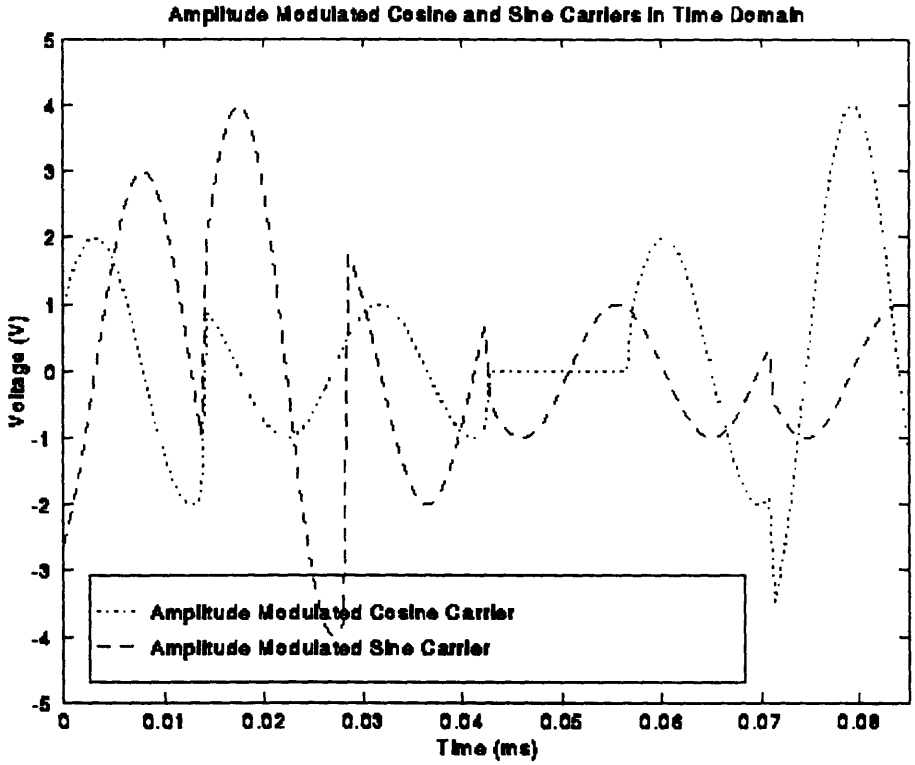


Figure 3.3-13: Amplitude Modulated Cosine and Sine Carriers in Time Domain

The QAM signal that is actually transmitted is the sum of the modulated carriers in Figure 3.3-13. An oscilloscope trace of the transmitted signal would look like Figure 3.3-14. Again, the transmitted signal appears to be a jumbled mess.

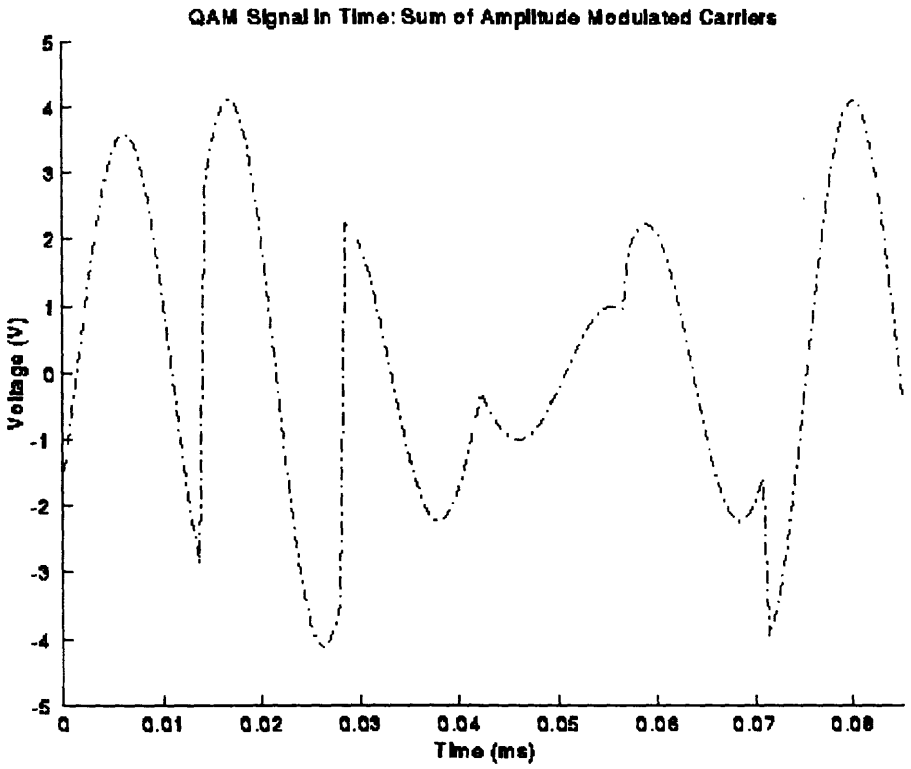


Figure 3.3-14: QAM Signal in Time: Sum of Modulated Carriers

The first step in demodulating this signal is to multiply it by the carrier sine waves. The QAM signal of Figure 3.3-14 times the cosine carrier is shown in Figure 3.3-15, and the QAM signal times the sine carrier is shown in Figure 3.3-16.

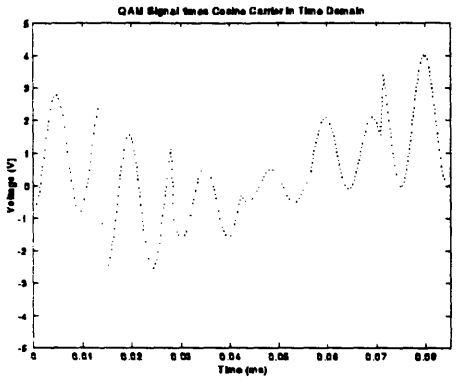


Figure 3.3-15: QAM Signal times Cosine Carrier in Time Domain

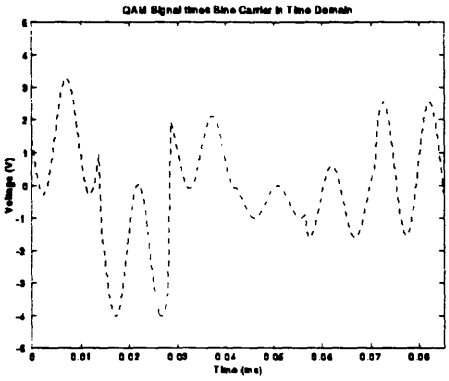


Figure 3.3-16: QAM Signal times Sine Carrier in Time Domain

These waveforms do not yet look like the coordinate data. As we saw in the frequency domain representations of Figure 3.3-9 and Figure 3.3-10, the signals have to be low pass filtered before the original coordinate spectra are recovered. The frequency representation of an ideal low pass filter is a sinc function. The time-domain filter with zero-crossings chosen to ensure a proper frequency passband is shown in Figure 3.3-17.

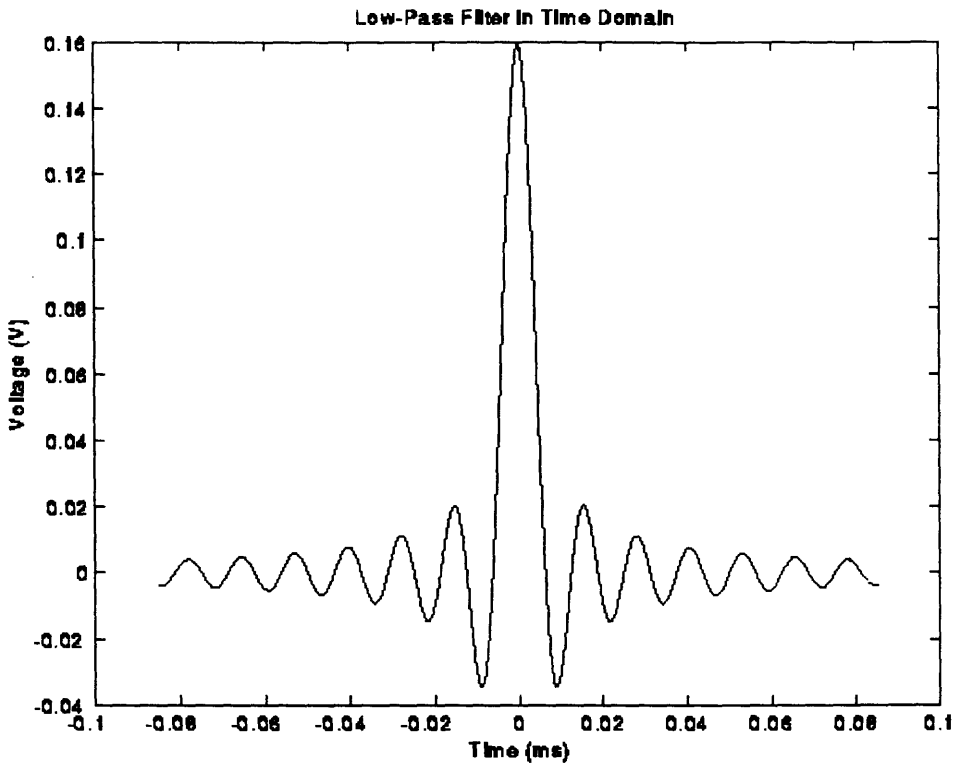


Figure 3.3-17: Low-Pass Filter in Time Domain

The result of convolving Figure 3.3-15 with the low pass filter of Figure 3.3-17 is shown in Figure 3.3-18. The waveform looks similar to the X data of Figure 3.3-11, but not exactly the same. The differences are due to the short time duration and infinite frequency components of our X data; it is not bandlimited and so the modulation has caused spectrum overlap. As we will discuss in Section 5, the X data should be low pass filtered before modulation (as well as after) to prevent overlap and aliasing. The result of convolving Figure 3.3-16 with the low pass filter is shown in Figure 3.3-19; again this looks similar to the Y data of Figure 3.3-12. Applying an ideal low pass filter before modulation would have enabled us to recover the data exactly.

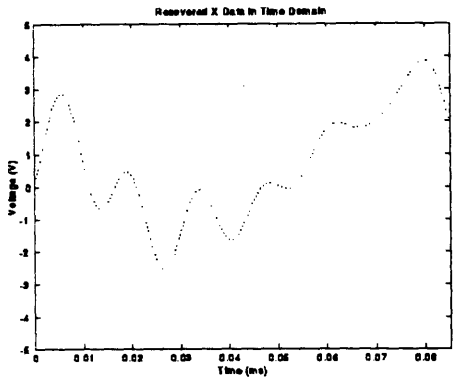


Figure 3.3-18: Recovered X Data in Time Domain

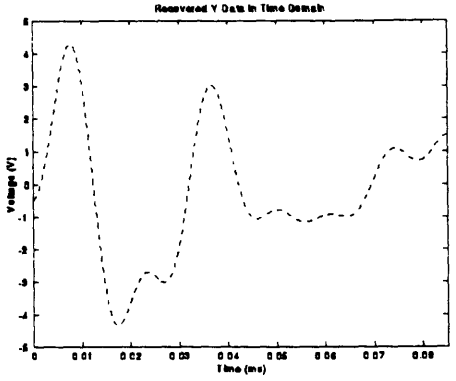


Figure 3.3-19: Recovered Y Data in Time Domain

3.4 DTS Implementation of QAM

The Digital Telemetry System uses QAM to encode uplink information. The carrier frequency is 52.5 kHz, and the symbol rate is 70 ksymbols/second. Each symbol may be set to represent either 3, 4, 5, or 6 bits. With 6 bits per symbol, DTS can transmit 420 kbits/s of raw data over a logging cable which only has a 70 kHz bandwidth. DTS operates two QAM uplink channels simultaneously to double the maximum raw data rate to 840 kbits/s. [Mayhugh 90, p. 6]

3.4.1 Sampling, Symbol, and Carrier Frequencies

As described and illustrated above, all bandlimited modulated information can be recovered exactly by multiplying by the carriers and a low pass filter. Unfortunately, this process is too computationally intensive to be used in DTS; DSPs are not fast enough to convolve a long sin(x)/x filter with the 210 ksamples/second data. Instead, Schlumberger developed and patented a technique to eliminate the need for signal reconstruction. [Montgomery 93]

The two carriers are at same frequency but 90° out of phase; when one carrier is at zero, the other is at its peak. DTS separates the carriers by sampling the signal four times a period: at the zero-crossings

of either carrier. A voltage measurement made at the zero-crossing of one carrier represents the peak amplitude of the other carrier. Because of this, alternate samples (modulo a periodic sign change) plot the amplitude data of each carrier. [DTS 91, p. 3-9]

The cosine and sine carriers are both at 52.5 kHz. Sampling at each of the two zero-crossings of each of the two carriers requires sampling at 4 * 52.5 kHz = 210 kHz. The symbol rate is defined to be 70 kHz; one-third of the 210 kHz sampling rate. Therefore, 3 of the 4 samples taken during one period of a carrier will have been made during one particular symbol. Of the group of 3, either 1 or 2 samples are zero-crossings and will be discarded, so 1 or 2 samples will represent a single symbol [DTS 91, p. 3-9]

The example of Section 3.2.3 is continued here. Instead of multiplying the QAM signal in Figure 3.3-14 with a cosine carrier and a sine carrier and then convolving with a sinc function, the X and Y data are recovered through synchronized sampling at a multiple of the carrier and symbol frequencies, as shown in Figure 3.4-20.

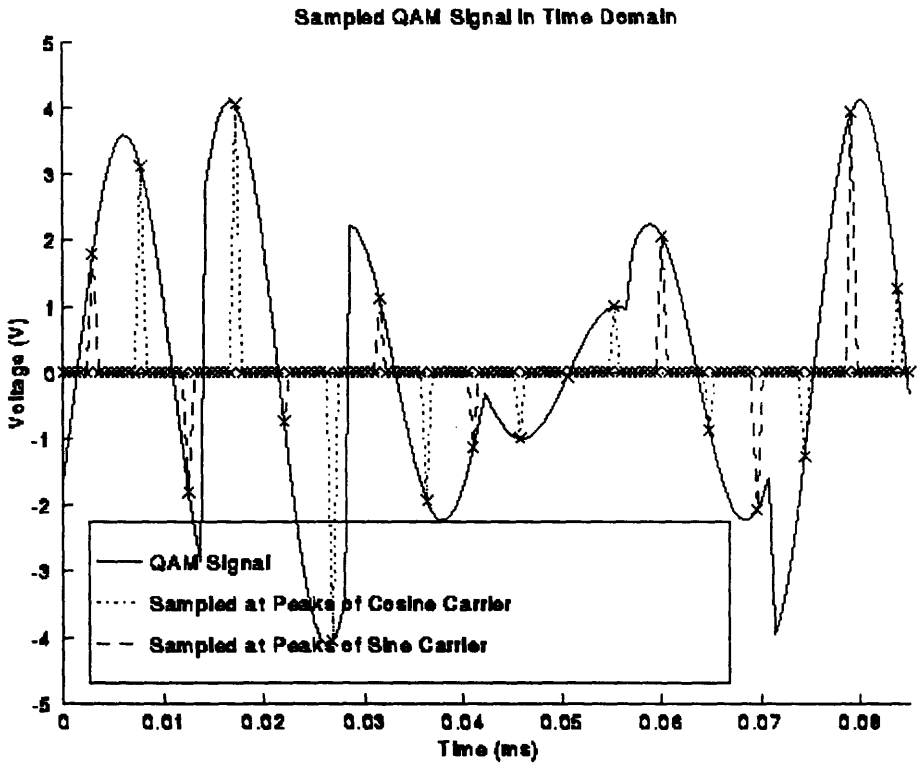


Figure 3.4-20: DTS Sampling of QAM Signal in Time

Sampling the 70 kHz symbols on a 52.5 kHz carrier at 210 kHz results in a 210 kHz sample stream. Alternate samples plot the amplitude data of each carrier (after removing the carrier's sign), so the 210 kHz sample stream separates into a 105 kHz X data stream and a 105 kHz Y data stream.

DTS does try to identify which samples come from which symbols. As explained in Section 5, the X-data (or Y-data) coordinate stream is recovered by convolving all 105 kHz samples with a low pass filter and sampling the output at 70 kHz.

3.4.2 Sweet Spot of the Cable

We are modulating our signal to move the transmitted data to a different location in the frequency spectrum. We chose the proportions between the carrier, symbol, and sample rates (3:4:12) to

reduce computation. Before proceeding, we must confirm that the absolute frequencies we've chosen, 52.5 kHz, 70 kHz, and 210 kHz respectively, do indeed cause us to transmit in the "sweet spot" of our channel. A quick look heptacable characterizations (reproduced in Figure 3.4-21) shows that the usable bandwidth of our channel lies between 5 and 90 kHz. [Montgomery 93, Background of Invention]

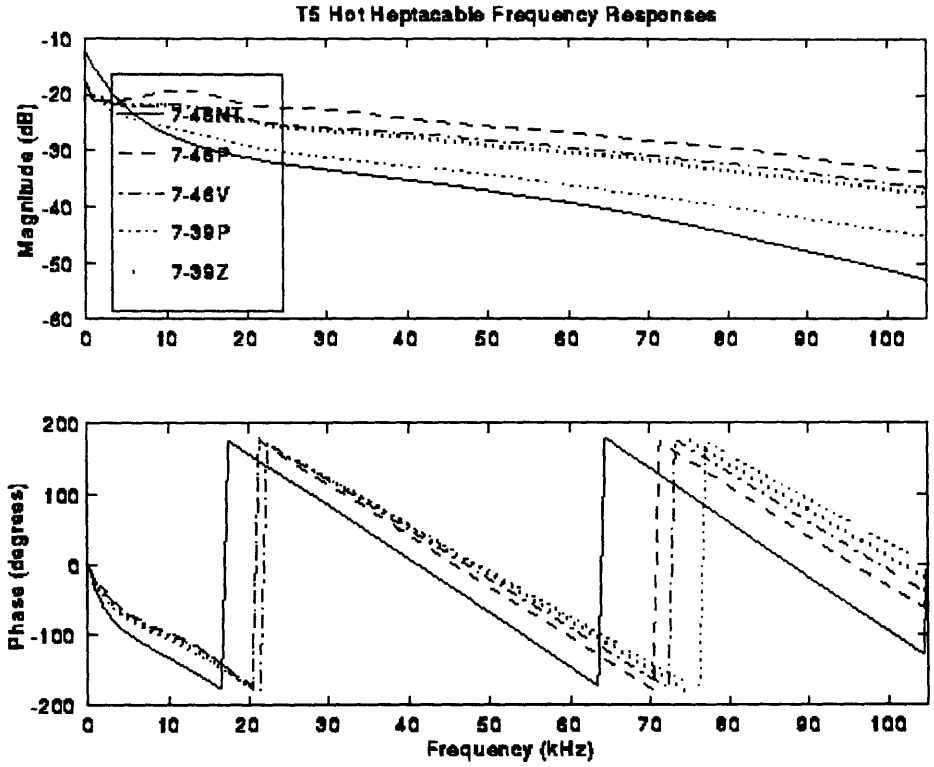


Figure 3.4-21: Heptacable Frequency Response

The carrier and symbol frequencies chosen in DTS result in transmission of a 70 kHz band around 52.5 kHz, so we are interested in cable loss and group delay between 17.5 kHz and 87.5 kHz -- inside the range of usable bandwidth. While there is definitely attenuation and delay in the band of interest, an adaptive equalizer (discussed in Section 6) is able to reverse this cable distortion.

3.4.3 Signal Spaces

DTS supports 3, 4, 5, and 6 bit per symbol encoding. This corresponds to QAM8, QAM16, QAM32, and QAM64 signal spaces, respectively. The default encoding is QAM32; smaller signal spaces are used for particularly bad cable SNRs, and the field engineer can choose QAM64 if higher data rates are needed on a very clean cable with a high SNR. [Montgomery 93, Description IIB] The signal spaces are shown in Figure 3.4-22 through Figure 3.4-25.

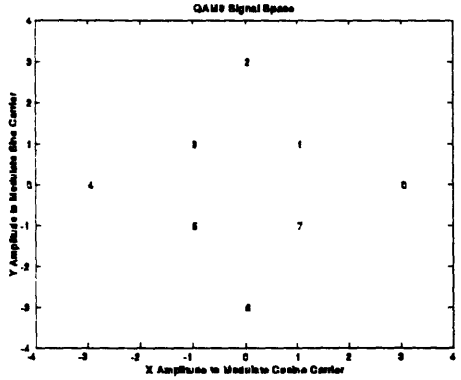


Figure 3.4-22: QAM8 Signal Space

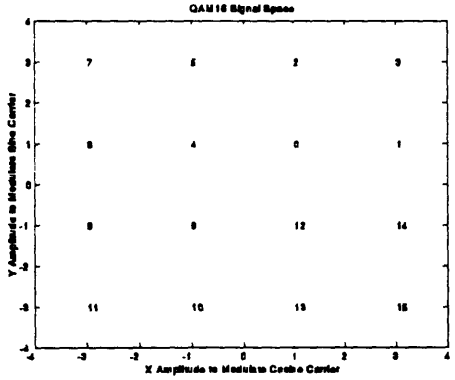


Figure 3.4-24: QAM16 Signal Space

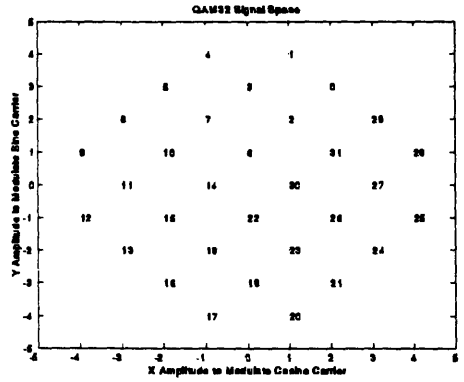


Figure 3.4-23: QAM32 Signal Space

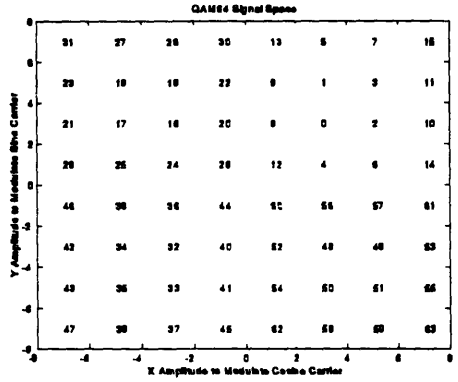


Figure 3.4-25: QAM64 Signal Space

3.4.4 Bit Error Rates

In order to assure a vanishing possibility of more than three errors in 800 hours of operation, it was decided that DTS should have a bit error rate no greater than 1E-7. The following required SNR values were derived from the graphs and charts of Section 3.2.3 [PCSI 88-1, p. 18]:

Bits/Symbol	Theoretically Required SNR (dB)
2	14
3	17.5
4	21
5	24
6	27

Assuming that the downhole transmitter delivers 2 Watts rms power to the cable, 28.3 dB is the worst case SNR due to random white noise expected on 7-46NT cable in T5 mode over 30 000 ft with an average cable temperature of 125 °C. [PCSI 88-1, p. 18] Given an implementation margin of 2 to 3 dB, it appears that worst case SNR will allow a BER of 1E-7 at our default of 5 bits/symbol, but not necessarily at 6 bits/symbol. Happily, we will usually not be dealing with worst case SNR, so 6 bits/symbol should also work most of the time.

4. Timing Recovery

4.1 Purpose of Timing Recovery

As we saw in Section 3.3, to demodulate the transmitted signal we need to multiply with a sine and a cosine wave at the same frequency and phase as the sine and cosine waves used to modulate the signal. This would be straight forward if a timing signal (containing information about carrier, symbol, and sample frequencies and phases) were sent with the modulated signal -- unfortunately, the timing signal would use up much of the precious channel. Instead, it is possible to derive the timing signals by appropriately filtering the modulated signal.

The optimal sampling time is that which maximizes the SNR of the sampled signal. Since the power of the sampled noise does not depend on the sampling time, the SNR is maximized by maximizing the power of the sampled signal. [Bingham 88, p. 211]

Because DTS assumes certain proportional relationships between carrier, symbol, and sample frequencies and phases, obtaining a lock on sample frequency and phase provides information about the carrier and symbol rates.

4.2 Theory behind Timing Recovery

To analytically determine the delay which maximizes the power of the sampled signal, we must first come up with an equation for the sampled signal.

For any signal s(t), with transform S(f), sampled at $t = nT+\tau$, the spectrum of the sampled signal is a superposition of an infinite number of shifted, phase-altered spectrums, as shown in Equation 4.2-1. [Bingham 88, p. 60]

$$S(f,\tau) = \sum_{k=-\infty}^{+\infty} S(f + kf_s)e^{j2\pi(f+kf_s)\tau}$$

Equation 4.2-1: Spectrum of s(t) sampled at $t = nT+\tau$

If we assume that s(t) is bandlimited to |f|<fs, as we assure by low pass filtering in Section 5, then only adjacent superimposing segments overlap. The sampled spectrum can therefore be defined on its primary period, as it is in Equation 4.2-2. [Bingham 88, p. 61]

$$S(f,\tau) = S(f - f_s)e^{j2\pi(f-f_s)\tau} + S(f)e^{j2\pi f\tau} + S(f + f_s)e^{j2\pi(f+f_s)\tau} \quad for\ |f| < \frac{f_s}{2}$$

Equation 4.2-2: Spectrum of Sampled s(t) over Primary Period

This equation is rewritten in Equation 4.2-3 to include the magnitudes and phases of the components explicitly. [Bingham 88, p. 216]

$$S(f,\tau) = |S(f - f_s)|e^{j[2\pi(f-f_s)\tau+\theta(f-f_z)]} + |S(f)|e^{j[2\pi f\tau+\theta(f-f_z)]} + |S(f + f_s)|e^{j[2\pi(f+f_s)\tau+\theta(f-f_z)]} \quad for\ |f| < \frac{f_s}{2}$$

Equation 4.2-3: Magnitude and Phase of Sampled s(t)

The spectrum of the sampled power is given in Equation 4.2-4.

$$|S(f,\tau)|^2 = \begin{cases} |S(f)|^2 + |S(f - f_s)|^2 + 2|S(f)S(f - f_s)|\cos[2\pi f_s\tau + \theta(f) - \theta(f - f_s)] & for\ f > 0 \\ |S(f)|^2 + |S(f + f_s)|^2 + 2|S(f)S(f + f_s)|\cos[2\pi f_s\tau - \theta(f) + \theta(f + f_s)] & for\ f < 0 \end{cases}$$

Equation 4.2-4: Spectrum of the Sampled Power of s(t)

If the delay of the channel were equalized before sampling, or, equivalently, the sampling time could be chosen separately for each frequency component, the argument of the cosines would be zero. Because this is not possible, sampling causes us to lose the power identified in Equation 4.2-5.

$$\Delta P(f,\tau) = \begin{cases} 2\left|S(f)S(f-f_s)\right|\{1-\cos[2\pi f_s\tau + \theta(f) - \theta(f-f_s)]\} & for\ f > 0 \\ 2\left|S(f)S(f+f_s)\right|\{1-\cos[2\pi f_s\tau - \theta(f) + \theta(f+f_s)]\} & for\ f < 0 \end{cases}$$

Equation 4.2-5: Power Lost due to Sampling

The amplitude S(f)S(f-fs) in Equation 4.2-5 is usually greatest at the band edges, |fs/2|. Equation 4.2-6 expands θ in a Taylor series around the band edges in φ, the argument of the cosine. [Bingham 88, p. 217]

$$\phi = 2\pi f_s\tau + \theta(f_s/2) - \theta(-f_s/2) + (d\theta_+ - d\theta_-)(f \pm f_s/2) + (d^2\theta_+ - d^2\theta_-)\frac{(f \mp f_s/2)^2}{2}$$

Equation 4.2-6: Argument of Cosine in Power Lost Equation

The notation indicates the first and second derivatives of θ evaluated at the upper and lower band edges.

We want the power lost to be as small as possible, so we want cosine to be close to 1, so we want the argument of the cosine to be close to zero. As a first approximation, this is achieved by setting the sampling delay according to Equation 4.2-7.

$$\tau = \frac{\theta(-f_s/2) - \theta(f_s/2)}{2\pi f_s}$$

Equation 4.2-7: Optimal Sampling Delay

By sampling the transmitted signal at t = nT+τ, where τ is proportional to the difference in the signal phase at the band edges, we maximize the power of the sampled signal and maximize the SNR.

4.3 DTS Implementation of Timing Recovery

In DTS, timing recovery is done in the passband, before demodulation. This requires a slight modification in Equation 4.2-7: the band edges are now centered around fc rather than zero, as updated in Equation 4.3-1. [PCSI 88-1, p. 7]

$$\tau = \frac{\theta(f_c - f_s/2) - \theta(f_c + f_s/2)}{2\pi f_s}$$

Equation 4.3-1: Optimal Sampling Delay in Passband

Unfortunately, Equation 4.3-1 is not directly useful for a practical implementation. The numerator requires knowledge of the phase of the incoming signal; the samples occur too rapidly for such calculations. Instead, it is common practice to use a technique called Band-Edge Component Maximization (BECM) to establish timing. [Bingham 88, p. 210]

As illustrated in Figure 4.3-1, BECM involves three bandpass filters. The signal goes through a filter centered at fc + fs/2 and through a filter centered at fc - fs/2 -- the two passband band edges. The outputs of these filters are multiplied together in time, and then passed through a final filter centered at fs. The output of the final filter is monitored. The phase of the original sampling is gradually shifted until the sampling times correspond to zero-crossings of the output of the final filter. We will now look at how this works.

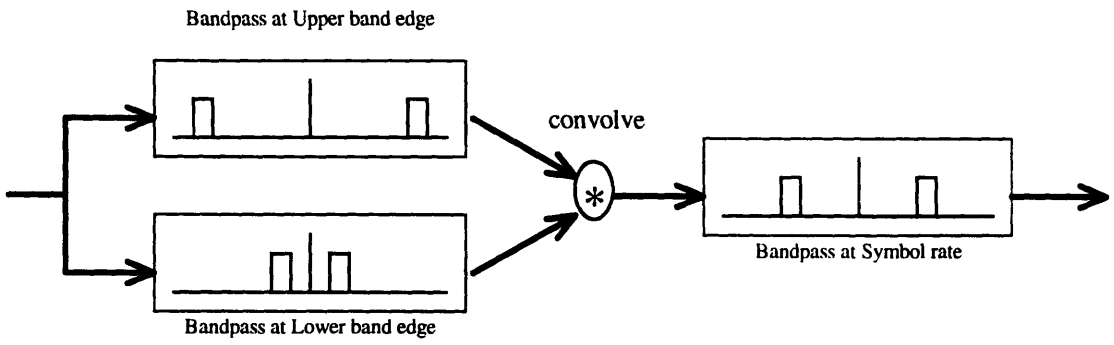


Figure 4.3-1: BECM Filters [PCSI 88-1, p. 29]

It is useful to look at the filtering operations in the frequency domain. Let us call the signal coming into the BECM system X. A narrow bandpass filter centered at fc + fs/2 will leave impulses of spectrum at fc + fs/2, and at -(fc + fs/2), as shown in Figure 4.3-2. Similarly, a narrow bandpass filter centered at fc - fs/2 will leave impulses of spectrum at fc - fs/2, and at - (fc - fs/2), as shown in Figure 4.3-3.

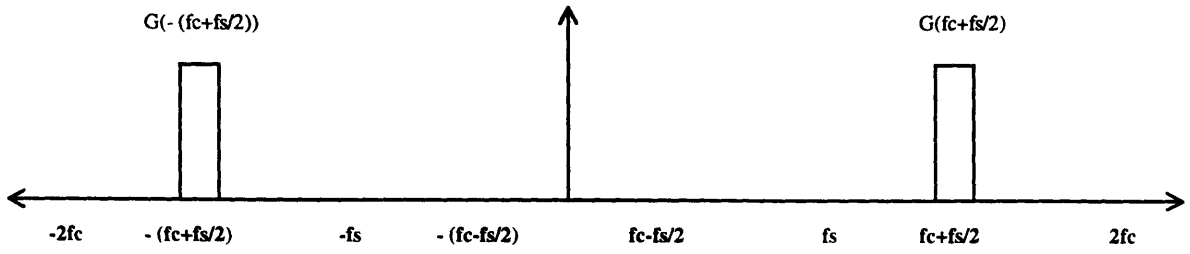


Figure 4.3-2: Signal after |fc + fs/2| Bandpass Filter

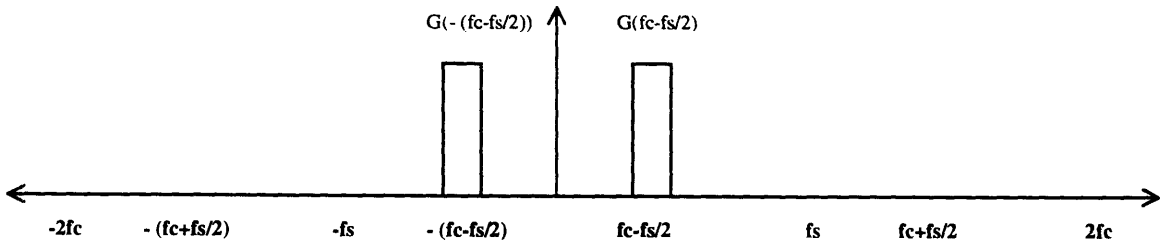


Figure 4.3-3: Signal after |fc - fs/2| Bandpass Filter

Multiplication in the time domain is equivalent to convolution in the frequency domain. Convolution between a superposition of scaled and shifted impulses is but a superposition of different scaled and shifted impulses. Convolving Figure 4.3-2 with Figure 4.3-3 results in Figure 4.3-4. As we would expect, the result has power both at |fs| and at other frequencies. The final bandpass filter isolates the components at fs and -fs.

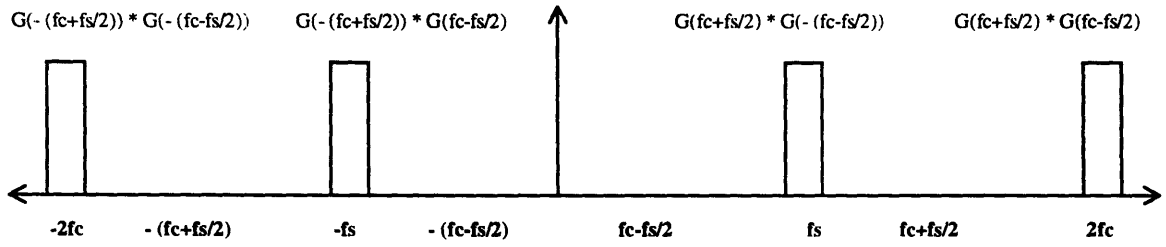


Figure 4.3-4: Signal after Convolving Bandpass Filters

The goal is to have an incoming signal X which causes the output of the final filter to cross zero. In crossing zero, the output will necessarily have had to have zero phase. As seen in Figure 4.3-4, the phase of the output is $\angle$ X(fc + fs/2) + $\angle$ X(-fc + fs/2). The signal X is simply the frequency representation of original real data convolved with a real carrier and then a real cable impulse response -- it is only after demodulation that the time-domain signal will have an imaginary component. Therefore, the time domain pair of X will be real, so X must be conjugate symmetric. Conjugate symmetric implies

an odd phase, so $\angle X(-fc + fs/2) = -\angle X(fc - fs/2)$. The phase of the filter output can be written as $\angle X(fc + fs/2) - \angle X(fc - fs/2)$, familiar as (modulo a sign) the numerator of Equation 4.3-1! As seen from the equation, when the phase is zero, no additional timing delay is necessary.

To ensure proper timing, the sampling clock is gradually shifted forwards or backwards in time based on the sign of the filter output. Sampling occurs with the proper delay when the filter output crosses zero.

It is interesting to look at the actual filters themselves. All the BECM filtering is done digitally at 210 kHz. To ease the computational burden, DTM takes advantage of symmetries to reduce the number of calculations that must be done for each filter output. There are two sets of bandpass filters: low Q filters are used initially, when timing may be quite inaccurate and the spectra blurred. High Q filters are used later, during steady state, when tracking adjustments are expected to be minimal. The low Q filters are plotted for the 17.5 kHz, 87.5 kHz, and 70.0 kHz bands, respectively, in Figure 4.3-5, Figure 4.3-7, and Figure 4.3-9. The high Q filters appear opposite, in Figure 4.3-6, Figure 4.3-8, and Figure 4.3-10.

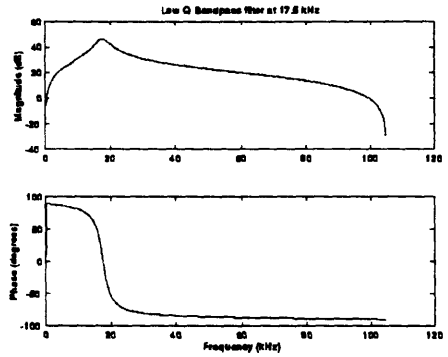


Figure 4.3-5: Low Q Bandpass filter at 17.5 kHz

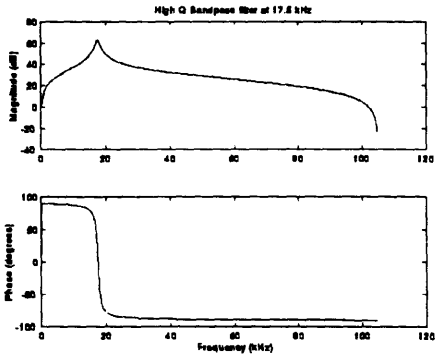


Figure 4.3-6: High Q Bandpass filter at 17.5 kHz

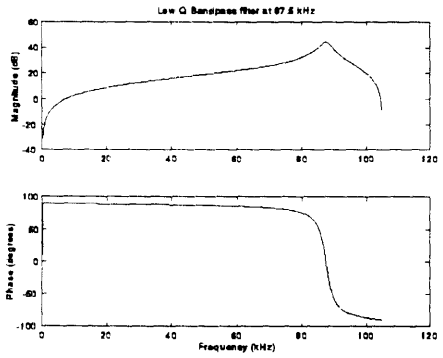


Figure 4.3-7: Low Q Bandpass filter at 87.5 kHz

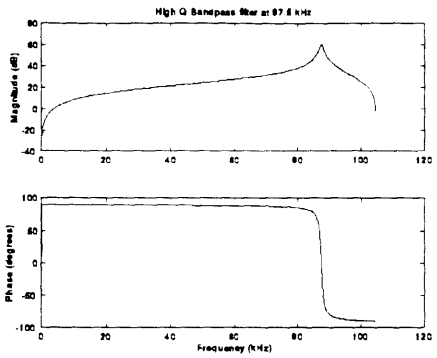


Figure 4.3-8: High Q Bandpass filter at 87.5 kHz

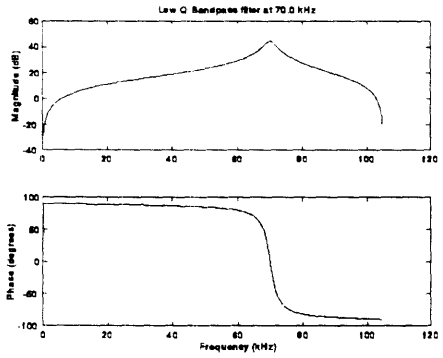


Figure 4.3-9: Low Q Bandpass Filter at 70.0 kHz

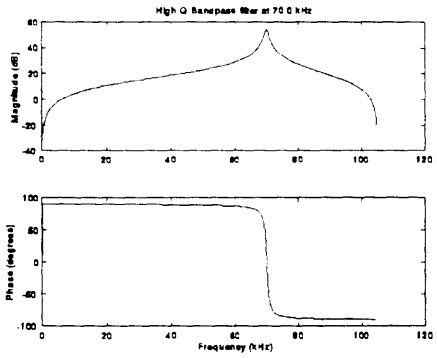


Figure 4.3-10: High Q Bandpass filter at 70.0 kHz

5. Low Pass Filter

5.1 Purpose of Low Pass Filter

The time domain modulation and demodulation we performed in Section 3.3.3 was not able to perfectly recover the original data. The original data, in the example, was a superposition of step functions, bandlimited in time. Bandlimited in time implies an infinite bandwidth in frequency. As can be seen in frequency domain steps of Section 3.3.2, modulating data with a bandwidth wider than the carrier frequency causes frequency overlap; the data can not be successfully recovered. For this reason, we must ensure that the data we modulate is bandlimited to the carrier frequency.

Assuming the DTS symbol frequency of 70 kHz, we can imagine that the original data is predominantly a square wave at 70 kHz. The strongest frequency component of this signal is at half the symbol frequency, or 35 kHz. This frequency component is significantly below the 52.5 kHz carrier frequency, so it should be possible to filter out the high frequency components of the original data and still preserve most of its information.

We must be careful in the design of our low pass filter. At an abstract level, low pass filters remove high frequency components, and thus blur the data. Applying a low pass filter to our data will blur one symbol into another, creating inter-symbol interference (ISI). [Bingham 88, p. 57] This blurring acts as noise: if severe enough, it could cause errors in symbol slicing.

Although a low pass filter will blur data, if we know ahead of time that we will be sampling the data and only observing it as certain times, we can arrange that those particular sample times are not subjected to ISI. We can thereby achieve a low pass filter which will not cause slicing errors.

5.2 Theory Behind Low Pass Filter

The time representation of an ideal low pass filter is a sinc function. By choosing the bandwidth of the filter such that the zeros of the sinc function occur at the symbol rate, we insure that the convolution of a given symbol with the low pass filter includes no contributions from neighboring symbols, and thus avoids inter-symbol interference. Such filters are called Nyquist filters. [Bingham 88, p. 58]

Unfortunately, an ideal low pass filter actually transforms to an infinitely long sinc function. Such a time convolution is not physically possible, so we must be willing to accept a truncated time filter with its associated non-ideal stop band.

A commonly used Nyquist filter is the raised-cosine. [Bingham 88, p. 62] The parameter α is used to determine the amount of excess bandwidth $\alpha fs/2$ of the filter. An α of 0.25 allows the passage of frequencies 25% higher than would an ideal filter. [PCSI 88-1, p. 6]

The frequency response of the filter is given in Equation 5.2-1 and plotted in Figure 5.2-1, while the filter impulse response is given in Equation 5.2-2 and plotted in Figure 5.2-2. [Bingham 88, p. 62]

$$G(f) = \begin{cases} T & for\ |f| < (1-\alpha)\frac{f_s}{2} \\ T\ \frac{1-\sin(\pi(f-f_s/2)/\alpha\ f_s)}{2} & for\ (1-\alpha)\frac{f_s}{2} < |f| < (1+\alpha)\frac{f_s}{2} \end{cases}$$

Equation 5.2-1: Low Pass Filter in Frequency Domain

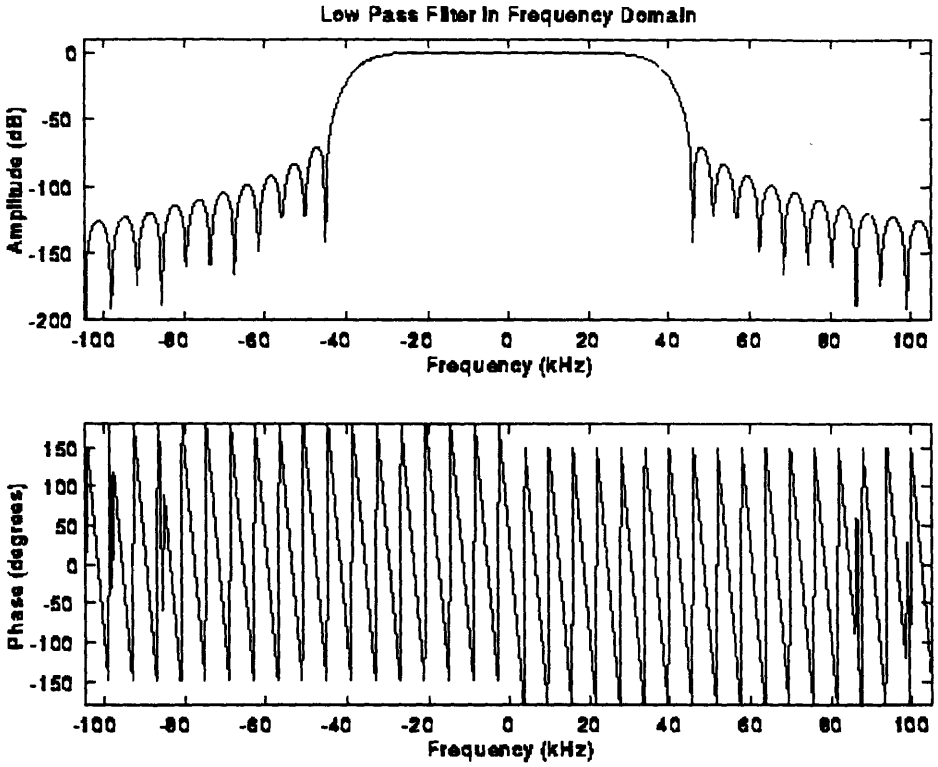


Figure 5.2-1: Frequency Response of Lowpass Filter [PCSI 88-1, p. 6]

$$g(t) = \mathrm{sinc}\left(\frac{t}{T}\right)\frac{\cos(\alpha\pi t/T)}{1-(2\alpha t/T)^2}$$

Equation 5.2-2: Low Pass Filter in Time Domain

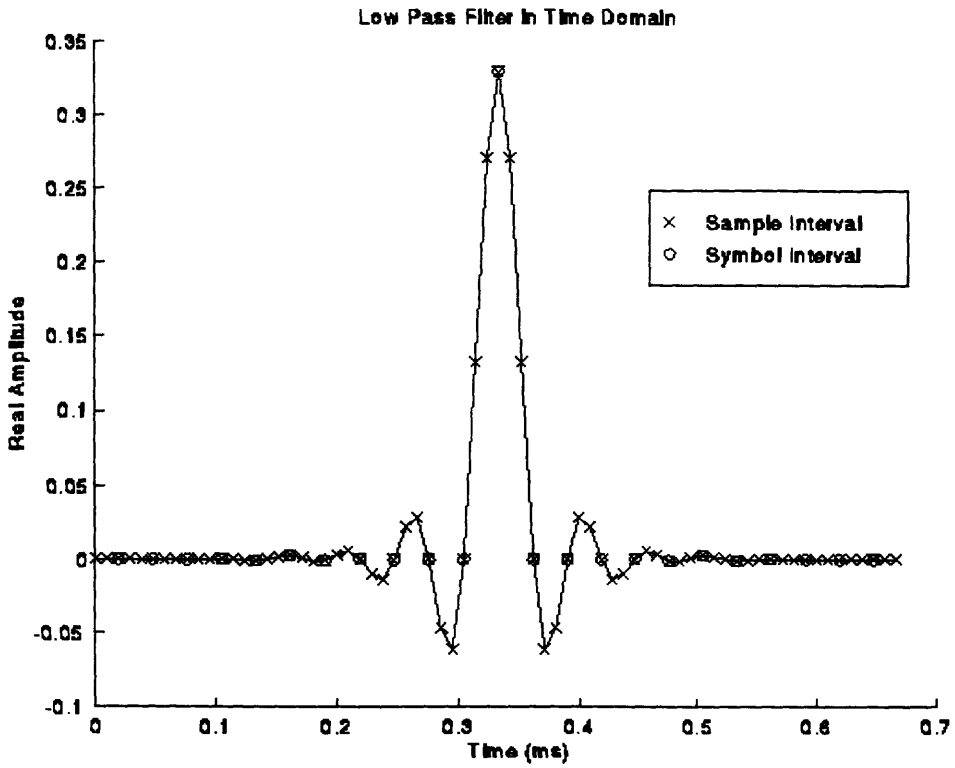


Figure 5.2-2: Impulse Response of Lowpass Filter

5.3 DTS Implementation of Low Pass Filter

Though the method is unintuitive at first glance, DTS implements the low pass filtering very efficiently. Minimizing calculation is crucial because the filter input is arriving at the sampling frequency of 210 kHz.

As is common practice, DTS splits the filter convolution between the transmitter and the receiver. The filter split is symmetrical, so that the time domain representation of the square root of the raised-cosine is applied both before modulation and after demodulation, as suggested by Equation 5.3-1. [Bingham 88, p. 64]

$$G_{transmitter}(f) = G_{receiver}(f) = \sqrt{G(f)}$$

Equation 5.3-1: Splitting Low Pass Filter between Transmitter and Receiver

This division serves two purposes. The primary reason for splitting the filter is to reduce the impact of added Gaussian noise in the channel. The split low pass filter acts as a matched filter, improving the SNR of the received signal. The second reason is specific to this application: it is costly to put complexity downhole. Splitting the filter allows less computation to be done in the DTC without burdening the DTM with the entire convolution. The actual split filter coefficients are plotted in Figure 5.3-1.

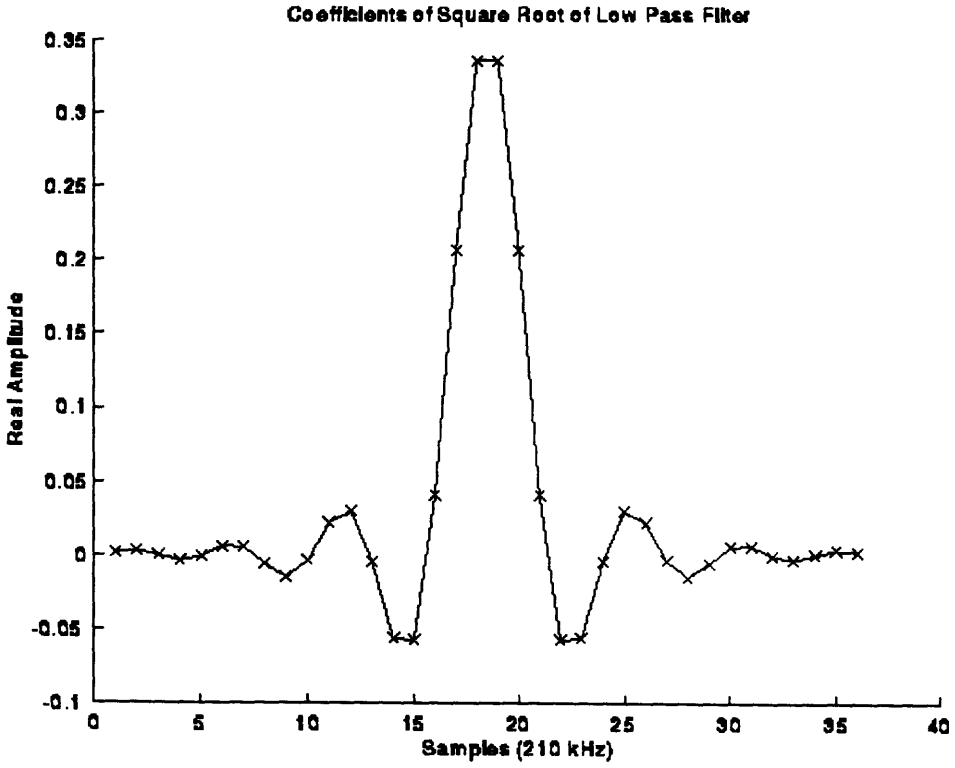


Figure 5.3-1: Split Lowpass Filter Coefficients

As with BECM filtering, the DTM takes advantage of the fact that although the incoming signal is at the sampling frequency of 210 kHz, the filter output need only be at the symbol rate of 70 kHz. The filter coefficients are at 210 kHz, but only a subset of them need be used during each calculation.

6. Adaptive Equalizer

6.1 Purpose of Adaptive Equalization

As described in Sections 2.1.1 and 3.4.2, logging cable is not an ideal channel. Even ignoring additive random noise, frequency-dependent cable attenuation and non-linear group delay cause the voltage levels received at the top of the cable to differ substantially from those sent at the bottom. The effects of the cable can be viewed as a source of noise, as we discussed in Section 3.2.3. The purpose of an adaptive equalizer is to reduce the cable noise until it becomes negligible compared to other sources of noise in the system. At that point, the cable and equalizer combination will mimic an ideal channel.

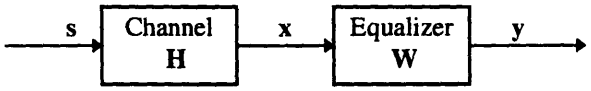


Figure 6.1-1: Signal through Channel and Equalizer

To summarize Figure 6.1-1, the channel distortion, H, causes the cable output, x, to differ from the input, s. We want to add an equalizer, W, after the channel so that the difference between the equalizer output, y, and the channel input, s, is negligible. Setting y equal to s, transfer function manipulation in Equation 6.1-1 gives us an ideal frequency response for W -- assuming that we know H.

$$
\begin{aligned}
Y(f) &= W(f)X(f) \\
&= W(f)\{H(f)S(f)\} \\
&= \{W(f)H(f)\}W(f) \\
\textit{if} \quad W(f) &= \frac{1}{H(f)} \\
\textit{then} \quad Y(f) &= S(f)
\end{aligned}
$$

Equation 6.1-1: Ideal Equalizer Weights in Frequency Domain

Because we do not want to assume that we know the channel response ahead of time, we must compute the equalizer adaptively.

6.2 Theory Behind Adaptive Equalization

We will start off assuming that the channel is real and does not change with time, and slowly develop a full-fledged QAM least mean squares equalization algorithm as we add these elements back in.

6.2.1 Mean Squared Error

We begin by assuming we know our channel baseband response h(t), and that it is represented by M significant samples taken at the symbol rate, as suggested by Equation 6.2-1. We will be equalizing this impulse response with a filter of length N.

$$h = h_0 + h_1 z^{-1} + \ldots + h_{M-1} z^{M-1}$$

$$\mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{M-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \text{ with N - 1 zeros}$$

Equation 6.2-1: Channel Baseband Impulse Response

We will call the input to the cable s[n]. A matrix containing the history of the input signal is defined in Equation 6.2-2.

$$\mathbf{s}[n] = \begin{bmatrix} s[n] \\ s[n-1] \\ \vdots \\ s[n-N-M] \end{bmatrix}$$

Equation 6.2-2: History of Transmitted Signal

The output of the cable is the baseband response (discussed in Section 7) convolved with the input signal. We will call it x[n]. Computation and history of x[n] are shown in Equation 6.2-3.

$$\begin{aligned} x[n] &= h * s[n] \\ &= \mathbf{h}^T \ \mathbf{s}[n] = \sum_{k=0}^{M-1} h_k \ s[n-k] \end{aligned}$$

$$\mathbf{x}[n] = \begin{bmatrix} x[n] \\ x[n-1] \\ \vdots \\ x[n-N-M] \end{bmatrix}$$

$$\mathbf{X}[n] = \begin{bmatrix} \mathbf{x}[n] & \mathbf{x}[n-1] & \cdots & \mathbf{x}[n-N-1] \end{bmatrix}$$

Equation 6.2-3: Cable Output

In order to equalize the channel, we select N weights, represented in Equation 6.2-4. We label these weights, also known as taps, w_k as k ranges from 0 to N-1.

$$w = w_0 + w_1 z^{-1} + \ldots + w_{N-1} z^{N-1}$$

$$\mathbf{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_{N-1} \end{bmatrix}$$

Equation 6.2-4: Equalizer Weights

The output of the equalizer, y[n], equals the input x[n] to the equalizer convolved with the equalizer tap weights w. It is computed in Equation 6.2-5.

$$\begin{aligned} y[n] &= w * x[n] \\ &= \mathbf{w}^T \mathbf{x}[n] = \sum_{k=0}^{N-1} w_k \; x[n-k] \end{aligned}$$

$$\mathbf{y}[n] = \begin{bmatrix} y[n] \\ y[n-1] \\ \vdots \\ y[n-N-M] \end{bmatrix}$$

Equation 6.2-5: Equalizer Output

We would ideally like the equalizer output to be equal to the signal input, so that y[n]=s[n] for all n. As seen in Equation 6.2-6, this means that we want to select out weights w so that w*h is an impulse. It is worthwhile to note that this is the time domain dual of Equation 6.1-1.

$$\begin{aligned} y[n] &= w * x[n] \\ &= w * (h * s[n]) \\ &= (w * h) * s[n] \end{aligned}$$

Equation 6.2-6: Ideal Equalizer Output

This optimal situation can be expressed as the matrix equation of Equation 6.2-7.

$$\mathbf{X}[n]\,\mathbf{w} = \mathbf{s}[n]$$

Equation 6.2-7: Ideal Equalizer Output in Matrix Form

Unfortunately, finding a weight vector to satisfy this equation might not be possible. For example, imagine we have a five tap equalizer and a channel with an impulse response of only four significant samples, as seen in Figure 6.2-1.

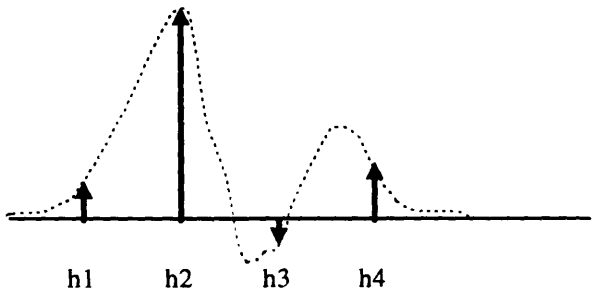


Figure 6.2-1: Example Channel Impulse Response

Because this is an impulse response, s[n] is an impulse, and x[n] = hn. The matrix operations of Equation 6.2-7 can be written out explicitly as [Bingham 88, p. 241]:

w1	**w2**	**w3**	**w4**	**w5**		**s**
h1	0	0	0	0	=	0
h2	h1	0	0	0	=	0
h3	h2	h1	0	0	=	0
h4	h3	h2	h1	0	=	1
0	h4	h3	h2	h1	=	0
0	0	h4	h3	h2	=	0
0	0	0	h4	h3	=	0
0	0	0	0	h4	=	0

This is an overdetermined set of equations -- there are only N (five in this example) unknowns to control the N+M-1 (eight in this example) equations resulting from the convolution of the input impulse response with the equalizer. It can only be solved approximately, hence the concept of minimum error that we shall examine in Section 6.2.2.

We can give the equalizer some help. We know that the values of s[n] come from a signal space; there are a limited number of possible values, and we know what they are. Therefore, we can slice y[n] to create d[n], rounding the equalizer output to the nearest possible symbol. A vector history of d[n] is defined in Equation 6.2-8.

$$d[n] = sliced\left\{y[n]\right\}$$
$$\mathbf{d}[n] = \begin{bmatrix} d[n] \\ d[n-1] \\ \vdots \\ d[n-N-M] \end{bmatrix}$$

Equation 6.2-8: Sliced Equalizer Output

Now we don't need w*h to be a perfect impulse: we need it to be close enough to an impulse that we always slice the equalizer output to the correct symbol and avoid slicing errors. We form an error signal in Equation 6.2-9 by subtracting the equalizer output from what we wish the equalizer output to be.

$$e[n] = s[n] - y[n]$$
$$\mathbf{e}[n] = \mathbf{s}[n] - \mathbf{y}[n] = \begin{bmatrix} e[n] \\ e[n-1] \\ \vdots \\ e[n-N-1] \end{bmatrix}$$

Equation 6.2-9: Error Signal

It is not actually this error that we wish to minimize -- we wish to minimize the mean square of this error, or MSE. The MSE is derived in Equation 6.2-10. [Mayhugh 92, p. 30]

$$
\begin{aligned}
MSE &= E\left[e^2[n]\right] \\
&= E\left[s^2[n]\right] + \mathbf{w}^T[n]\, E\left[\mathbf{x}[n]\,\mathbf{x}^T[n]\right]\mathbf{w}[n] + 2\,\mathbf{w}^T[n]\, E\left[s[n]\,\mathbf{x}[n]\right] \\
&= E\left[s^2[n]\right] + \mathbf{w}^T[n]\,\mathbf{R}\,\mathbf{w}[n] + 2\,\mathbf{w}^T[n]\,\mathbf{p}
\end{aligned}
$$

$$
\text{where } \mathbf{R} = E\left[\mathbf{x}[n]\,\mathbf{x}^T[n]\right]
$$

$$
\text{and } \mathbf{p} = E\left[s[n]\,\mathbf{x}[n]\right]
$$

Equation 6.2-10: Mean Square Error

R is called the auto-correlation matrix of the equalizer input vector x.

6.2.2 Optimal Weights

The optimal solution to minimize the MSE is found by solving:

$$
\frac{\partial\, MSE}{\partial\, \mathbf{w}} = 0
$$

The derivative of the MSE Equation 6.2-10 results in the optimum weight vector Equation 6.2-11. [Mayhugh 92, p. 31]

$$
\begin{aligned}
\mathbf{R}\ \mathbf{w}_{opt} &= \mathbf{p} \\
\mathbf{w}_{opt} &= \mathbf{R}^{-1}\,\mathbf{p} \\
&\cong (\mathbf{X}^{\mathbf{T}}[n]\,\mathbf{X}[n])^{-1}\,\mathbf{X}^{\mathbf{T}}[n]\,\mathbf{s}[n]
\end{aligned}
$$

Equation 6.2-11: Optimal Weight Calculation

This is called a normal equation for a linear least squares solution; it is a standard linear algebra method for finding the best solution to an overconstrained problem.

It is also interesting to look at the derivative from another point of view -- that of the first line of Equation 6.2-10. Combining Equation 6.2-9, Equation 6.2-5, and Equation 6.2-3, we arrive at Equation 6.2-12. [Verghese 94]

$$
\begin{aligned}
MSE &= E\left[e^2\right] \\
&= E\left[(d[n]-y[n])^2\right] \\
&= E\left[(s[n]-\mathbf{w}^T\mathbf{x}[n])^2\right] \\
\frac{\partial MSE}{\partial \mathbf{w}} &= 0 \\
0 &= -2\,\mathbf{x}[n]\,(s[n]-\mathbf{w}_{opt}{}^T\mathbf{x}[n]) \\
0 &= -2\,\mathbf{x}[n]\,e[n] \\
\mathbf{x}[n]\,s[n] &= \mathbf{x}[n]\,y[n]
\end{aligned}
$$

Equation 6.2-12: Derivation of Error Orthogonality

If the vectors x[n], x[n-1], x[n-2], etc. are considered the basis vectors, the last line of Equation 6.2-12 shows that for the optimal weight vector, the output of the equalizer is a projection of the input to the channel onto the basis vector space.

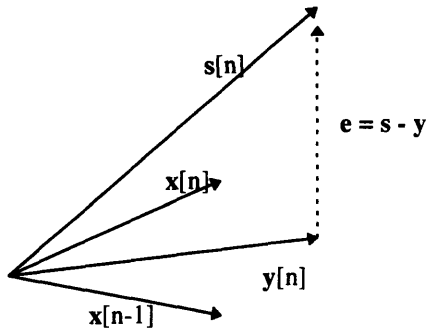


Figure 6.2-2: Error Orthogonality [Verghese 94]

Equivalently, the second to last line of Equation 6.2-12 shows that with the optimal weight vector, the error is orthogonal to all the basis vectors. [Verghese 94] This orthogonality condition, depicted in Figure 6.2-2. is again a common theme in linear algebra.

6.2.3 Time-Varying Channel

If our channel response were constant, we could just measure the impulse response, compute the optimal equalizer weights using Equation 6.2-11, and hardwire the filter into the receiver. Unfortunately, the channel response changes with time (as different cables are used, and as the cables heat up and cool down), so our equalizer weights must also change with time:

$$\mathbf{h} \rightarrow \mathbf{h}(t), \quad \text{so} \quad \mathbf{w} \rightarrow \mathbf{w}[n]$$

We could solve the matrix Equation 6.2-11 for each time step, as indicated in Equation 6.2-13.

$$\mathbf{R}[n]\ \mathbf{w}_{opt}[n] = \mathbf{p}[n]$$
$$\mathbf{w}_{opt}[n] = \mathbf{R}^{-1}[n]\ \mathbf{p}[n]$$

Equation 6.2-13: Block Weight Calculation

However, this calculation is very time consuming. A widely used Least Mean Square (LMS) algorithm adapts the weights on a sample-by-sample basis. This method avoids the complicated computation of R^{-1} and p, and so is a practical method for finding close approximate solutions for the optimal weight vector in real time. In the LMS algorithm, the next weight vector w[n+1] is increased by a change proportional to the negative gradient of the MSE performance surface of Equation 6.2-13, shown in Equation 6.2-14. [TI 90, p. 204]

$$\mathbf{w}[n+1] = \mathbf{w}[n] - \mu\ \nabla$$

Equation 6.2-14: Calculating Weights by Steepest Descent

The adaptation step size, μ, controls stability and convergence rate. The gradient at the nth iteration, Λ[n], is estimated by assuming the squared error $e^2[n]$ as an estimate of the MSE. We computed the derivative of the MSE in Equation 6.2-12 and do it again in Equation 6.2-15. [TI 90, p. 204]

$$\nabla[n] = \frac{\delta\, e^2[n]}{\delta\, \mathbf{w}[n]} = -2\, e[n]\, \mathbf{x}[n]$$

Equation 6.2-15: Gradient Estimate

Substitution of this instantaneous gradient estimate into the steepest descent Equation 6.2-14 yields the Widrow-Hoff LMS algorithm (with μ substituted for 2μ) in Equation 6.2-16. [TI 90, p. 204]

$$\mathbf{w}[n+1] = \mathbf{w}[n] + \mu\, e[n]\, \mathbf{x}[n]$$

Equation 6.2-16: Widrow-Hoff LMS Algorithm

Minimum error is achieved when the gradient becomes zero.

6.2.4 Decision Directed Training

We don't know the cable input s[n], so we have to use the sliced equalizer output d[n] as our best guess for the correct equalizer output. This is called decision directed training. Substituting d[n] for s[n], the practical error signal is expressed in Equation 6.2-17. [Johnson 95, p. 31]

$$e[n] = d[n] - y[n]$$

Equation 6.2-17: Practical Error Calculation

6.2.5 Complex Channel

An equalizer for a one-dimensional baseband system has real input signals and tap coefficients. As suggested by Equation 6.2-18, in two-dimensional QAM systems both the input and the taps are complex, so all operations must use complex arithmetic -- usually requiring four times as many multiplications. The complex conjugate of the transpose should be used in vector and matrix operations. [Bingham 88, p. 239].

$$s \rightarrow sr + j\,si, \text{ so } \mathbf{x} \rightarrow \mathbf{xr} + j\,\mathbf{xi} \quad \text{and} \quad e \rightarrow er + j\,ei,$$
$$\text{so } \mathbf{w} \rightarrow \mathbf{wr} + j\,\mathbf{wi}$$

Equation 6.2-18: Complex Data and Error

Complex weights are calculated in Equation 6.2-19. [PCSI 88-1, p. 28]

$$\begin{aligned}
\mathbf{w}[n+1] &= \mathbf{w}[n] + \mu\, e[n]\mathbf{x}^*[n] \\
&= \mathbf{w}[n] + \mu\,(er + j\,ei)(\mathbf{xr} - j\,\mathbf{xi}) \\
&= \mathbf{w}[n] + \mu(er\,\mathbf{xr} + ei\,\mathbf{xi} + j\,ei\,\mathbf{xr} - j\,er\,\mathbf{xi}) \\
\mathbf{wr}[n+1] &= \mathbf{wr}[n] + \mu\,(er[n]\,\mathbf{xr}[n] + ei[n]\,\mathbf{xi}[n]) \\
\mathbf{wi}[n+1] &= \mathbf{wi}[n] + \mu\,(er[n]\,\mathbf{xi}[n] - er[n]\,\mathbf{xi}[n])
\end{aligned}$$

Equation 6.2-19: Calculation of Complex Weights

6.2.6 Noise Enhancement

In the process of equalizing attenuation distortion in the received signal, linear equalizers will enhance the additive noise. [Bingham 88, p. 243] Because the sampled input noise is assumed to be white, the total noise enhancement can be defined by Equation 6.2-20.

$$\text{Noise power enhancement} = \frac{\text{input SNR}}{\text{output SNR}}$$
$$= \mathbf{w}^T\mathbf{w}$$
$$= \sum_{k=0}^{N-1} w_k^{\,2}$$

Equation 6.2-20: Noise Power Enhancement

6.2.7 Least Mean Square Parameters

Equalizer performance depends on its parameters. As we've seen, more equalizer taps will decrease the residual equalizer error, but require more computation. The convergence parameter also involves a trade-off: a large μ increases the convergence rate, but increases the steady-state error. Furthermore, too large a convergence parameter could cause instability. Convergence is guaranteed if μ remains in the bounds given by Equation 6.2-21. [Mayhugh 90, p. 34]

$$0 < \mu < \frac{1}{N\,E[x^2[n]]}$$

Equation 6.2-21: Maximum Convergence Parameter

The convergence parameter is usually chosen to be close to the maximum. The relationship between convergence rate and convergence parameter is given in the time constant of Equation 6.2-22. [Mayhugh 90, p. 34] Note that because the convergence time constant is inversely proportional to the convergence parameter, it is also implicitly proportional to the number of taps, N.

$$\tau_{MSE} = \frac{1}{4\mu\,E[x^2[n]]}$$

Equation 6.2-22: Convergence Time Constant

6.3 DTS Implementation of Adaptive Equalizer

6.3.1 Equalization Calculations

At startup, the imaginary equalization weights are initialized to all zeros and the real equalization weights are initialized to zeros with an impulse of 1 in the center position. Subsequently, the equalizer output is calculated each symbol time according to Equation 6.3-1. [PCSI 88-1, p. 28]

$$yr[n] = \sum_{k=0}^{N-1} wr_k \; xr[n-k] - wi_k \; xi[n-k]$$
$$yi[n] = \sum_{k=0}^{N-1} wi_k \; xr[n-k] + wr_k \; xi[n-k]$$

Equation 6.3-1: Calculation of Complex Equalizer Output

This output is compared to the constellation thresholds and sliced to give d[n], which is then used to compute the error, as shown in Equation 6.3-2. [PCSI 88-1, p. 28]

$$er[n] = dr[n] - yr[n]$$
$$ei[n] = di[n] - yi[n]$$

Equation 6.3-2: Calculation of Complex Error.

Finally, the equalizer weights are updated according to Equation 6.3-3. [PCSI 88-1, p. 28]

$$\mathbf{wr}[n+1] = \mathbf{wr}[n] + \mu\,(er[n]\,\mathbf{xr}[n] + ei[n]\,\mathbf{xi}[n])$$
$$\mathbf{wi}[n+1] = \mathbf{wi}[n] + \mu\,(er[n]\,\mathbf{xi}[n] - er[n]\,\mathbf{xi}[n])$$

Equation 6.3-3: Calculation of Complex Weights

The process then begins again with the next incoming symbol and the new equalizer weights.

6.3.2 Training Sequences

DTS uses two types of training sequences, long training and short training. During initial power-up training, no a priori knowledge of the proper equalization tap settings is assumed. A pseudo-random sequence of two points is sent. The receiver does not try to sync onto the sequence; it takes advantage of the limited two-symbol signal space to perform decision-directed training. Long training sends 1024 symbols at 70 ksymbols/second, taking 14.63 ms. [PCSI 88-1, p. 32]

Because DTS operates in half-duplex mode, periods of uplink communication are interrupted by periods of downlink communication. The cable response might have altered during the downlink segment, so a short training session at the beginning of each uplink segment reestablishes proper equalization. The receiver stores the equalizer tap settings from the previous receiving session and uses them as the initial setting values for the short training. The same pseudo-random sequence is used as in long training. Short training sends 128 symbols at 70 ksymbols/second, taking 1.83 ms. [PCSI 88-1, p. 33]

The equalizer continues to be updated during steady state transmission, although the convergence parameter is decreased. It is assumed that convergence rate is no longer an issue, so reducing steady state error and sensitivity to noise become important. Steady state data is scrambled before transmission to avoid a repetitive pattern of symbols which might confuse the equalizer. [DTS 91, p. 3-13]

6.3.3 Amount of Equalization

As we recall from Section 6.1, the purpose of the adaptive equalizer is to make the effects of cable distortion negligible compared to other sources of noise in the system, and,of course, well below that theoretically required to achieve the desired BER. Repeated from Section 3.4.4, we assume worst case

added Gaussian noise of 28.3 dB, and have the following required SNR values for a BER of 1E-7 [PCSI 88-1, p. 18]:

Bits/Symbol	Theoretically Required SNR (dB)
2	14
3	17.5
4	21
5	24
6	27

Therefore, we need to equalize such that the negative of the MSE (normalized by the signal power) is 6 dB larger than 27 or 28 dB.

It is estimated that with 15 taps, DTS is able to achieve a -35 dB MSE on worst-case cable. As we shall demonstrate with the simulation in Section 10.4, fewer taps would not result in enough equalization, while more taps would be useless overkill.

PART III -- Simulation

The purpose of this thesis is to simulate the telemetry process. This is useful for several reasons:

- A correctly-working simulation will verify our understanding of the process.
- We will be able to test a wide variety of cables (including worst case) that aren't available in the lab.
- We will be able to optimize telemetry parameters after learning about trade-offs between performance and implementation cost.

It was decided that the simulation should have the following adjustable parameters:

- Carrier, symbol, and sample frequencies
- Number of bits/symbol
- Number of equalizer taps
- Initialization of taps
- Convergence rate factor
- Length of training

Although, at present, Schlumberger is most interested in the operation of the DTM receiver, in order to simulate the receiver we will also have to simulate the transmitter and channel.

7. Cable Impulse Response Baseband Equivalent

7.1 Purpose of Baseband Equivalent

Perhaps the most critical part of the simulation is an accurate model of the channel. Not only must the cable impulse response be known, but the effects of filtering, sampling, and modulation on the transmitted signal must all be taken into account. This is most intuitively done by including each of the systems separately and explicitly in the simulation, i.e., convolving with filters, computing band-edge timing synchronization, and applying and removing carriers. Alternatively, the systems can be combined in series to form one end-to-end system, called a baseband equivalent. [Bingham 88, p. 79] Convolving the raw data with this one transfer function produces an output equivalent to that obtained by directly modulating, convolving, sampling, filtering, and demodulating the input stream on the cosine carrier.

Although counterintuitive, there are advantages to combining the cable impulse response, filtering, sampling, and modulation all into one baseband response:

- Most simulations in the literature use a baseband response. Simulating the same way as the literature eases debugging.
- As discussed in Section 7.3.5, the existence of a single impulse response allows the calculation of ideal equalizer taps.
- The baseband impulse response is only calculated once for each cable. Baseband simulations have fewer computation steps, and therefore run much faster.

7.2 Theory behind Baseband Equivalent

Our goal is to derive a baseband impulse response. [PCSI 88-1, p. 47] First, we identify the passband cable impulse response in Equation 7.2-1.

$$h'(t) = \int_{-\infty}^{+\infty} G(f)\, e^{j(2\pi ft)} df$$

$$h'(t) = \int_{-\infty}^{+\infty} |G(f)|\, e^{j(2\pi ft + \angle G(f))} df$$

Equation 7.2-1: Passband Cable Impulse Response

The integral from minus infinity to infinity over frequency of H(f) times $e^{j2\pi ft}$ takes the inverse Fourier transform of the cable transfer function G(f) to form the passband cable impulse response h(t).

As we saw in Section 3.3.2, if we assume an ideal channel (where G(f) = 1 for all f), then modulation (multiplying by e^{jwt}) and demodulation (multiplying by e^{-jwt}) can be considered simple frequency shifts up to a passband and down again to baseband; the end-to-end transfer function of the system is equivalent to that of two independent undistorted baseband channels. Unfortunately, the channel is not ideal; the data is distorted by the attenuation and phase in the passband of G(f). Demodulation translates the passband section of G(f) down to the baseband as G(f + fc).

This frequency shift can be accomplished by a substitution of variables within the cable transfer function, as shown in Equation 7.2-2. [Bingham 88, p. 79]

$$h''(t) = \int_{-\infty}^{+\infty} G(f + f_c)\, e^{j(2\pi ft)} df$$

$$h''(t) = \int_{-\infty}^{+\infty} |G(f + f_c)|\, e^{j(2\pi ft + \angle G(f + f_c))} df$$

Equation 7.2-2: Demodulated Impulse Response

We wish to include the effects of sampling in our impulse response, so must incorporate the timing delay we discussed in Section 4. Equation 7.2-3 reproduces this calculation.

$$\tau = \frac{\theta(-f_s/2) - \theta(f_s/2)}{2\pi f_s}$$

Equation 7.2-3: Optimal Timing Delay

The actual implementation applies a delay of τ in time to the sampling clock. This is equivalent to running a fixed sampling clock but applying a delay of τ to the transmitted signal. This is again equivalent to applying a delay of τ to the cable response. Applying a delay of τ in time can be accomplished by multiplying by $e^{-j2\pi tf}$ in frequency. We can, therefore, incorporate the sampling delay into our impulse response, as shown in Equation 7.2-4.

$$h'''(t) = \int_{-\infty}^{+\infty} G(f + f_c)\, e^{j(2\pi f(t+\tau))} df$$

$$h'''(t) = \int_{-\infty}^{+\infty} |G(f + f_c)|\, e^{j(2\pi f(t+\tau) + \angle G(f+f_c))} df$$

Equation 7.2-4: Demodulated Impulse Response with Timing Delay

The purpose of the low pass filtering, as discussed in Section 5, is to bandlimit the data spectrum. The filter applied at baseband, before modulation, can be viewed as a pulse-shaping block for the input signal. For this reason, the low pass filter should be cascaded as the input signal will be: by simply multiplying the transfer function of the filter by the transfer function of the demodulated cable response. Equation 7.2-5 demonstrates.

$$h''''(t) = \int_{-\infty}^{+\infty} L(f) G(f + f_c)\, e^{j(2\pi f(t+\tau))} df$$

$$h''''(t) = \int_{-\infty}^{+\infty} |L(f)|\,|G(f + f_c)|\, e^{j(2\pi f(t+\tau) + \angle L(f) + \angle G(f+f_c))} df$$

Equation 7.2-5: Lowpass Demodulated Impulse Response with Timing Delay

The last component to be incorporated into our baseband response is carrier phase. Ideally, the carrier phase should result in a baseband response in which the largest sample is purely real. In the actual implementation, an adaptive rotator computes the ideal frequency phase. For the simulation, we can simply calculate the carrier phase needed to compensate for the phase added by the cable and filters. Calling this phase θ_c, we arrive at the complete baseband impulse response in Equation 7.2-6. [Bingham 88, p. 79]

$$h(t) = \int_{-\infty}^{+\infty} L(f)G(f+f_c)\, e^{j(2\pi f(t+\tau)+\theta_c)} df$$

$$h(t) = \int_{-\infty}^{+\infty} |L(f)||G(f+f_c)|\, e^{j(2\pi f(t+\tau)+\theta_c+\angle L(f)+\angle G(f+f_c))} df$$

Equation 7.2-6: Baseband Response

Finally, to enable the baseband response to be convolved with the input signal, the baseband response must be sampled at the symbol rate. This will produce an output at the symbol rate, as promised.

7.3 Calculation of Baseband Response

The calculation of a baseband response is illustrated in Figure 7.3-1.

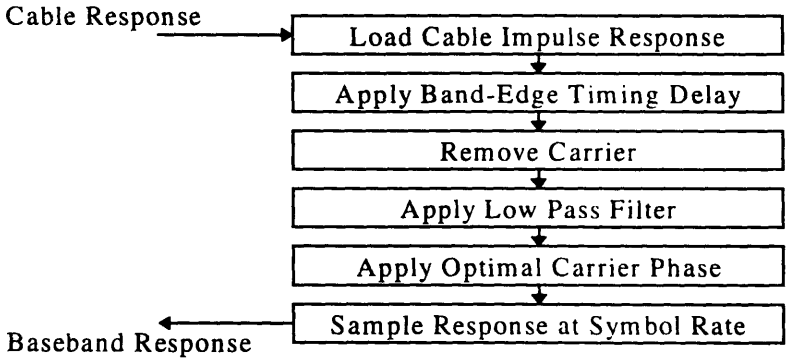


Figure 7.3-1: Calculation of Baseband Response

7.3.1 Initialize Baseband Parameters

Before we begin calculating the baseband response, baseband parameters must be initialized. Code 7.3-1 summarizes the typical settings.

```
input_sample_freq =  10000000;
output_sample_freq = 1000000;
fsample = 210000;
fcarrier = 52500;
fsymbol = 70000;

oversample = fsample/fsymbol;
fbeupper = fcarrier + fsymbol/2;
fbelower = fcarrier - fsymbol/2;
fsample_index = length(cable_fft);
fcarrier_index = floor(fcarrier/freq_delta);
fbeupper_index = floor(fbeupper/freq_delta);
fbelower_index = floor(fbelower/freq_delta);
```

Code 7.3-1: Baseband Parameters

7.3.2 Load Cable Impulse Response

The first step in calculating a baseband response is to load the cable impulse response. The cable impulse response of 27 kft of hot 7-46NT cable is given in Figure 7.3-2.

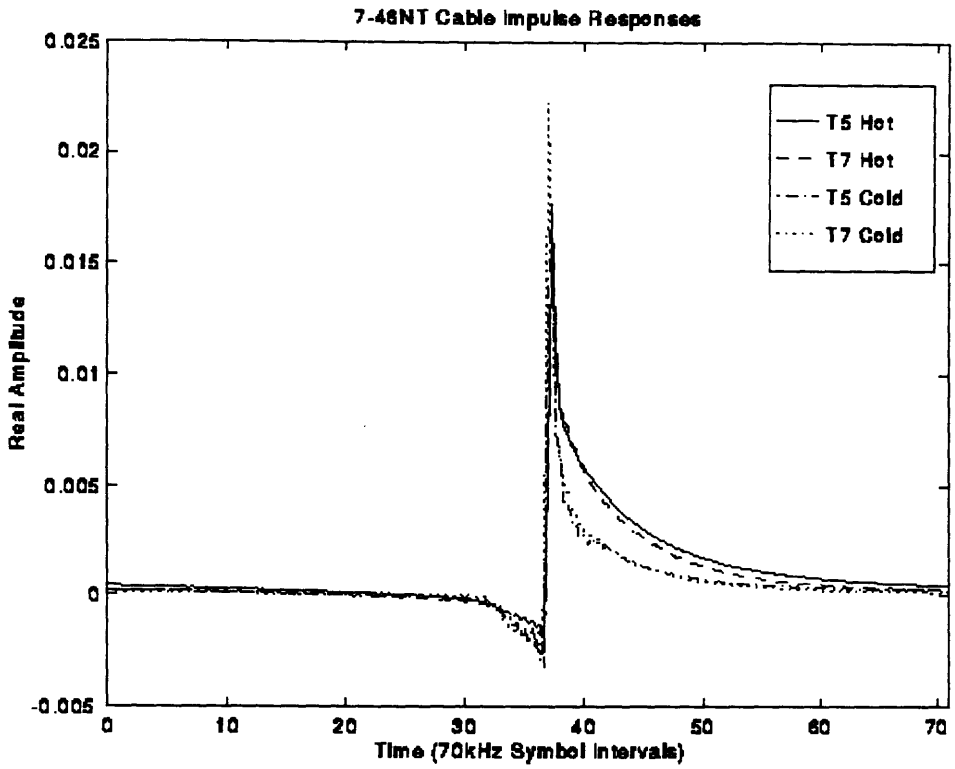


Figure 7.3-2: Cable Impulse Response of 27 kft hot 7-46NT

The cable with the most attenuation and non-linear group delay is 27 kft of hot 7-46NT (actually, worst case is 30 kft -- we model the additional 3 kft with our completed simulation); if telemetry works on this cable, it should work on all cables.

Cable impulse responses are gathered by inputting a pulse into the top of a cable and measuring the output at the bottom. The output of the cable is a function of both the input pulse and the cable response. The cable response is calculated by dividing the frequency representation of the output signal by that of the input pulse. Care must be taken: the pulse must be narrower than the cable response time intervals of interest, because too wide a pulse can place an irremovable sinc null in the middle of the output signal, thereby resulting in an inaccurate cable representation. It is difficult to generate a very narrow pulse with which to measure; the narrower the pulse in time, the higher the voltage level must be

to maintain the same level of power into the cable. Too little power can result in an inaccurate measurement due to noise.

The cable data available used different sampling rates for the input and the output measurements. The simulation zero-pads and interpolates to compute the actual cable responses, though not included in Code 7.3-2.

```
sim_disp('Loading Impulse Responses', debug);

%  AVG files are 1024 samples long, at 1 microsecond sample intervals
%  INP files are 1024 samples long, at 100 nanosecond sample intervals
%  Will need to interpolate AVG files to sampling interval of 100 ns
%    so that can divide FFT's.
%  cable_outlong is now 10240 samples long, at 100 ns sample intervals
%  100 ns sample intervals means that FFT will go up to 10000 kHz.
%  10240 point FFT's means that each FFT point is 10000 kHz/10240 = 977 Hz

%  Divide the frequency representations to get the transfer function of
%    the cable alone
%  freq_delta is the frequency spacing in Hz between FFT points = 977 Hz

sim_disp('Dividing Output FFT by Input FFT to get Cable Transfer Function', debug);
cable_long_fft = cable_outlong_fft ./ cable_inlong_fft;
freq_delta = input_sample_freq / length(cable_long_fft);
```

Code 7.3-2: Dividing Output Cable Measurement by Input

The cable responses are not interesting above 105 kHz, so the high frequencies are discarded in Code 7.3-3.

```
%  Know that cable response really only goes up to about 105 kHz for heptacable,
%    so will discard the rest.
%  Impulse responses are real, so know that frequency representations must be
%    conjugate symmetric.
%  highest_freq/freq_delta = 105kHz/977Hz = 107 data points, 10240-106-1=10133

sim_disp('Keeping Frequency Response up to 105kHz', debug);
cable_fft = ...
        [cable_long_fft(1:(highest_freq/(freq_delta*2) + 1)).' ...
        cable_long_fft((length(cable_long_fft)-(highest_freq/(freq_delta*2))+1): ...
                        length(cable_long_fft)).' ].';
```

Code 7.3-3: Eliminate Frequencies above those of Interest

An inverse FFT is used to compute the cable impulse response from the transfer function, as demonstrated in Code 7.3-4.

```
%  Compute Inverse FFT to get Cable Impulse Response
%  Expect these to be purely real, because these cable impulse responses
%     convolved with purely real inputs have to give purely real outputs

sim_disp('Computing Cable Impulse Response', debug);
cable_ifft = fftshift(ifft(cable_fft));
```

Code 7.3-4: Channel Frequency Response from Impulse Response

The cable impulse for 27 kft of hot 7-46NT cable is shown above in Figure 7.3-2. The cable impulse response is given in Figure 7.3-3.

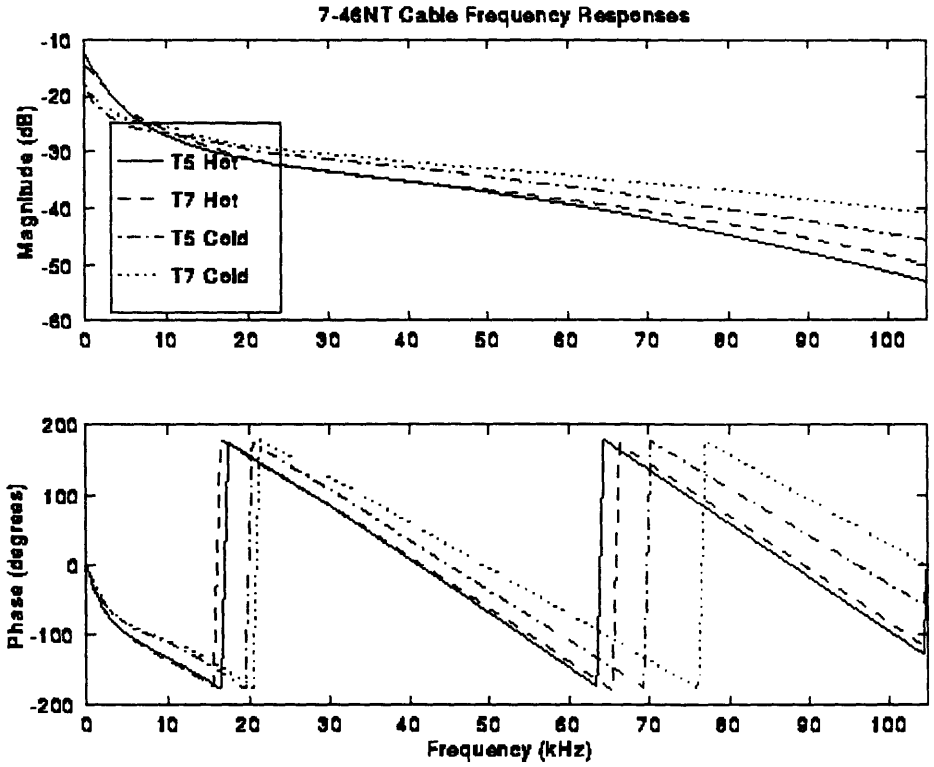


Figure 7.3-3: Cable Frequency Response of 27kft hot 7-46NT

7.3.3 Apply Timing Delay

In Code 7.3-5, the optimal timing delay is calculated according to Equation 7.2-3.

```
%  Calculate sample timing delay

sim_disp('Calculating Sample Timing Delay', debug);
CABLE_PHASE = unwrap(angle(cable_fft));
tau = (CABLE_PHASE(fbeupper_index) - CABLE_PHASE(fbelower_index)) / ...
        (2*pi*fsymbol);
```

Code 7.3-5: Calculate Optimal Timing Delay

The timing delay is applied to the cable frequency response by multiplying the response by a frequency shift in Code 7.3-6.

```
%  Sample delay in time corresponds to rotation by e^(-j*2*pi*f*tau) in frequency
%  Apply band edge timing delay by multiplying by rotation

sim_disp('Applying Band Edge Timing Delay', debug);
delay_vector = exp(-j*2*pi*tau*(0:freq_delta:((length(cable_fft)-1)*freq_delta)).');
CABLE_BE = cable_fft .* delay_vector;
```

Code 7.3-6: Apply Timing Delay

7.3.4 Remove Carrier

The carrier is removed simply by shifting the spectrum centered around fc down to DC. More details can be found in the comments of Code 7.3-7.

```
%  Remove carrier by shifting frequencies centered at carrier frequency down
%    to be centered at DC.
%  This involves shifting frequencies from fcarrier->2*f_carrier to 0->fcarrier
%    and shifting 0->fcarrier to -fcarrier->0 and zeroing out other frequencies
%  Discarding frequency information above |fcarrier|=52.5kHz is not a problem
%    because frequencies above |fsymbol*1.25|=(35kHz*1.25)=43.75kHz will be
%    attenuated by pulse shaping low pass filter with alpha=0.25 anyway
%  Represent -fcarrier->0 periodically in frequency band
%    (fsample-fcarrier)->fsample

sim_disp('Removing Carrier', debug);
CABLE_DEMOD = zeros(length(cable_fft), 1);
CABLE_DEMOD(1:(fcarrier_index+1)) = ...
        CABLE_BE(fcarrier_index:(2*fcarrier_index));
CABLE_DEMOD((fsample_index-fcarrier_index+2):fsample_index) = ...
        CABLE_BE(1:(fcarrier_index-1));
```

Code 7.3-7: Demodulate from Passband to Baseband

7.3.5 Apply Low Pass Filter

Because the low pass filter implementation is split between the transmitter and the receiver, we have to compute the original filter coefficients by recombining the two parts. The convolution is done in Code 7.3-8.

```
%  Part of filter applied in transmitter, the other in reciever
%  Convolve time filter coefficients of two filters together to apply both at once

Nyquist_filter = conv(DTC_shaping_filter(:), DTM_shaping_filter(:));
```

Code 7.3-8: Compute Low-Pass Filter

The filter is applied by multiplying the frequency representations together, as seen in Code 7.3-9.

```
%  Apply pulse shaping low pass filter

%  Take FFT of combined filter so that can apply it in frequency domain
%  Low pass filters have samples at 210kHz, so FFT will go up to 210kHz
%  Need to take FFT of length CABLE_length so that freq points represent 977Hz

sim_disp('Applying Pulse Shaping Low Pass Filter', debug);
LOWPASS = fft(Nyquist_filter, length(cable_fft));
CABLE_LOWPASS = CABLE_DEMOD .* LOWPASS
```

Code 7.3-9: Apply Low Pass Filter

The frequency domain low pass filter is given in Figure 7.3-4 -- its raised-cosine characteristics are hidden in the logarithmic scaling.

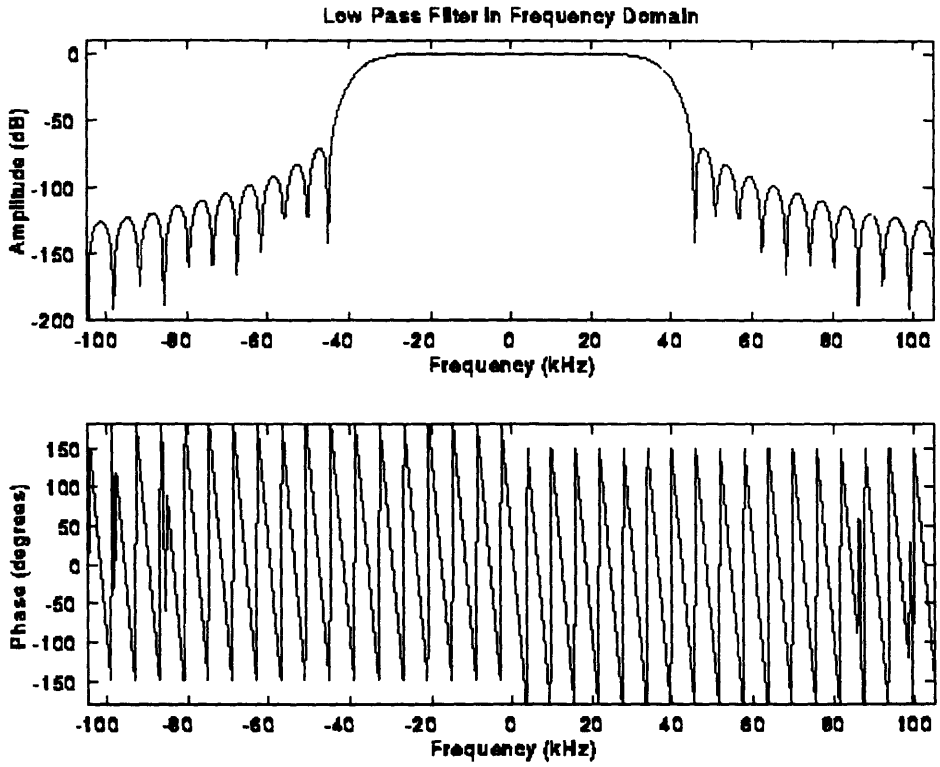


Figure 7.3-4: Low Pass Filter in Frequency Domain

Graphed in Figure 7.3-5, an inverse Fourier transform of the frequency domain raised-cosine filter results in the Nyquist low pass filter familiar from Section 5.

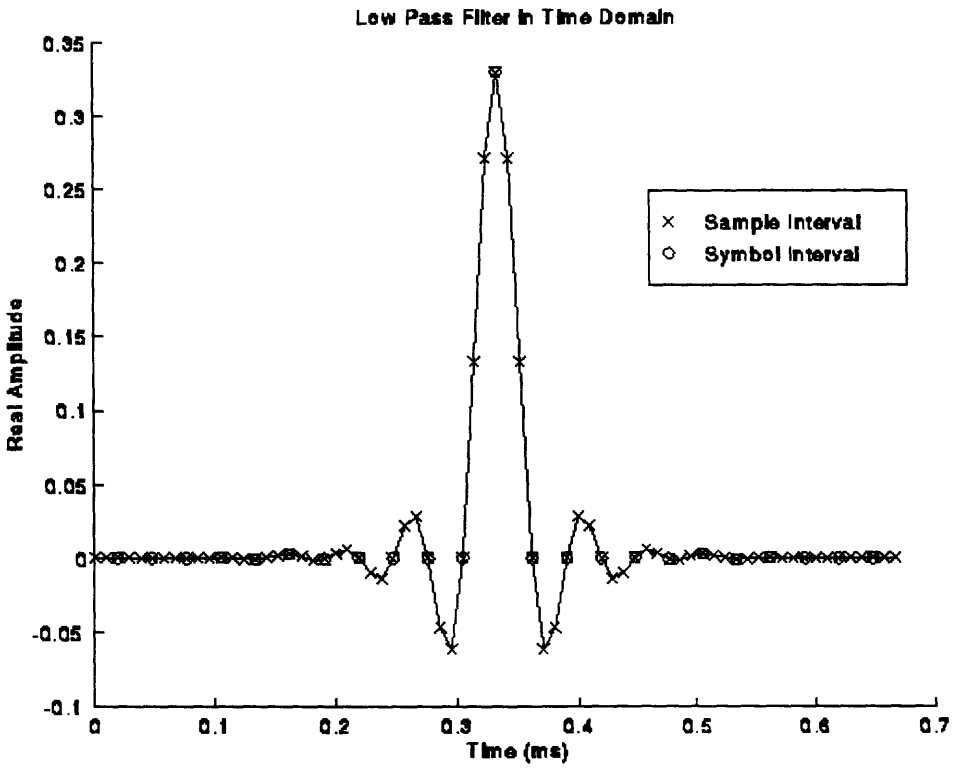


Figure 7.3-5: Low Pass Filter in Time Domain

7.3.6 Apply Carrier Phase

The carrier phase is calculated in Code 7.3-10 such that the largest sample is purely real. This is important to ensure successful decision-directed training.

```
% Apply carrier phase so that largest sample is purely real

sim_disp('Applying Optimal Carrier Phase', debug);
clear j;
cable_max = max(cable_lowpass);
carrier_phase = -atan2(imag(cable_max), real(cable_max));
baseband_sample = cable_lowpass * exp(j*carrier_phase);
```

Code 7.3-10: Apply Optimal Carrier Phase

7.3.7 Sample Response at Symbol Rate

Finally, the baseband response must be decimated down to the symbol rate. This will enable it to be convolved directly with the input symbols, and produce an appropriately sampled output signal. As seen in Code 7.3-11, we want to decimate such that we include the maximum baseband response sample.

```
% Sample cable response at 70kHz to get baseband response at the symbol rate

oversample = fsample/fsymbol;
baseband = baseband_sample((rem(cable_max, oversample) + oversample): ...
                oversample:length(baseband_sample));
```

Code 7.3-11: Sample Response at Symbol Rate

The frequency representation of the baseband cable response is shown in Figure 7.3-6, while the time domain baseband impulse response is plotted in Figure 7.3-7.

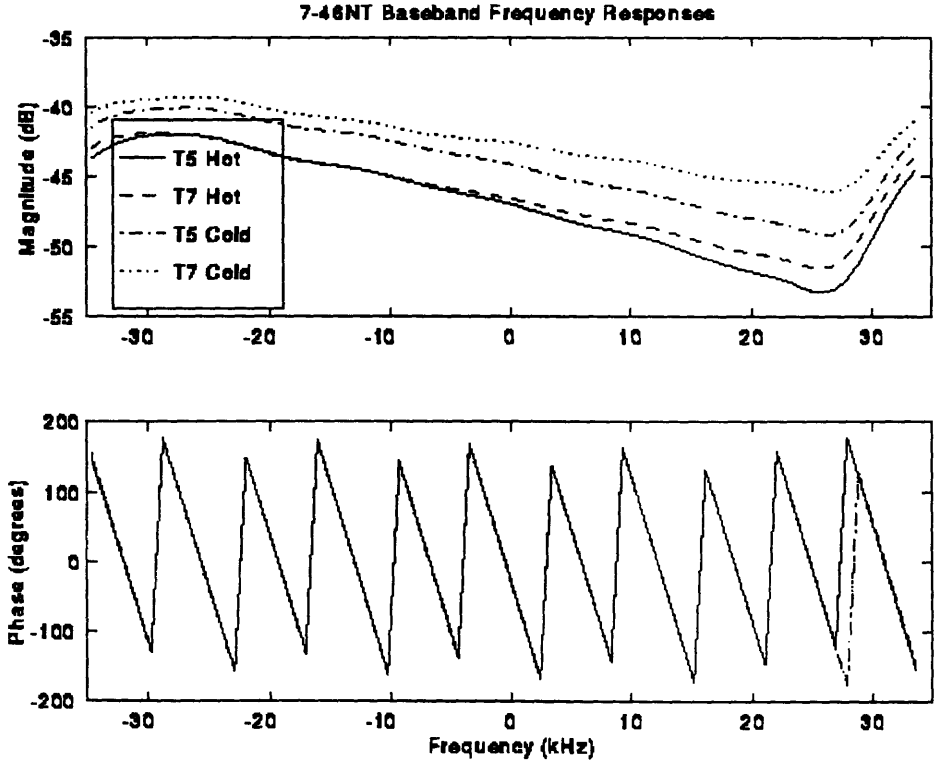


Figure 7.3-6: Baseband Cable Response; Frequency Domain

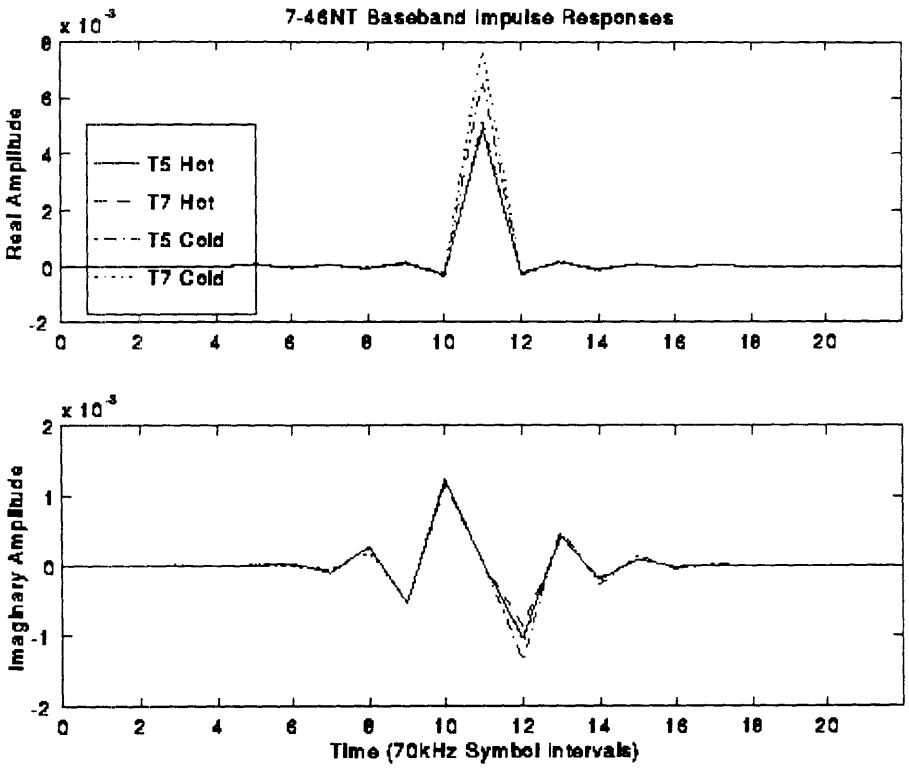


Figure 7.3-7: Baseband Cable Response; Time Domain

8. Simulation Steps

The simulation steps are illustrated in Figure 7.3-8. All simulations run through a warm-up phase, a training phase, and then a steady-state phase.

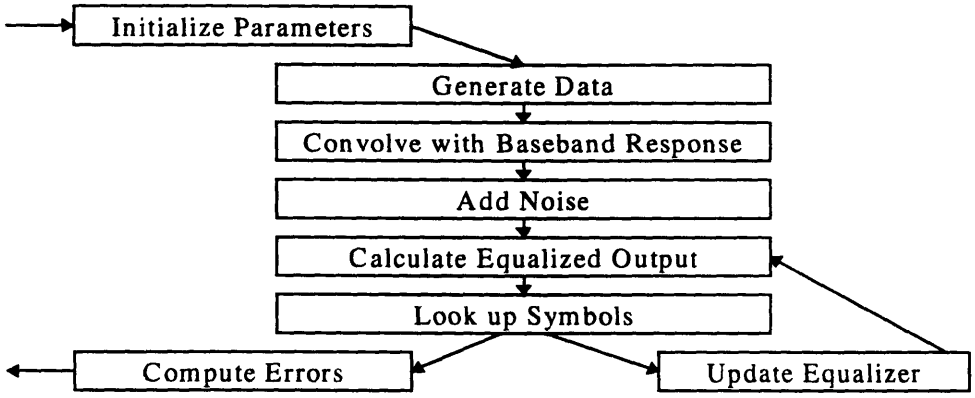


Figure 7.3-8: Simulation Steps

8.1 Initialize Parameters

Each simulation begins by initializing its parameters. The parameters in the top half of Code 8.1-1 govern the cable environment and telemetry set-up, while the parameters in the bottom are simply cleared before calculation begins. Code 8.1-1 demonstrates a typical simulation set-up.

```
cable_name = 'T5hot746NT'
raw_length = 27647;
desired_length = 30000;
SNR = 0;
QAM = 64;
NC = 15;
TRAIN_NOISE = 0;
alpha_scale = 1;
C_init = 'impulse';
center_tap = (NC+1)/2;
Npn = 500*(NC/15);
Nss = 1000*(NC/15);
pn = zeros(7, (Ninit+Npn+Nss));
pn(:, 1) = [0 1 0 1 0 1 0]';

clear XMOD;
clear XSNEAK;
clear XCABLE;
C = zeros((Ninit+Npn+Nss), NC);
Z = zeros((Ninit+Npn+Nss), 1);
E = zeros((Ninit+Npn+Nss), 1);
D = zeros((Ninit+Npn+Nss), 1);
```

```
rand('seed', 0);
randn('seed', 0);
```

Code 8.1-1: Initialize Parameters

8.2 Generate Data

The next simulation step is to generate data. During training in the real DTS, a pseudo-random number (PN) sequence is generated in the DTC and then transmitted; Code 8.2-1 generates the same PN sequence. Steady state data would normally come from the tools, through the DTC. The simulation models this by simply generating a random data stream from the available signal space.

```
%  Generates symbols at 70kHz
%  Random training sequence, then either or random QAM8, 16, 32, or 64

sim_disp('Generating Input Signal', debug);
for NSYMBOLS=2:(Ninit+Npn)
  pn(1, NSYMBOLS) = xor(1, xor(pn(6, (NSYMBOLS-1)), pn(7, (NSYMBOLS-1))));
  pn((2:7), NSYMBOLS) = pn((1:6), (NSYMBOLS-1));
  if (pn(1, NSYMBOLS) > 0)
    XMOD(NSYMBOLS) = (+3) + i*(-1);
  else
    XMOD(NSYMBOLS) = (-3) + i*(+1);
  end;
end;

for NSYMBOLS=((Ninit+Npn+1):(Ninit+Npn+Nss))
  if (QAM==64)
    XMOD(NSYMBOLS) = (fix(rand*8)*2 - 7) + i*(fix(rand*8)*2 - 7);
  elseif (QAM==16)
    XMOD(NSYMBOLS) = (fix(rand*4)*2 - 3) + i*(fix(rand*4)*2 - 3);
  end;
end;
```

Code 8.2-1: Generate Data

8.3 Convolve with Baseband Response

Modulation, demodulation, filtering, and sampling are modeled by convolving the generated data with the baseband response calculated in Section 7. Calculating the cable output is simple, as seen in Code 8.3-1.

```
%  Convolve with filtered baseband cable response

sim_disp('Convolving with Cable', debug);
XCABLE = conv(baseband, XMOD);
[peak latency] = max(baseband);
XCABLE = XCABLE(latency:(latency + length(XMOD) - 1));
```

Code 8.3-1: Convolve with Baseband Response

8.4 Add Noise

In order to compute bit error rates and to test that convergence is assured even in the presence of noise, a random white gaussian signal is added to the output of the cable in Code 8.4-1.

```
%  Calculate Received Signal Energy so that can choose good alpha
%  Also used to add appropriate amount of noise

sigpn_energy = mean(XCABLE(2:(Ninit+Npn)).*conj(XCABLE(2:(Ninit+Npn))));
sigss_energy = mean(abs(XCABLE((Ninit+Npn+1):(Ninit+Npn+Nss))).^2);

%  Add noise

if (TRAIN_NOISE)
  sim_disp('Adding Noise', debug);
  noise_variance = (((QAM-1)/3) * 2 /sigss_energy) * 10^(SNR/10);
  X = XCABLE + sqrt(noise_variance) * randn(size(XCABLE));
else % if (TRAIN_NOISE)
  X = XCABLE;
end  % if (TRAIN_NOISE)
```

Code 8.4-1: Add noise

8.5 Calculate Equalized Output

A few more parameters must be set before calculating the equalized output. The equalizer taps are initialized either with a real impulse or with the complex vector C_init. The convergence parameter alpha (called μ in Section 6) depends on the energy of the cable output. As seen in Code 8.5-1, the training alpha is set to be the maximum guaranteed to converge, while the steady state alpha is only set to one-third of its possible maximum.

```
if (strcmp(C_init, 'impulse'))
  C(Ninit, center_tap) = sqrt(mean(XMOD.*XMOD)/mean(X.*X));
else
  C(Ninit, :) = C_init.';
end;
```

```
%  PN Training for Ninit+Npn symbols

sim_disp('PN Training', debug);
alpha_max = 1/(NC*sigpn_energy);
alpha = alpha_scale * alpha_max;



%  Steady state for Nss symbols

sim_disp('Steady State Transmission', debug);
alpha_max = 1/(NC*sigss_energy);
alpha = alpha_scale * alpha_max/3;
```

Code 8.5-1: Setup for Calculating Equalized Output

The equalizer output is calculated in Code 8.5-2 by convoluting the weights by the equalizer input.

```
Z(NSYMBOLS) = 0;
for k=1:NC
    Z(NSYMBOLS) = Z(NSYMBOLS) + C(NSYMBOLS,k) * X(NSYMBOLS-k);
end;
```

Code 8.5-2: Calculate Equalized Output

8.6 Look Up Symbols

The signal space used during symbol look-up depends on whether the system is in training or steady state. During training, the symbol decision is made by slicing based upon the real component of the equalizer output. During steady state, the more complicated decision is made by finding the symbol point closest (in Euclidian distance) to the equalized output. This slicing is demonstrated for the square 64-point symbol space in Code 8.6-1.

```
if (Z(NSYMBOLS) > 0)
  D(NSYMBOLS) = (+3) + i*(-1);
else
  D(NSYMBOLS) = (-3) + i*(+1);
end;



if (QAM==64)
   D(NSYMBOLS) = (round((real(Z(NSYMBOLS))+7)/2)*2-7) + ...
                        i*(round((imag(Z(NSYMBOLS))+7)/2)*2-7);
   if (real(D(NSYMBOLS)) > 7)
     D(NSYMBOLS) = (+7) + i*imag(D(NSYMBOLS));
   elseif (real(D(NSYMBOLS)) < -7)
     D(NSYMBOLS) = (-7) + i*imag(D(NSYMBOLS));
   end;
```

```
   if (imag(D(NSYMBOLS)) > 7)
     D(NSYMBOLS) = real(D(NSYMBOLS)) + i*(+7);
   elseif (imag(D(NSYMBOLS)) < -7)
     D(NSYMBOLS) = real(D(NSYMBOLS)) + i*(-7);
   end;
end;  % if QAM
```

Code 8.6-1: Look Up Symbols

8.7 Update Equalizer

The equalizer is updated according to the equations developed in Section 6. Matlab's ability to manipulate complex numbers simplifies the calculations, as illustrated by Code 8.7-1.

```
E(NSYMBOLS) = D(NSYMBOLS) - Z(NSYMBOLS);
for k=1:NC
  C(NSYMBOLS+1, k) = C(NSYMBOLS, k) + alpha*(E(NSYMBOLS) * conj(X(NSYMBOLS-k)));
end;
```

Code 8.7-1: Update Equalizer

8.8 Compute Errors

Finally, we compute the error metrics of the simulation. Code 8.8-1 not only computes the number of slicing errors (for use in BER calculations), but also the mean squared error (for use in evaluating the equalizer).

```
%  Needed Calculations

C_final = C((Ninit+Npn+Nss), :);
slicing_errors_real = real(XSNEAR((Ninit+Npn+1):(Ninit+Npn+Nss)) - ...
                      D((Ninit+Npn+1):(Ninit+Npn+Nss)));
slicing_errors_imag = imag(XSNEAR((Ninit+Npn+1):(Ninit+Npn+Nss)) - ...
                      D((Ninit+Npn+1):(Ninit+Npn+Nss)));
Nerrors = sum(slicing_errors_real| slicing_errors_imag);
MSE = 10*log10(mean(abs(E((Ninit+Npn+1):(Ninit+Npn+Nss))).^2) / ...
       mean(abs(XMOD((Ninit+Npn+1):(Ninit+Npn+Nss))).^2))
```

Code 8.8-1: Compute Errors

9. Ideal Equalizer Taps

9.1 Purpose of Ideal Equalizer Taps

It is critical to know whether the receiver is successfully equalizing the cable. Convergence can be gathered by looking at the tap variations over time, and convergence to the proper set of weights can be assumed by successful demodulation output. It would be useful, however, to have a theoretical mechanism for predicting the optimal tap weights. This is possible in our simulation world because, unlike in real life, we have complete knowledge of our baseband impulse response.

9.2 Theory Behind Ideal Equalizer Taps

The complex baseband output is calculated as a complex convolution of the baseband response with the signal input, as demonstrated in Equation 9.2-1.

$$
\begin{aligned}
x[n] &= h * s[n] \\
&= \sum_{k=0}^{N-1} h_k \; s[n-k] \\
&= \sum_{k=0}^{N-1} (hr + jhi)_k \; (sr + jsi)[n-k] \\
&= \sum_{k=0}^{N-1} hr_k \; sr[n-k] - hi_k \; si[n-k] + j\, hi_k \; sr[n-k] + j\, hr_k \; si[n-k] \\
xr[n] &= \sum_{k=0}^{N-1} hr_k \; sr[n-k] - hi_k \; si[n-k] \\
xi[n] &= \sum_{k=0}^{N-1} hi_k \; sr[n-k] + hr_k \; si[n-k]
\end{aligned}
$$

Equation 9.2-1: Calculation of Complex Baseband Output

It can be seen from Equation 7.2-6 that, in general, h(t) is complex and can be defined as $h_{real}(t)+j\, h_{imag}(t)$. This means that a signal that is input to one channel in the transmitter appears (distorted

of course) at the output of both channels. This relationship for a linear channel is summarized in Figure 9.2-1.

$$\mathbf{sr} \xrightarrow{hr} xr$$
$$\mathbf{sr} \xrightarrow{hi} xi$$
$$\mathbf{si} \xrightarrow{-hi} xr$$
$$\mathbf{si} \xrightarrow{hr} xi$$

Figure 9.2-1: Complex Baseband Transfer Functions

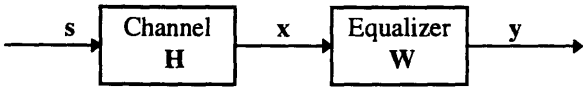
Calculation of the complex equalizer output follows Equation 9.2-1 exactly, if the weight vector w is substituted for the baseband vector h, the equalizer input x is substituted for the cable input s, and the equalizer output z is substituted for the baseband output x. We can, therefore, mimic Figure 9.2-1 in Figure 9.2-2.

$$\mathbf{xr} \xrightarrow{wr} zr$$
$$\mathbf{xr} \xrightarrow{wi} zi$$
$$\mathbf{xi} \xrightarrow{-wi} zr$$
$$\mathbf{xi} \xrightarrow{wr} zi$$

Figure 9.2-2: Complex Equalizer Transfer Functions

Following the logic presented in Section 6.2, we want the equalizer output Z to equal the cable input S, and so the weight vector W must equalize the baseband response H. By comparing Figure 9.2-1 with Figure 9.2-2, we further note that zr must equalizer hr, and zi must equalize hi.

Knowing the baseband response H, there are two ways to compute the ideal weight vector W. The first is what we can call the Frequency Response Reciprocal method. As reproduced from Section 6.1, Equation 9.2-2 gives a simple calculation for ideal frequency domain weights.



$$
\begin{aligned}
Y(f) &= W(f)X(f) \\
&= W(f)\,\{H(f)S(f)\} \\
&= \{W(f)H(f)\}\,W(f)
\end{aligned}
$$

$$
\textit{if} \quad W(f) = \frac{1}{H(f)}
$$

$$
\textit{then} \quad Y(f) = S(f)
$$

Equation 9.2-2: Frequency Response Reciprocal

Slightly more practically, we can derive an ideal time domain weight vector from the table reproduced from Section 1.2.1:

w1	**w2**	**w3**	**w4**	**w5**		**s**
h1	0	0	0	0	=	0
h2	h1	0	0	0	=	0
h3	h2	h1	0	0	=	0
h4	h3	h2	h1	0	=	1
0	h4	h3	h2	h1	=	0
0	0	h4	h3	h2	=	0
0	0	0	h4	h3	=	0
0	0	0	0	h4	=	0

As mentioned in Section 1.2.1, this is an overconstrained set of equations. The most common linear algebra solution to an overconstrained set of equations is a least squares fit, as given in Equation 9.2-3. (Please note that, in this equation, H does not represent the frequency domain representation of the

baseband response, nor is d the sliced equalizer output. Instead, variables represent values in the above chart.)

$$\mathbf{H}\,\mathbf{w} = \mathbf{d}$$
$$\mathbf{H}^T\mathbf{H}\,\mathbf{w} = \mathbf{H}^T\mathbf{d}$$
$$(\mathbf{H}^T\mathbf{H})^{-1}\,\mathbf{H}^T\mathbf{H}\,\mathbf{w} = (\mathbf{H}^T\mathbf{H})^{-1}\,\mathbf{H}^T\mathbf{d}$$
$$\mathbf{w} = (\mathbf{H}^T\mathbf{H})^{-1}\,\mathbf{H}^T\,\mathbf{d}$$

Equation 9.2-3: Linear Algebra Method

9.3 Computation of Ideal Equalizer Taps

9.3.1 Frequency Response Reciprocal Method

The ease with which Matlab performs FFTs makes the computation of ideal weights via the Frequency Response method very straight forward, as shown in Code 9.3-1. For historical (PCSI documentation) reasons, the weight vector is called C rather than W.

```
%  FFT Reciprocal method

BASEBAND_MSE = fft(baseband, length(baseband)*2);
INV_BASEBAND_MSE = 1./BASEBAND_MSE;

Copt_long = fftshift(ifft(INV_BASEBAND_MSE));

projection_long = conv(baseband, Copt_long);

MSE_min = 10*log10((sum(abs(projection_long).^2) - max(abs(projection_long)).^2) / ...
                        max(abs(projection_long).^2));
```

Code 9.3-1: Ideal Weights via FFT Reciprocal Method

9.3.2 Linear Algebra Method

The Linear Algebra method is more appropriate for computing the ideal weight vector when the number of weights is limited, as it is in practical applications. Code 9.3-1 demonstrates the calculation of an ideal weight vector of length NC.

```
for k=1:(NC)
  H(k,:) = [zeros(1,(k-1)) baseband.' zeros(1,(NC-k))];
end;
H = H.';

[peak delay] = max(real(baseband));

desired_short = zeros(1,(NC+length(baseband)-1));
desired_short(delay+NC/2-1) = 1;
desired_short = desired_short.';

Copt_short = inv(H' * H) * H' * desired_short;

projection_short = H * Copt_short;

MSE_min = 10*log10(sum(abs(desired_short - projection_short).^2) / ...
                       max(abs(projection_short).^2));
```

Code 9.3-1: Ideal Weights via Linear Algebra Method

10. Simulation Results

The simulation has the following adjustable parameters:

- Carrier, symbol, and sample frequencies
- Number of bits/symbol
- Number of equalizer taps
- Initialization of taps
- Convergence rate factor
- Length of training

By varying these parameters, the simulation was verified, the effects of cable material, cable length, and number of equalizer taps were studied, and the prototype of a monocable bit rate feasibility simulation was developed.

10.1 Simulation Validation

The simulation was validated in several ways:

- Achieved PCSI's documented MSE and convergence results given the same inpute parameters and data.
- Converged to optimal weights, as computed in Section 9.
- Bounded BER to be less than the required 1E-7.
- Varies MSE as expected: increases with temperature, length of cable, number of bits/symbol, fewer taps, smaller SNR, and use of T5 mode over T7.

10.2 Cable Material

The effect of cable material on telemetry can be analyzed by changing the cable impulse response while keeping all other parameters constant. Code 10.2-1 demonstrates the implementation of a simulation.

```
[input_sample_freq, output_sample_freq, fsample, fcarrier, fsymbol, Nyquist_filter] = ...
       sim_Initializing;

for cable_count = 20:24
  if (cable_count==20)
    cable_name = 'T5hot739P';
    raw_length = 22857;
  elseif (cable_count==21)
    cable_name = 'T5hot739Z';
    raw_length = 18000;
  elseif (cable_count==22)
    cable_name = 'T5hot746NT';
    raw_length = 27647;
  elseif (cable_count==23)
    cable_name = 'T5hot746P';
    raw_length = 21549;
  elseif (cable_count==24)
    cable_name = 'T5hot746V';
    raw_length = 24224;
  end;

  [cable_fft, cable_ifft, freq_delta] = ...
       sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, cable_name);

  cable_fft = cable_fft .^ (desired_length/raw_length);
  cable_ifft = fftshift(ifft(cable_fft));

  [baseband, BASEBAND] = ...
       sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
         cable_ifft, cable_fft);
  [C_final, MSE, XMOD, XSNEAK, XCABLE, X, C, Z, E, D, Nerrors] = ...
       sim_Equalizing(baseband, NC, QAM, SNR, TRAIN_NOISE, Npn, Nss, C_init,alpha_scale);

end;  % for cable_count
```

Code 10.2-1: Simulate with Different Cables

The results of the simulation are presented below:

```
%  EQUALIZING ALL CABLES;  DTS parameters

% Simulation took 680 seconds = 11.4 minutes
% Average of 680 seconds / 34 cables = 20 seconds per cable

>> NC=15; QAM=64; SNR=0;  % SNR = 0 = no noise
>> tic; sim_runeqtest; toc

cable_name      MSE (dB)

COAXcold223ZT   -33.9038
COAXhot223ZT    -17.5985
MONOcold122ZT   -17.6791
MONOhot122ZT    -17.6729
```

```
T2cold739P      -17.6721
T2cold739Z      -25.5885
T2cold746N      -24.2662
T2cold746P      -26.7557
T2cold746V      -25.9876
T2hot739P       -19.3932
T2hot739Z       -29.2527
T2hot746NT      -25.1265
T2hot746P       -28.1266
T2hot746V       -29.4305

T5cold739P      -37.7115
T5cold739Z      -38.4023
T5cold746NT     -39.0524
T5cold746P      -38.4691
T5cold746V      -38.9581
T5hot739P       -39.1296
T5hot739Z       -40.6525
T5hot746NT      -36.2091
T5hot746P       -40.9090
T5hot746V       -41.9450

T7cold739P      -43.5416
T7cold739Z      -43.5684
T7cold746NT     -42.8854
T7cold746P      -41.2000
T7cold746V      -44.8026
T7hot739P       -39.9288
T7hot739Z       -42.6404
T7hot746NT      -37.8501
T7hot746P       -43.8160
T7hot746V       -46.5796
```

As expected, T5 is a worse channel than T7, hot cable is more difficult to equalize than cold cable, and 7-46NT is the worst cable of the lot. The T2 cable mode is not presently used for telemetry. The simulation suggests that running standard DTS on coax or monocable would not be successful -- the equalization is not able to meet SNR requirements for 4 bits/symbol.

10.3 Cable Length

It is important to be able to model changes in cable length for at least two reasons: it is useful to understand how telemetry will operate on cables longer and shorter than those available for characterization, and cable characterizations are not done at a consistent cable lengths, so analyzing the impact of cable material requires that the lengths be normalized.

A cable characterization of raw length is modeled to be of desired length by simply raising its frequency representation to the (desired length/raw length) power, as demonstrated in Code 10.3-1.

```
[input_sample_freq, output_sample_freq, fsample, fcarrier, fsymbol, Nyquist_filter] = ...
       sim_Initializing;
[cable_fft, cable_ifft, freq_delta] = ...
       sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, cable_name);

for desired_length=(27000:500:33000)

  cable_fft_raw = cable_fft;
  cable_ifft_raw = cable_ifft;
  cable_fft = cable_fft .^ (desired_length/raw_length);
  cable_ifft = fftshift(ifft(cable_fft));

  [baseband, BASEBAND] = ...
       sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
         cable_ifft, cable_fft);
  [C_final, MSE, XMOD, XSNEAK, XCABLE, X, C, Z, E, D, Nerrors] = ...
       sim_Equalizing(baseband, NC, QAM, SNR, TRAIN_NOISE, Npn, Nss, C_init,alpha_scale);

end;  % for desired_length
```

Code 10.3-1: Simulate Different Cable Lengths

The results show that equalization performance does indeed degrade with length:

```
% Simulation took 681 seconds = 11.4 minutes
% Average of 681 seconds / 13 lengths = 52.3 seconds per length

>> tic; sim_runLtest; toc

NC=15;
QAM=64;
TRAIN_NOISE = 0;
SNR = 0;
T5hot746NT
raw_length =       27647

       desired length  MSE (dB)
       27000           -36.0333
       27500           -35.8020
       28000           -35.5736
       28500           -35.3481
       29000           -35.1255
       29500           -34.9057
       30000           -34.6886
       30500           -34.4742
       31000           -34.2623
       31500           -34.0530
       32000           -33.8462
       32500           -33.6419
       33000           -33.4400
```

We have been using a worst case cable characterization of 27 kft because we did not have a 30 kft characterization available. Happily, we see in the results above that a worst case cable of 30 kft (in fact, even 33 kft) still meets DTS specs.

10.4 Number of Equalizer Taps

The original primary purpose of the simulation was to determine how many equalizer taps DTS truly requires to meet its specs. The current implementation has 15 tap weights. It was not known whether fewer tap weights could acheive the same specs while decreasing computation cost, or whether more tap weights could enable telemetry on poorer channels at the expense of increased computation cost. The simulation of Code 10.4-1 experimented with four sets of tap weights.

```
[input_sample_freq, output_sample_freq, fsample, fcarrier, fsymbol, Nyquist_filter] = ...
        sim_Initializing;
[cable_fft, cable_ifft, freq_delta] = ...
        sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, cable_name);
[baseband, BASEBAND] = ...
        sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
          cable_ifft, cable_fft);

for NC=[7 15 31 63]

  Npn = 500*(NC/15);
  Nss = 1000*(NC/15);
  [C_final, MSE, XMOD, XSNEAK, XCABLE, X, C, Z, E, D, Nerrors] = ...
        sim_Equalizing(baseband, NC, QAM, SNR, TRAIN_NOISE, Npn, Nss, C_init,alpha_scale);
  eval(['C_NC' int2str(NC) ' = C_final;']);
  eval(['MSE_NC' int2str(NC) ' = MSE;']);

end;  % for NC
```

Code 10.4-1: Simulate Different Numbers of Equalizer Taps

The results were comforting; 15 is the appropriate number of taps for DTS. More taps would decrease equalizer error, but the impact would be negligible because random gaussian error would still dominate. Fewer taps would not provide enough equalization to run at 5 bits/symbol.

```
%  Simulation took  365.7 seconds = 6.1 minutes
%  No average reasonable because increasing number of taps increases computation time

>> QAM=64; SNR=0    %  SNR = 0 = no noise
>> tic; sim_runNCtest; toc

Number Taps     MSE (dB)

     7          -22.8196
    15          -36.2091
    31          -52.6582
    63          -54.3257
```

10.5 Bit Error Rate

As mentioned in Section 10.1, a bit error rate bound helped to validate the simulation. Computing the true BER is expensive: to be 68% confident that the BER was within 10% of 10^{-7} would require approximately 10^7/sqrt(0.10) = 10^9 symbols to be processed. [Wolaver 95, p. 90] At about 1 second/1000 symbols (equalizer not updated during BER runs), 10^9 symbols would take 10^6 seconds = 11.6 days of computation. Computing the true BER is also unnecessary.

We are only interested in a bound on the BER. Assuming that errors arrive with a Poisson distribution, then the probability of observing no errors in N symbols is $p(0)=e^{-N/BER}$. Setting the probability p(0) equal to 10%, we find that N = 2.3/BER = 2.3 * 10^7 symbols. [Wolaver 95, p. 91] At 1 second/1000 symbols, 2.3 * 10^7 symbols would take 2.3 * 10^4 seconds = 6.4 hours; much more reasonable. Code 10.5-1 provides the foundation for BER bound measurements.

```
BER = 10^(-7);
confidence = .99
SNR = 26;
total_errors = 0;
N_total = -log(1-confidence)/BER
Nperblock = 2^17
Nrepeat = ceil(N_total/Nperblock);

[input_sample_freq, output_sample_freq, fsample, fcarrier, fsymbol, Nyquist_filter] = ...
    sim_Initializing;
[cable_fft, cable_ifft, freq_delta] = ...
    sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, cable_name);
[baseband, BASEBAND] = ...
    sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
        cable_ifft, cable_fft);
[C_final, MSE, XMOD, XSNEAK, XCABLE, X, C, Z, E, D, Nerrors] = ...
    sim_Equalizing(baseband, NC, QAM, SNR, TRAIN_NOISE, Npn, Nss, C_init,alpha_scaling);

[XMOD, XCABLE, sig_energy] = ...
       sim_ComputingBERIn(baseband, NC, QAM, Nperblock);

for counter = (1:(Nrepeat-1))
  [MSE, X, Z, E, D, Nerrors] = ...
       sim_ComputingBEROut(XMOD, XCABLE, sig_energy, NC, QAM, SNR, Nperblock, C_final);
  total_errors = total_errors + Nerrors;
end;

total_errors
errors = D - XMOD;
Nerrors = length(find(errors));
```

Code 10.5-1: Compute BER Bound

10.6 Monocable

As seen in Figure 10.6-1, monocable is a much worse channel than heptacable. A glance at the sharp channel degradation with frequency suggests that a monocable telemetry system use a lower carrier frequency than DTS's 52.5 kHz.

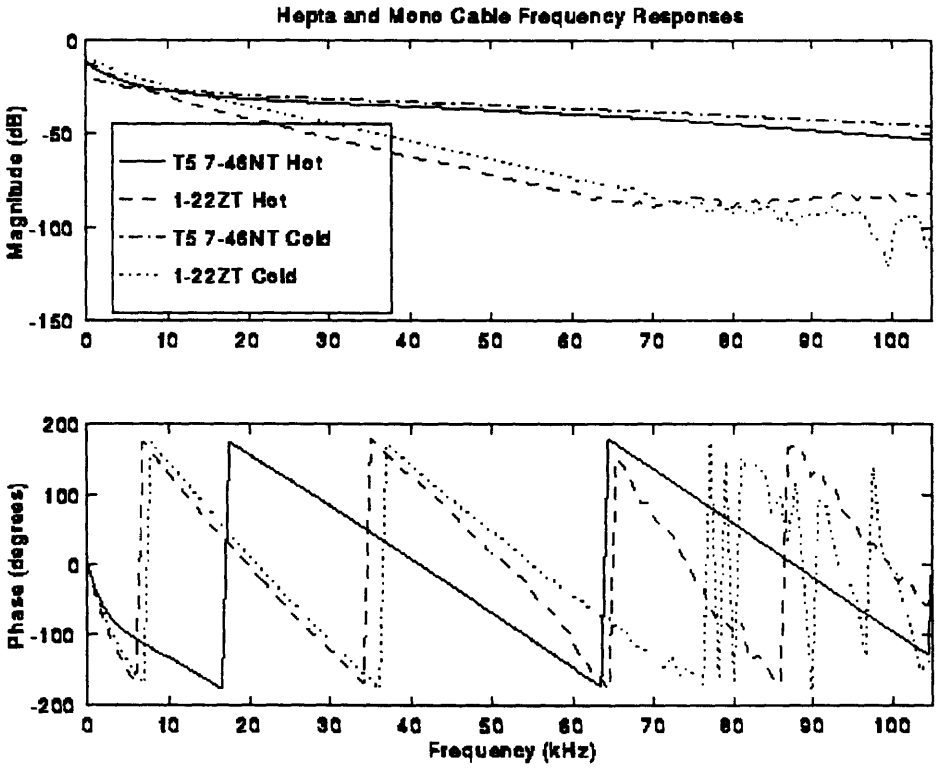


Figure 10.6-1: Hepta and Mono Cable Frequency Responses

It would be convenient if monocable telemetry could be developed as a frequency-scaled version of DTS: simply slow down the clock and thereby lower and narrow the passband into monocable's "sweet spot." This bandwidth narrowing necessarily reduces the bit rate of the telemetry system. In addition, because monocable is such a poor channel, more equalizer taps will be necessary to achieve the same level

of equalization and therefore the ability to pack the same number of bits/symbol. Code 10.6-1 loops through the various frequency scaling and equalization options.

```
cable_name = 'MONOhot122ZT'
tapinit_cable_name = 'MONOcold122ZT'
SNR = 0;
TRAIN_NOISE = 0;
QAM = 16;
C_init_MONOcold = 0;
alpha_scale = 1;

for freq_scale=0.6:(-0.05):0.3

  [input_sample_freq, output_sample_freq, fsample, fcarrier, fsymbol, Nyquist_filter] =...
       sim_Initializing;

  fcarrier = fcarrier*freq_scale;
  fsymbol = fsymbol*freq_scale;
  fsample = fsample*freq_scale;
  fbelower = fcarrier - fsymbol/2
  fbeupper = fcarrier + fsymbol/2
  bits_per_sec = log2(QAM)*fsymbol
  [cable_fft, cable_ifft, freq_delta] = ...
       sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, cable_name);
  if (C_init_MONOcold)
    [tapinit_cable_fft, tapinit_cable_ifft, noop] = ...
       sim_LoadingCable(input_sample_freq, output_sample_freq, fsample, ...
       tapinit_cable_name);
  end;

  baseband, BASEBAND] = ...
       sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
         cable_ifft, cable_fft);
   if (C_init_MONOcold)
    [tapinit_baseband, tapinit_BASEBAND] = ...
       sim_FindingBaseband(fsample, fcarrier, fsymbol, freq_delta, Nyquist_filter, ...
         tapinit_cable_ifft, tapinit_cable_fft);
  end;

  for NC=[31 26 21 18 15]

      Npn = 10*500*(NC/15);
      Nss = 1000*(NC/15);

      [C_min, C_NC, MSE_min, MSE_NC] = ...
         sim_OptimizingTaps(baseband, NC);
      MSE_NC

      if (C_init_MONOcold)
        [noop, C_init, noop, noop] = ...
          sim_OptimizingTaps(tapinit_baseband, NC);
      else
        C_init = 'impulse';
      end;

      [C_final, MSE, XMOD, XSNEAK, XCABLE, X, C, Z, E, D, Nerrors] = ...
        sim_Equalizing(baseband, NC, QAM, SNR, TRAIN_NOISE, Npn, Nss, C_init,alpha_scale);
      Nerrors

  end; % for NC
end;   % for freq_scale
```

Code 10.6-1: Monocable Simulation

The results of the monocable simulation are summarized in Figure 10.6-2. The MSE was calculated for each frequency scaling/number of taps option. This MSE was used to compute the maximum number of bits/symbol which can be transmitted at a BER of 1E-7. Taking the narrowed bandwidth into account, the maximum data rate is computed for each option. It is clear that there are several possibilities for which more than 120 kbits/second can be achieved.

MONOhot122ZT	30000ft	250F	78.2ohm term				THEORETICAL						
							Convergence issues not included						
Theoretical QAM SNR Requirements							(but should not be a problem)						
Need to add 3 dB implementation margin													
QAM	*64*	*32*	*16*	*8*	*4*								
Bits/Symbol	*6*	*5*	*4*	*3*	*2*								
SNR (dB)	27	24	21	17.5	14								
Frequency Scaling factor	**0.8**	**0.75**	**0.7**	**0.65**	**0.6**	**0.55**	**0.5**	**0.45**	**0.4**	**0.35**	**0.3**	**0.25**	**0.2**
Number of Taps													
31	22	24	26	28	30	32	33	37	36	39	45	44	21
26	20	22	24	25	28	29	30	33	33	35	40	38	20
21	19	20	22	23	25	25	26	29	28	30	33	30	15
18	17	19	20	21	23	23	24	27	26	28	31	27	14
15	16	18	18	19	21	21	22	24	23	25	27	25	13
31 Tap SNR	22	24	26	28	30	32	33	37	36	39	45	44	21
Meets QAM requirement	17.5	21	21	24	27	27	27	27	27	27	27	27	17.5
Max Bits/Symbol	3	4	4	5	6	6	6	6	6	6	6	6	6
Data Rate (kbit/second)	168	210	196	228	252	231	210	189	168	147	126	105	84
26 Tap SNR	20	22	24	25	28	29	30	33	33	35	40	38	20
Meets QAM requirement	14	17.5	17.5	21	24	24	24	27	27	27	27	27	27
Max Bits/Symbol	2	3	3	4	5	5	5	6	6	6	6	6	6
Data Rate (kbit/second)	112	158	147	182	210	193	175	189	168	147	126	105	84
21 Tap SNR	19	20	22	23	25	25	26	29	28	30	33	30	15
Meets QAM requirement	14	14	17.5	17.5	21	21	21	24	24	27	27	27	27
Max Bits/Symbol	2	2	3	3	4	4	4	5	5	6	6	6	6
Data Rate (kbit/second)	112	105	147	137	168	154	140	158	140	147	126	105	84
18 Tap SNR	17	19	20	21	23	23	24	27	26	28	31	27	14
Meets QAM requirement	14	14	14	17.5	17.5	17.5	21	21	21	24	27	27	27
Max Bits/Symbol	2	2	2	3	3	3	4	4	4	5	6	6	6
Data Rate (kbit/second)	112	105	98	137	126	116	140	126	112	123	126	105	84
15 Tap SNR	16	18	18	19	21	21	22	24	23	25	27	25	13
Meets QAM requirement	0	14	14	14	17.5	17.5	17.5	17.5	17.5	21	24	21	0
Max Bits/Symbol	0	2	2	2	3	3	3	3	3	4	5	4	0
Data Rate (kbit/second)	0	105	98	91	126	116	105	95	84	98	105	70	0

Figure 10.6-2: Maximum Monocable Data Rates

11. Conclusions

The realization of a new wireline acquisition front end has made it possible for Schlumberger to redesign its uphold receiver. During the re-design process, it became clear that a simulation of the quadrature amplitude demodulation system would be of use. The purpose of this thesis was to simulate the telemetry demodulation process.

As outlined in Section 1.1, the thesis had three goals:

1. To document an understanding of DTS demodulation.
2. To create a simulation of the demodulation process.
3. To demonstrate the use of the simulation as a design tool.

The project deliverables ensured that all three goals were met. In addition, the project resulted in valuable simulation experience -- a fourth deliverable.

11.1 Telemetry Understanding

This thesis and the simulation code document an understanding of QAM, timing recovery, low pass filtering, and adaptive equalization. The understanding was gained through reading background literature, DTS manuals and documentation, and assembly code.

11.2 Telemetry Simulation

The simulation has been verified. It is able to measure MSE and BER for a QAM telemetry system, given a cable impulse response and values of carrier, symbol, and sample frequencies, number of bits/symbol, cable length, length of training, number of taps, tap initialization, and SNR.

The simulation was used to confirm that the current DTS parameters are appropriate to meet DTS specifications.

11.3 Telemetry Design Tool

The prototype monocable simulation showed that the simulation will make a useful design tool. It could play an important role in each of the three projects telemetry projects mentioned in Section 1.1:

1. The simulation has already forced a deep understanding of the DTS implementation. It has also verified the DTS parameters. Perhaps there will be future simulation work in optimizing parameters to reduce computation as all the demodulation functionality is moved from eight TI TMS320C2x fixed-point DSPs [TI 90] to two Analog Devices SHARC DSPs [Motorola 95].

2. It is expected that the simulation will be used to do feasibility studies on using adaptive equalization with CTS. Though listed as Non-Requirements, the Built-In Telemetry Requirements Document mentions using signal processing and adaptive equalization to increase default DTS operation to 6 bits/second [Booker 95, Section 2.3.2] and increase the cable range of CTS [Booker 95, Section 2.3.3]. The simulation can help achieve these Nice-To-Haves.

3. The Requirements Document also speaks of designing Built-In Telemetry such that it could run a new telemetry system over monocable. [Booker 95, Section 2.3.1]. A prototype simulation has already been developed to help with monocable telemetry system design and data rate feasibility. This work is expected to continue, especially after more monocable frequency response data has been gathered.

11.4 Simulation Experience

This simulation experience uncovered three pieces of information that will likely be valuable to Schlumberger in the future:

1. Through trial-and-error, it was decided that Matlab is better than i-Logix for signal processing simulation. Matlab is faster, has built-in signal processing functions, and has better graphing capabilities.

2. Also through trial-and-error, the importance of an accurate cable model was confirmed. Great care must be taken to use narrow input pulses when measuring cable characteristics.

3. Finally, the baseband response was discovered as an important telemetry concept.

The development of the QAM telemetry simulation provided a useful, interesting, and challenging thesis project.

11.5 Acknowledgments

I would like to thank:

- Lloyd Clark and Terry Mayhugh, for their insight, patience, and encouragement.
- Guy Vachon, Steve Kush, and Peter Highnam, for providing the thesis topic and resources.
- Supporters of the MIT VI-A program, for providing this opportunity to do real engineering work in such a favorable environment.
- John A. C. Bingham, for writing The Theory and Practice of Modem Design -- it is not clear that I could have completed the simulation without it.
- Friends and family, for listening during my struggles and cheering during my victories.

12. References

General

[Bingham 88] Bingham, John A. C. The Theory and Practice of Modem Design. A Wiley-Interscience Publication, John Wiley & Sons, Inc. New York. 1988.

[Campbell 95] Campbell, Heather A. "Implementation of Quadrature Amplitude Demodulation in Schlumberger's Digital Telemetry System." Thesis Proposal for Master of Engineering. Massachusetts Institute of Technology. August, 1995.

[Gardner 94] Gardner, Wallace R. and Goodman, Kenneth R. "Adaptive Telemetry System for Hostile Environment Well Logging." United States Patent number 5,365,229. November 15, 1994.

[Gardner 95] Gardner, Wallace, R., Goodman, Kenneth R., and Pickett, Robert D. "High Data Rate Wireline Telemetry System." United States Patent number 5,387,907. February 7, 1995.

[Lucky 68] Lucky, R. W., Saltz, J., and Weldon, E. J. Jr. "Equalization of the Baseband System." Principles of Data Communication.. McGraw-Hill. New York. 1968. pp. 128-165.

[Mayhugh 92] Mayhugh, Terry. "Anadrill Acoustic Telemetry Bus Feasibility Report." Internal Documentation. Schlumberger Austin Systems Center. August 5, 1992. pp. 19-39.

Digital Telemetry System

[Booker 95] "WAFE Built-In Telemetry Requirements Document. Version 0.1." Schlumberger Austin Systems Center. August, 1995.

[Clark 95] Clark, Lloyd. "The Evolution of Wireline Telemetry Systems." Internal Documentation. Schlumberger Austin Systems Center. 1995.

[DTS 91] Digital Telemetry System (DTS) Maintenance Manual. Volume 1. Schlumberger Austin Systems Center. C300129. April, 1991.

[Kelly 91] Kelly, Ronald C. and Montgomery, Michael A. "Digital Telemetry System. High Level Design. Revision H." Schlumberger Austin Systems Center. March, 1991.

DTS Uplink Implementation

[Jansen 8x] Jansen, Errol and Mayhugh, Terry. Cable impulse response and noise measurements. Lab notebook. Schlumberger Austin Systems Center. February through April, 1988.

[Mayhugh 90] Mayhugh, Terry. "DTS Modulation Techniques." Internal Memorandum. Schlumberger Austin Systems Center. August 22, 1990.

[Montgomery 93] Montgomery, Michael A. "Method and Apparatus for Quadrature Amplitude Modulation of Digital Data using a Finite State Machine." United States Patent number 5,253,271. October 12, 1993.

[Montgomery 9x] Montgomery, Michael A. "Finite State Machine Implementation of a Quadrature Amplitude Modulator." Internal Memorandum. Schlumberger Austin Systems Center.

[PCSI 8x] Pacific Communication Sciences, Inc. "Proposal For Development of a Prototype Cable Modem for the Digital Telemetry System." Internal Memorandum. Schlumberger Austin Systems Center.

[PCSI 88-1] Pacific Communication Sciences, Inc. "High Level Design. Digital Telemetry System Cable Modem Prototype Development Program." Internal Memorandum. Schlumberger Austin Systems Center. February 9, 1988.

[PCSI 88-2] Pacific Communication Sciences, Inc. "High Level Design. URD Board for DTS. Production Development Program." Internal Memorandum. Schlumberger Austin Systems Center. November 19, 1988.

[PCSI 89] Pacific Communication Sciences, Inc. "Uphole Receiver Digital Board for DTS. Low Level Design." Internal Memorandum. Schlumberger Austin Systems Center. February 6, 1989.

Quadrature Amplitude Modulation

[Samueli 94] Samueli, Henry. "A Closer Look at QAM." Electronic Design. November 7, 1994. pp. 73-74.

Adaptive Equalization

[Johnson 95] Johnson, C. R. "On the Interaction of Adaptive Filtering, Identification, and Control." IEEE Signal Processing Magazine. March 1995. pp 22-37.

[TI 90] Texas Instruments. "Implementation of Adaptive Filters with the TMS320C25 and TMS320C30." DSP Applications, Volume 2. TI. 1990. pp. 191-272.

[Verghese 94] Verghese, George. "Deterministic Least Squares." 6.011 Introduction to Communication, Control, and Signal Processing, Lecture 12. Massachusetts Institute of Technology. Fall 1994.

Simulation Software

[Burrus 94] Burrus, C. Sidney, et al. Computer-Based Exercises for Signal Processing Using Matlab®. Matlab Curriculum Series. Prentice Hall. New Jersey. 1994.

[i-Logix 91] i-Logix. The Languages of Statemate. i-Logix. January, 1991.

[i-Logix 93] i-Logix. Statemate User Reference Manual. Volumes I and II, Version 5.0. i-Logix. June, 1993.

Bit Error Rates

[Wolaver 95] Wolaver, Dan H. "Measure Error Rates Quickly and Accurately." Electronic Design. May 30, 1995. pp. 89-98.

Digital Signal Processor Hardware and Software

[Motorola 95] ADSP-21062 SHARC Super Harvard Archetecture Computer.
http://www.mwmedia.com/tpvs/analog/dsp/sdsp21062.htm

[TI 90] Texas Instruments. TMS320C2x User's Guide. Texas Instruments. 1990.

[TI 95] TI TMS320C3x Floating-point DSPs.
http://www.ti.com/sc/docs/dsps/prodinfo/newc3x.htm

Appendices

1. DTS Specifications

The following specifications were made during the high-level design of DTS [Kelly 91]:

- Data rate on 30 000 ft of hot 7-46NT cable: 500 kbits/s uplink and 6.3 kbits/s downlink variable to 128.7 kbits/s uplink and 40 kbits/s downlink.
- Maximum tools per string: 15
- Modulation type: QAM uplink and Biphase downlink
- Carrier Frequency: 52.5 kHz
- Symbol Rate: 70 kHz
- Sample Rate: 210 kHz
- Transmission mode: half-duplex
- Training time: 575.3 ms for long training, 10.0 ms for short training.
- Latency: as long as 322 ms (data transmission to reception of MAXIS response)
- Error rate: >800 hours mean time between data loss
- Hot cable bit error rate of 10^{-7} during uplink transmission rate of 700 kbits/second.
- Hot cable bit error rate of 10^{-6} during downlink transmission rate of 70 kbits/second.
- Performance under noise:

Data Rate (kbits/s)	SNR (dB)
210	27.0
280	30.0
350	33.0
420	37.0

- Full operation is guaranteed over all cable lengths from zero to the maximums listed below. The 175°C temperature limitations are the result of the DTC, while the 150°C limitations are due to the logging cable.

Cable Type	Maximum Length (kft)	Maximum Temperature (°C)
7-52NT	30	175
7-46P	30	150
7-46NT	30	175
7-46V	30	175
7-46P + NT stinger	22 + 8	175
7-39P	TBD	150
7-39Z	TBD	175

- Time stamps: +/- 45 microsecond accuracy worst case; +/- 20 microsecond if the tool pauses before writing a new message.

- Tool to Tool Synchronization: +/- 2 microseconds
- Link margins: assured when the electrical noise on the logging cable does not exceed 2.26 microvolts rms per square root Hertz into 50 ohms.
- Will not support any analog-transmitted measurement, except Spontaneous Potential and Casing Collar Locators.
- No current developments to support: CSU-D, hostile environment logging, monocable, or coaxial logging cable on DTS.

2. Simulation Function Headers

Headers for the core Matlab functions are listed below:

```
% sim_Initializing() initializes parameters
%
% OUTPUTS       input_sample_freq
%               output_sample_freq
%               fsample
%               fcarrier
%               fsymbol
%               Nyquist_filter
% INPUTS         [none]

% sim_LoadingCable() loads cable impulse responses
%
% OUTPUTS       cable_fft
%               cable_ifft
%               freq_delta
% INPUTS        input_sample_freq
%               output_sample_freq
%               highest_freq
%               cable_name

% sim_FindingBaseband() computes the baseband cable response
%
% OUTPUTS       baseband
%               BASEBAND
% INPUTS        fsample
%               fcarrier
%               fsymbol
%               freq_delta
%               Nyquist_filter
%               cable_ifft
%               cable_fft

% sim_Equalizing() equalizes the cable
%
% OUTPUTS       C_final
%               MSE
%               XMOD
%               XSNEAK
%               XCABLE
%               X
%               C
%               Z
%               E
%               D
%               Nerrors
% INPUTS        baseband
%               NC
%               QAM
%               SNR
%               TRAINING_NOISE
%               Npn
%               Nss
%               C_init
%               alpha_scale;

% sim_OptimizingTaps() calculates the optimal tap weights
%
% OUTPUTS       Copt_long
%               Copt_short
%               MSE_long
%               MSE_short
% INPUTS        baseband
%               NC
```

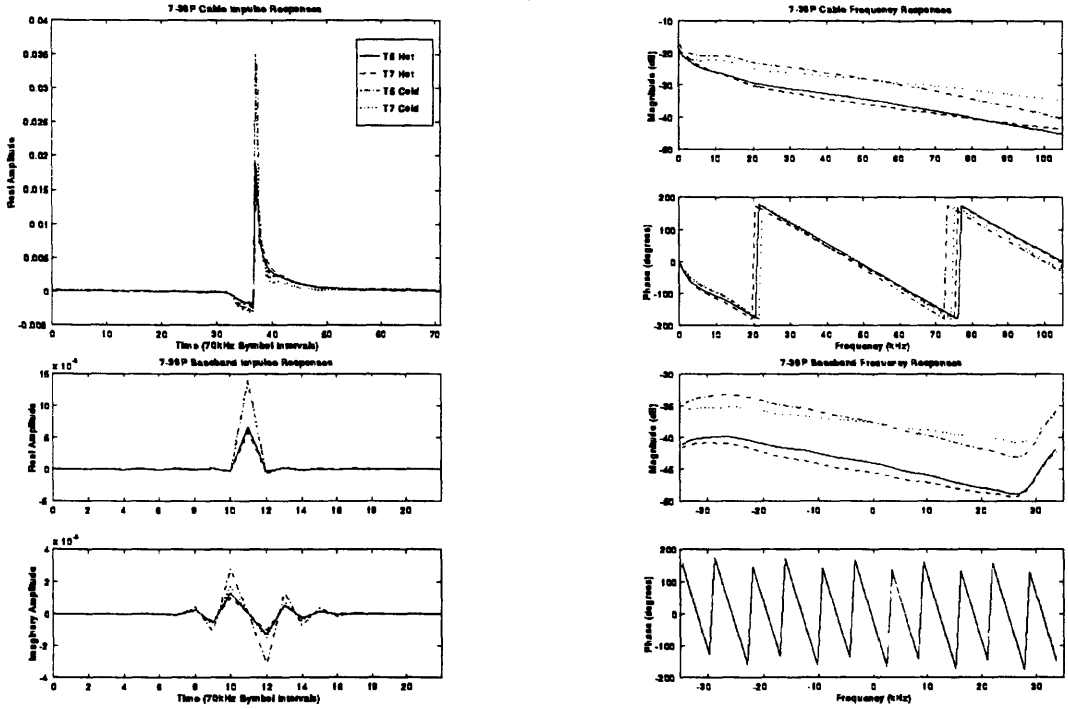
```
% sim_ComputingBERIn() computes the input to the cable and convolves with baseband
%
% OUTPUTS       XMOD
%               XCABLE
%               sig_energy
% INPUTS        baseband
%               QAM
%               Nss

% sim_ComputingBEROut() counts the number of slicing errors
%
% OUTPUTS       C_final
%               MSE
%               X
%               Z
%               E
%               D
%               Nerrors
% INPUTS        XMOD
%               XCABLE
%               sig_energy
%               NC
%               QAM
%               SNR
%               Nss
%               C_final
```

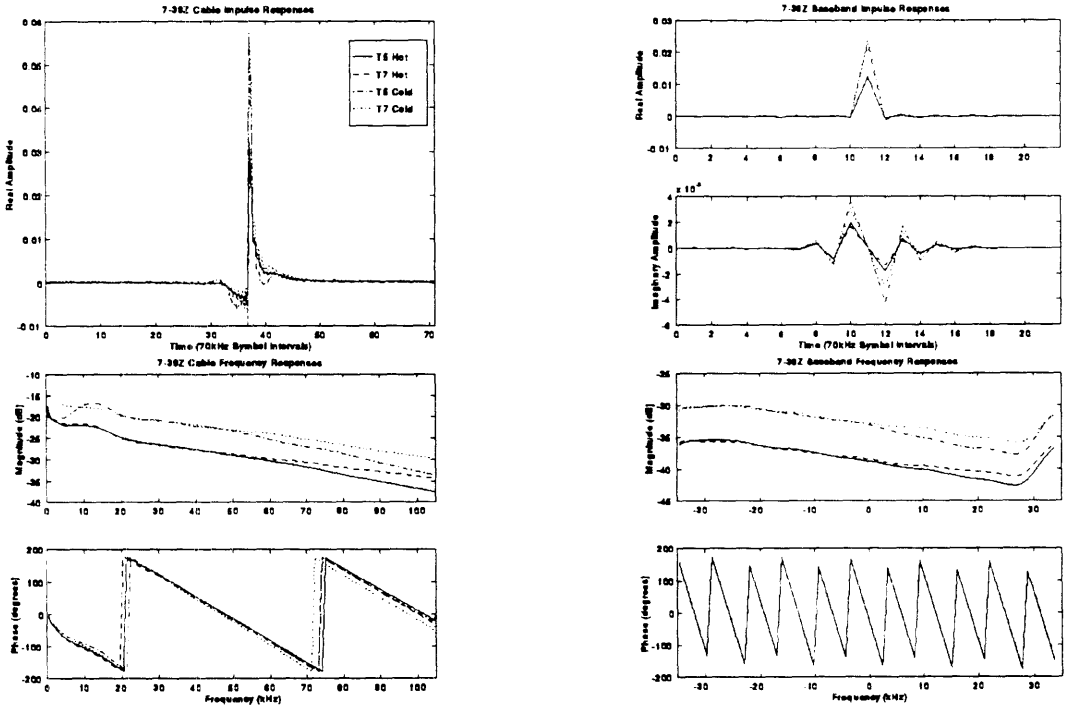
3. Cable Characterizations

Schlumberger had cable characterization data in a raw format [Jansen 8x]; during the simulation process I computed true cable characterizations from all these raw measurements. The plots are included here because they might be useful for future telemetry investigations. The baseband plots were all calculated using standard DTS parameters.

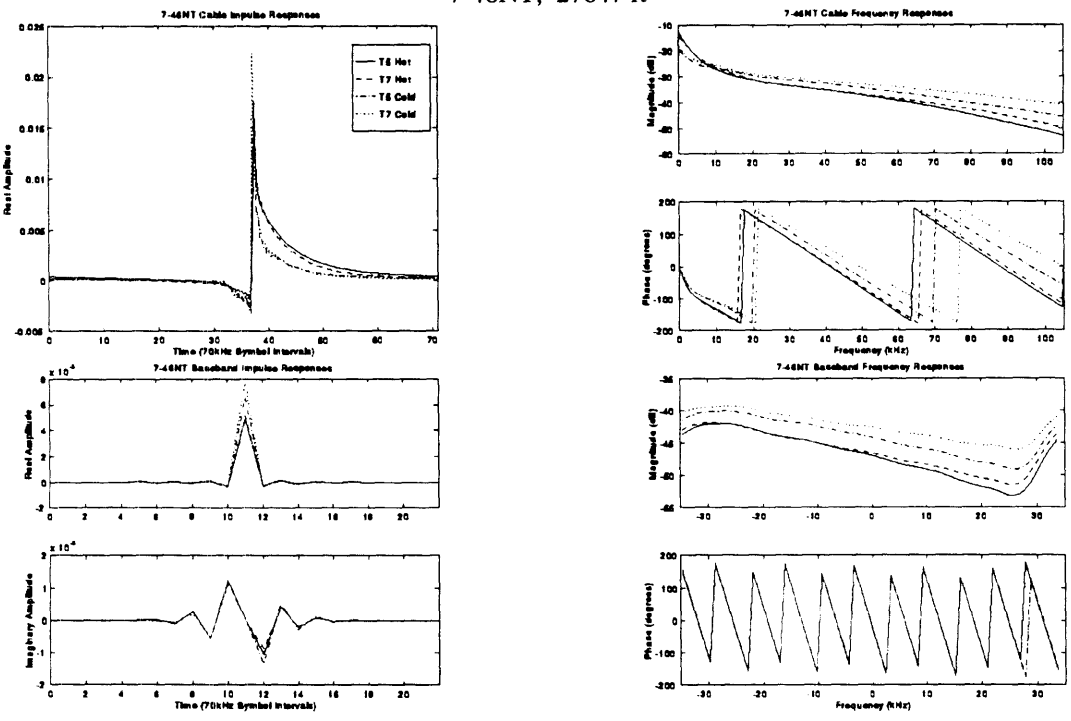
7-39P; 22857 ft



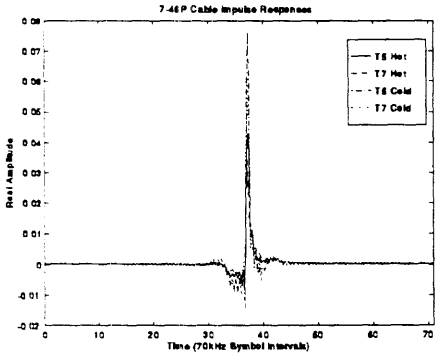
7-39Z; 18000 ft

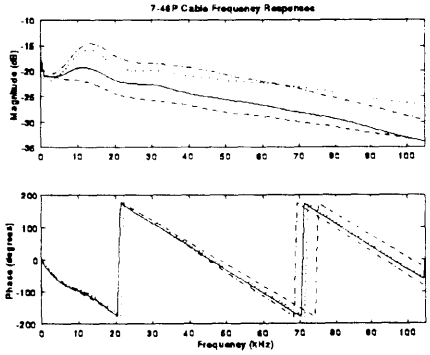


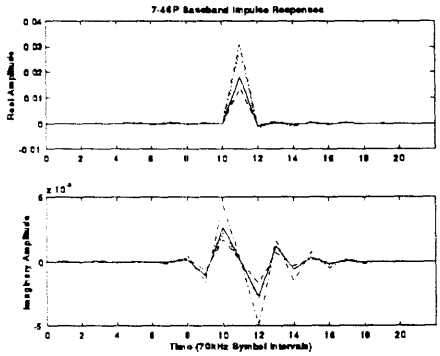
7-46NT; 27647 ft

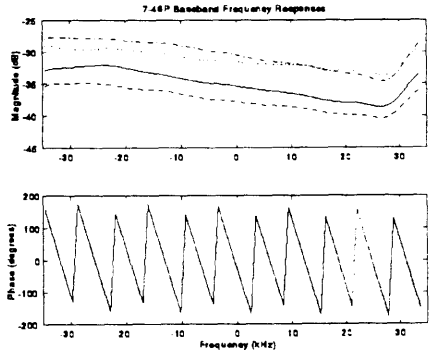


7-46P; 21549 ft

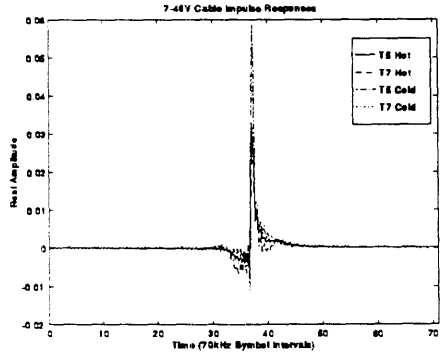


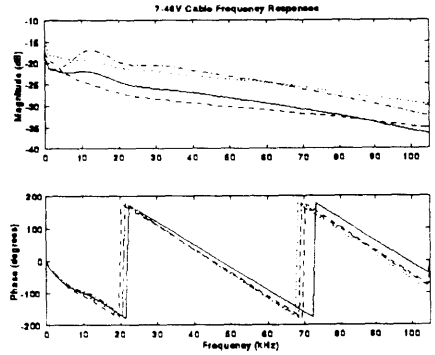


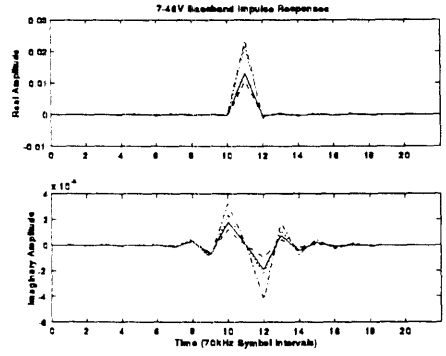


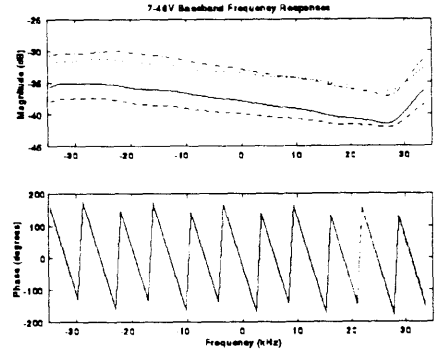


7-46V; 24224 ft

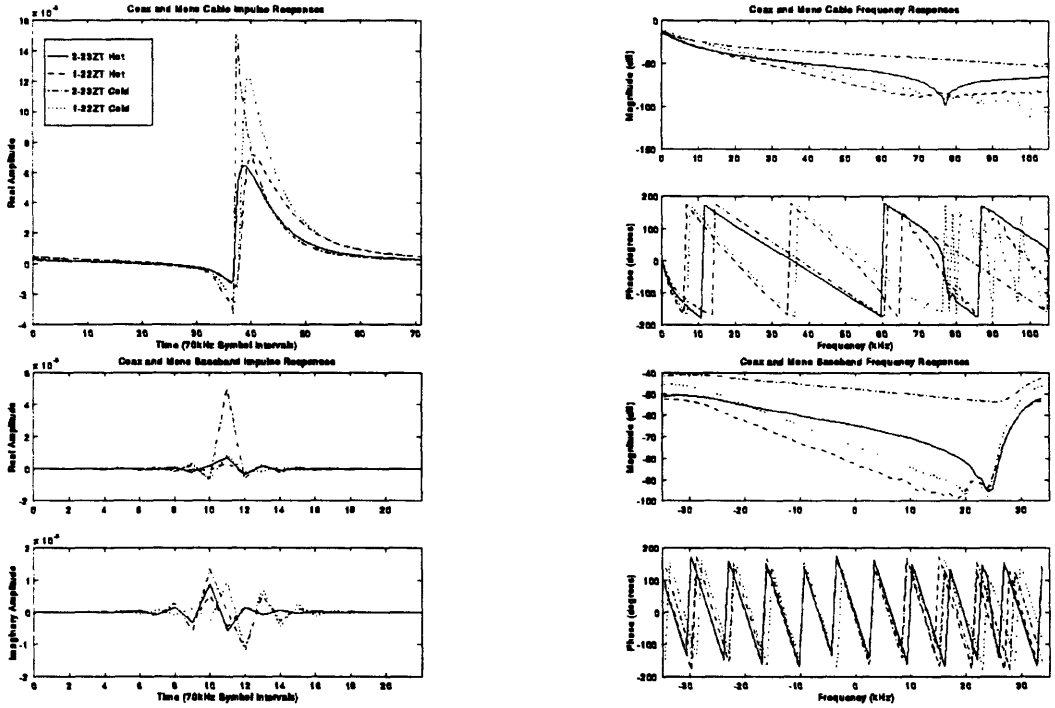




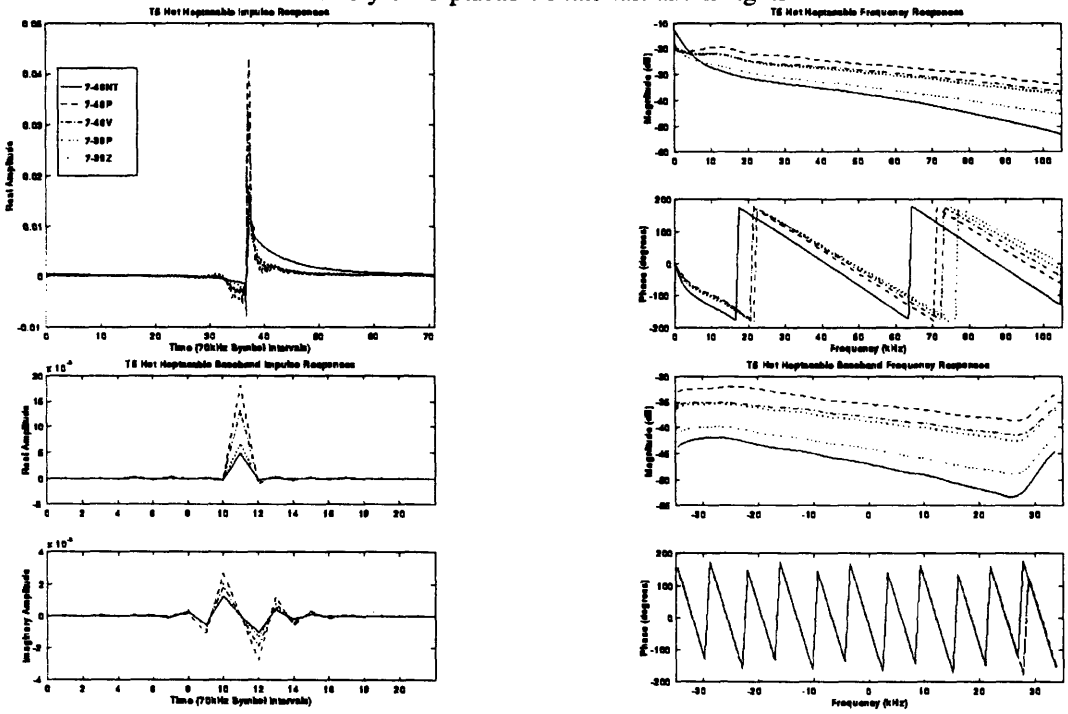




Coax and Monocable; 30000 ft



Variety of Heptacable Materials and Lengths



Hepta and Monocable Frequency Responses