Time Reversal Methods for Wireless Communications

BY

Steven Seiden

BS, Binghamton University, 2003

THESIS

Submitted in partial fulfillment of the requirements for
the degree of Master of Science in Electrical Engineering
in the Graduate School of
Binghamton University
State University of New York
2004

Accepted in partial fulfillment of the requirements for
the degree of Master of Science in Electrical Engineering
in the Graduate School of
Binghamton University
State University of New York
2004

Mark Fowler _____ May 3rd, 2004
Department of Electrical Engineering

Richard Plumb _____ May 3rd, 2004
Department of Electrical Engineering

Xiaohua Li _____ May 3rd, 2004
Department of Electrical Engineering

Abstract

Time reversal mirrors have the ability to focus energy back to its origin. Current wireless systems are designed with great consideration given to their ability to provide services to multiple users. Current technologies distinguish between users in the time, frequency, and code domains; however they ignore the spatial domain. Time reversal mirrors theoretically allow users in different locations to transmit and receive their signals at the same time using the same frequencies. Time reversal mirrors can also provide secure transmissions and help to simplify receiver design.

Current wireless systems go to great lengths to counteract the effects of the small scale fading using channel equalization, diversity techniques, and error correction codes. All these methods attempt to extract the desired signal from the replicas created by multipath propagation. Time reversal mirrors, on the other hand, attain greater accuracy in the presence of multipath propagation. The presence of reflectors directs more energy on the receiving array, in essence providing more information about the location of the receiver, which allows the array to focus even more strongly on the receiver.

This paper discusses the established uses of time reversal mirrors as well as current wireless technology. The gap between the two fields is bridged and tested. Several different electromagnetic simulations are used to test the usefulness of a time reversal mirror. A graphical user interface (GUI) is developed that implements the finite difference time domain method in various environments. The GUI is used to conduct experiments on different parameters of a time reversal mirror as well as different modulation formats. Future simulations are outlined, and a real test setup is presented in order to outline major areas of the field that can still be evaluated.

Acknowledgements

I would like to thank Professor Fowler and Professor Plumb for their help and inspiration. Not only were they a great help on this thesis, but they provided me with guidance throughout my graduate and undergraduate career.

I would also like to thank everyone that asked me about my thesis and listened patiently even if it sounded like a different language to them. Especially Aviva, who spent that time to read my thesis and correct it when she was very busy with her own work. My family was also very important to me in this entire process. They always stood behind me and encouraged me to follow my interests. They made me feel that no matter what the outcome of this process, their love for me would never change.

Table of Contents

1. Introduction 1
   1.1 Multiple Access Techniques 1
      1.1.1 Time Division Multiple Access (TDMA) 3
      1.1.2 Frequency Division Multiple Access (FDMA) 4
      1.1.3 Code Division Multiple Access (CDMA) 5
      1.1.4 "Spotlight" Multiple Access (SMA) vs. Space Division Multiple Access (SDMA) 8
   1.2 Multipath Propagation 9
      1.2.1 Overcoming Multipath Effects 12
      1.2.2 – Using Multipath Propagation to Our Advantage 14
   1.3 Applications of Spatial Focusing 15
      1.3.1 Security 15
      1.3.2 Multiple Access 16
      1.3.3 – Reduced Receiver Complexity 16
2. Background 18
   2.1 Time Reversal Theory 19
   2.2 TRM for Underwater Communications 20
   2.3 TRM for Radar Image Processing 21
   2.4 Array Spacing, Shape and Size 22
   2.5 Inhomogeneous Media 23
   2.6 Receiver Motion 24
3. Digital Modulation Techniques 26
   3.1 Amplitude Shift Keying (ASK) 26
   3.2 Phase Shift Keying 27
   3.3 Frequency Shift Keying (FSK) 29
   3.4 Pulse Shaping Filter 30
   3.5 Modulation Trade-offs 32
4. Ray Model Simulation 33
   4.1 Single Reflections 33
   4.2 Double Reflections 38
   4.3 Limitations of Ray Model 40
5. Electromagnetic Simulation 42
   5.1 FDTD Method 42
      5.1.1 Yee Cell 44
      5.1.2 Determining the Cell Size 45
      5.1.3 Determining the Time Step 46
      5.1.4 Specifying the Incident Field 47
      5.1.5 Building an Object 48
      5.1.6 Absorbing Boundary Conditions 48
      5.1.7 Two Dimensional Simulation 50
   5.2 Test Cases 51
   5.3 Graphical User Interface (GUI) Development 55
6. Simulation Results 63
   6.1 Modulation Format 63

6.2 Length of the Array 65
6.3 Off Axis Focusing 68
6.4 Line of Sight Environment 70
6.5 No Line of Sight Environment 72
7. Future Considerations 74
7.1 Test Scenarios 74
7.2 Real Testing 75
Appendix 77
A.1 Ray Simulation (Single Reflections) 77
A.2 Ray Simulation (Double Reflections) 80
A.3 FDTD Code Part 1 84
A.4 FDTD Code Part 2 85
A.5 GUI Code 88
Bibliography 95

List of Figures

FIGURE 1.1– TDMA SCHEME [1] 3
FIGURE 1.2 – FDMA SCHEME [1] 5
FIGURE 1.3 – CDMA SCHEME [1] 6
FIGURE 1.4 –FH- CDMA SCHEME [SKLAR] 6
FIGURE 1.5 – MULTIPLE ACCESS SCHEMES OF WIRELESS SYSTEMS [1] 8
FIGURE 1.6 - RAYLEIGH DISTRIBUTION AND RICEAN DISTRIBUTIONS 11
FIGURE 1.7 – RAKE RECEIVER [4] 13
FIGURE 2.1- (A) FORWARD PROPAGATION (B) REVERSE PROPAGATION WITH FOCUSING [12] 23
FIGURE 3.1 – ASK MODULATED SIGNAL 27
FIGURE 3.2 – PSK MODULATED SIGNAL 28
FIGURE 3.3 – FSK MODULATED SIGNAL 29
FIGURE 3.4 – ROOT RAISED COSINE FILTER IMPULSE RESPONSE 31
FIGURE 3.5 – RAISED COSINE FILTER FREQUENCY RESPONSE 32
FIGURE 4.1 – SINGLE REFLECTIONS SETUP 34
FIGURE 4.2 – RECEIVED SIGNAL AT THE ARRAY 35
FIGURE 4.3 – RECEIVED SIGNAL AT RECEIVER 37
FIGURE 4.4 –RESULT FROM SINGLE REFLECTIONS 37
FIGURE 4.5– DOUBLE REFLECTIONS, SIGNAL RECEIVED AT THE ARRAY 38
FIGURE 4.6 – DOUBLE REFLECTIONS, SIGNAL RECEIVED AT TRANSMITTER 39
FIGURE 4.7 – DOUBLE REFLECTIONS, COMPARING TRANSMITTED AND RECEIVED SIGNALS 39
FIGURE 5.1 – YEE CELL [15] 44
FIGURE 5.2 – LEAPFROGGING DIAGRAM [15] 45
FIGURE 5.4 – TEST CASE: SPHERICAL WAVE 52
FIGURE 5.5 – SPHERICAL WAVE, OVERHEAD VIEW 53
FIGURE 5.6– TEST CASE: WAVEGUIDE PROPAGATION 53
FIGURE 5.7 – DIRECTED 5 ELEMENT ARRAY 54
FIGURE 5.8 – GUI INTERFACE 55
FIGURE 5.9 – USER DEFINED REFLECTORS 57
FIGURE 5.10 – GUI INTERFACE, MODULATION SCHEME 58
FIGURE 5.11 – GUI INTERFACE, RECEIVED SIGNAL AT THE ARRAY 59
FIGURE 5.12 – GUI INTERFACE, TIME REVERSED SIGNAL 60
FIGURE 5.13 – GUI INTERFACE, RECEIVED SIGNAL AT RECEIVER 61
FIGURE 6.1 - FSK MODULATION, FREE SPACE 64
FIGURE 6.2 - PSK MODULATION, FREE SPACE 65
FIGURE 6.3- ASK MODULATION, FREE SPACE 66
FIGURE 6.4 - FSK MODULATION, 9 ELEMENTS 67
FIGURE 6.5 - FSK MODULATION, 25 ELEMENTS 68
FIGURE 6.6 - FSK MODULATION, 10 DEGREES OFF AXIS 69
FIGURE 6.7 - FSK MODULATION, 20 DEGESS OFF AXIS 70
FIGURE 6.8 - FSK MODULATIOON, LINE OF SIGHT 71
FIGURE 6.9 - PSK MODULATION, LINE OF SIGHT 71
FIGURE 6.10 - FSK MODULATION, NO LINE OF SIGHT 73

1. Introduction

Most engineering problems boil down to determining the interaction between design parameters and making the correct decision in terms of tradeoffs to optimize a system under certain criteria. Wireless systems are no exception. When designing a wireless system, the ideals are high data rates, a high number of users, decreased receiver complexity to facilitate longer battery life, and perfect signal detection. In order to accommodate as many users as possible resources need to be divided among the users. In order to reduce the complexity of the receivers, additional processing can be performed at the base station. Much of the complexity is necessary to counteract the effects of small scale fading which are caused by multipath effects. There are many different ways to divide the system resources and counteract small scale fading. Time reversal methods provide another way to divide the system resources while at the same time counteracting multipath effects.

1.1 Multiple Access Techniques

In Wireless Communications, systems operate at a certain frequency. These systems have a certain bandwidth which is dependent on the data rate of the system. Increasing the data rate of a system will increase the bandwidth of the system. Theoretically, there is an infinite amount of spectrum that is available but that fact is very deceiving. Large parts of the spectrum are unsuitable for wireless communications. The radio spectrum starts around 10 kHz and goes up to about 100 GHz. The entire spectrum is controlled and allocated by the Federal Communications Commission. Another issue

to keep in mind is that the information to be transmitted has a finite duration, which means that it cannot be bandlimited. There are thousands of wireless applications that require varying amounts of bandwidth. Garage door openers, for one, require a few bits of information so they can have low bit rates and thus will require a small amount of bandwidth. Cellular phones need to send voice data in real time so they have bit rate requirements in the 10kbps range. Newer cell phones now offer the ability to send pictures and video which greatly increases the bit rates. In wireless local area networks (WLAN) computers are networked together which requires data rates over 1Mbps. Each of these systems requires an increasing amount of bandwidth respectively. In order to accommodate all of these devices, use of the radio spectrum has to be extremely efficient. There are several techniques discussed here that try to efficiently use their allotted spectrum to accommodate the largest number of users with the highest signal quality.

Some types of wireless communications require only a simplex channel while others require a duplex channel. In a simplex channel, information is sent in one direction at a time from the base station to the mobile user or vice versa. Some examples of information traveling in one direction on a simplex channel are garage door openers or paging systems. An example of a bi-directional simplex channel is an intercom or a walkie-talkie. With a duplex channel, information needs to be transmitted in both directions simultaneously. There is a forward channel which transmits information from the base station to the mobile user and the reverse channel which transmits information in the other direction. Duplexing is necessary in applications such as cell phones and wireless networks. There are two different methods for creating a duplex channel from two simplex channels. In frequency division duplexing (FDD) the forward channel

occupies a certain frequency while the reverse channel occupies another frequency. Both the base station and the mobile have duplexers that break the signal apart. The other method of duplexing is time division duplexing (TDD). In TDD, the forward channel and reverse channel each have a time slot where the signal is transmitted. In a TDD scheme the forward and reverse channels are not actually duplexed, however if the time slots are small enough then there is no perceptible delay to the user. TDD is generally used for stationary systems due to the fact that motion will cause varying propagation delays and could possibly cause the signal to overlap [1]. With these two duplexing methods, several types of multiple access schemes are available.

1.1.1 Time Division Multiple Access (TDMA)

Multiple access techniques attempt to maximize the number of users given a certain amount of allocated bandwidth. In a TDMA scheme, each user utilizes the entire bandwidth. In order to maximize the number of users, each one gets a time slot in which to transmit their data. In figure 1.1, you can see how the users are delineated in time and use all of the allotted bandwidth.

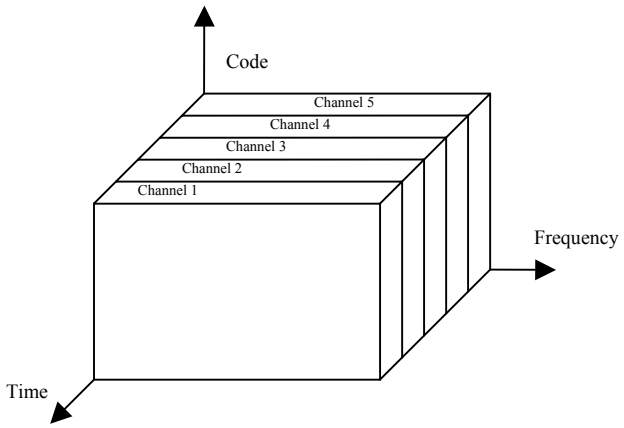

Figure 1.1– TDMA Scheme [1]

When implementing a TDMA scheme there are several points that need to be considered. There should be guard bands in between each channel to account for any propagation delays as the signal propagates. Another key issue is synchronization. The base station and the mobile both need to be able to keep track of time accurately. Having any discrepancy between the times of the base station and mobile will cause interference between channels. The size of each time slot is also limited by the data rate. More users can be accommodated with higher data rate and which user smaller time slots. However, the data rate is limited by the allotted bandwidth, so there are only a finite number of channels that can be created.

A TDMA scheme can be implemented with either TDD or FDD. In the TDD case, each user will have two different time slots. One time slot will be used for the forward channel and the other will be used for the reverse channel. For the FDD case, the user will get one time slot, but within that time slot the user will use one frequency to transmit over the forward channel and another frequency to transmit over the reverse channel.

1.1.2 Frequency Division Multiple Access (FDMA)

In an FDMA scheme, each user is assigned a frequency band and can transmit and receive a signal at any point in time. There still needs to be guard bands between each of the frequency bands in order to minimize the amount of interference between adjacent bands. Figure 1.2 shows an FDMA scheme on a time vs. frequency axis.

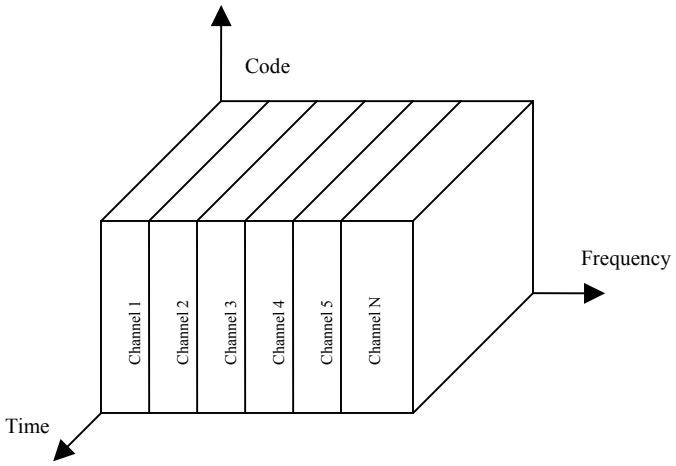

Figure 1.2 – FDMA Scheme [1]

When assigning frequency bands, the spacing between bands also has to take into account the fact that ideal filters are unattainable so the signals will have spectral content outside of the intended range. When the FDMA scheme is used with FDD each user has two different frequency bands, one for the forward channel and one for the reverse channel. Most applications assign forward and reverse channels at opposite ends of the available spectrum to avoid interference between channels. When a TDD method is used, each user will have one frequency band to operate with and two different time slots for the forward and reverse channels.

1.1.3 Code Division Multiple Access (CDMA)

There are several communications schemes that can fall under the heading of CDMA. The basic idea involves assigning each mobile a unique code. Each of these codes is mutually orthogonal to each other. Figure 1.3 shows the basic idea behind a CDMA scheme.

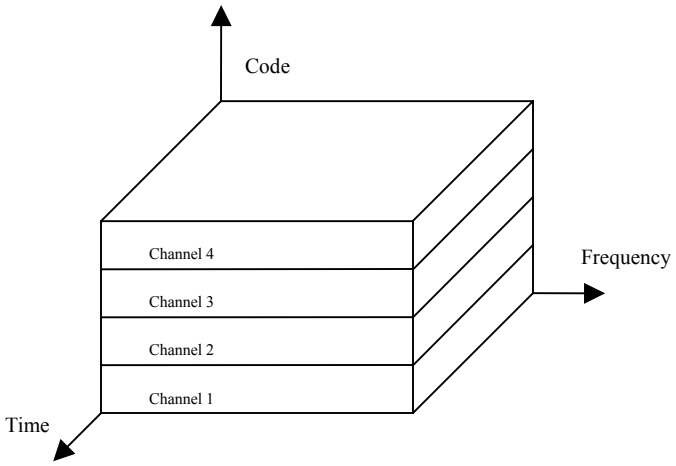

Figure 1.3 – CDMA scheme [1]

In this case, each user is allotted the entire frequency band and can transmit at any time. There are other CDMA schemes that use the system resources to offer more channels. One such scheme, depicted in Figure 1.4, shows a type of CDMA scheme called frequency hopping (FH-CDMA).

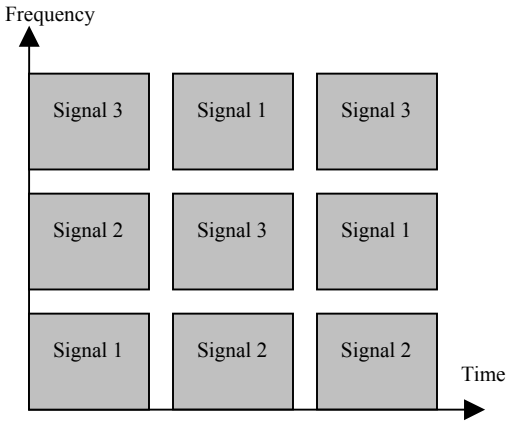

Figure 1.4 –FH- CDMA scheme [Sklar]

The frequency hopping CDMA scheme falls under the category of spread spectrum techniques. Each user has a pseudo-random (PN) code that is used to modulate the signal to be transmitted. As a result, each user will be transmitting at a different frequency, which makes this similar to FDMA. Over time, the PN code changes so that for each

time slot, the user will be transmitting over a different frequency which is similar to TDMA.

There are four main advantages to using a CDMA scheme [2]. When signals are transmitted through a random media there can be an excessive amount of mulitpath propagation. Small scale fading is due to this effect. One type of fading channel is a frequency selective channel which causes certain frequencies to be attenuated more than others. Due to this fact, received signal power can vary greatly within this type of channel. In an FDMA scheme, if a user is unlucky they may be assigned a frequency band that the channel constantly attenuates to a lower than acceptable level which will greatly reduce their signal quality. In a FH-CDMA scheme, a user which operates in a poor frequency band will only operate in that band for a short period of time. As a result, CDMA schemes can help counteract fading channels. Another advantage to a CDMA code is the privacy that it can afford a user. When a user has a constant frequency band, any receiver can pick up the same signal that the user is transmitting and receiving. Using a CDMA scheme will allow the user to switch frequency bands in a random manner making it extremely hard to intercept that signal. Since the user is not constantly operating at the same frequency, it becomes harder to jam that signal with artificial noise at one specific frequency. Another advantage to a CDMA scheme is that the orthogonal codes are not perturbed by propagation delays, which removes the requirement of precise synchronization that TDMA schemes require.

There are trade-offs that occur when using each system. As a result, different types of wireless systems utilize various types of multiple access schemes. Figure 1.5 shows some of the applications and the schemes that they use.

Wireless System	Multiple Access Scheme
Advanced Mobile System (AMPS)	FDMA/FDD
Global System for Mobile (GSM)	TDMA/FDD
US Digital Cellular (USDC)	TDMA/FDD
Pacific Digital Cellular (PDC)	TDMA/FDD
Cordless Telephone (CT2)	FDMA/TDD
Digital European Cordless Telephone (DECT)	FDMA/TDD
US Narrowband Spread Spectrum (IS-95)	CDMA/FDD
W-CDMA (3GPP)	CDMA/FDD CDMA/TDD
cdma2000 (3GPP2)	CDMA/FDD CDMA/TDD

Figure 1.5 – Multiple Access Schemes of Wireless Systems [1]

As can be seen from this table, certain situations determine which aspects of a multiple access scheme are more important to the design. There is not one scheme that is best suited for all scenarios.

1.1.4 "Spotlight" Multiple Access (SMA) vs. Space Division Multiple Access (SDMA)

In most scenarios, the base stations in wireless transmitters broadcast a signal isotropically. In an SDMA scheme, the antenna has directivity; it can direct the signal that it wants to transmit in any direction. According to the reciprocity theory of antennas, they also receive signals with the same pattern with which they transmit. This allows a frequency band to be used by several users if they are located at different azimuths from the base station. SMA can be considered a specific type of SDMA. SMA theoretically has the ability to locate the users not only by their azimuth, but by their range as well. Each user will have a unique code that identifies that user. Periodically, the user will transmit the full signal constellation representing all of the possible bits that can be sent. That signal will propagate though a channel unique to that user and be received by an

array of receivers at the base station. That signal will be stored at the base station as the prototype signal constellation for that user at that specific location. When information is to be transmitted over the forward channel, instead of using the original signal constellation, the base station will transmit a time reversed version of the prototype signal it received from the user. These time reversed signals will propagate back through the same channel they were received and have a strong enough power at the receiver to be detectable. At other locations, the multipath propagations will add destructively causing the signal power to be too small for detection. This can be viewed as a spotlight shining on the location of the user, while other locations will be dark. The area that the spotlight covers will depend of the size of the antenna array as well as the carrier frequency of the transmitted signal. As a result, each user can use the same frequency and time slot to communicate since the signal will only be detectable at their location [3].

1.2 Multipath Propagation

When an electromagnetic wave is propagating, it will interact with the surrounding environment. Some objects will absorb the waves, some will let the waves pass through, and others will reflect the waves in different directions. In environments with a lot of reflecting objects there will be several versions of a signal that will be incident on a receiver. Each version will have a different phase due to the propagation delay of the wave traveling on its specific path. There can also be small scale fading due to destructive interference between various versions of the received wave. These effects can be further exacerbated by motion of the receiver which will cause a Doppler shift in the received wave. The Doppler shift can be calculated from equation 1.1. In this

equation, v is the velocity of the receiver, λ is the wavelength of the transmitted signal, and θ is the angle between the direction of motion of the receiver and a direct line to the transmitter.

$$f_d = \frac{v}{\lambda}\cos\theta \tag{1.1}$$

A channel can be described as fast when the rate of change of the channel due to f_d is greater than the symbol rate of the transmitted signal. Otherwise, the channel is considered slow. Another parameter that is important when characterizing the propagation channel is the *rms* delay spread, σ_τ. The *rms* delay spread is the standard deviation of the power delay profile for a channel given a certain noise threshold. The power delay profile is a spatial or temporal average of the impulse response received at a specific location [1]. From σ_τ, the coherence bandwidth of a channel can be determined. It is proportional to the inverse of the *rms* delay spread. The coherence bandwidth determines the range of frequencies over which the channel will have a flat frequency response. From these two parameters, there are four possible channel types, flat slow fading, flat fast fading, frequency selective slow fading, and frequency selective fast fading.

In order to characterize a multipath channel statistically, there are two different cases to consider. When there is a line of sight path between the transmitter and receiver then there will be one dominant signal received and the power will be distributed according to a Ricean distribution. When there is not a line of sight path then there will not be a dominant signal at the receiver, and the signal power will be distributed according to a Rayleigh distribution. Figure 1.6 shows a Rayleigh distribution and several Ricean distributions. They come from equations 1.2 and 1.3 respectively.

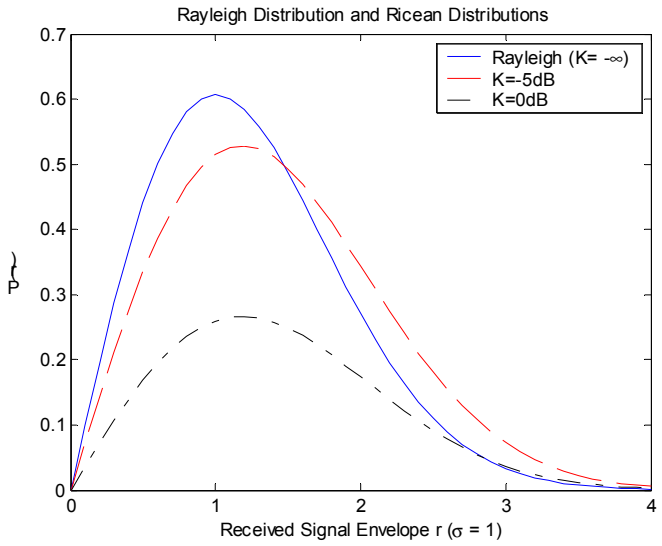

Figure 1.6 - Rayleigh Distribution and Ricean Distributions

$$P(r)_{Rayleigh} = \frac{r}{\sigma^2}\exp\left(-\frac{r^2}{2\sigma^2}\right) \tag{1.2}$$

$$P(r)_{Ricean} = \frac{r}{\sigma^2}\exp\left(-\frac{r^2+A^2}{2\sigma^2}\right)I_0\left(\frac{Ar}{\sigma^2}\right) \tag{1.3}$$

In equation 1.3, A represents the amplitude of the dominant received signal and I_0 is the modified Bessel function of the first kind. The parameter K in figure 1.6 is found according to equation 1.4 and represents the SNR of the dominant received signal in dB with respect to the multipath noise.

$$K(dB) = 10\log_{10}\left(\frac{A^2}{2\sigma^2}\right) \tag{1.4}$$

From figure 1.6 it can be seen that as K goes to -∞ the Ricean distribution becomes the Rayleigh Distribution. This makes sense since a small value of K corresponds to an absence of a direct line of sight path.

1.2.1 Overcoming Multipath Effects

Current Wireless Systems need a method of overcoming the various effects of multipath propagation. There are several methods that attempt to achieve this task such as equalization, diversity, and channel coding [1]. All methods attempt to utilize the statistical nature of the channels.

Channel equalization is an attempt to characterize the channel using adaptive signal processing. Its main objective is to eliminate inter-symbol interference (ISI). ISI occurs when the coherence bandwidth of the channel is smaller than the bandwidth of the transmitted signal, causing bit errors in the detected signal. A channel equalizer attempts to model the channel as a filter and estimate the filter coefficients. This process is performed frequently due to the fact that the channel can vary greatly in short periods of time. In order to estimate the taps a training signal is used. This training signal is transmitted to the receiver so that the receiver has a reference signal with which to test the channel. A recursive algorithm is used to calculate the coefficients that will help mitigate the distortion created by the multipath effects. Equalization methods require an increase in the number of bits transmitted due to the training set, as well as extra processing time due to calculating the filter coefficients at certain intervals. The inverse of the channel filter can then be applied to the signal in order to counteract the effects of the channel. Equalization techniques are most effective in TDMA schemes were the data is already broken into fixed length segments which facilitates the insertion of a training signal.

Using diversity to combat multipath effects takes advantage of the statistical nature of multipath propagation. Since multipath effects fluctuate rapidly within a short

distance, receivers that are relatively close to each other can receive very different signals. While one receiver is experiencing extreme fading, another receiver that is nearby might be receiving a strong signal. A method of implementing diversity is using a RAKE receiver in a CDMA scheme. The block diagram for a RAKE receiver with three branches is shown in figure 1.7. In the RAKE receiver, the received signal is delayed by the chip rate twice to create three different versions of the received signal.

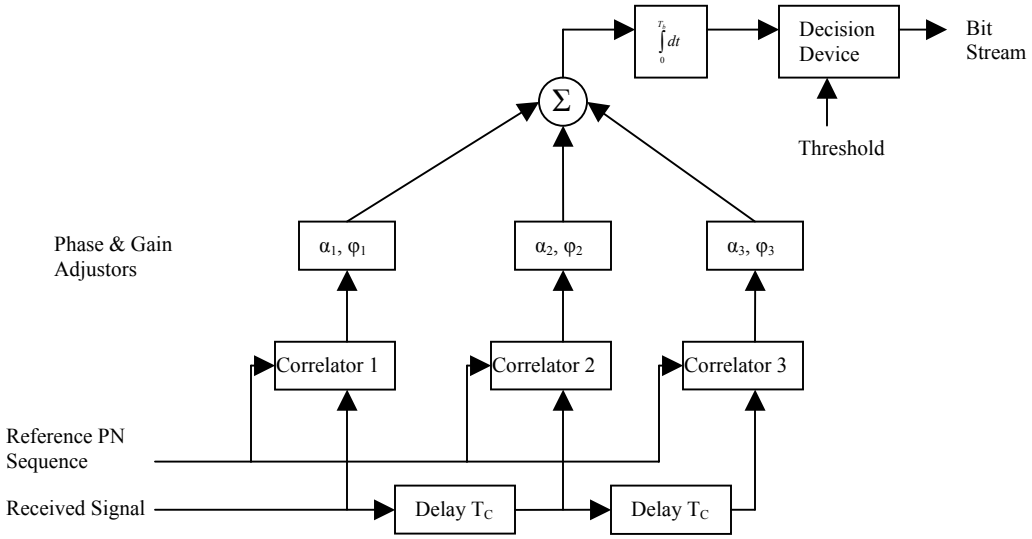

Figure 1.7 – RAKE Receiver [4]

In the case when the multipath components are delayed by more than a chip rate, the signals will be uncorrelated with each other [1]. Instead of having spatial diversity as described above you have temporal diversity. Each of the delayed signals is correlated with a reference sequence and weighted. Signals that experience large fading will be decreased with a smaller weight while strong signals will be boosted. This allows a CDMA receiver to achieve diversity in detecting a signal and improve its performance.

The third method of improving signal detection in the presence of multipath effects is the use of coding. There are several available coding schemes which all

introduce redundancies in the data so that the receiver has multiple versions of the data. These multiple version can be compared with each other to obtain the optimal detected signal. One type of coding is block coding. Block codes group the data into packets and add parity bits to create codewords. Block codes reduce the data rate because they insert extra bits that do not carry any extra information. Some examples of block codes are Hamming codes, Hadamard codes, Golay codes, cyclic codes, BCH codes, and Reed-Solomon codes. Each code has advantages and disadvantages. A convolutional code is another type of code that continuously encodes data instead of breaking it up into blocks. Another type of code is turbo code which rearranges data in a different order and transmits the ordered and jumbled versions together with parity check bits. At the receiver, the transmission is broken down and two different versions of the data are available. The two versions can be compared and an optimal decoded stream can be determined. The basic idea is that each bit in the jumbled version will experience different multipath effects so that each bit has a greater chance of avoiding deep fades.

1.2.2 – Using Multipath Propagation to Our Advantage

All of the previous methods attempt to model the channel and counteract it or introduce redundancy to avoid the effects of multipath propagation. With time reversal methods, the multipath channel can be used as an advantage. A signal containing the full signal constellation can be transmitted by the receiver to the base station. The base station records that signal at an antenna array. After those signals are time reversed they are saved as prototype signals in a new constellation as previously mentioned. When data is to be transmitted to the user, the new prototype constellation can used. The new

signal will travel back through the same channel as it was received. The multipath effects that affected the signal in the forward path will have an inverse effect in the reverse path. As a result, there is destructive interference in most locations while the receiver will have a strong signal due to constructive interference. As long as the prototype signals are updated faster than the channel the time reversal will act as a matched filter and allow the signal to be detected by the receiver.

1.3 Applications of Spatial Focusing

Achieving spatial focusing can have many advantages. By using a time reversal method, many key design criteria can be achieved. Signals can be transmitted with greater security, receiver complexity can be reduced, multiple users can be accommodated, and energy can be saved.

1.3.1 Security

Many wireless users desire to have covert communications. This idea causes problems because the information is being broadcast through the air. Anyone can theoretically receive a broadcast without knowing the location of the receiver. With wired transmission, anyone wishing to steal information would need to be able to tap into the connection between the transmitter and receiver. With ordinary wireless systems, each user's signal is broadcast in all directions. Knowing the user's frequency band, time slot, or code will allow someone to decode the information intended for that user. Using time reversal methods, the power of the transmitted signal will be extremely low in most locations. In order to have enough signal power to decode the signal, the receiver would

need to be relatively close to the original user. This provides a great deal of privacy and security to mobile users.

1.3.2 Multiple Access

Other multiple access schemes divide users in the time domain, frequency domain, or code domain. Using time reversal methods, users can further be divided spatially. An antenna array at the base station can store different prototype signals for each user. That would enable the base station to transmit signals at the same frequency during the same time period to different users simply by using the different prototypes. This will allow a large number of users to be accommodated while still offering very high bit rates. High bit rates will help to allow such 3G applications such as streaming video.

1.3.3 – Reduced Receiver Complexity

In order to reduce the small scale fading due to multipath effects, a considerable amount of overhead is required. Using equalization techniques requires the processing power to estimate channel parameters. Diversity techniques require the processing power to implement structures such as the RAKE receiver. Using codes, the receiver will require the processing power to decode the codes. Using a time reversal method will eliminate the need for all of this processing power. All of the processing just described takes place on the receiver side and acts on the signal after it is received. A time reversal technique transfers the processing requirements to the base station. As a result the receiver will detect a signal that has already been corrected for the channel. Moving the processing to the base station involves a one time cost, while excess processing power

required at the receiver is paid for each time a new receiver is built. Reducing the processing required at the receiver will also reduce its energy requirements. Reduced energy requirements will lead to longer battery lives, which is a great advantage.

2. Background

Time reversal theory was first developed by M. Fink in the mid 1990s [5]. Since then, the theory has been applied to help solve several different engineering problems including underwater communications with acoustic waves and ground penetrating radar. Time reversal theory allows engineers to characterize inhomogeneous media that increase the complexity of many problems. In an underwater environment, the speed of sound can fluctuate greatly based on different topographical features. The surface of the water and the seabed can also have varying effects on the propagation of acoustic waves. As a result, significant processing is usually needed to model the channel and counteract its effects.

In ground penetrating radar, the objective is to get a clear picture of objects that are underneath the surface of the earth. The goal might be to locate landmines that have been left behind after a battle has taken place. In order to locate the objects, the ground around it has to be modeled correctly. Certain areas can have very inhomogeneous soil that varies greatly in its electromagnetic properties. These properties make it difficult to locate the desired objects. Time reversal methods can be implemented to help get a clearer picture of what is underneath the surface.

2.1 Time Reversal Theory

Time reversal theory can be thought of as applying a matched filter to counteract the effects of your channel. Acoustic waves and electromagnetic waves both satisfy Helmholtz's differential wave equation, which is shown as equation 2.1.

$$\nabla^2 G(r,r') + k(r)^2 \nabla G(r,r') = -\delta(r-r') \tag{2.1}$$

In equation 2.1, **k** represents the wavenumber, which is a function of the frequency of the wave as well as the speed the wave travels in the medium. The function **G** must conform to the boundary conditions of the problem. Solving partial differential equations can be quite difficult.

Instead of analyzing signal transmission using electromagnetic theory, matched filter ideas can show the same results. The receiver will transmit a signal *x(t)* in the forward direction. As the wave travels through an environment, only its magnitude and phase will change assuming that the environment is motionless and does not alter frequencies. The effects of the channel can be modeled as a filter with impulse response *h(t)*. The received signal at an array element can be shown in equation (2.2) where *i* represents array element and ranges from 0…..N-1 where N is the number of array elements.

$$r_i(t) = x(t) \otimes h_i(t) \tag{2.2}$$

Equation (2.2) can be converted into the frequency domain where convolution becomes multiplication. This is show in equation (2.3).

$$R_i(f) = X(f) H_i(f) \tag{2.3}$$

Time reversal of the time domain signal will result in $r_i(-t)$ or $R_i^*(f)$ in the frequency domain, where * represents the complex conjugate. Assuming that the channel has the

same response *h(t)* in the reverse direction, transmitting *r(t)* back through the channel will result in equation (2.4) in the frequency domain.

$$Y(f) = R_i^*(f)H_i(f) \tag{2.4}$$

Substituting equation (2.3) into (2.4) yields:

$$Y(f) = X^*(f)H_i^*(f)H_i(f) \tag{2.5}$$

In equation (2.5), since the channel was assumed to behave as a linear filter, multiplication the filter by its complex conjugate will result in the magnitude squared. If the frequency response of the channel is flat over the desired range of frequencies, the output will be:

$$Y(f) = AX^*(f) \tag{2.6}$$

In equation (2.6) A represents a scaling factor that takes into account the gain of the filter as well as the losses associated with propagation over a certain distance. Time reversing the received signal Y(f) and then converting back into the time domain will result in the same signal that was transmitted, x(t).

2.2 TRM for Underwater Communications

Underwater communications using acoustic waves has had trouble achieving high data rates due to the time varying nature of the dispersive multipath environments [6], [7]. Previously, adaptive channel equalization has been implemented in order to reduce the ISI present in underwater communications. Time reversal has two effects; first temporal compression that reduces dispersion caused by the channel, second the spatial focusing that mitigates the effect of fading [8]. These characteristics also eliminate the need for diversity techniques such as multiple receiving antennas. In July of 1999,

Edelman conducted underwater testing of a time reversal mirror for communications off the west coast of Italy. The experiment was conducted at 3.5 kHz using binary phase shift keying (BPSK) in three different underwater environments: an absorptive bottom, a reflective bottom, and a sloping bottom [8].

One experiment performed involved lowering a receiver array into the water to detect the received signal at locations other than the intended spot. Two other methods of underwater communication were used: one way single source communications, and one way broadside communications. The receiver transmitted a short pulse that was recorded at the mirror. The signal was time reversed and then propagated back through the same channel originally traveled. The experiment transmitted 50 bits to the receiving array and decoded them. Not only was the TRM the only method to decode all 50 bits correctly at the intended location, the TRM also caused more detection errors at other locations. This result shows that the TRM can provide an extent of security that the other methods could not provide. Edelman also showed that the time reversal technique can provide an adequate matched filter to the channel impulse response which will mitigate the effects of ISI. The TRM also allowed signal energy to be focused in both range and depth [8].

2.3 TRM for Radar Image Processing

Plumb and Leuschen applied a TRM to solve a remote sensing problem [9]. In ground penetrating radar, antennas transmit a pulse from the surface and then the signal is recorded either in the same location, monostatic case, or in a different location, bistatic case. The desired outcome of ground penetrating radar is to get an accurate picture of the object space. This requires a mapping, or migration, from the image space in which the

signals are recorded to the desired object space. Plumb and Leuschen proposed using time reversal as a matched filter to model the dielectric properties of the ground [9].

The theory was tested using a large sandbox with submerged PVC pipes. A radar system was placed on the surface of the box and the signal reflections were recorded in both monostatic and bistatic cases. The recorded signals were then fed into a Finite Difference Time Domain (FDTD) algorithm in order to virtually back propagate the waves into the sandbox. The results presented show that the PVC pipes can clearly be detected [9]. The pipes are detected due to the fact that the TRM refocused the energy that was reflected by the pipes back to its original location. This experiment shows the flexibility of time reversal theory. It also expands the theory beyond acoustic waves to waves at frequencies close to 1 GHz. It provides insight into a method for simulating time reversal communications. An FDTD algorithm can easily be adapted to receive signals at many different locations as well as reintroducing them in different locations.

2.4 Array Spacing, Shape and Size

The effectiveness of an array in focusing energy on the receiver will depend on the shape of the array, the size of the array, and the spacing of the elements in the array.

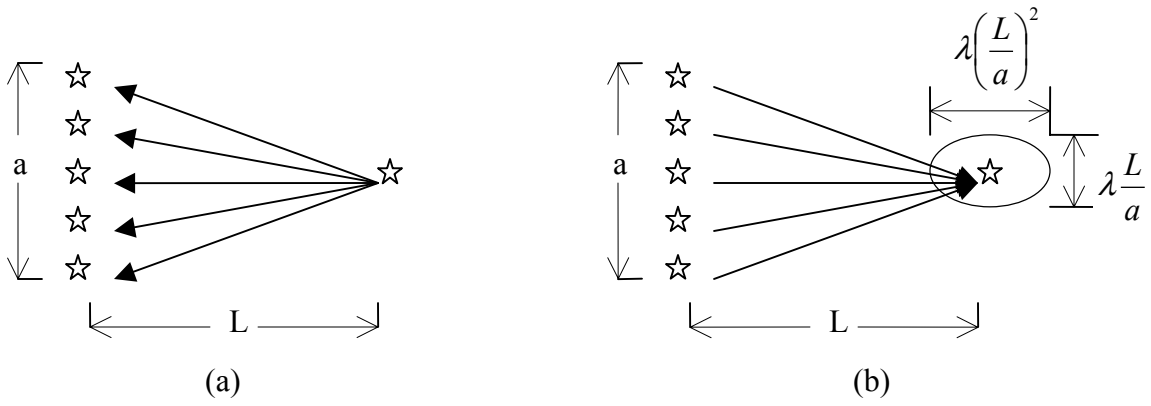

Figure 2.1- (a) Forward Propagation (b) Reverse Propagation with Focusing [12]

Figure 2.1(a) shows the forward propagation of a signal from a receiver to an array. Figure 2.1(b) shows the time reversed signal after reverse propagation back through the same environment. The time reversed signal will focus energy around the receiver in an ellipse that is aligned along the range axis. The major axis will have an approximate length of $\lambda\left(\frac{L}{a}\right)^2$ and the minor axis will have an approximate length of $\lambda\frac{L}{a}$ [10]. In the equations, λ is the wavelength of the signal that is transmitted back and forth. It was shown by Steinberg that when the array spacing is λ/2, the antenna array will act as array aperture with a size of $(N-1)\frac{\lambda}{2}$ where N is the number of array elements [11]. That particular spacing will ensure that each element will not act like a separate entity as well as minimize the interference between the elements [11].

2.5 Inhomogeneous Media

In most cases, inhomogeneous environments cause problems for communication systems. Replicas of the signal are incident on the receiver due to reflections, resulting in small scale fading. This problem is compounded when the environment or the user is in motion which results in Doppler shifts. In the case of time reversal methods, an inhomogeneous media will actually improve the accuracy of the communication. Random reflectors throughout the environment will focus more energy onto the antenna array. As a result, the effective size of the array, a_e will be greater than the original size of the array, a. Since $a_e > a$, the energy ellipse from figure 2.1(b) will tighten in both directions due to the fact that they are inversely proportional to the array size. In this case, the extra focusing power is called super-resolution [12].

2.6 Receiver Motion

In order for time reversal methods to be applicable to more wireless communications systems, it is important to account for the fact that the receiver may be in motion. Many new wireless systems transmit signals to and from receivers that are constantly in motion, and may be traveling at speeds varying from 1km/hr for a person walking to 100km/hr for a person driving in a car all the way up to 1000km/hr for someone flying in an airplane. Motion on the receivers end will result in a Doppler shift causing a change in frequency from the original signal. Systems that cannot account for this phenomenon are not suitable for mobile wireless communications.

In underwater communications, Gomes has shown that time reversal theory will still result in a focusing of the transmitted energy at the location of the receiver even if the receiver is in motion [13]. The multipath canceling properties of the array will be

maintained for digital communications purposes; the only effects being non-uniform rotation and magnitude fluctuations in the received signal constellation [13]. This phenomenon can be handled through the use of wavefront segmentation as shown in by Gomes [13]. This key result shows that motion of the receiver does not destroy the advantages of a time reversal mirror. This broadens the class of wireless systems for which the time reversal mirror can be utilized.

3. Digital Modulation Techniques

Transmitting data requires modulation. The information can be used to modify or modulate a signal so that a receiver can detect that alteration and convert the received signal, or demodulate it, and recover the original data. In digital modulation, instead of modulating a continuous time signal data bits such as 1's and 0's are modulated. Many different modulation schemes are available, all of which have various advantages and disadvantages. The important aspects of the modulation scheme are its bandwidth, its susceptibility to error, and its complexity on the receiving end. Each input data bit will be represented by a certain number of samples of a discrete signal. For the examples in this section there will be 100 samples per bit. This section describes three modulation schemes, amplitude shift keying, phase shift keying, and frequency shift keying. It also discusses the effects of a pulse shaping filter.

3.1 Amplitude Shift Keying (ASK)

In an ASK system, the amplitude of a signal that is transmitted will be modified according to the input bit. The transmitted signal can be represented by equation 3.1.

$$s[n] = g[n]\cos[2\pi f_c n + \theta] \tag{3.1}$$

In equation 3.1, g[n] will be a 1 whenever the input bit is a 1, and will have a value of -1 when the input bit is a zero. f_c is the carrier frequency of the transmitted signal, and θ is the phase. With an input sequence of [1 0 1 1], figure 3.1 shows the appropriate ASK signal with f_c = 900 MHz.

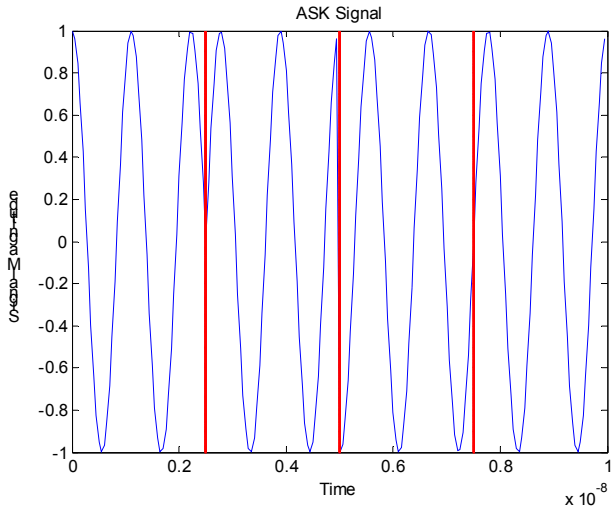

Figure 3.1 – ASK Modulated Signal

ASK modulation is not customarily implemented in this manner. It does however show up as a specific case of phase shift keying. As a result, its susceptibility to error and spectral bandwidth will be discussed in the next section.

3.2 Phase Shift Keying

When phase shift keying is implemented, the phase of the modulated signal is altered based on the data. In this section, binary phase shift keying (BPSK) will be discussed which has two different possible phase shifts of $\pm\pi$. In this case, the signal can be represented by equations (3.2a) and (3.2b) representing a 1 or a 0 respectively.

$$s[n] = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c n + \theta_c) \tag{3.2a}$$

$$s[n] = \sqrt{\frac{2E_b}{T_b}}\cos(2\pi f_c n + \theta_c + \pi) \tag{3.2b}$$

E_b represents the energy per bit and T_b represents the period of each bit [Rap]. θ_c represents the phase of the signal. An interesting note here is that the BPSK signal

constellation is the same as an ASK signal with two levels. In Equation (3.2b) the shift of π can be thought of as a multiplication by -1. Figure 3.2 shows a BPSK signal for the data [1 0 1 1]. The bandwidth of a BPSK signal will be $2R_b$, where R_b is the bit rate.

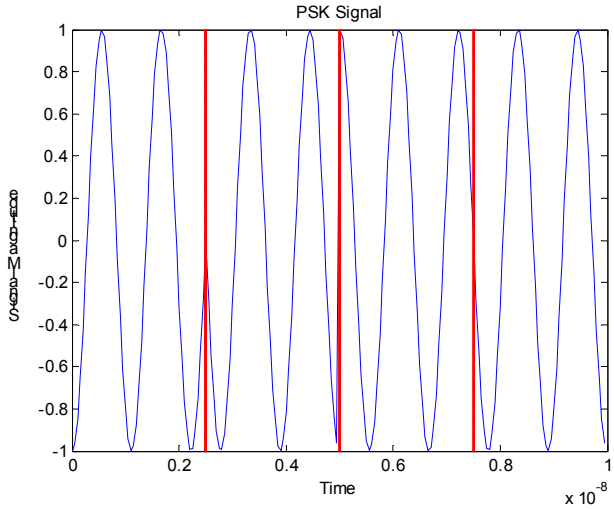

Figure 3.2 – PSK Modulated Signal

The probability of an error is given by equation (3.3), where Q(…) represents the Q-function.

$$P_e = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) \tag{3.3}$$

The ratio of E_b over N_0 represents the signal to noise ratio of the signal [1]. Equation (3.3) holds under the case when the noise is additive, white, and Gaussian (AWGN). A BPSK signal needs to be demodulated using a synchronous receiver. This means that a local carrier needs to be generated with the correct frequency and phase. Depending on the presence of a pilot signal, the local carrier can be determined using a phase locked loop (PLL), a Costas Loop or a squaring loop [1].

3.3 Frequency Shift Keying (FSK)

In an FSK modulation scheme the frequency of the transmitted signal is varied according to the input bit that needs to be sent. In the case of binary FSK equations (3.4a) and (3.4b) show the waveforms for a 1 or a 0, respectively [1].

$$s[n] = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c n + 2\pi \Delta f n + \theta_c) \tag{3.4a}$$

$$s[n] = \sqrt{\frac{2E_b}{T_b}} \cos(2\pi f_c n - 2\pi \Delta f n + \theta_c) \tag{3.4b}$$

In equations (3.4a) and (3.4b) Δf is the change in frequency from the carrier frequency, f_c. Figure 3.3 shows a plot of the data [1 0 1 1]. The portion of the signal that carries a 1 has a higher frequency than the portion that contains a 0.

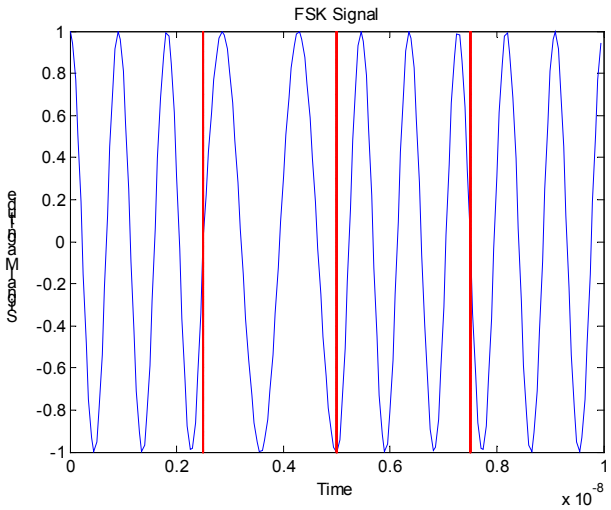

Figure 3.3 – FSK Modulated Signal

The bandwidth of an FSK signal, determined by Carson's rule, is equal to $2(\Delta f + R_b)$. Unlike PSK, FSK signals can be detected without a coherent carrier.

$$P_e^{Coherent} = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.5}$$

$$P_e^{Non-Coherent} = \frac{1}{2}\exp\left(-\frac{E_b}{2N_0}\right) \tag{3.6}$$

When coherent detection is used the probability of detecting an error is given by equation (3.5), however when non-coherent detection is used the probability of detecting an error is given by equation (3.6). The error under coherent detection is lower than the error under non-coherent detection; however this tradeoff comes at the cost of higher receiver complexity.

3.4 Pulse Shaping Filter

When square wave signals are transmitted, they will begin to spread due to the limited bandwidth of the channel as well as varying time delays. As a result, symbols will spread and overlap one another causing Inter Symbol Interference (ISI). In order to counteract this effect a pulse shaping filter can be used to change the pulse from a square pulse to another shape that is more immune to channel effects. The main consideration is that the impulse response of the filter is zero at every sampling point other than the desired point.

$$h_{RC} = \frac{\sin\left(\frac{\pi t}{T_s}\right)}{\pi t} \cdot \frac{\cos\left(\frac{\pi \alpha t}{T_s}\right)}{1-\left(\frac{4\alpha t}{2T_s}\right)^2} \tag{3.7}$$

This criterion is known as the Nyquist condition. One type of filter that satisfies the Nyquist condition is the raised cosine filter with an impulse response given in equation (3.7) and shown in figure 3.4. The Frequency Response is given by equation (3.8) and shown in figure 3.5.

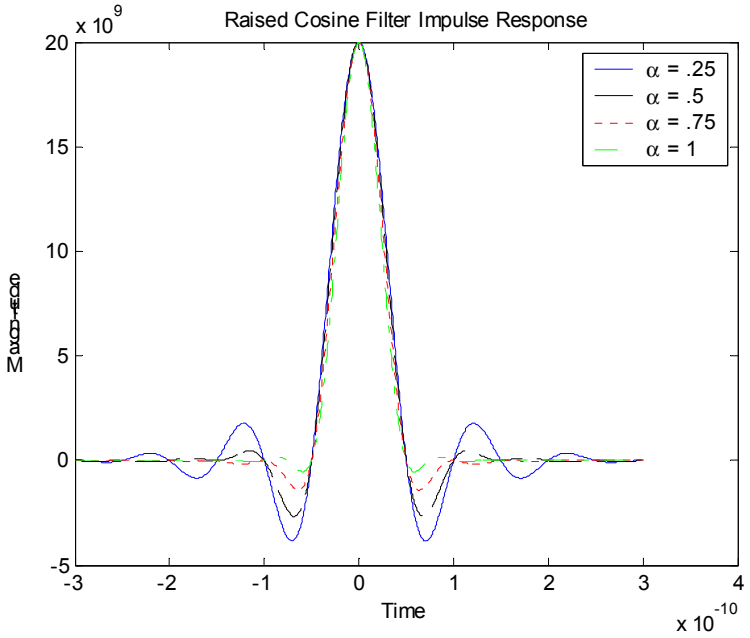

Figure 3.4 – Root Raised Cosine Filter Impulse Response

$$H_{RC} = \begin{cases} 1 & 0 \le |f| \le \dfrac{(1-\alpha)}{2T_s} \\ \dfrac{1}{2}\left[1+\cos\left[\dfrac{\pi\left(|f|\cdot 2T_s - 1+\alpha\right)}{2\alpha}\right]\right] & \dfrac{(1-\alpha)}{2T_s} \le |f| \le \dfrac{(1+\alpha)}{2T_s} \\ 0 & |f| > \dfrac{(1+\alpha)}{2T_s} \end{cases} \tag{3.8}$$

The parameter α is a scaling factor that can be used to alter the raised cosine filter. When implementing a raised cosine filter the greater immunity to ISI comes at the price of increased signal bandwidth. When α = 0 the raised cosine filter becomes a square wave with high ISI but the lowest possible bandwidth. As α increases up to a value of 1, the ISI will decrease but the bandwidth will increase.

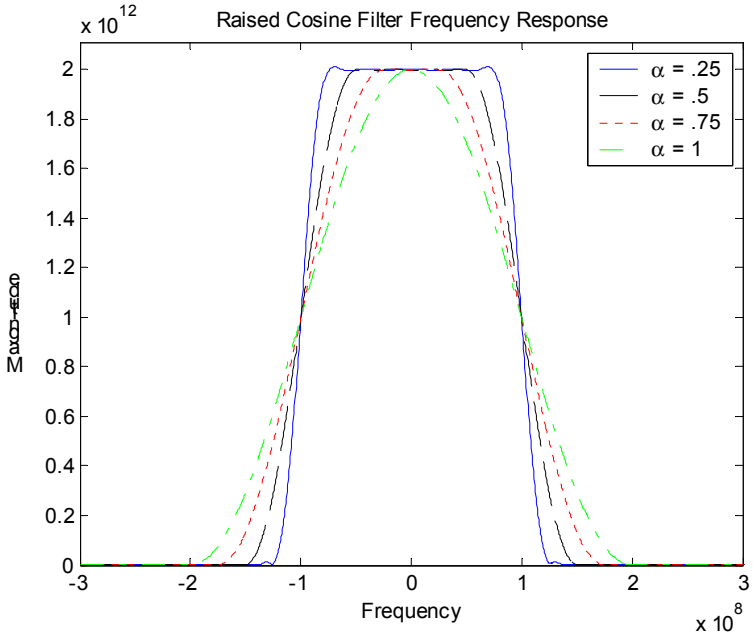

Figure 3.5 – Raised Cosine Filter Frequency Response

3.5 Modulation Trade-offs

Each modulation scheme has advantages and disadvantages. Only a few schemes were presented here to give the basic idea behind each modulation scheme. PSK was only shown in the binary case which had the advantages of a low bandwidth and low error rates, however it requires coherent detection witch necessitates a high level of receiver complexity. In order to improve the PSK scheme, the number of levels can be increased from 2 to 4 to create without increasing the bandwidth but with a higher error rate. PSK can also be encoded differentially which allows non-coherent detection but has the drawback of increased error. We also saw that FSK has a higher bandwidth with higher error rates, but allows the receiver to be less complex using non-coherent detection.

4. Ray Model Simulation

In order to test the ideas behind time reversal arrays and their effects on communications, simulations need to be employed. In order to begin testing the theory, electromagnetic propagation was modeled using ray propagation. According to this theory, the electromagnetic waves will travel in a straight line from the transmitter to the receiver. If there are reflector at some locations in the simulation, then the wave will travel in a straight line, or ray, to that reflector and then directly to the receiver. Theoretically, the reflected wave will propagate from the first reflector to every other reflector and then to the transmitter. This process will continue depending on the number of reflectors. Creating all of these rays can become exhausted so the process can be cut short after a certain number of reflections. This can provide an accurate model due to the fact that the reflector will absorb some energy from the signal and each successive reflection will have less of an effect on the total received signal.

4.1 Single Reflections

This test assumes that there are several point reflectors that will reflect the original signal and create multiple paths of propagation between the sender and receiver. For this simulation, the size of the test area is 100 x 100 units. The transmitter is located at (0,0) and the receiver is located at (0,100). The points are randomly spaced over the range x=[0,100] and y=[-50,50]. Figure 4.1 is a diagram that shows the layout of the simulation.

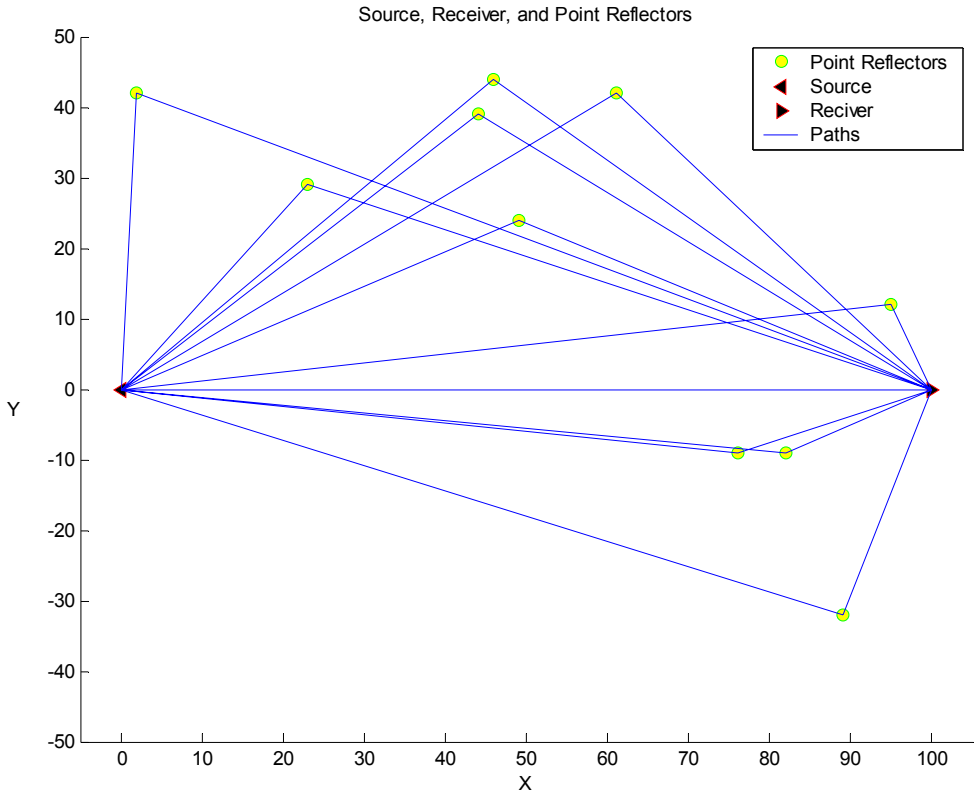

Figure 4.1 – Single Reflections Setup

Assuming that the transmitter broadcasts its signal in all directions, the point reflectors will reflect the signal in all directions. The lines in figure 4.1 show the multipath characteristic of the simulation. The test signal that will be sent is: $g(t) = \cos(2\pi f_c t)$ with f_c equal to 5 MHz.

The signal received at the receiver will be the sum of the direct signal plus all of the reflected signals. It is assumed that the magnitude will decrease proportionally to the inverse of the distance traveled. Each received signal will also have a phase shift. The total received signal will be:

$$r_{total} = \sum_{n=1}^{points+1} \frac{1}{4\pi d_n^{\,2}} g(t - \phi_n) \qquad (4.1)$$

In equation (4.1), d_n is the total distance traveled by each signal and ϕ_n is the phase shift for each signal. *Points* is the number of point reflectors in the simulation. When the original signal is transmitted, the phase is equal to:

$$\phi_n = \frac{2\pi \bmod(t,T)}{T} \tag{4.2}$$

where t is the time it took for the signal to travel from transmitter to receiver and T is the number of samples per period of the cosine wave. T for this simulation is equal to F_s/f_c, with F_s=500MHz, so T=100. Figure 4.2 shows two plots, the top plot shows each of the received signals, and the bottom plot shows r_{total}.

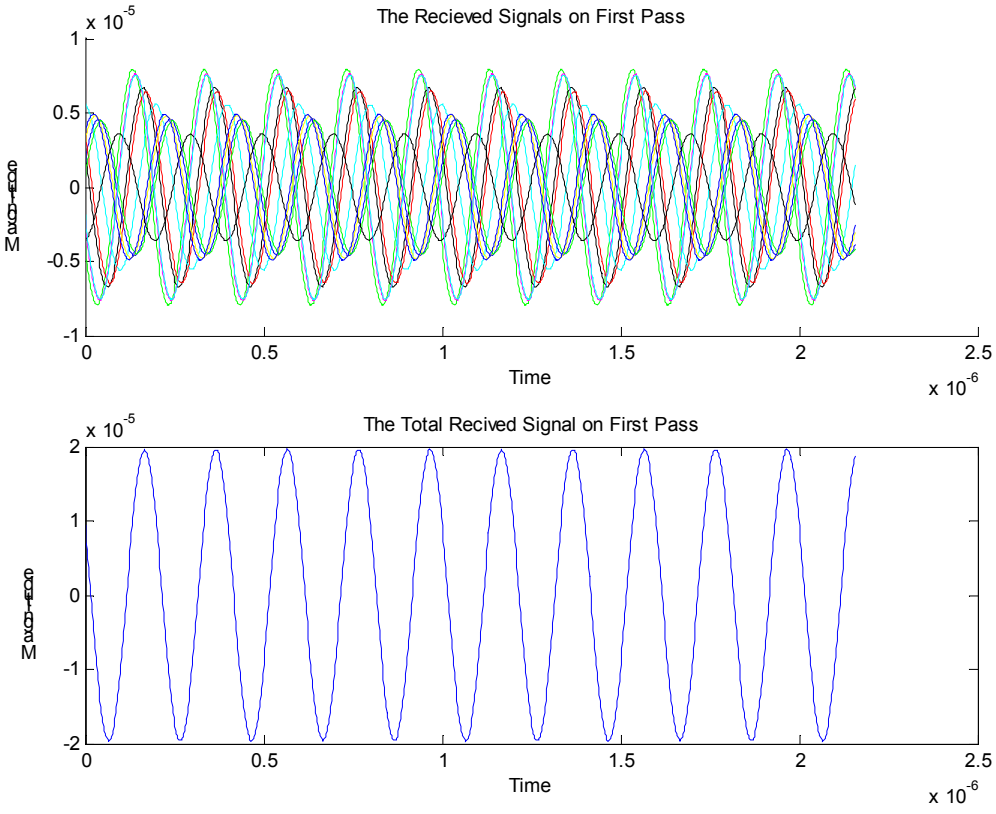

Figure 4.2 – Received Signal at the Array

At this point you can time reverse the total received signal r_{total}. That time reversed signal will then be propagated back from the receiver to the transmitter.

$$r_{total}^{TR} = \sum_{n=1}^{point+1} A(d_n) g(-t + \phi_n) \tag{4.3}$$

In this case, $A(d_n)$ is the magnitude as a function of distance traveled. The same effects will change the transmitted signal in this case, but the phase needs to be calculated differently. There is no longer a closed form equation for the time reversed signal so another method of phase shift needs to be employed. The time taken to travel back to the original transmitter will be the same as the time it took for the wave to travel its first path. Converting that time into a number of samples allows you to delay the received signal by the necessary phase. This method is only accurate to 3.6 degrees due to the 100 samples per period. Since the wave is traveling though the same environment, it will have the same phase shift as before, so the received signal is:

$$s(t) = \sum_{n=1}^{point+1} A'(d_n) g(-t + \phi_n - \phi_n) \tag{4.4}$$

It is clear that the phases will cancel each other out. $A'(d_n) = A(A(d_n))$. Figure 4.3 is a plot of each individual received signal and the sum $s(t)$.

After canceling out the phases, $g(t)$ is no longer dependent on n, so it can be brought outside the summation. The argument of the summation is just a number, so if you normalize it to 1, $s(t) = g(-t)$. Applying another time reversal to the signal $s(t)$ that is received back at the original transmitter, you should obtain the original sent signal. Figure 4.4 shows that result.

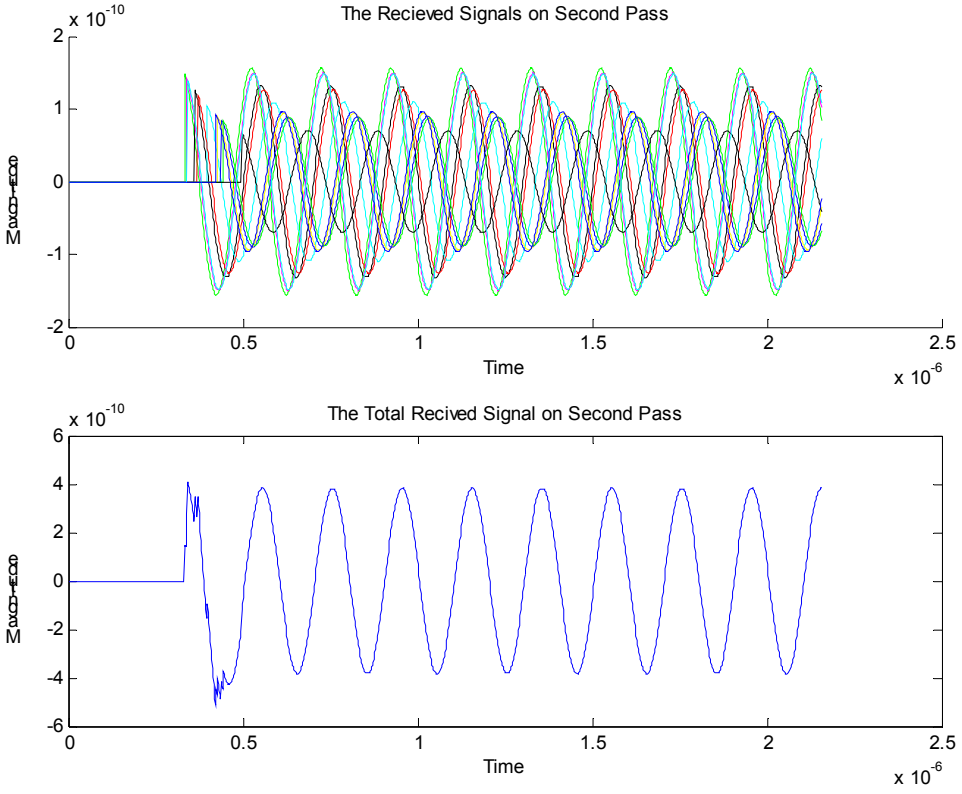

Figure 4.3 – Received Signal at Receiver

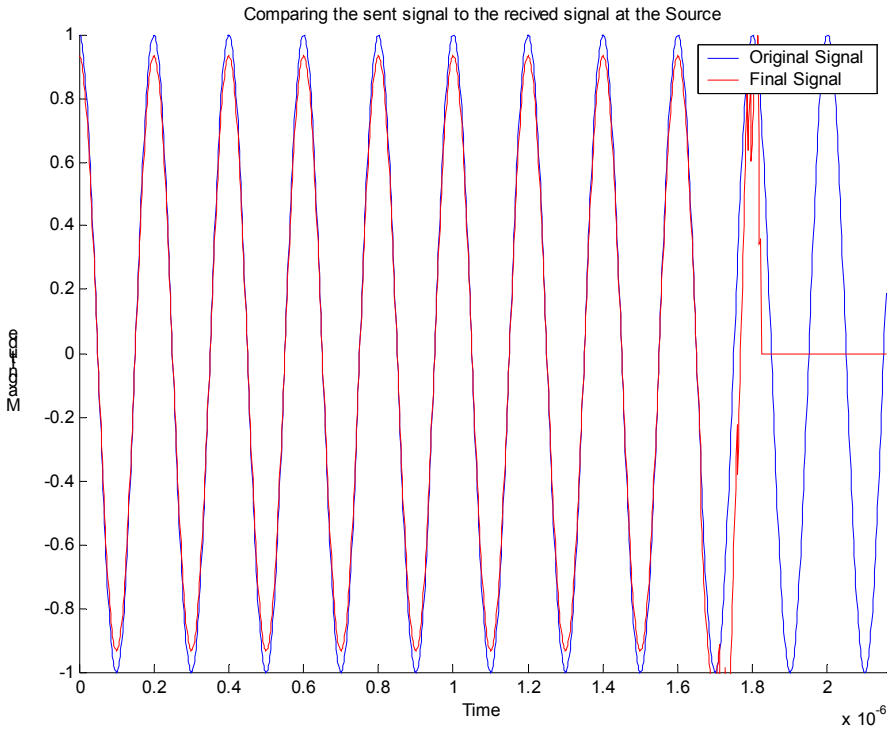

Figure 4.4 –Result from Single Reflections

The flat line at the end accounts for the delay in the propagation of the signal from the receiver back to the transmitter.

4.2 Double Reflections

At this point, the model can be improved by taking into account all of the instances in which the reflected signal will encounter another reflector on its path towards the array. A reflection that does not occur in the direction of propagation towards the array will not be counted. It is assumed that those reflections will travel much longer distances due to the fact that they travel in the wrong direction for part of their trip as well as the fact that the reflectors will most likely absorb more energy under those conditions.

The following figures show the results of the double reflections simulation.

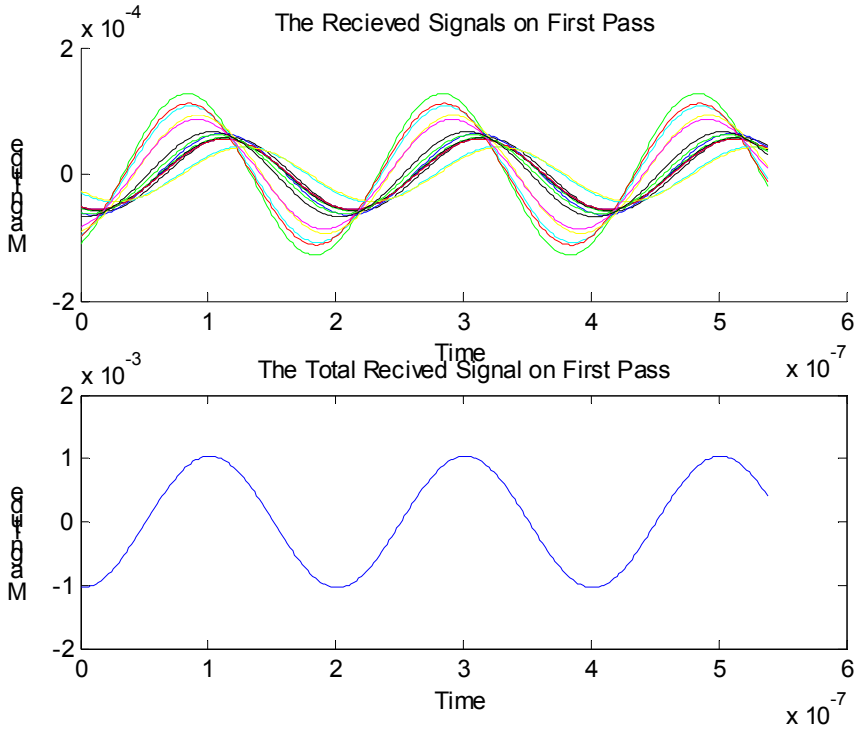

Figure 4.5– Double Reflections, Signal Received at the Array

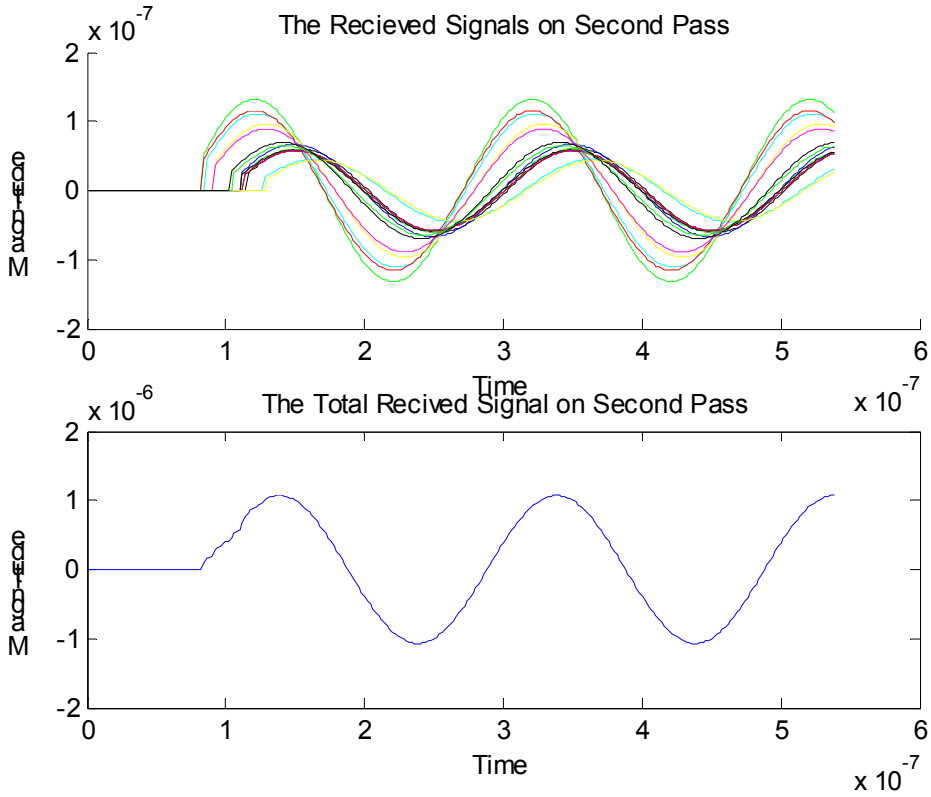

Figure 4.6 – Double Reflections, Signal Received at Transmitter

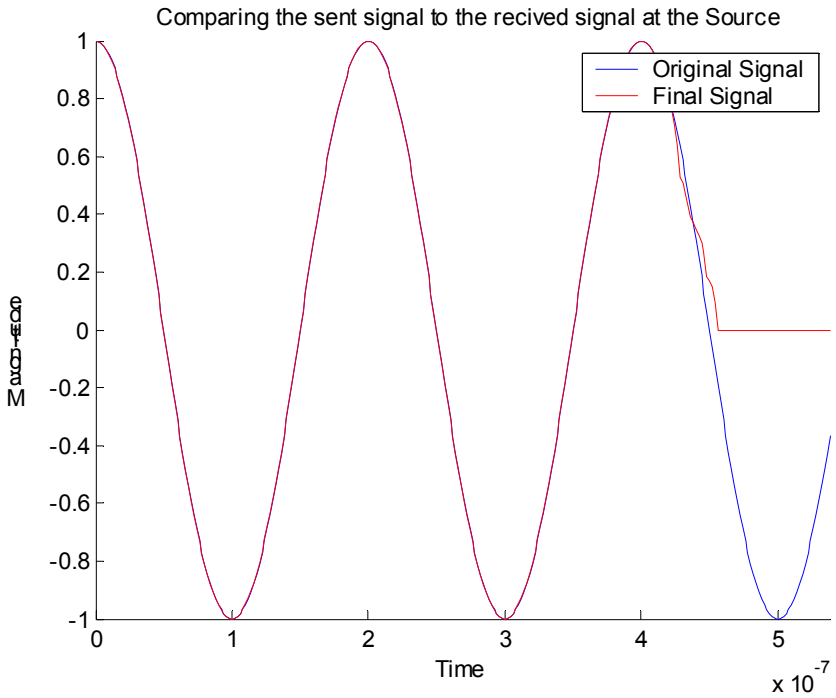

Figure 4.7 – Double Reflections, Comparing Transmitted and Received Signals

Figure 4.5 shows the signal that is received at the array after is has been reflected by 5 point reflectors accounting for double reflections. That total signal is time reversed and propagated back through the same environment with the 5 reflectors. Figure 4.6 shows the received signal at the original transmitter. Figure 4.7 shows the results of the experiment after the signal received at the transmitter is time reversed again. This figure shows that the transmitter is able to fully reconstruct the original signal.

4.3 Limitations of Ray Model

This simulation only shows the rudimentary workings of the time reversal method. It shows that the signal can be sent back and forth without any loss of information. The simulation has several drawbacks. The ray method does not accurately model the interaction between electromagnetic waves and objects. In this simulation the objects were approximated by perfectly reflecting points. A model that better captures these effects is necessary to get a better understanding of time reversal theory.

This simulation also neglects to consider different modulation schemes when transmitting data. For this experiment, a sinusoid was transmitted back and forth in order to achieve a certain level of understanding of the phenomena that are occurring. Future simulations need the ability to implement various modulation techniques.

This simulation also uses a single point to receive the signal. Time reversal theory requires the use of an antenna array to accurately locate and focus energy on the receiver. The current simulation assumes that the waves will travel directly from the source to the receiver or from the source to the reflector to the receiver. In actuality, the sources are transmitting spherical waves that are propagating in all directions. One of the

major points of time reversal theory is that it has the ability to focus energy. This simulation cannot measure the electromagnetic fields at points other then the source or the receiver. A new method of determining the value of the fields at every point in the simulation is needed. The next chapter will discuss the finite difference time domain method for electromagnetic simulations which can answer a lot of the questions raised by this simulation.

5. Electromagnetic Simulation

In order to fully explore the ideas behind time reversal methods, more elaborate models need to be used. The finite difference time domain method (FDTD) accurately models many more of the real phenomena that occur during electromagnetic propagation than a simple ray model. As a result, FDTD simulations will be used to continue to test time reversal methods for communications.

5.1 FDTD Method

The Finite Difference Time Domain method provides a technique for evaluating electromagnetic waves. The Electromagnetic model involves three equations. They are Maxwell's Equations, the Constitutive Parameters, and the Lorentz Force Equations. Maxwell's Equations and the Constitutive parameters are:

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \qquad \nabla \cdot \vec{D} = \rho \qquad \vec{D} = \varepsilon \vec{E}$$

$$\nabla \times \vec{H} = \frac{\partial \vec{D}}{\partial t} + \vec{J} \qquad \nabla \cdot \vec{B} = 0 \qquad \vec{B} = \mu \vec{H}$$

The total electric field can be defined as $\vec{E}^{total} \equiv \vec{E}^{incident} + \vec{E}^{scattered}$. This is done for several reasons. The first is that the incident field can be defined in the problem as it is known before hand. The second is that the scattered field will be observed, and also needs to be absorbed at the simulation boundaries [14]. This arrangement allows us to manipulate all the elements of the field that require alterations. The magnetic fields are

defined in the same manner $\overrightarrow{H}^{total} \equiv \overrightarrow{H}^{incident} + \overrightarrow{H}^{scattered}$. Both of these fields must satisfy Maxwell's Equations.

If you assume free space, then $\sigma = 0$, $\mu = \mu_0$, and $\varepsilon = \varepsilon_0$. Knowing that $\vec{J} = \varepsilon \vec{E}$ and solving these equations for $\vec{E}$ and $\overrightarrow{H}$, you get:

$$\frac{\partial \overrightarrow{H}^{scat}}{\partial t} = -\frac{1}{\mu}\left(\nabla \times \overrightarrow{E}^{scat}\right) \tag{5.1a}$$

$$\frac{\partial \overrightarrow{E}^{scat}}{\partial t} = \frac{1}{\varepsilon}\left(\nabla \times \overrightarrow{H}^{scat}\right) \tag{5.1b}$$

Now the equations are differenced. The finite differencing replaces the derivatives with differences [14].

$$\frac{\partial f}{\partial t} \equiv \lim_{\Delta t \to 0} \frac{f(x,t_2) - f(x,t_1)}{\Delta t} \approx \frac{f(x,t_2) - f(x,t_1)}{\Delta t} \tag{5.2a}$$

$$\frac{\partial f}{\partial x} \equiv \lim_{\Delta x \to 0} \frac{f(x_2,t) - f(x_1,t)}{\Delta x} \approx \frac{f(x_2,t) - f(x_1,t)}{\Delta x} \tag{5.2b}$$

In equations (5.2a) and (5.2b) Δt and Δx are finite rather than infinitesimal. Applying these approximations to Maxwell's equations, you get six different equations, 2 each in the x, y, and z directions [14]. Two of the equations are:

$$\frac{\overrightarrow{E}_x^{scat,n} - \overrightarrow{E}_x^{scat,n-1}}{\Delta t} = \frac{1}{\varepsilon_0}\left[\frac{\Delta \overrightarrow{H}_z^{scat,n-\frac{1}{2}}}{\Delta y} - \frac{\Delta \overrightarrow{H}_y^{scat,n-\frac{1}{2}}}{\Delta z}\right] \tag{5.3a}$$

$$\frac{\overrightarrow{H}_y^{scat,n+\frac{1}{2}} - \overrightarrow{H}_y^{scat,n-\frac{1}{2}}}{\Delta t} = \frac{1}{\mu_0}\left[\frac{\Delta \overrightarrow{E}_z^{scat,n}}{\Delta x} - \frac{\Delta \overrightarrow{E}_x^{scat,n}}{\Delta y}\right] \tag{5.3b}$$

5.1.1 Yee Cell

Equations (5.3a) and (5.3b) are applicable in three dimensions. In order to create a leapfrogging phenomenon, the electric fields and magnetic fields have to be staggered. In this manner, the electric fields will induce a magnetic field, then the magnetic field will induce another electric field and the cycle continues. This type of organization was first visualized by Kane Yee in 1966 in *IEEE Transactions on Antennas and Propagation* [15]. The following is a diagram of his cell:

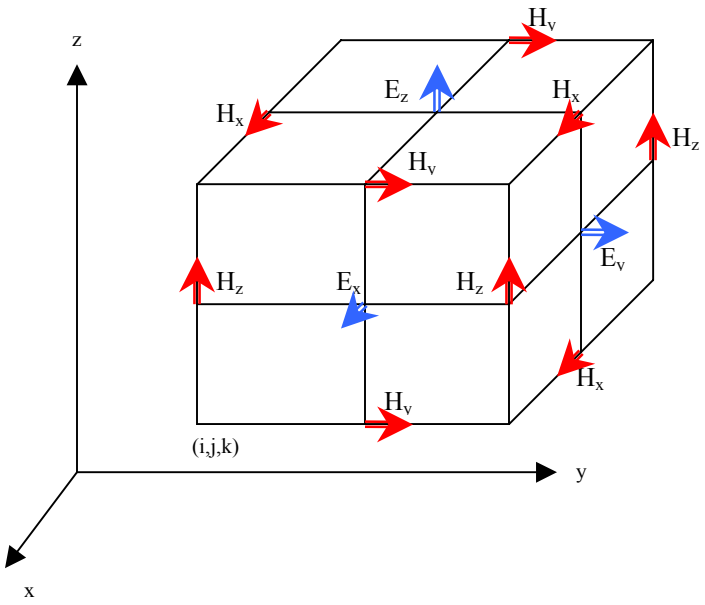

Figure 5.1 – Yee Cell [15]

This allows for the proper leapfrogging of the electric and magnetic fields. This can also be seen in the following diagram.

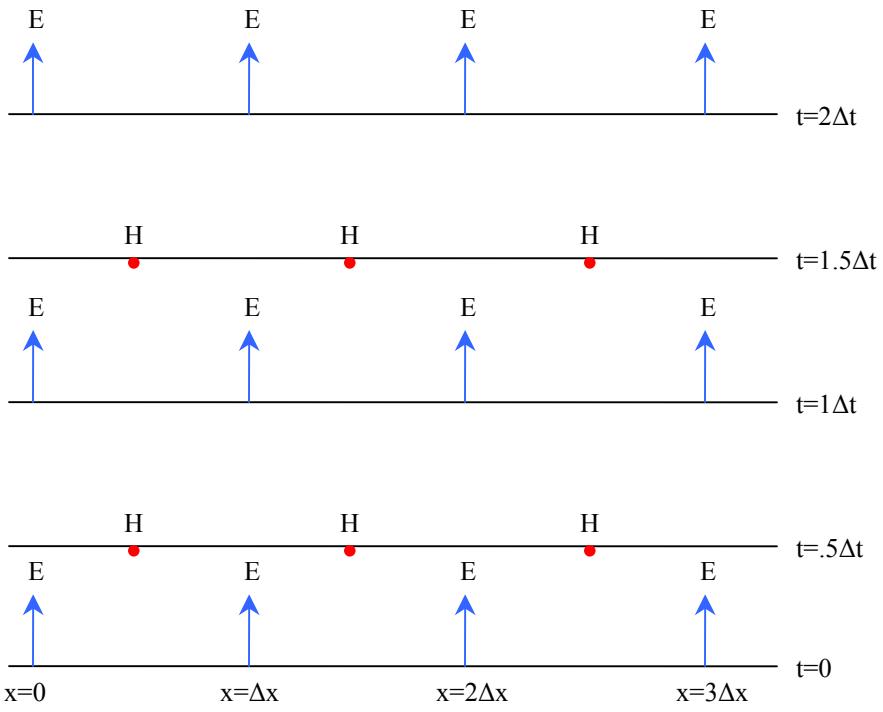

Figure 5.2 – Leapfrogging Diagram [15]

5.1.2 Determining the Cell Size

The next step in the process is to determine the appropriate cell size in order to achieve the proper representation of what is really occurring. When determining the cell size, there are two considerations that need to be considered. The cells need to be small enough to permit accurate results at the highest frequency of interest, yet large enough to allow actual computations to be made [14]. The acceptable cell size is approximately 10 cells per wavelength. For greater accuracy, cell sizes of 1/20 can be used. The lower limit is set by the Nyquist rate, so at least 2 cells per wavelength are needed. Another reason that cell sizes need to be small is to accurately model the geometry of the problem. The cell size has to be able to model the smallest detail of the object that is being used in the simulation.

5.1.3 Determining the Time Step

After determining the cell size, the time step can be determined from the Courant condition. The reason for this is that in one time step, any point on a wave must not pass through more than one cell. During one time step, the FDTD algorithm can only propagate the wave from one cell to its nearest neighbors [13]. In three dimensions, assuming that the speed of light is c, the time step must satisfy:

$$\Delta t = 1 \Big/ c\sqrt{\frac{1}{(\Delta x)^2} + \frac{1}{(\Delta y)^2} + \frac{1}{(\Delta z)^2}} \tag{5.4}$$

Using this time step, accurate results can be obtained. By decreasing the time step the amount of improved accuracy is negligible; so this value is called the "magic" time step.

The relationship for the "magic" time step is derived in the following manner. The wave number is $k = \pm w/c$. In order to eliminate error from the approximations it is necessary for the numerical wave number, $\tilde{k}$, to be equal to the regular wave number, k [15]. The finite difference approximation of the scalar wave equation is:

$$u_i^{n+1} \cong (c\Delta t)^2 \left[\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} \right] + 2_i^n - u_i^{n-1} \tag{5.5}$$

Setting $u_i^n = u(x_i, t_n) = e^{j(wn\Delta t - \tilde{k} i\Delta x)}$, which is a sinusoidal traveling wave, then the next term will be:

$$e^{j[w(n+1)\Delta t - \tilde{k} i\Delta x]} = \left(\frac{c\Delta t}{\Delta x}\right)^2 \begin{Bmatrix} e^{j[n\Delta t - \tilde{k}(i+1)\Delta x]} \\ -2e^{j[wn\Delta t - \tilde{k} i\Delta x]} \\ +e^{j[wn\Delta t - \tilde{k}(i-1)\Delta x]} \end{Bmatrix} + \begin{Bmatrix} 2e^{j[wn\Delta t - \tilde{k} i\Delta x]} \\ -e^{j[w(n-1)\Delta t - \tilde{k} i\Delta x]} \end{Bmatrix} \tag{5.6}$$

Factoring out the term $e^{j(wn\Delta t - \tilde{k} i\Delta x)}$ on both sides:

$$e^{jw\Delta t} = \left(\frac{c\Delta t}{\Delta x}\right)^2 \cdot \left(e^{-j\tilde{k}\Delta x} - 2 + e^{j\tilde{k}\Delta z}\right) + \left(2 - e^{-jw\Delta t}\right) \tag{5.7}$$

Separating the temporal terms from the spatial terms and dividing by 2 yields:

$$\frac{e^{jw\Delta t}+e^{-jw\Delta t}}{2}=\left(\frac{c\Delta t}{\Delta x}\right)^2\cdot\left(\frac{e^{j\tilde{k}\Delta z}+e^{-j\tilde{k}\Delta z}}{2}-1\right)+1 \tag{5.8}$$

Now using Euler's Identity, the numerical dispersion formula is:

$$\cos(w\Delta t)=\left(\frac{c\Delta t}{\Delta x}\right)^2\cdot\left[\cos(\tilde{k}\Delta x)-1\right]+1 \tag{5.9}$$

Equation (5.9) allows for the calculation of the dispersion that is created by the finite difference time domain method.

Using the "magic" time step, $c\Delta t=\Delta x$ in the equation (5.9) yields

$$\cos(w\Delta t)=1\cdot\left[\cos(\tilde{k}\Delta t)-1\right]+1=\cos(\tilde{k}\Delta t) \tag{5.10}$$

In order for equation (5.10) to hold, the following must be true

$$\tilde{k}\Delta x=\pm w\Delta t\rightarrow\tilde{k}=\pm\frac{w\Delta t}{\Delta x}=\pm\frac{w\Delta t}{c\Delta t}=\pm\frac{w}{c} \tag{5.11}$$

This shows that when using the "magic" time step, dispersion is not a factor. This result is for the one dimensional case, however the result at the beginning of this section holds for three dimensional space.

5.1.4 Specifying the Incident Field

The incident field can be specified quite easily. In order to create a uniform plane wave, every value on the edge of the surface is set to a sine or cosine function. As time marches on, the value on the edge will change and the previous value will move into the simulation space according to the central differencing equations. When specifying a point source, a magnetic loop is created. This magnetic loop will create a current through

the loop according to Ampere's Law. The result will be an electric field inside the loop which can simulate a point source. Using a sinusoidal source, the point charge can be created to emit a spherical wave.

5.1.5 Building an Object

Objects are built by specifying their permittivity and conductivity at every point in which the object is located. This process also needs to be done within the structure of the Yee Cells. This allows for objects with complicated structures as small as one cell size to be modeled.

5.1.6 Absorbing Boundary Conditions

Radiation boundary conditions are extremely important to the finite difference time domain method. In this leapfrogging technique, the fields are calculated using a central differencing method. This requires the adjacent fields to be known. This works when a wave guide is being simulated because the tangential electric fields at the boundaries have to be zero. In free space however, there are no such limitations. When the propagating waves get to the boundaries, reflections occur. These are due to the fact that the FDTD algorithm is only defined over a certain space. Outside that space every value is assumed to be zero. The FDTD algorithm will approximate the outer edges of the simulation incorrectly. In order to fix this problem, the leapfrogging must be stopped before it reaches the outer edges or a different approximation needs to be used. Stopping the time stepping is really not an option so a different method of approximating the outer

electric fields needs to be used. A simple solution to this problem is the first order Mur approximation [14]. The approximation for $\vec{E}_z$ at the x=0 boundary is:

$$\vec{E}_z^{n+1}(0, j, k+½) = \vec{E}_z^{n}(1, j, k+½) + \frac{c\Delta t - \Delta x}{c\Delta t + \Delta x}\left[\vec{E}_z^{n+1}(1, j, k+½) - \vec{E}_z^{n}(0, j, k+½)\right] \quad (5.12)$$

The current term for $\vec{E}_z$ is calculated from the $\vec{E}_z$ field at the point one index closer into the simulation from the previous time step, plus a constant times the current value at that closer point minus the previous value at the edge. This is a first order approximation because it only goes one term into the simulation space. A second order approximation is available which uses terms from the two fields that are further into the simulation space.

For this simulation a perfectly matched layer (PML) is used. The main objectives of a PML are to reduce the reflections from the edge of the simulation area as well as attenuate the waves that travel through the PML. The PML will have a finite thickness, so attenuating the signal ensures that by the time the wave reaches the true boundary of the simulation and bounces back to the desired simulation space it has been attenuated by a great deal. In order to achieve these properties, a layer around the original simulation space is defined so that it matches the impedance of the simulation space [16]. This idea was proposed in 1994 by Berenger. The fields are attenuated at the boundaries regardless of the angle of incidence.

Implementing the PML boundary conditions comes at a price. It is necessary to break the fields into components in both the x and y directions which doubles the number of variables. When equations (5.14) and (5.15) hold, the PML can be matched to free space, or any other environment.

$$\frac{\sigma_y}{\varepsilon} = \frac{\sigma_y^*}{\mu} \qquad \text{in the y direction} \quad (5.14)$$

$$\frac{\sigma_x}{\varepsilon} = \frac{\sigma_x^*}{\mu} \qquad \text{in the x direction} \tag{5.15}$$

In equations (5.14) and (5.15) σ represents the conductivity of the simulation space, and σ^* represents the conductivity of the PML. The main advantages of the PML are that it provides roughly a 10^3 improvement over other 2nd and 3rd order boundary conditions and that it operates strictly in the time domain.

5.1.7 Two Dimensional Simulation

This section tests the FDTD theory in MATLAB. The simulation runs in two dimensional space in the TM mode. This means that the electric field will always be in the z direction, and the magnetic field will be in the x-y plane. To test the algorithm, several different cases were run, each attempting to test different aspects of the algorithm. Figure 5.3 below shows a flow chart of the code.

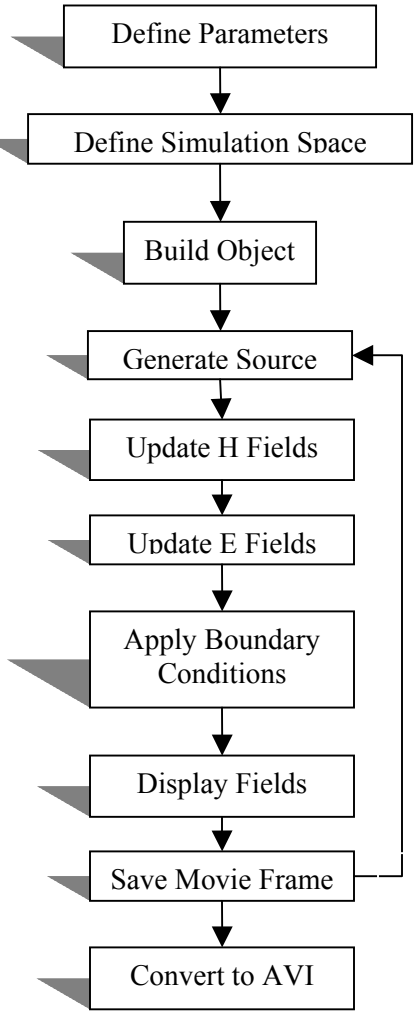

Figure 5.2– FDTD Flow Chart [14]

The program begins by defining several constants such as the permittivity, permeability, and the speed of light in a vacuum. The next step is to create the simulation space. The spatial

increments are set up to be one tenth of a wavelength to provide accurate sampling of the

fields. The time step is calculated to satisfy Courant's stability equation. The size of the space is set up by determining the number of cells to be used. The coefficients for Maxwell's Equation are also defined due to the fact that they will not change. They are defined at every point in the simulation space so that objects can be introduced in the next step. Objects are introduced now by changing the values of the coefficients at the required points in the coefficient matrices.

The next step is to initialize the electric fields and magnetic fields to zero to make sure that the space is available, and can be indexed properly. Following this step, variables are set up to implement the PML boundary conditions. At this point, the setup has been completed, and the time stepping can begin.

Now the incident field is defined. After the source is generated, the electric fields are updated using the Maxwell's equations in differential form. According to Yee's cell, the magnetic fields occur at half a time step after the electric fields. The theory is followed here, because the electric field is defined first.

5.2 Test Cases

In order to verify that the simulation works properly certain test cases can be applied. By running experiments in which the outcome is already known, it allows for the verification of the process. One simple test case is to simulate a point source. This will show that the source is defined properly. The result of a point source is a spherical wave. Figure 5.4 shows the result of this test case.

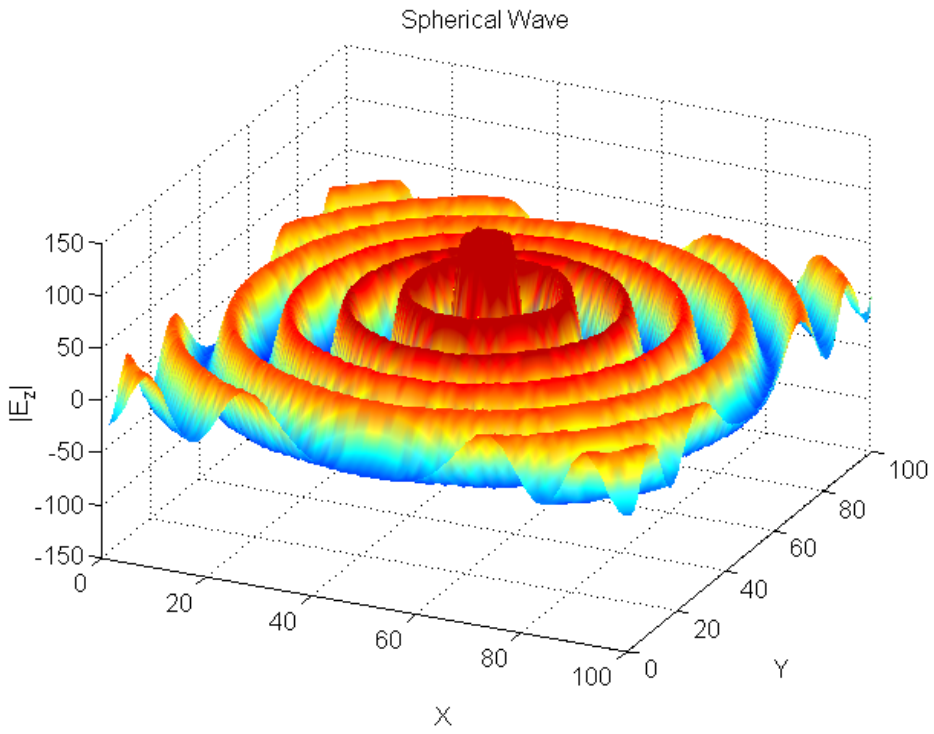

Figure 5.3 – Test Case: Spherical Wave

This test case also shows another important aspect of the simulation. Looking at the edges of the simulation space the lack of reflections is apparent. The spherical wave has its peak value at the source and then the magnitude decreases according to an inverse square law as expected. The field magnitude is also constant at a given radius which defines a spherical wave. Figure 5.5 shows an overhead view of the spherical wave. Since the simulation is only two dimensional, the amplitude can be represented by a color instead of a distance above or below the X-Y plane. In the graphical user interface that is developed later, the results will be shown from this vantage point.

Another important aspect of the simulation to test is the introduction of objects. In figure 5.6, a waveguide is placed around the source and array to guide the wave towards the array. The length of the waveguide is apparent from the figure. The wave is only guided to the array, and then the wave begins to spread out as it should.

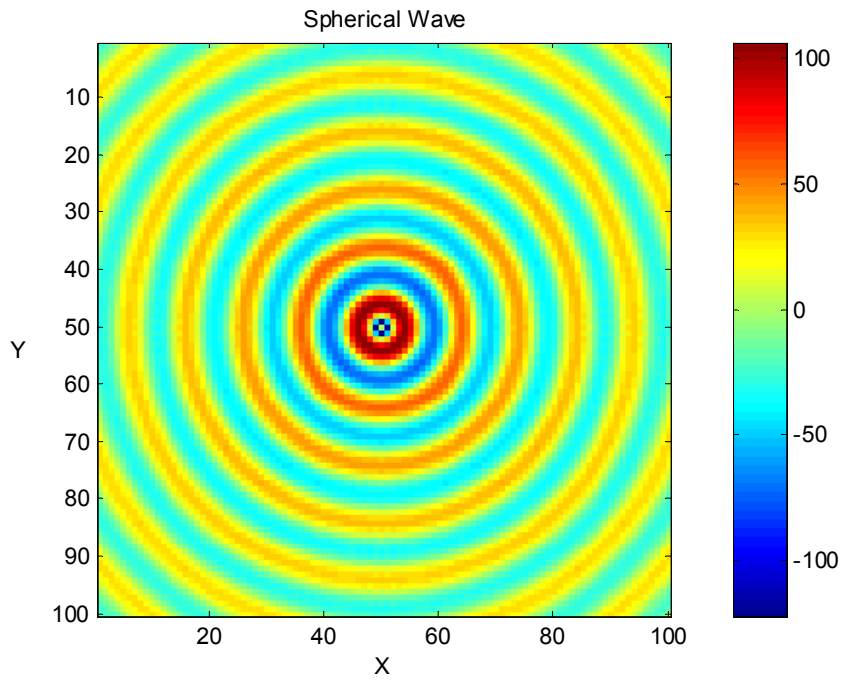

Figure 5.4 – Spherical Wave, Overhead View

This occurs at an X coordinate of 50. It is also encouraging that the waveguide completely reflects the signal and does not allow the signal to penetrate it.

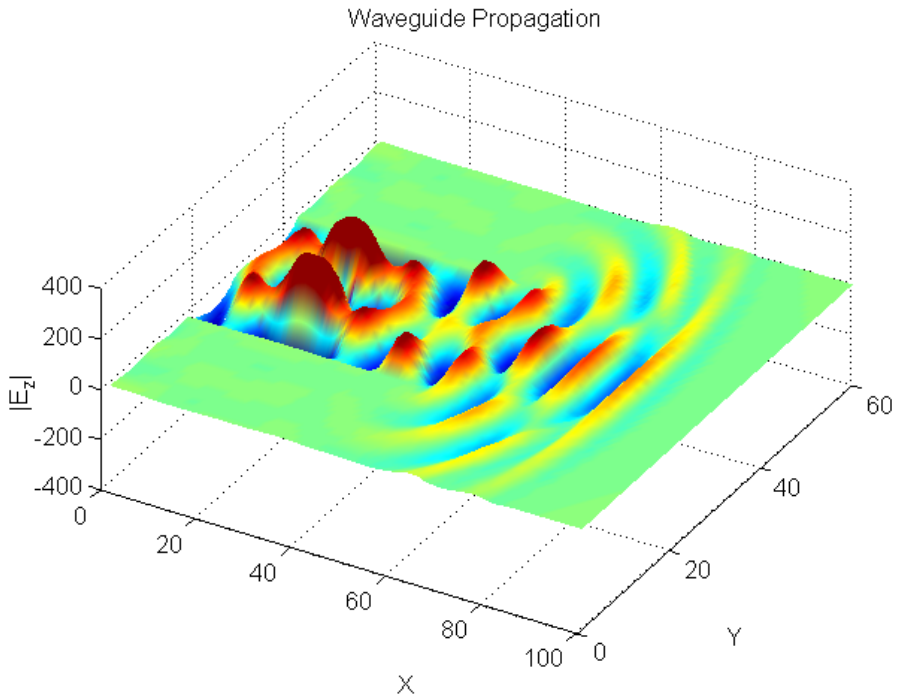

Figure 5.5– Test Case: Waveguide Propagation

A third aspect of the simulation that needs to be tested is the ability to introduce a different signal at each point in the simulation. Without this important feature, time reversal theory cannot be tested. Figure 5.7 shows several different signals being introduced at different points. The points are located in a linear array at an X coordinate of 50. The signal that was introduced was delayed by several samples at each successive element which adds directivity to the array focusing.

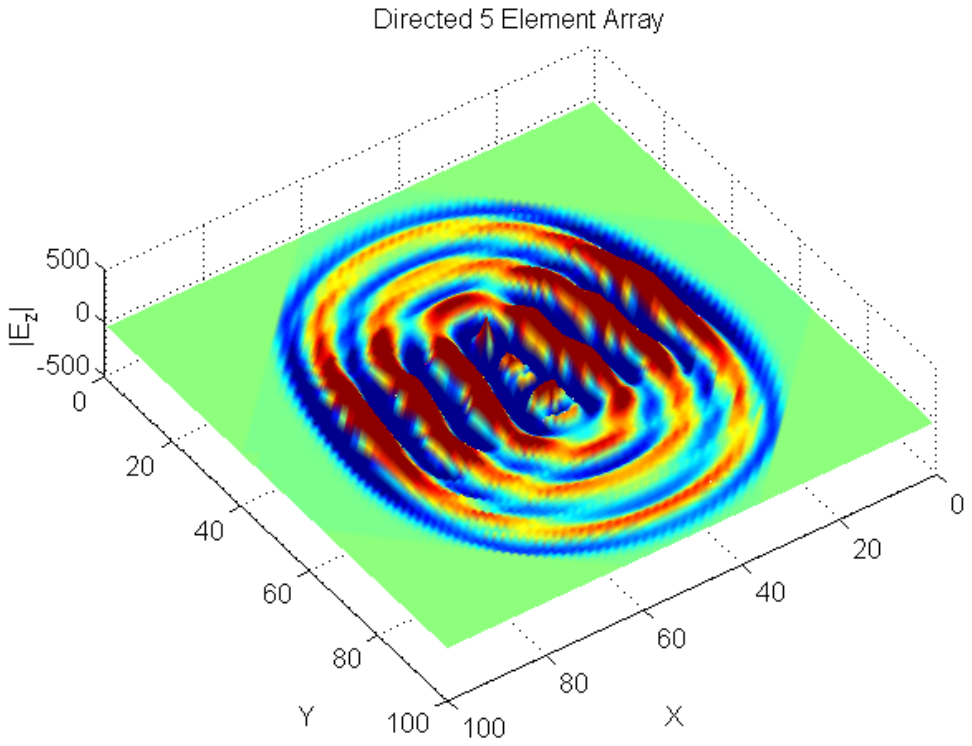

Figure 5.6 – Directed 5 Element Array

These three test cases show that the FDTD algorithm can adequately model the key criteria that are necessary to testing the theories of time reversal communications.

5.3 Graphical User Interface (GUI) Development

Developing a GUI allows the important aspects of the simulation to be altered easily. There is not a need to understand the code in which the language is written and anyone can use the application. In order for the GUI to be helpful, it needs to provide all of the necessary information to test time reversal theories, it has to allow the information to be saved so that it can be analyzed at a later date and it has to allow a user to change the parameters of the simulation.

The GUI was developed using the GUIDE function in Matlab. A .fig file is developed that shows the graphical nature of the program. An m-file is then developed that provides the functionality to the interface. Figure 5.8 shows the interface for the program.

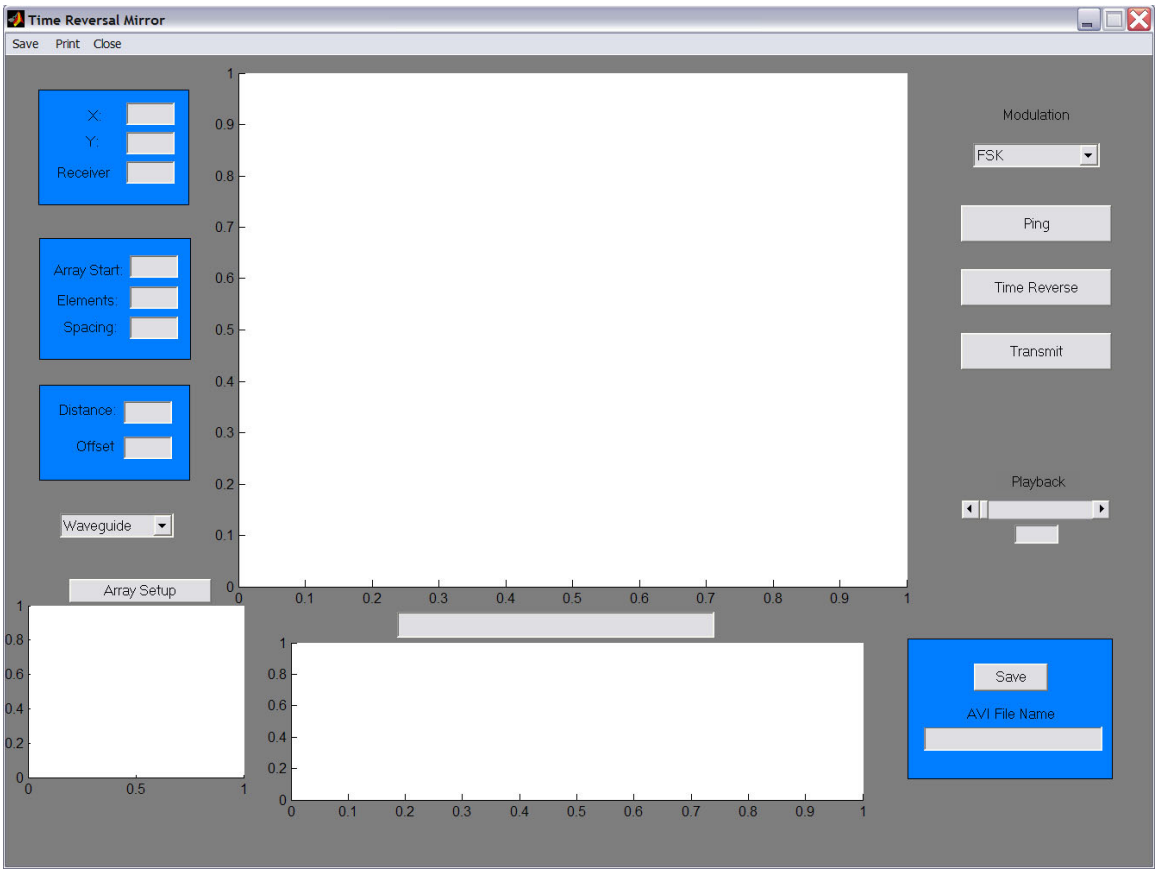

Figure 5.7 – GUI Interface

The left side of the GUI allows the user to input the various parameters of the simulation. The dimensions of the simulation space in the X and Y direction can be entered, as well as the location of the receiver in the X direction. The second grouping of inputs allows the user to enter the number of elements in the receiving array, the location of the first element in the X direction, and the spacing between the elements. The third block of inputs lets the user define the offset or Y coordinate of the receiver and the distance between the receiver and the array. The last user input on the left side of the GUI allows the user to introduce objects into the simulation. There are five choices: 1) Waveguide, 2) Line of Sight, 3) No Line of Sight, 4) Free Space, and 5) User Defined. The waveguide options places two reflecting walls on either side of the array. The line of sight and no line of sight options place 20 random reflectors throughout the simulation space. The points were chosen randomly using the randn() function in Matlab, but do not change from simulation to simulation. The randn() function distributes the points according to a normal distribution with unit variance. They are 9 cells by 9 cells wide and are placed based on the size of the simulation space. The difference between those two options is that the no line of sight option places an additional 9 by 9 cell reflector directly in the path between the receiver and the array. The free space option introduces nothing into the simulation space. The last option, user defined, opens another window. In that window the location of the receiver and array elements are plotted. The user can then place reflectors on that plot using the left mouse button in any location that they desire. When the user finishes they can click the right mouse button which closes the window. This function will introduce reflectors wherever the user chooses to do so.

Figure 5.9 shows the screen that enables the user to input the reflectors. The receiver can be seen at the point (30, 10) and the array is has 5 elements lined up at Y = 90. The reflectors have been placed in a diagonal line through the simulation.

The space on the bottom left of the GUI is used to display the simulation setup. After the user has defined all of the necessary parameters and selected the objects to introduce they may press the array setup button. This button will show a plot with the location of the receiver, the array elements, and the reflectors in the simulation. This is a helpful visual tool when using the GUI. It helps the user remember where the key elements of the simulation are as the rest of the program runs.

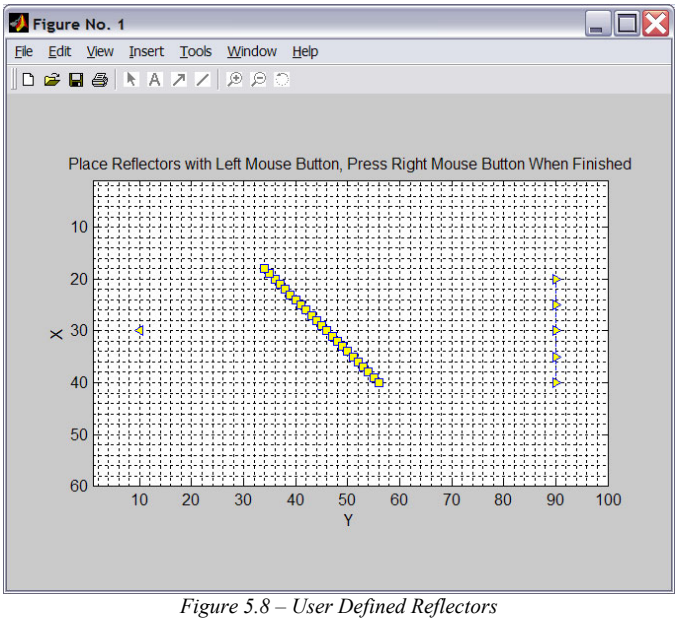

Figure 5.8 – User Defined Reflectors

After the user has set up the simulation they can choose the modulation type. The upper right hand corner has an input which lets the user choose between four options: 1) FSK, 2) ASK, 3) PSK, and 4) No Modulation. The first three options will modulate two bits, a 1 and a 0, in order to test the entire signal constellation. The no modulation option allows the user to transmit a simple sinusoid. When the user chooses a modulation

option, the signal that will be transmitted is plotted in the area at center of the bottom of the GUI. The line above the plot will give a textual update that describes the current plot. This area will be used to plot several things, so the textual display helps the user keep track of the plots. Figure 5.10 displays the GUI at this point in the process. The parameters have been entered on the left and the Line of Sight option has been selected. The array setup button has been already been pressed and the simulation setup can be seen in the lower left hand corner. The receiver is on the left and the array is on the right. The reflectors are the red squares that are randomly placed throughout the space.

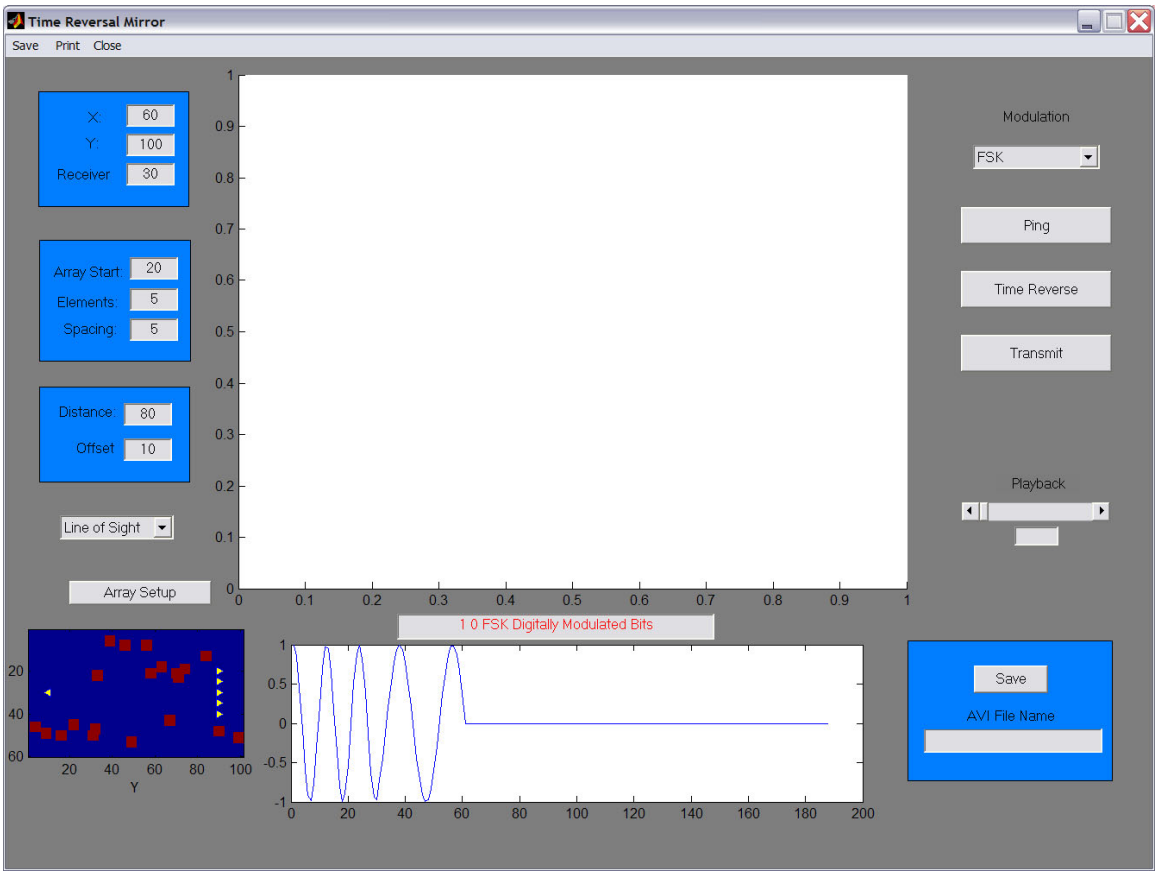

Figure 5.9 – GUI Interface, Modulation Scheme

The FSK modulation option was selected, so the bottom plot shows a Binary FSK signal representing a 1 and a 0. The text box above the bottom plot states reads: "1 0 FSK

Digitally Modulated Bits." The signal is then zero padded so that the simulation will run for a long enough time period.

After the user has selected a modulation scheme, they may press the Ping button. This will initiate the FDTD algorithm. The signal that was chosen is introduced into the simulation at the location of the receiver. After each time step of the FDTD algorithm the electric field will be plotted in the large area in the center of the GUI. The result is a slideshow that depicts the wave propagation from the receiver to the array. When the entire signal has been transmitted, the signal that is received at the array is plotted at the bottom. Figure 5.11 shows the GUI after the forward propagation. At this point, the user can press the time reverse button. This removes the critical part of the received signal and time reverses it. The result of this process is again plotted at the bottom.

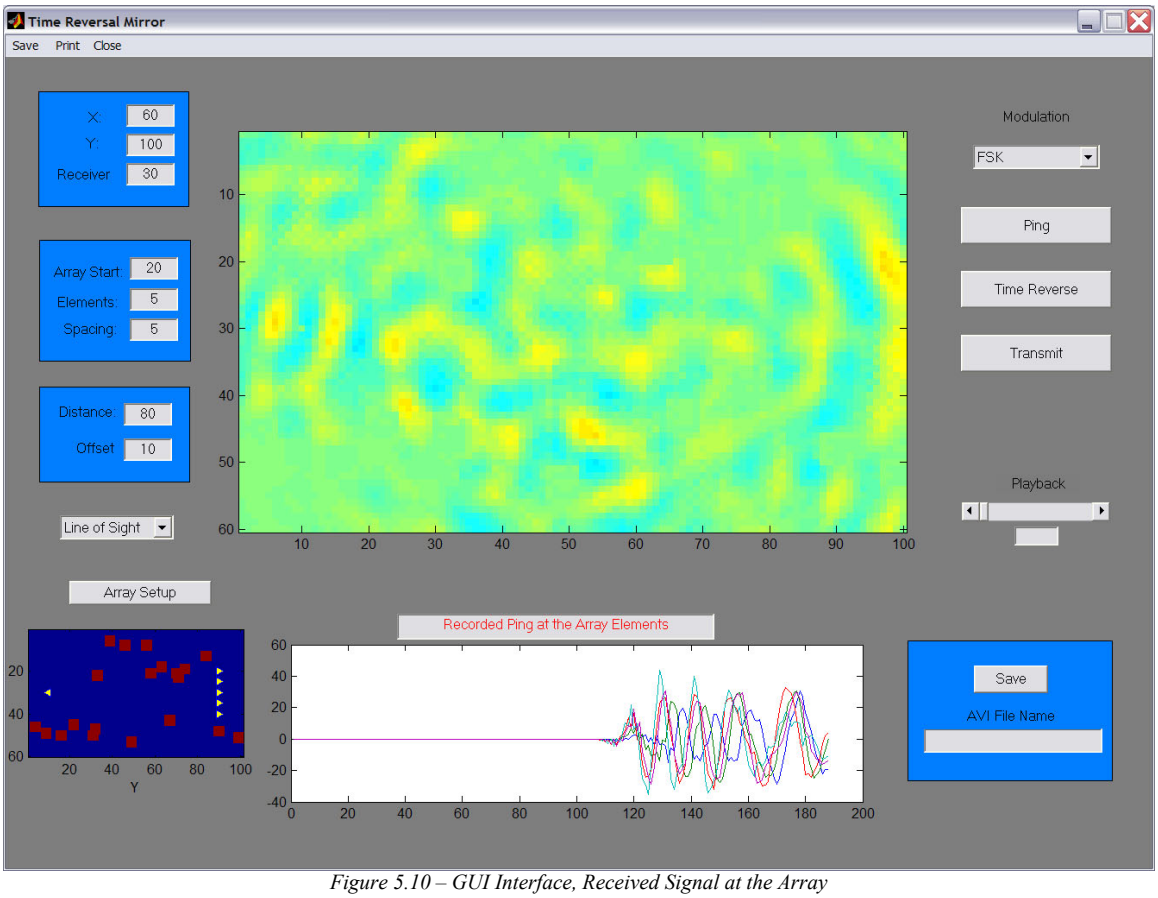

Figure 5.10 – GUI Interface, Received Signal at the Array

Figure 5.12 shows the time reversed signal in the bottom plot area. The text box notifies the user that the signal is the time reversed signal to be transmitted by the array.

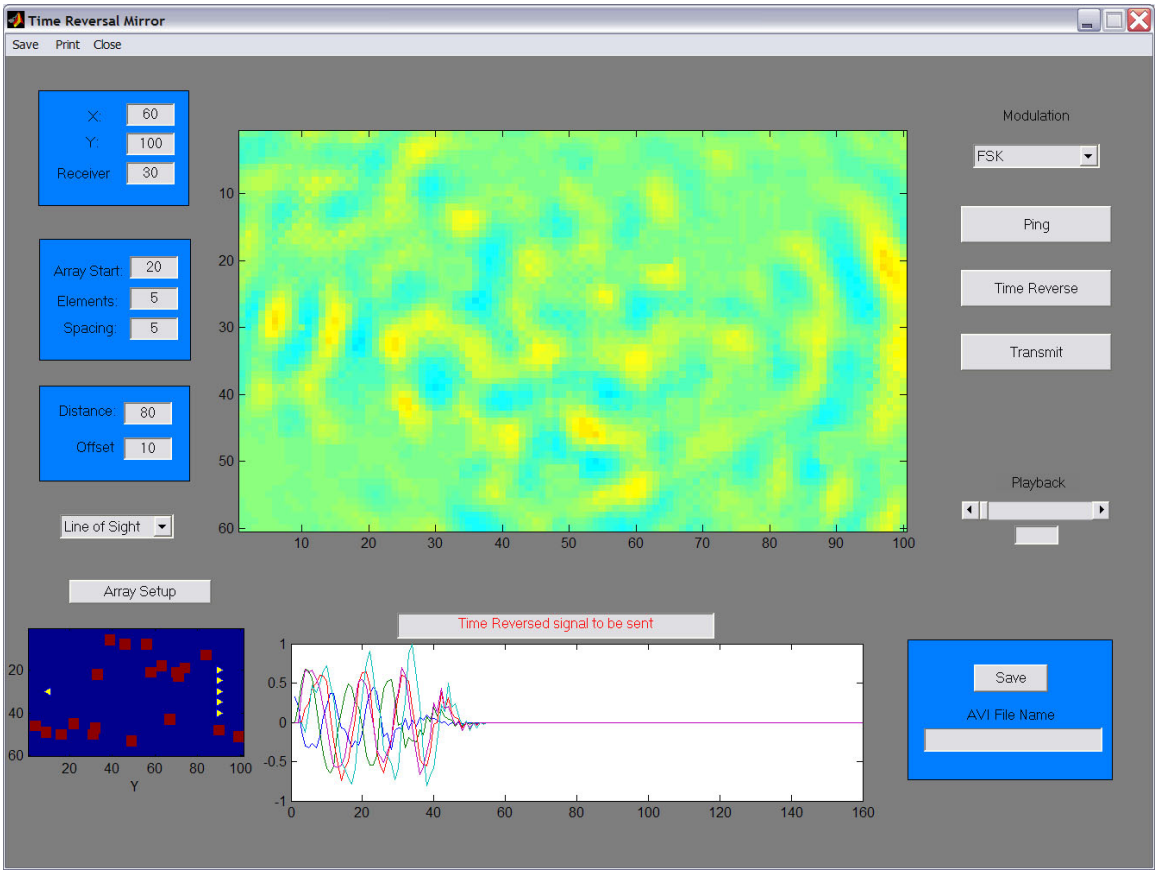

Figure 5.11 – GUI Interface, Time Reversed Signal

At this point, the user can press the transmit button which will transmit the time reversed signal to the user. The signal will propagate back through the same environment through which the forward propagation occurred. Each step in the process will again show up as a slideshow in the main plot section. In order to ensure that simulation will run for a long enough period of time in the reverse direction, the time reversed signal was zero padded. When the simulation finishes, the received signal at the receiver is plotted at the bottom with an appropriate text message. The final image that shows up in the main plot area will not necessarily show the time instant when the signal is at the

receiver. In most cases the signal will continue to propagate past the receiver in order to determine whether or not the signal attenuates at further distances. In order to determine the focusing around the receiver a playback function is available. A slider located on the right side of the GUI allows the user to cycle through the all of the pictures that made up the slideshow. In Figure 5.13, the box under the slider shows that the current plot is frame number 319. This seems to be an appropriate frame to show the ability of the time reversal mirror to focus energy. The user can playback the entire process stopping at any frame to check certain aspects of the simulation.

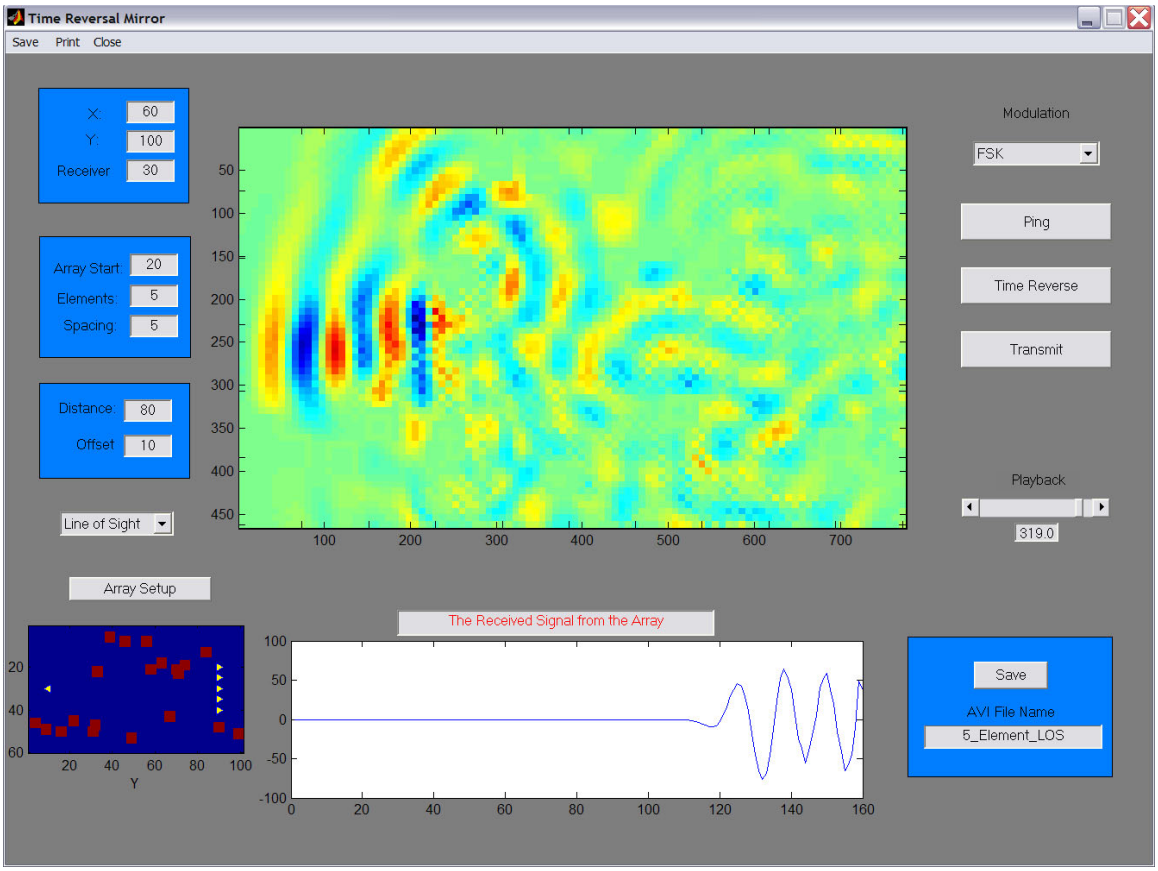

Figure 5.12 – GUI Interface, Received Signal at Receiver

The last key feature of the GUI is the box in the lower right hand corner. This area allows the user to save the slide show as a movie in AVI format. The user can type

in a descriptive name and press the save button. The file will be saved in the working directory of Matlab.

The GUI provides a lot of functionality that enables anyone to test the rudimentary aspects of time reversal theory. No experience with Matlab is necessary to run the program. It also provides the ability to test various components of the simulation while holding all of the other variables constant. For example, the location of the receiver can be changed while holding every other variable constant. That type of experiment would show whether or not the array has better focusing ability in certain directions. In another test, one might want to vary the spacing between the elements and hold all of the other parameters constant. This could be used to determine the optimal spacing for between the array elements. The next section will use this GUI to perform various experiments.

6. Simulation Results

Using the GUI, several of the theories of time reversal mirrors are tested. The first test shows how different modulation formats will behave when they are used in conjunction with a time reversal mirror. In the binary case, ASK and PSK modulated signals are practically identical. Multiplication by a -1 is equivalent to a phase shift of $\pm\pi$, so the two signals are equivalent. Due to this fact, only the first experiment tests all three modulation schemes; the following experiments test PSK and FSK. The experiments here will try and determine the effects of the number of elements in the array, the effects of different environments on the focusing ability, and the ability of the array to focus energy in directions other than broadside to the array.

6.1 Modulation Format

It is very important to test different modulation formats with the time reversal array to determine whether the array will have different effects on each signal. With PSK and ASK signals, the phase shift that is used to differentiate between bits needs to remain in the signal as it propagates along its path. If this information is lost, those formats are not suitable for use with a time reversal mirror. When sending an FSK signal, the information is encoded in the frequency of the signal, so in order for that modulation to be useful, the frequency cannot be altered by the time reversal mirror.

All three experiments that were run to test the modulation format were done at a distance of 40λ with 17 array elements. The receiver was located broadside to the

array. The experiments were performed in free space. Figure 6.1 shows the results of the FSK test.

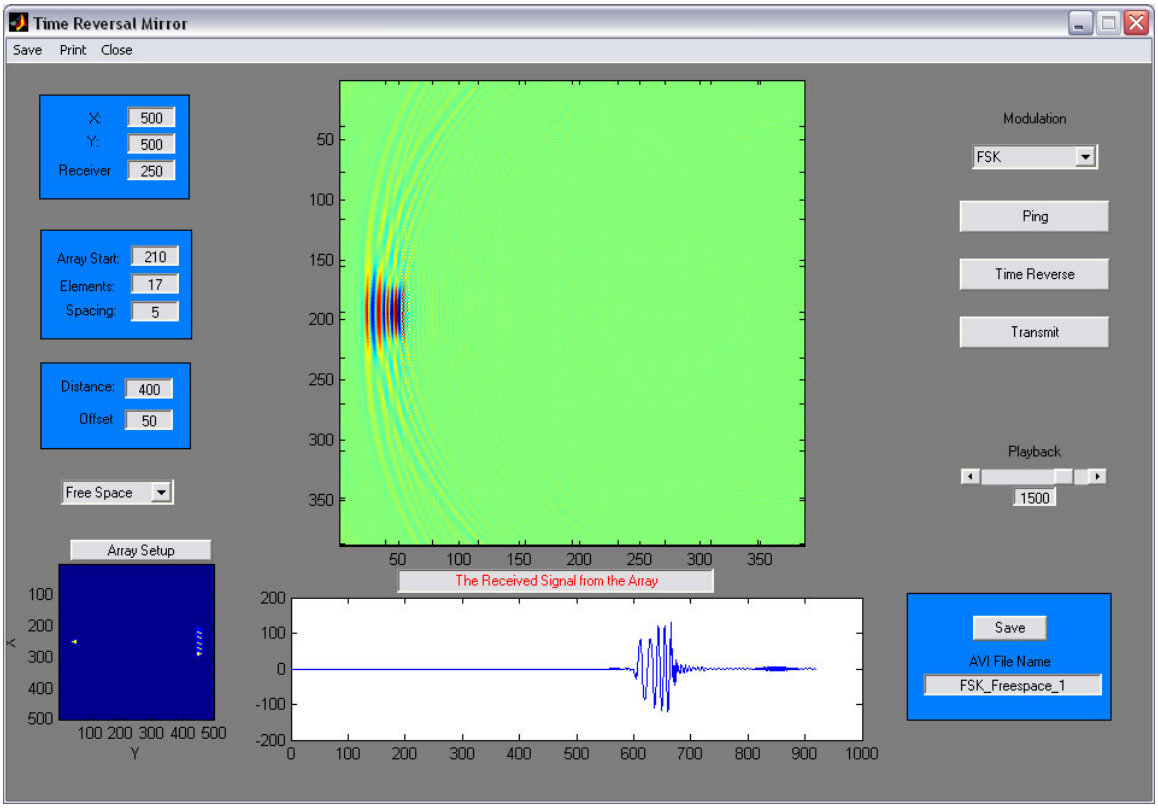

Figure 6.1 - FSK Modulation, Free Space

The results of this experiment show that the FSK signal was received by the receiver with the frequency modulation intact. Figure 6.2 show the result of the PSK test, and figure 6.3 shows the result of the ASK test. Both of these tests show that the phase modulation remained intact throughout the process.

6.2 Length of the Array

This experiment tests the theory from chapter 2 that the energy that the array transmits will be focused in an ellipse with a range axis of $\lambda\left(\frac{L}{a}\right)^2$ and a cross range axis equal to $\lambda\left(\frac{L}{a}\right)$. In this case, L will be 40λ and the length of the array, *a* will vary.

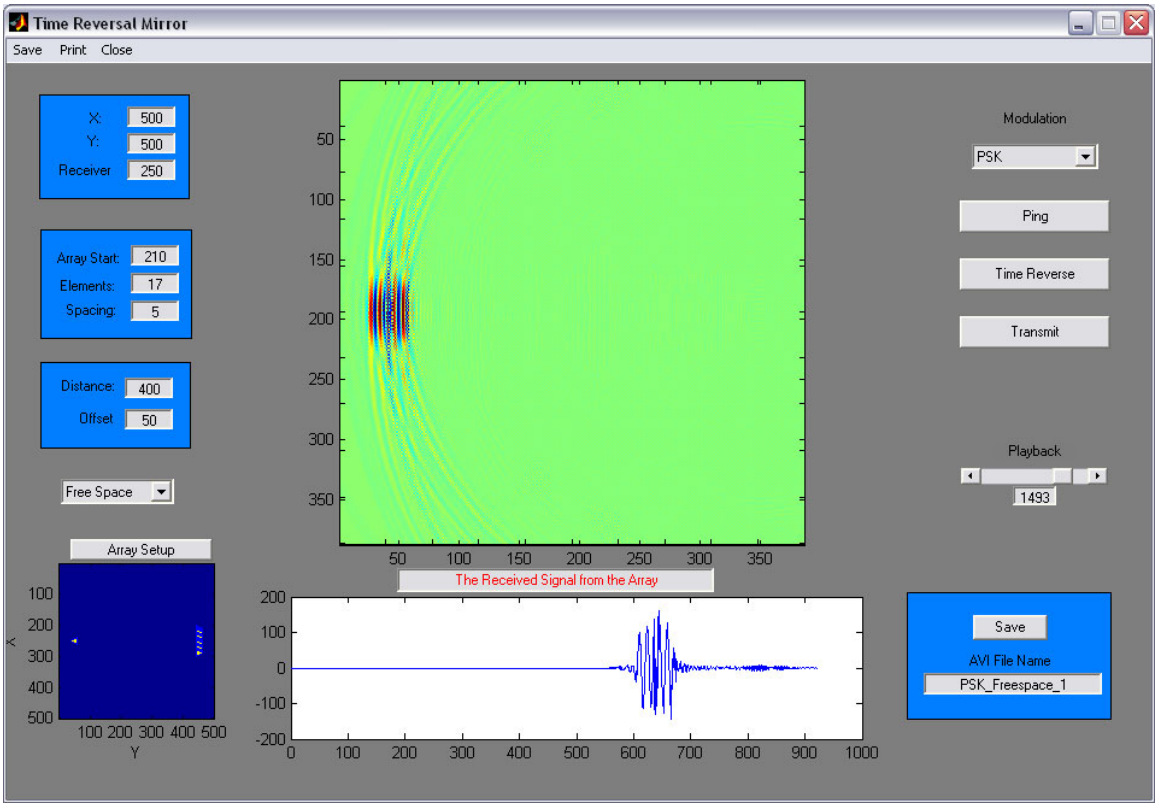

Figure 6.2 - PSK Modulation, Free Space

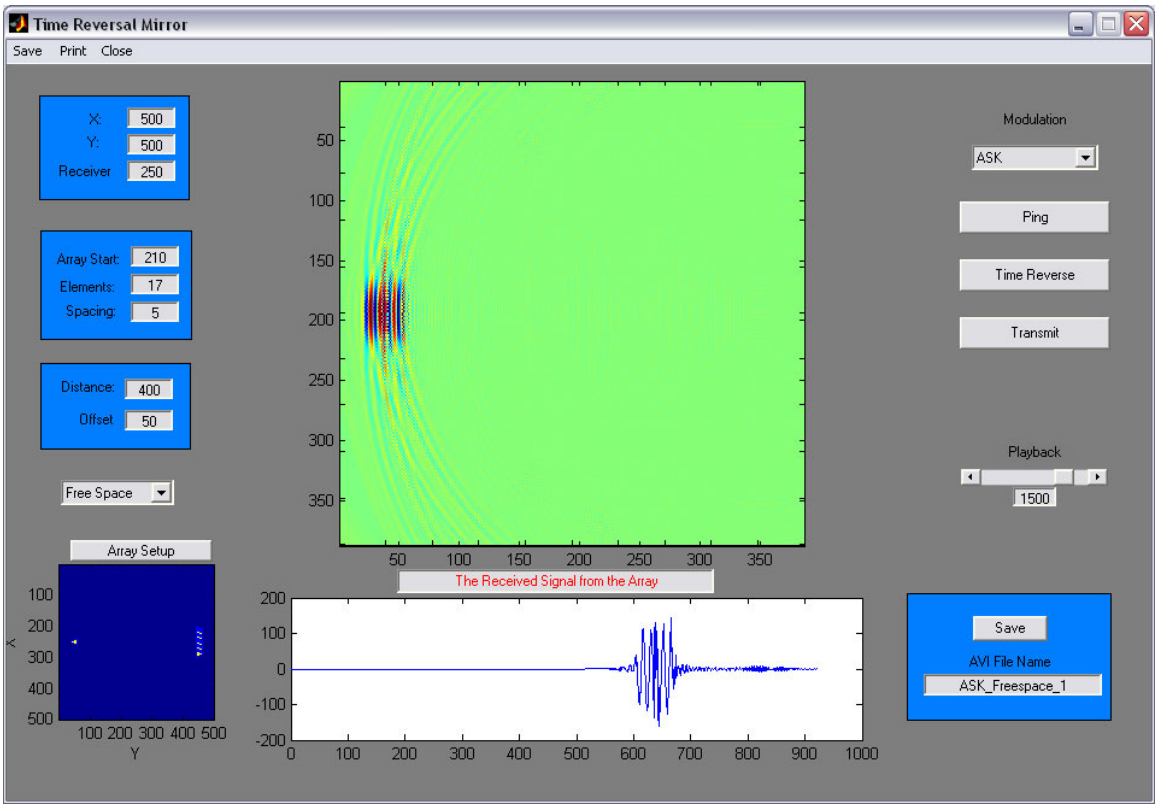

Figure 6.3- ASK Modulation, Free Space

The cross range axis will be easily measured from the figures provided. The range axis is not measured because it is too large for the simulation space. The smallest array used will have a length of 4λ. This would result in a range axis of 100λ which would require a simulation space that was over 1000 cells long. The experiments for this section are performed using FSK modulation and are performed in free space.

Figure 6.4 shows the results of the simulation with an array which has 9 elements. The length of this array will be 4λ. The theoretical cross range axis for this setup should be 10λ or 100 cells. The original simulation space had dimensions of 500 by 500. When the simulation is played back, it is scaled to 400 by 400. In this space, a length of 100 cells will now be 80 cells.

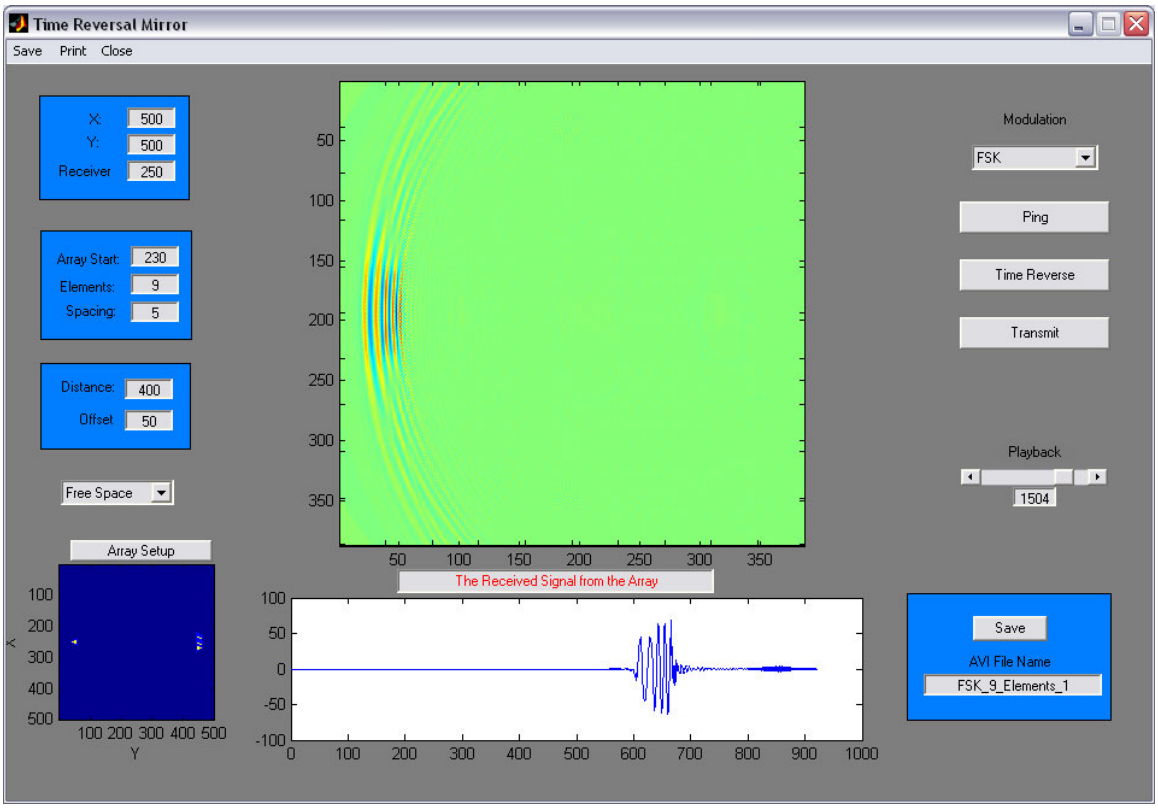

Figure 6.4 - FSK Modulation, 9 Elements

The figure shows that the theoretical cross range axis is achieved.

For an array with 17 elements, refer back to figure 6.1. The length of the array in this case is 8λ which should result in a cross range axis of 5λ which in the figure should show up as 40 cells which it does.

Figure 6.5 shows the result when there are 25 elements in the array which results in a length of 12λ. The theoretical cross range axis should be $\frac{10}{3}\lambda$ which would be a little more than 25 cells in the figure and again it does.

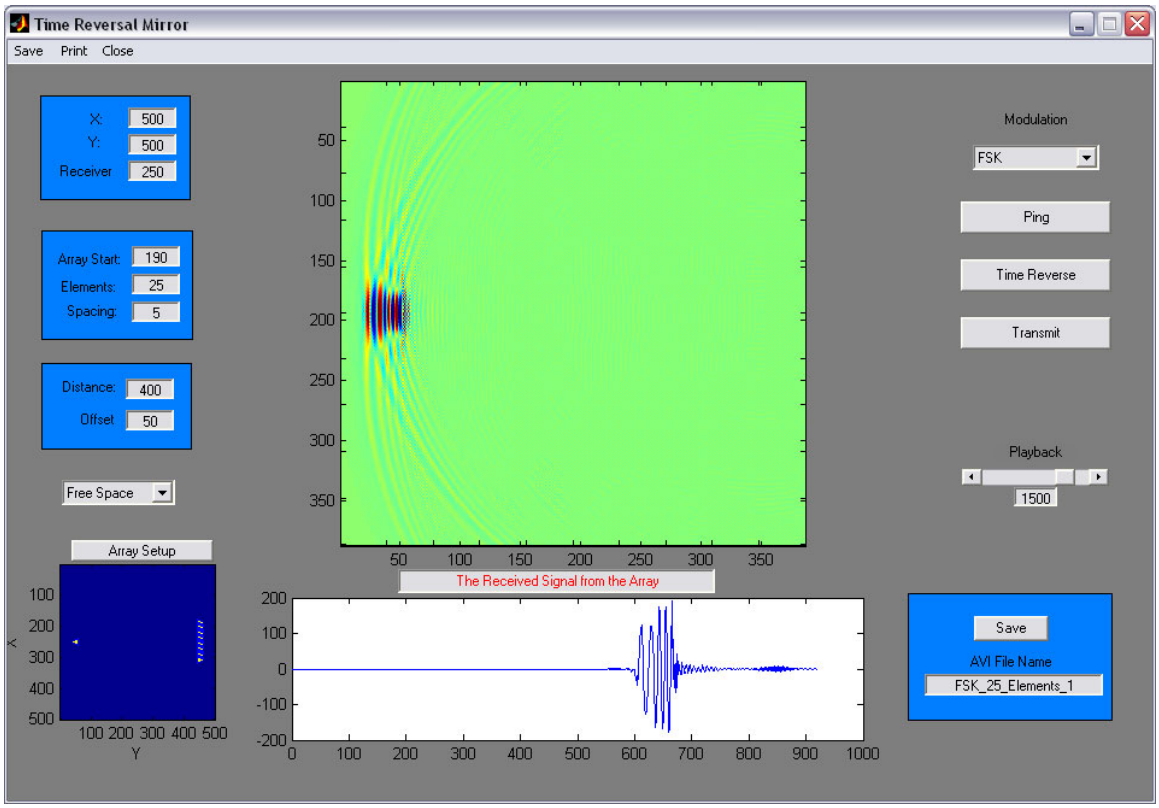

Figure 6.5 - FSK Modulation, 25 Elements

Not only is the focused field smaller, the amplitude of the received field is also larger. From some of the figures, it appears that the focused field is a little larger than predicted by the theory; however the increased amplitude of the signal has an effect as well. The width of the signal should be determined by a percentage of energy within the certain bounds. Since the amplitude is higher for larger arrays, the width of the signal will be slightly less than pictured due to the amplitude increase.

6.3 Off Axis Focusing

This experiment tests whether the array can focus energy in various directions. Figure 6.6 shows the results when the receiver is at a 10 degree angle from the array and figure 6.7 shows the results when the receiver is at an angle of 20 degrees. Both of

the experiments were performed in a free space environment with arrays with 17 elements. Both figures show that the array was able to focus energy in other directions.

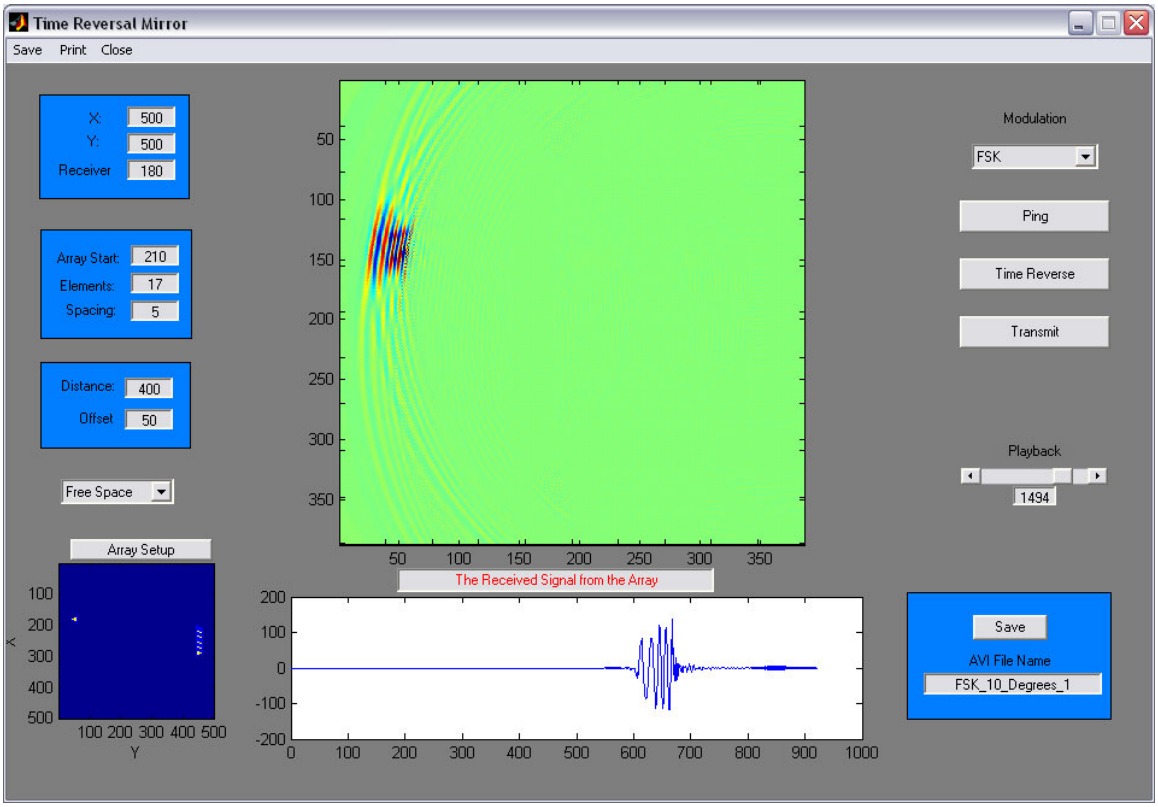

Figure 6.6 - FSK Modulation, 10 Degrees off Axis

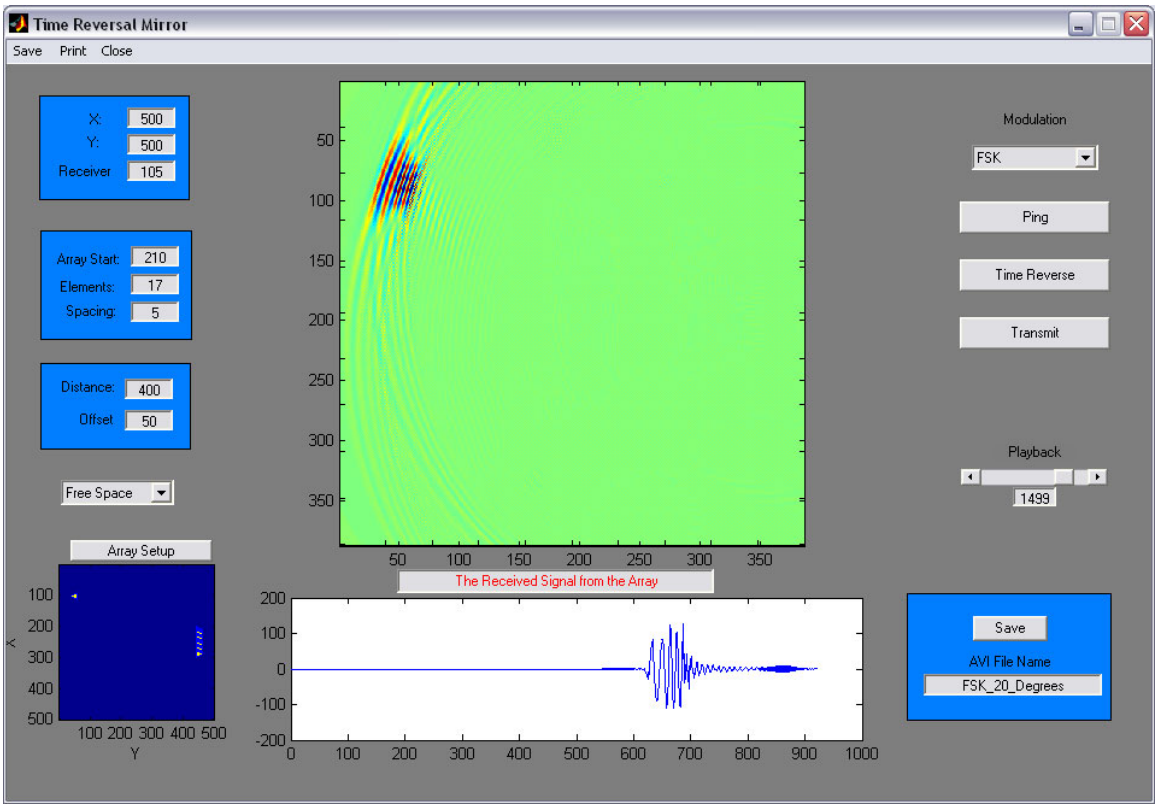

Figure 6.7 - FSK Modulation, 20 Degess off Axis

6.4 Line of Sight Environment

A major theoretical advantage to using a time reversal mirror is its ability to utilize multipath propagation to enhance its focusing ability. In order to test this idea, the array placed in environment with random reflectors. This setup does allow a direct line of sight path from the receiver to the array. Due to the fact that there are reflectors present, more energy should be focused on the array increasing its effective aperture. This, in turn, should enable to array to focus more strongly on the receiver.

For this experiment, both FSK and PSK transmissions were tested. The distance between the array and the receiver is 40λ and there are 17 elements in the array. Figures 6.1 and 6.2 show the focusing ability of the same arrays without any reflectors for FSK and PSK signals respectively.

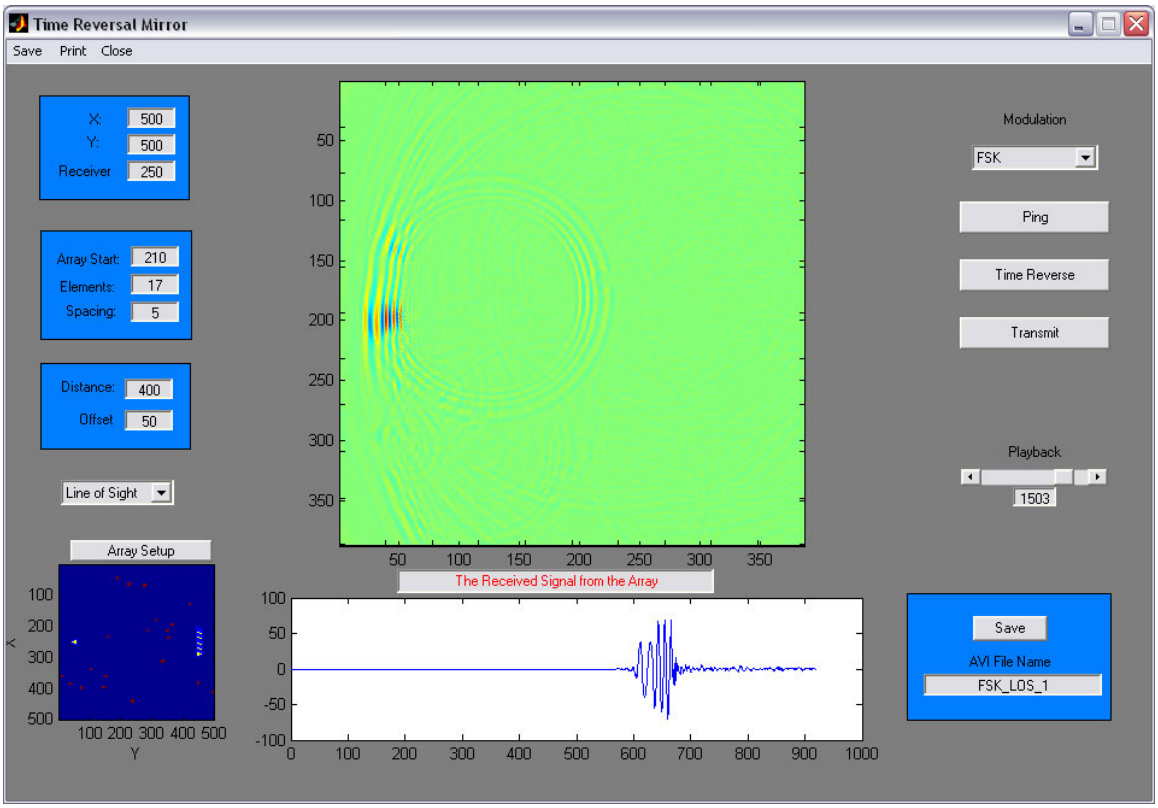

Figure 6.8 - FSK Modulatioon, Line of Sight

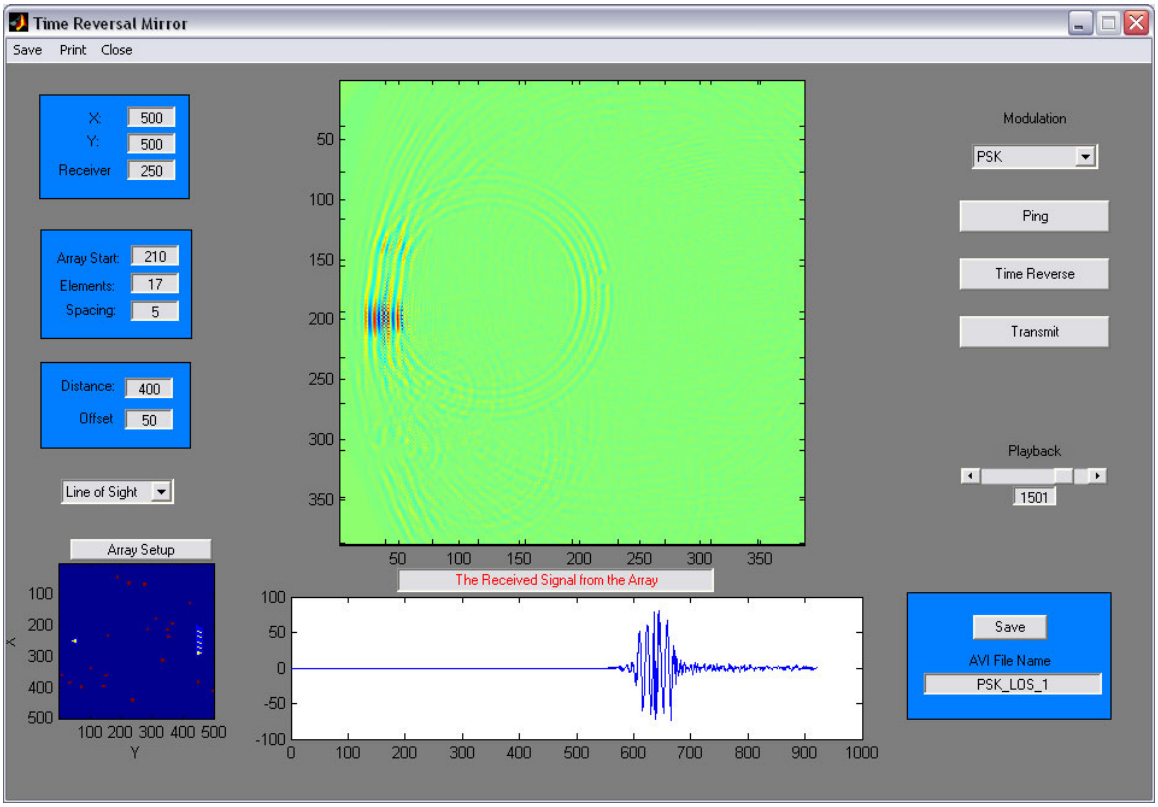

Figure 6.9 - PSK Modulation, Line of Sight

Figures 6.8 and 6.9 show the results of transmitting an FSK and PSK signal respectively in a line of sight environment. From the figures, the ability of the array to focus energy is greatly increased in the presence of reflector for both modulation formats. Both figures 6.8 and 6.9 show a small amount of signal energy located at another location slightly above and to the right of the desired location. This is most likely due to the specific layout of the reflectors and would vary based on the randomness of the environment.

6.5 No Line of Sight Environment

This experiment tests the ability of the array to focus on a reflector when there is an obstacle in the direct path between the receiver and the array. This reflector will block a large portion of the signal energy from the array which should degrade the focusing ability. Figure 6.10 shows the results of this experiment. Looking at the received signal, the FSK format was preserved. The signal is focused at the location of the receiver; however the amplitude is decreased compared the figure 6.8 which show the same experiment performed with a direct line of sight path. It is encouraging that the array was still able to focus on the receiver even when there was a lot less energy collected.

6.6 Summary of Results

These experiments where very successful. The theories behind time reversal mirrors held up under the testing. A major idea that came out of the testing was that all three tested modulation formats performed equally well under the given conditions.

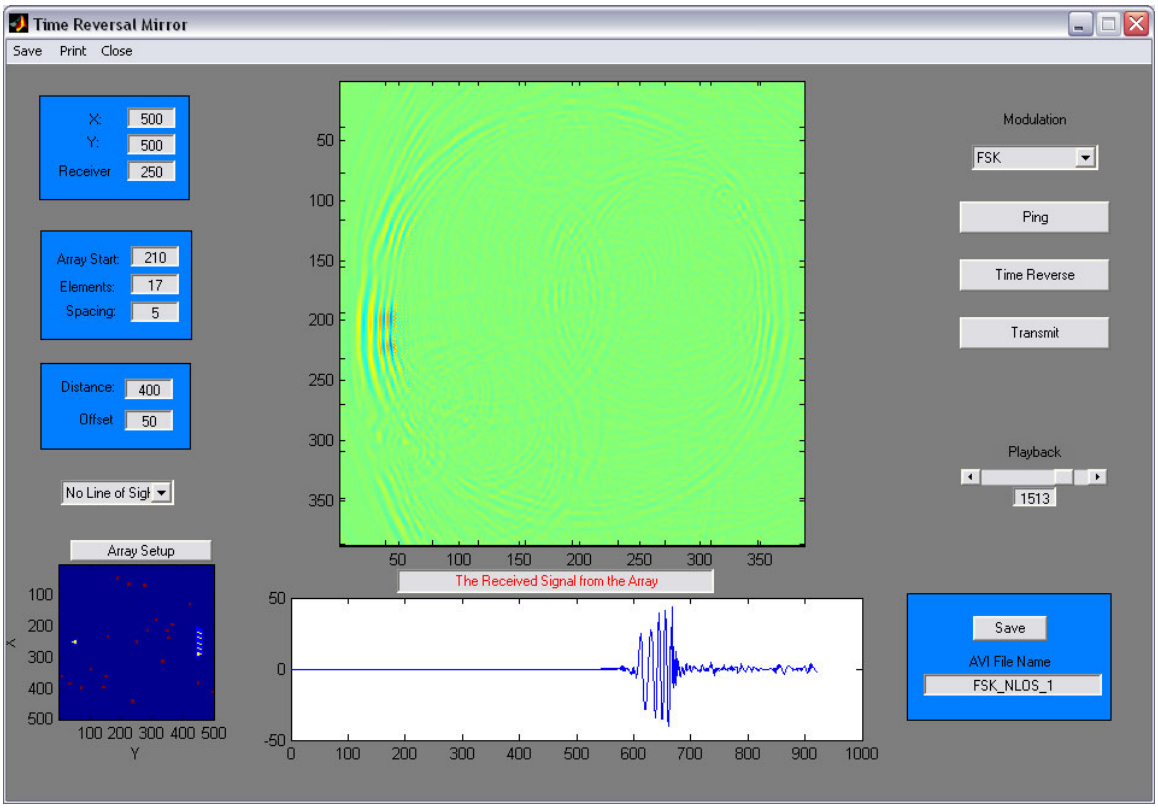

Figure 6.10 - FSK Modulation, No Line of Sight

The ability of the array to focus energy in various locations was demonstrated. Better focusing can be achieved through the use of a larger array. It was also shown that randomly placed reflectors will enhance the ability of the array to focus energy. This key fact can possibly lead to wireless systems that can be implemented without channel equalizers, or diversity techniques.

7. Future Considerations

Using time reversal methods for wireless communications is in the early stages of development. This paper deals with only a small number of parameters than can be evaluated to help enhance the performance of a time reversal mirror. A lot of work is still necessary at the simulation stage of the technology before testing is performed on real equipment. This chapter presents some of the aspects that can be further explored using the existing software that was developed for this paper with only slight modifications. The second part of this chapter presents a possible scenario perform actual tests of time reversal mirrors in real situations.

7.1 Test Scenarios

This paper deals with the basic principles of time reversal theory. The focusing ability of the array was tested and positive results were obtained. In order to determine whether or not this method was viable for communications, only the signal constellation was transmitted. Another major step that can be taken to further test time reversal theory would be to actually encode a number of bits using the desired prototype signals. Processing can be performed at the array that will extract the signal constellation for the specific user and store that information. That information can then be used to transmit actual data back to the receiver. Processing can then be performed as the receiver to demodulate the received signal. Using that code, bit error rates can be calculated

experimentally to determine whether certain modulation formats perform better than others. Noise can be added to the simulation is order to facilitate the calculations.

For this paper, the shape of the array was linear. In the future, other array shapes can be tested using this simulation to determine whether different shapes can improve the focusing ability of the array. Another experiment that can be performed to test the bounds of the focusing ability would be to place two receivers in the simulation space. This might prove to be somewhat difficult. Each user would need to have a specific code that was unique to that user. The array would need to be able to differentiate between the two receivers when it receives their prototype signals. The array would have to determine the prototype signals for the two different users. At that point, the array could transmit data to both users. Tests could be performed to determine how close the users can get to one another before the interference between the two became an important factor.

Those are just some of the scenarios that could be tested with the basic structure that has been set up for this paper. Receiver motion can also be simulated with this code by changing the location of the receiver at each time step. There is a lot of room for growth in this field, and simulations can be a very useful tool in fleshing out the nuances.

7.2 Real Testing

After sufficient simulations are performed the next logical step would be to perform actual testing of a time reversal mirror. The ideal situation would be an inexpensive test setup that was portable. A portable setup would enable testing in many different environments.

A theoretical test setup could utilize 900 MHz transmitter and receiver pairs that are used to transmit audio files from computers to stereo equipment. The equipment can be accessed through the sound card and the processing can be performed using Matlab. The transmitters and receivers can be collocated so that they can be used as a transceiver. An array can be setup using multiple receiver transmitter pairs. The main difficulty in this setup would involve multiplexing the signals from each transceiver pair so that the processing can be performed on one computer. The same type of setup can be used for the receiver. An array can be used here as well to monitor the signal intensity at other locations besides receiver location. The cost of this experiment could be relatively low, including the price of the two computers and each transmitter receiver pair costing under $50.

This is a growing field and will most likely yield results in many different areas. Underwater communications and ground penetrating have already displayed the usefulness of this algorithm. Wireless communications will most likely benefit as well from this technology.

Appendix

A.1 Ray Simulation (Single Reflections)

```
%Steven Seiden
%November 13,2003
%Time Reversal Methods in Communications

%point reflectors and a cosine wave transmitted
clear;
points=10;
dimension=100;
%generate randomly placed point reflectors
x=round(dimension*rand(1,points));
y=round(dimension*rand(1,points)-dimension/2);
reflectors=[x;y]';

%place the receiver and transmitter
source=[0 0];
reciever=[dimension 0];

%--------------------------------Plot Geometry----------------------------
% figure(4)
% hold on
% plot(reflectors(:,1),reflectors(:,2),'o','markeredgecolor','g','markerfacecolor','y')
% plot(source(:,1),source(:,2),'<','markeredgecolor','r','markerfacecolor','k')
% plot(reciever(:,1),reciever(:,2),'>','markeredgecolor','r','markerfacecolor','k')
% line([reciever(1,1) source(1,1)],[reciever(1,2) source(1,2)]);
% for i=1:points
%     line([reflectors(i,1) source(1,1)],[reflectors(i,2) source(1,2)])
%     line([reciever(1,1) reflectors(i,1)],[reciever(1,2) reflectors(i,2)])
% end
% title('Source, Receiver, and Point Reflectors')
% xlabel('X')
% ylabel('Y')
% legend('Point Reflectors','Source','Reciver','Paths')
% axis([-5 dimension+5 -(dimension/2) dimension/2])
% hold off
%-------------------------------------------------------------------------
%transmit a cosine wave
Fs=50E7;
%determine the maximum distance for a wave to travel
v=3E8;  %speed of light
maximum=(dimension/2)+sqrt((dimension/2)^2+dimension^2);
tmin=2*maximum/v;
t=0:1/Fs:2*tmin;
fc=5E6;
T=1/fc;
x_t=cos(2*pi*fc*t);

%The recived signal will have a different amplitude
%and phase.  The phase will be determined by time taken
%to reach the reciever divied by the period of the wave
%and then modulus by T. The amplitude will decrease by
%a inverse square law

%calculate the distances travelled first from source
%to reciever
distance1=sqrt((sum([reciever-source].^2,2)));
%find the distances travelled by reflected waves
%first find distance from source to point
distance2=sqrt(sum([reflectors(:,1)-source(1) reflectors(:,2)-source(2)].^2,2));
%then find distance from point to reciever
distance3=sqrt(sum([reciever(1)-reflectors(:,1) reciever(2)-reflectors(:,2)].^2,2));
```

```
%add the distances together
distance4=distance2+distance3;
%create a column of distances starting with direct path
distance=[distance1;distance4];

%Now apply the inverse square law
A_x=1./(4*pi*distance.^2);

%determine the phase differences in radians

time=distance./v;
phases=2*pi*mod(time,T)./T;

%the recieved signal
for i=1:points+1
   r_t(i,:)=A_x(i).*cos(2*pi*fc*t-phases(i));
end

%----------plot the recieved signals on First Pass------
figure(3)
subplot(2,1,1)
hold on
for k=1:points+1

   switch mod(k,7)
      case 0
         plot(t,r_t(k,:),'m')
      case 1
         plot(t,r_t(k,:),'g')
      case 2
         plot(t,r_t(k,:),'k')
      case 3
         plot(t,r_t(k,:),'c')
      case 4
         plot(t,r_t(k,:),'b')
      case 5
         plot(t,r_t(k,:),'r')
      case 6
         plot(t,r_t(k,:),'y')
   end
end
title('The Recieved Signals on First Pass')
ylabel('Magnitude')
xlabel('Time')
hold off

%add each of the recieved signals together
subplot(2,1,2)
r_total=sum(r_t,1);
plot(t,r_total);
title('The Total Recived Signal on First Pass')
ylabel('Magnitude')
xlabel('Time')
%--------------------------------------------------------------------------

%Time Reverse the total recieved signal
TR_t=fliplr(r_total);

%Propagate the reversed signal back to the reciever
%the distances travelled will be the same, so the amplitudes
%and phases should change by the same amount.

%time shift the reversed signal by travel time from reciever to
%source by discarding first time*Fs number of samples
TR_t_shift=zeros(points,length(t));
delay=round(time*Fs);
for g=1:points+1
   TR_t_shift(g,:)=[zeros(1,delay(g)) TR_t(1:length(TR_t)-delay(g))];
end
```

```
%only have length(t)- the maximum shift samples to use

%decrease the amplitude according to the inverse square law
for f=1:points+1
   y_t(f,:)=A_x(f)*TR_t_shift(f,1:length(t));
end

%add up all of the signals recived at the source
y_total=sum(y_t,1);

%normalize the signal
y_norm=y_total/max(y_total);

%--------------Recieved Signals after 2nd Pass------------------
figure(2)
t1=0:1/Fs:(length(t)-1)/Fs;
subplot(2,1,1)
hold on
for k=1:points+1
   switch mod(k,7)
      case 0
         plot(t1,y_t(k,:),'m')
      case 1
         plot(t1,y_t(k,:),'g')
      case 2
         plot(t1,y_t(k,:),'k')
      case 3
         plot(t1,y_t(k,:),'c')
      case 4
         plot(t1,y_t(k,:),'b')
      case 5
         plot(t1,y_t(k,:),'r')
      case 6
         plot(t1,y_t(k,:),'y')
   end
end
title('The Recieved Signals on Second Pass')
ylabel('Magnitude')
xlabel('Time')
hold off

%add each of the recieved signals together
subplot(2,1,2)
plot(t1,y_total);
title('The Total Recived Signal on Second Pass')
ylabel('Magnitude')
xlabel('Time')

%--------------Compare Orignal Signal with Final Signal----------
figure(1)
hold on
plot(t1,x_t(1:length(t)))
plot(t1,fliplr(y_norm),'r')
title('Comparing the sent signal to the recived signal at the Source')
xlabel('Time')
ylabel('Magnitude')
legend('Original Signal','Final Signal')
axis([0 t1(length(t1)) -1 1])
hold off

%----------------------------------------------------------------
```

A.2 Ray Simulation (Double Reflections)

```
%Steven Seiden
%November 23,2003
%Time Reversal Methods in Communications

%point reflectors and a cosine wave transmitted
clear;
points=5;
dimension=25;
factor=.9;
%generate randomly placed point reflectors
x=round(dimension*rand(1,points));
y=round(dimension*rand(1,points)-dimension/2);
reflectors=[x;y]';

%place the receiver and transmitter
source=[0 0];
receiver=[dimension 0];
checkpoint=[0,1];

%---------------------------------Plot Geometry----------------------------
figure(4)
hold on
plot(reflectors(:,1),reflectors(:,2),'o','markeredgecolor','g','markerfacecolor','y')
plot(source(:,1),source(:,2),'<','markeredgecolor','r','markerfacecolor','k')
plot(receiver(:,1),receiver(:,2),'>','markeredgecolor','r','markerfacecolor','k')
line([receiver(1,1) source(1,1)],[receiver(1,2) source(1,2)]);
for i=1:points
   line([reflectors(i,1) source(1,1)],[reflectors(i,2) source(1,2)])
   line([receiver(1,1) reflectors(i,1)],[receiver(1,2) reflectors(i,2)])
end
title('Source, Receiver, and Point Reflectors')
xlabel('X')
ylabel('Y')
legend('Point Reflectors','Source','Reciver','Paths')
axis([-5 dimension+5 -(dimension/2) dimension/2])
hold off
%-------------------------------------------------------------------------
%transmit a cosine wave
Fs=500E6;
%determine the maximum distance for a wave to travel
v=3E8;  %speed of light
maximum=(dimension/2)+sqrt((dimension/2)^2+dimension^2);
tmin=2*maximum/v;
t=0:1/Fs:2*tmin;
fc=5E6;
T=1/fc;
x_t=cos(2*pi*fc*t);

%The recived signal will have a different amplitude
%and phase.  The phase will be determined by time taken
%to reach the receiver divied by the period of the wave
%and then modulus by T. The amplitude will decrease by
%a inverse square law

%calculate the distances travelled first from source
%to receiver
distance1=sqrt((sum([receiver-source].^2,2)));

%----------------------------Single Reflections-------------------------
%find the distances travelled by reflected waves
%first find distance from source to point
distance2=distance2d(source,reflectors);
%then find distance from point to receiver
distance3=distance2d(reflectors,receiver);
```

```
%add the distances together
distance4=distance2+distance3;

%----------------------------Double Reflections-------------------------
%sorts the reflectors from left to right
orderedlr=sort(reflectors);

%calculates the distance from each node to each of the nodes that are to its right
doubledistance=zeros(points-1,points-1);
for i=1:points-1
   for j=1:points-i

doubledistance(j,i)=distance2d(source,orderedlr(i,:))+distance2d(orderedlr(i,:),orderedlr(i+j,:))+distance2d(order
edlr(i+j,:),receiver);
   end
end


%create a column of distances starting with direct path
total_distance=[distance1;distance4;reshape(doubledistance,[],1)];

%get rid of all of the zeros
[dummy1,dummy2,total_distance]=find(total_distance);

%determines the number of signals that can be received assuming only two reflections in the forward direction
signals=length(total_distance);

%Now apply the inverse square law
A_x=1./(4*pi*total_distance.^2);

%each signal will attenuate due to the number of reflections it takes
Ar_x=ones(signals,1);
%No reflections
Ar_x(1)=1;
%One Reflections
Ar_x(2:points+1)=factor;
%Two Reflections
Ar_x(points+2:signals)=factor^2;



%determine the phase differences in radians
time=total_distance./v;
phases=2*pi*mod(time,T)./T;

%the recieved signal
for i=1:signals
   r_t(i,:)=Ar_x(i)*A_x(i)*cos(2*pi*fc*t-phases(i));
end


%add each of the recieved signals together
r_total=sum(r_t,1);

%---------------plot the recieved signals on First Pass------
figure(3)
subplot(2,1,1)
hold on
for k=1:signals

   switch mod(k,7)
      case 0
         plot(t,r_t(k,:),'m')
      case 1
         plot(t,r_t(k,:),'g')
      case 2
         plot(t,r_t(k,:),'k')
      case 3
         plot(t,r_t(k,:),'c')
      case 4
```

```
            plot(t,r_t(k,:),'b')
        case 5
            plot(t,r_t(k,:),'r')
        case 6
            plot(t,r_t(k,:),'y')
    end
end
title('The Recieved Signals on First Pass')
ylabel('Magnitude')
xlabel('Time')
hold off


subplot(2,1,2)
plot(t,r_total);
title('The Total Recived Signal on First Pass')
ylabel('Magnitude')
xlabel('Time')
%-------------------------------------------------------------------------

%Time Reverse the total recieved signal
TR_t=fliplr(r_total);

%Propagate the reversed signal back to the receiver
%the distances travelled will be the same, so the amplitudes
%and phases should change by the same amount.

%time shift the reversed signal by travel time from receiver to
%source by discarding first time*Fs number of samples
TR_t_shift=zeros(signals,length(t));
delay=round(time*Fs);
for g=1:signals
    TR_t_shift(g,:)=[zeros(1,delay(g)) TR_t(1:length(TR_t)-delay(g))];
end

%only have length(t)- the maximum shift samples to use


%decrease the amplitude according to the inverse square law
for f=1:signals
    y_t(f,:)=Ar_x(f)*A_x(f)*TR_t_shift(f,1:length(t));
end

%add up all of the signals recived at the source
y_total=sum(y_t,1);

%normalize the signal
y_norm=y_total/max(y_total);

%--------------Recieved Signals after 2nd Pass------------------
figure(2)
t1=0:1/Fs:(length(t)-1)/Fs;
subplot(2,1,1)
hold on
for k=1:signals
    switch mod(k,7)
        case 0
            plot(t1,y_t(k,:),'m')
        case 1
            plot(t1,y_t(k,:),'g')
        case 2
            plot(t1,y_t(k,:),'k')
        case 3
            plot(t1,y_t(k,:),'c')
        case 4
            plot(t1,y_t(k,:),'b')
        case 5
            plot(t1,y_t(k,:),'r')
        case 6
            plot(t1,y_t(k,:),'y')
```

```
   end
end
title('The Recieved Signals on Second Pass')
ylabel('Magnitude')
xlabel('Time')
hold off

%add each of the recieved signals together
subplot(2,1,2)
plot(t1,y_total);
title('The Total Recived Signal on Second Pass')
ylabel('Magnitude')
xlabel('Time')




%--------------Compare Orignal Signal with Final Signal----------
figure(1)
hold on
plot(t1,x_t(1:length(t)))
plot(t1,fliplr(y_norm),'r')
title('Comparing the sent signal to the recived signal at the Source')
xlabel('Time')
ylabel('Magnitude')
legend('Original Signal','Final Signal')
axis([0 t1(length(t1)) -1 1])
hold off

%----------------------------------------------------------------------
```

A.3 FDTD Code Part 1

```
function [ezx,ezy,hx,hy] =
fdtdsteve(ezx,ezy,hx,hy,alphaex,alphaey,alphamx,alphamy,betaex,betaey,betamx,betamy);

[m,n]=size(ezx);


%advance electric field Ezx
for j=1:n
   for i=1:m-1
      ezx(i,j)=alphaex(i,j)*ezx(i,j)+betaex(i,j)*(hy(i+1,j)-hy(i,j));
   end
end

%advance the electric field Ezy
for j=1:n-1
   for i=1:m
      ezy(i,j)=alphaey(i,j)*ezy(i,j)-betaey(i,j)*(hx(i,j+1)-hx(i,j));
   end
end

%advance the magnetic field Hx
for j=2:n
   for i=1:m
      hx(i,j) = alphamy(i,j)*hx(i,j)-betamy(i,j)*(ezx(i,j)+ezy(i,j)-ezx(i,j-1)-ezy(i,j-1));
   end
end

%advance the magentic field Hy
for j=1:n
   for i=2:m
      hy(i,j) = alphamx(i,j)*hy(i,j)+betamx(i,j)*(ezx(i,j)+ezy(i,j)-ezx(i-1,j)-ezy(i-1,j));
   end
end
```

A.4 FDTD Code Part 2

```
function [E, V, frames] = fdtd05(imax, jmax, del, signal, start, space, array, rstart, rspace, depth, offset, media,
ER)
%   Reverse Time Migration - Modified on August 1, 2002
%
%   Inputs:
%         imax:   size of grid in x-direction (in number of cells)
%         jmax:   size of grid in y-direction (in number of cells)
%         del:    size of cell in meters( needs to be less than lambda/10)
%         signal: matrix with row vectors of signal (Input Signal)
%         space:  the number of cells between transmitters
%         start:  the cell at which the first trasnmitter is located
%         array:  the number of elements in the receiving array
%         rstart: the x-coordinate of the first receiver
%         rspace: the spacing between receivers
%         depth:  the distance between the receiving array and the transmitting array
%         offset: the distance from zero in the x-direction where the simulation will start
%         media: value between 1 and 5 that deterines the type of environment in which the expermient takes
place
%         ER: when media=5, a user defined environment is provided
%
%   Outputs:
%         E: Outputs the final electric field
%         V: Outputs the received signal at certain points (array elements for forward migration, receiver for
reverse migration)
%         frames: a slide show that shows the evolution of E over time.  Can be converted into AVI format using
movie2avi command

% constitutive parameters
e0 = (1e-9)/(36*pi);
u0 = (4e-7)*pi;
c = 1./sqrt(e0*u0);
%The time step is determined by the cell size accroding to the courant stability equation
dt=(1/(c*sqrt((1/del)^2+(1/del)^2)));
%Allows for the input of several signals at a time (tr = array elements, Nt = length of signal)
[tr,Nt] = size(signal);
m = signal;
nb = 32;                                        % PML: number of layers

% Compute total grid size, including PML layers
itotal = imax+2*nb;
jtotal = jmax+2*nb;
er = ones(itotal, jtotal);                      % relative permittivity
ur = ones(itotal, jtotal);                      % relative permeability
order = 3;                                      % order of PML layer number
r = 1e-7;                                       % reflection factor R(0);
smax = log(r)*(-(order+1)/2)*(e0*c)/(nb*del);   % max conductivity

tt = dt:dt:Nt*dt;                               % time vector
sig = 0.00*ones(itotal, jtotal);                % conductivity
sigex = zeros(itotal, jtotal);                  % electric conductivity on left and right PML
sigey = zeros(itotal, jtotal);                  % electric conductivity on upper and lower PML

%Determines the type pf propagation that is occuring and declares variables that help map the environment
if space==1;  %forward Propagation
   center=rstart+((array-1)/2)*rspace;
   edge1=rstart;
   edge2=rstart+(array-1)*rspace;
   length=(offset+depth);
   left=offset;
   right=offset+depth;
end

if rspace==1; %Reverse Propagation
   center=start+((tr-1)/2)*space;
   edge1=start;
   edge2=start+(tr-1)*space;
   length=offset;
   left=offset+depth;
```

```
   right=offset;
end

%random points used to define the environment with LOS or no LOS propagation
pointsy1=round((jmax-3)*[ 0.7271    0.3093    0.8385    0.5681    0.3704    0.7027    0.5466    0.4449    0.6946
0.6213])+3;
pointsy2=round((jmax-3)*[  0.1365    0.0118    0.8939    0.1991    0.2987    0.6614    0.2844    0.4692    0.0648
0.9883])+3;
pointsx1=round((imax/2-10)*[ 0.7948    0.9568    0.5226    0.8801    0.1730    0.9797    0.2714    0.2523
0.8757    0.7373])+3;
pointsx2=round((imax/2-10)*[0.5828    0.4235    0.5155    0.3340    0.4329    0.2259    0.5798    0.7604    0.5298
0.6405])+(imax/2+8);
points=[[pointsx1 pointsx2]' [pointsy1 pointsy2]'];
scatter=20;

%Determines which type of environment to run the simulation under
switch media
   case 1 %waveguide
      er(nb+edge1-2,nb+(0:length-1))=100*ones(1,length);
      er(nb+edge2+2,nb+(0:length-1))=100*ones(1,length);
   case 2  %Line of Sight
      for i=1:scatter
         er(nb+points(i,1)-2:nb+points(i,1)+2,nb+points(i,2)-2:nb+points(i,2)+2)=100*ones(5,5);
      end
   case 3    %No Line of Sight
      er(nb+center-4:nb+center+4,nb+left+(right-left)/2-4:nb+left+(right-left)/2+4)=100*ones(9,9);
      for i=1:scatter
         er(nb+points(i,1)-2:nb+points(i,1)+2,nb+points(i,2)-2:nb+points(i,2)+2)=100*ones(5,5);
      end
   case 4   % Free Space
   case 5   %User Input
      if nargin==13
         er(nb:nb+imax-1,nb:nb+jmax-1)=ER;
      end
end

% Electric conductivity
for i=1:nb,
  sigex(i,:) = smax*((nb-i+1)/nb)^order;
  sigey(:,i) = smax*((nb-i+1)/nb)^order;
  sigex(imax+nb+i, :) = smax*(i/nb)^order;
  sigey(:, jmax+nb+i) = smax*(i/nb)^order;
end;
sigmx = sigex*u0./e0;                          %magnetic conductivity on left and right PML
sigmy = sigey*u0./e0;                          %magnetic conductivity on upper and lower PML

% Coefficients for Maxwell's equations
alphaex = ones(itotal, jtotal);
alphaey = ones(itotal, jtotal);
alphamx = ones(itotal, jtotal);
alphamy = ones(itotal, jtotal);
betaex = ones(itotal, jtotal);
betaey = ones(itotal, jtotal);
betamx = ones(itotal, jtotal);
betamy = ones(itotal, jtotal);

%input boundary coefficients for PML
%***********************************************
alphaex = exp(-(sigex*dt)./e0);                  %left,right
alphaey = exp(-(sigey*dt)./e0);                  %upper,lower
alphamx = exp(-(sigmx*dt)/u0);                    %left,right
alphamy = exp(-(sigmy*dt)/u0);                    %upper,lower

% left component
betaex(1:nb,:) = (1-alphaex(1:nb,:))./(er(1:nb,:).*sigex(1:nb,:)*del);
% right component
betaex(imax+nb+1:itotal,:) = (1-
alphaex(imax+nb+1:itotal,:))/(er(imax+nb+1:itotal,:).*sigex(imax+nb+1:itotal,:)*del);

% upper component
```

```
betaey(:,1:nb) = (1-alphaey(:,1:nb))./(er(:,1:nb).*sigey(:,1:nb)*del);
% lower component
betaey(:,jmax+nb+1:jtotal) = (1-
alphaey(:,jmax+nb+1:jtotal))/(er(:,jmax+nb+1:jtotal).*sigey(:,jmax+nb+1:jtotal)*del);

% left component
betamx(1:nb,:) = (1-alphamx(1:nb,:))./(sigmx(1:nb,:)*del);
% right component
betamx(imax+nb+1:itotal,:) = (1-alphamx(imax+nb+1:itotal,:))/(sigmx(imax+nb+1:itotal,:)*del);

% upper component
betamy(:,1:nb) = (1-alphamy(:,1:nb))./(sigmy(:,1:nb)*del);
% lower component
betamy(:,jmax+nb+1:jtotal) = (1-alphamy(:,jmax+nb+1:jtotal))/(sigmy(:,jmax+nb+1:jtotal)*del);

% Input coefficiences in main FDTD grid
%************************************************
dtdu = dt./(ur*u0*del);
dtde = dt./(er*e0*del);
loss = 1 - ((sig*dt)./(er*e0));

alphaex(nb+1:imax+nb,:) = loss(nb+1:imax+nb,:);
alphaey(:,nb+1:jmax+nb) = loss(:,nb+1:jmax+nb);
alphamx(nb+1:imax+nb,:) = 1;
alphamy(:,nb+1:jmax+nb) = 1;
betaex(nb+1:imax+nb,:) = dtde(nb+1:imax+nb,:);
betaey(:,nb+1:jmax+nb) = dtde(:,nb+1:jmax+nb);
betamx(nb+1:imax+nb,:) = dtdu(nb+1:imax+nb,:);
betamy(:,nb+1:jmax+nb) = dtdu(:,nb+1:jmax+nb);
% Initialize electromagnetic fields
ezx = zeros(itotal,jtotal);
ezy = zeros(itotal,jtotal);
hx = zeros(itotal,jtotal);
hy = zeros(itotal,jtotal);

% Start the main loop for FDTD calculations
%******************************************************
% The outer loop enters each step, one at a time, while the inner loop inputs that
% step for each trace.
for j = 1:Nt,
       for i = 1:tr;
         % enter the signal from the last time step to the first (reversed in time)
         %The Electric field is initiated by creating a magnetic loop
           hx((i-1)*space+nb+start,offset+nb)   =  signal(i,j);
           hy((i-1)*space+nb+start,offset+nb)   = -signal(i,j);
           hx((i-1)*space+nb+start,offset+nb+1) = -signal(i,j);
           hy((i-1)*space+nb+start+1,offset+nb) =  signal(i,j);
           ezy((i-1)*space+nb+start,offset+nb)=0;
           ezx((i-1)*space+nb+start,offset+nb)=0;
       end;

  % Advance the fields using code 'fdtdsteve'
    [ezx,ezy,hx,hy] =
fdtdsteve(ezx,ezy,hx,hy,alphaex,alphaey,alphamx,alphamy,betaex,betaey,betamx,betamy);
    %The extra layers from the boundary conditions are removed
    E=(ezx+ezy);
    [a,b] = size(E);
    E = E(33:(a-32), 33:(b-32));
       %The desired signal is recorded
    V(1:array,j)=[E(rstart:rspace:rstart+(array-1)*rspace,offset+depth)];
    %The current electric field is plotted and saved for the movie
    colormap(jet)
    imagesc(E)
    caxis([-100 100])
    axis image
    frames(j)=getframe;
 end;
%#########################################################################
```

A.5 GUI Code

```
function varargout = TimeReversalMirror_11(varargin)
%Time Reversal Mirror Version 1.1
%Developed by Steven Seiden Spring 2004
%
%Electromagnetic simulation to test the ideas behind time reversal methods for communications.
%The program is complete graphics based and allows users to control many different aspects of the program.
%The sizae of the simulation, the placement of the recevier and array can be defined.  Several preset
%enviroments can be chosen, or the user can input their own enviroment aby hand.  The user can test
%three types of digital modulation, ASK, PSK, and FSK.  The full signal constellation is trasnmitted to the
%array.  The array reversed the signal in time and stores it as the new prototype signal constellation.
%The user can then transmit the prototype signals back to the receiver through the same enviroment.
%Throughout the process, the electric field intensity is plotted on a log scale.  The entire process can
%be recorded and then converted to an AVI file.  The EM simulation is performed using the FDTD algorithm
%With PML boundary conditions that were coded by Mary Alt.


%---------------------------------------GUI-----------------------------------------------------------

% TIMEREVERSALMIRROR_11 Application M-file for TimeReversalMirror_11.fig
%    FIG = TIMEREVERSALMIRROR_11 launch TimeReversalMirror_11 GUI.
%    TIMEREVERSALMIRROR_11('callback_name', ...) invoke the named callback.

% Last Modified by GUIDE v2.0 12-Apr-2004 18:55:29

if nargin == 0  % LAUNCH GUI

	fig = openfig(mfilename,'reuse');

	% Generate a structure of handles to pass to callbacks, and store it.
	handles = guihandles(fig);
	guidata(fig, handles);

	if nargout > 0
		varargout{1} = fig;
	end

elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK

	try
		if (nargout)
			[varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
		else
			feval(varargin{:}); % FEVAL switchyard
		end
	catch
		disp(lasterr);
	end

end


%| ABOUT CALLBACKS:
%| GUIDE automatically appends subfunction prototypes to this file, and
%| sets objects' callback properties to call them through the FEVAL
%| switchyard above. This comment describes that mechanism.
%|
%| Each callback subfunction declaration has the following form:
%| <SUBFUNCTION_NAME>(H, EVENTDATA, HANDLES, VARARGIN)
%|
%| The subfunction name is composed using the object's Tag and the
%| callback type separated by '_', e.g. 'slider2_Callback',
%| 'figure1_CloseRequestFcn', 'axis1_ButtondownFcn'.
%|
%| H is the callback object's handle (obtained using GCBO).
%|
%| EVENTDATA is empty, but reserved for future use.
%|
%| HANDLES is a structure containing handles of components in GUI using
```

```
%| tags as fieldnames, e.g. handles.figure1, handles.slider2. This
%| structure is created at GUI startup using GUIHANDLES and stored in
%| the figure's application data using GUIDATA. A copy of the structure
%| is passed to each callback.  You can store additional information in
%| this structure at GUI startup, and you can change the structure
%| during callbacks.  Call guidata(h, handles) after changing your
%| copy to replace the stored original so that subsequent callbacks see
%| the updates. Type "help guihandles" and "help guidata" for more
%| information.
%|
%| VARARGIN contains any extra arguments you have passed to the
%| callback. Specify the extra arguments by editing the callback
%| property in the inspector. By default, GUIDE sets the property to:
%| <MFILENAME>('<SUBFUNCTION_NAME>', gcbo, [], guidata(gcbo))
%| Add any extra arguments after the last argument, before the final
%| closing parenthesis.

%=======================VARIABLES=====================================

% -------------------------X Dimension--------------------------
function varargout = X_Callback(h, eventdata, handles, varargin)
handles.x= str2double(get(h,'string'));
guidata(h,handles);
% -------------------------Y Dimension--------------------------
function varargout = Y_Callback(h, eventdata, handles, varargin)
handles.y = str2double(get(h,'string'));
guidata(h,handles);
% -------------------------Recevier in X--------------------------
function varargout = receiver_Callback(h, eventdata, handles, varargin)
handles.receiver=str2double(get(h,'string'));
guidata(h,handles);
% --------------------Location of first array element---------
function varargout = arraystart_Callback(h, eventdata, handles, varargin)
handles.arraystart = str2double(get(h,'string'));
guidata(h,handles);
% --------------------# of elements in the array----------
function varargout = elements_Callback(h, eventdata, handles, varargin)
handles.elements = str2double(get(h,'string'));
guidata(h,handles);
% --------------------Sapcing between arrayelements------
function varargout = spacing_Callback(h, eventdata, handles, varargin)
handles.arrayspace = str2double(get(h,'string'));
guidata(h,handles);
% -----------------distance between array and recevier-----
function varargout = distance_Callback(h, eventdata, handles, varargin)
handles.depth = str2double(get(h,'string'));
guidata(h,handles);
% -----------------offset from x=0 and recevier----------
function varargout = offset_Callback(h, eventdata, handles, varargin)
handles.offset = str2double(get(h,'string'));
guidata(h,handles);
% --------------------Shape of hte array----------------------
function varargout = shape_Callback(h, eventdata, handles, varargin)
handles.shape=get(h,'value');
guidata(h,handles);
% -------------------environment-------------------------
function varargout = random_media_Callback(h, eventdata, handles, varargin)
handles.media=get(h,'value');
guidata(h,handles);
% -----------------movie name-----------------------------
function varargout = movie_name_Callback(h, eventdata, handles, varargin)
handles.movie_name=get(h,'string');
guidata(h,handles);
% --------------------convert movie to AVI---------------------
function varargout = avi_Callback(h, eventdata, handles, varargin)
frames1=handles.frames1;
frames2=handles.frames2;
movie_name=handles.movie_name;
movie2avi([frames1 frames2],movie_name)
```

```
%==========================PROCESSING================================

% ----------------------------First Transmission--------------------
function varargout = ping_Callback(h, eventdata, handles, varargin)
X=handles.x;
Y=handles.y;
depth=handles.depth;
offset=handles.offset;
arraystart=handles.arraystart;
arrayspace=handles.arrayspace;
elements=handles.elements;
receiver=handles.receiver;
media=handles.media;
shape=handles.shape;

if media==5
   er=handles.er;
else
   er=0;
end


fc=1E9;
c=3E8;
lambda=c/fc;
%length in meters of each cell
del=lambda/10;
dt=(1/(c*sqrt((1/del)^2+(1/del)^2)));
Fs=1/dt;
handles.del=del;
ping_signal=handles.send_data;

%plot the signal to transmit
axes(handles.axes4);
plot(ping_signal);
set(handles.signal,'string','Pinging Signal sent from the Receiver to the Array');
media=handles.media;

%Transmit the original signal from the recevier
axes(handles.axes1)
[E1, V1, frames1] = fdtd05(X, Y, del, ping_signal, receiver, 1, elements, arraystart, arrayspace, depth, offset,
media,er);
handles.E1=E1;
handles.V1=V1;
handles.frames1=frames1;
set(handles.signal,'string','Recorded Ping at the Array Elements')
axes(handles.axes4)
plot(1:length(V1),V1)
guidata(h,handles);

% ----------------------------Time Reversal------------------
function varargout = TR_Callback(h, eventdata, handles, varargin)
elements=handles.elements;
V1=handles.V1;
send_data=handles.send_data;
depth=handles.depth;
arrayspace=handles.arrayspace;



%find the delay amount for each array element
for k=1:elements
   delay(k)=min(find(abs(V1(k,:))>1E-3));
end

%d_hat represents the delay from the start of the simulation to the time when the first element records a signal
d_hat=min(delay);
%determines the delay for each signal relative to the first one
delay=delay-d_hat;
```

```
stop=d_hat+120;
if stop>length(V1)
   stop=length(V1);
end

%create the signal to send to the receiver
for k=1:elements
   temp2(k,:)=[ zeros(1,round(2*depth)) V1(k,d_hat:stop)];
end

%Flip the signal and normalize it to 1
send=fliplr(temp2/max(max(abs(temp2))));
handles.send=send;
axes(handles.axes4)
plot(1:length(send),send)
set(handles.signal,'string','Time Reversed signal to be sent');
guidata(h,handles)

% ----------------------------Perform the modulation---------------------------------
function varargout = modulation_Callback(h, eventdata, handles, varargin)
depth=handles.depth;
fc=1E9;
c=3E8;
lambda=c/fc;
%length in meters of each cell
del=lambda/10;
dt=(1/(c*sqrt((1/del)^2+(1/del)^2)));
Fs=1/dt;
t=0:1/Fs:29/Fs-1/Fs;

%Regular Sine wave without modulation
source=sin(2*pi*fc*t);
ping_signal=[source];

value=get(h,'value');
data=[1 0];
if value==1 % FSK
   send_data=dmod(data,fc,Fs/30,Fs,'fsk');
   set(handles.signal,'string','1 0 FSK Digitally Modulated Bits')
elseif value==2 % ASK
   send_data=dmod(data,fc,Fs/30,Fs,'ask');
   set(handles.signal,'string','1 0 ASK Digitally Modulated Bits')
elseif value==3 %PSK
   send_data=dmod(data,fc,Fs/30,Fs,'psk');
   set(handles.signal,'string','1 0 PSK Digitally Modulated Bits')
elseif value==4 %No Modulation
   send_data=ping_signal;
   set(handles.signal,'string','The Original Pining Signal (No Modulation)')
end

%Form the original signal to be trasnmitted
send_data=[send_data zeros(1,round(2*depth))];
handles.send_data=send_data;
axes(handles.axes4)
plot(send_data)

guidata(h,handles);

% -------------------------Re-Trasnmit the Time Revered Signal-----------------------------
function varargout = transmit_Callback(h, eventdata, handles, varargin)
X=handles.x;
Y=handles.y;
del=handles.del;
send=handles.send;
arraystart=handles.arraystart;
arrayspace=handles.arrayspace;
receiver=handles.receiver;
depth=handles.depth;
offset=handles.offset;
```

```
media=handles.media;
shape=handles.shape;
frames1=handles.frames1;

if media==5
   er=handles.er;
else
   er=0;
end

%Transmit
axes(handles.axes1);
[E2, V2, frames2] = fdtd05(X, Y, del,send, arraystart, arrayspace, 1, receiver, 1, -depth, offset+depth,media,er);
handles.frames2=frames2;

maximum=length(frames1)+length(frames2);

slider_step(1) = 1/(maximum-1);
slider_step(2) = 1/(maximum-1);

set(handles.slider1,'sliderstep',slider_step,'max',maximum,'min',1,'Value',1)

%Plot the Recevied signal
set(handles.signal,'string','The Received Signal from the Array')
axes(handles.axes4)
plot(V2)
guidata(h,handles)

%----------------------Show the Simulation Set-Up-----------------------------------
function varargout=array_setup_Callback(h, eventdata, handles, varargin)
X=handles.x;
Y=handles.y;
receiver=handles.receiver;
elements=handles.elements;
arraystart=handles.arraystart;
arrayspace=handles.arrayspace;
depth=handles.depth;
offset=handles.offset;
media=handles.media;
shape=handles.shape;


%Determine where the array is?
center=arraystart+((elements-1)/2)*arrayspace;
edge1=arraystart;
edge2=arraystart+(elements-1)*arrayspace;
length=(offset+depth);
left=offset;
right=offset+depth;

%Define the realtive permativitty of each point in the simulation
er=ones(X,Y);
pointsy1=round((Y-3)*[ 0.7271    0.3093    0.8385    0.5681    0.3704    0.7027    0.5466    0.4449    0.6946
0.6213])+3;
pointsy2=round((Y-3)*[  0.1365    0.0118    0.8939    0.1991    0.2987    0.6614    0.2844    0.4692    0.0648
0.9883])+3;
pointsx1=round((X/2-10)*[ 0.7948    0.9568    0.5226    0.8801    0.1730    0.9797    0.2714    0.2523    0.8757
0.7373])+3;
pointsx2=round((X/2-10)*[0.5828    0.4235    0.5155    0.3340    0.4329    0.2259    0.5798    0.7604    0.5298
0.6405])+(X/2+8);
points=[[pointsx1 pointsx2]' [pointsy1 pointsy2]'];

scatter=20;

%Provide the desired environment
switch media
   case 1 %waveguide
      er(edge1-2,(1:length))=100*ones(1,length);
      er(edge2+2,(1:length))=100*ones(1,length);
   case 2  %line of Sight
```

```
    for i=1:scatter
        er(points(i,1)-4:points(i,1)+4,points(i,2)-4:points(i,2)+4)=100*ones(9,9);
    end
  case 3    %No Line of Sight
    er(center-4:center+4,left+(right-left)/2-4:left+(right-left)/2+4)=100*ones(9,9);
    for i=1:scatter
        er(points(i,1)-4:points(i,1)+4,points(i,2)-4:points(i,2)+4)=100*ones(9,9);
    end
  case 4   % Free Space
  case 5   %User Defined
    figure(1)
    plot(offset,receiver,'marker','<','markerfacecolor','y','markeredgecolor','b','linestyle',':', 'markersize', 5)
    hold on
    plot((offset+depth)*ones(1,elements),arraystart:arrayspace:arraystart+(elements-1)*arrayspace,...
    'marker','>','markerfacecolor','y','markeredgecolor','b','linestyle',':','markersize', 5)

    axis equal
    axis([1 Y 1 X])
    title('Place Reflectors with Left Mouse Button, Press Right Mouse Button When Finished')
    xlabel('Y')
    ylabel('X')
    view([0 270])

    figure(1)
    check=1;
    grid on
    grid minor

    %Allow the user to place their own reflectors
    while check ==1
        [x,y,button]=ginput(1);
        if button==3
            check=0;
        else
            plot(round(x),round(y),'marker','square','markerfacecolor','r','markeredgecolor','k','markersize', 5)
            axis([1 Y 1 X])
            er(round(y),round(x))=1000;
        end
      end
    hold off
    close

  end

handles.er=er;
guidata(h,handles)



axes(handles.axes2)

%Plot the simulation space
colormap(jet)
imagesc(er)
caxis([0 100])
hold on
plot(offset,receiver,'marker','<','markerfacecolor','y','markeredgecolor','b','linestyle',':', 'markersize', 5)
plot((offset+depth)*ones(1,elements),arraystart:arrayspace:arraystart+(elements-1)*arrayspace,...
'marker','>','markerfacecolor','y','markeredgecolor','b','linestyle',':','markersize', 5)

axis([1 Y 1 X])
xlabel('Y')
ylabel('X')
axis image
hold off


%================================MENU FUNCTIONS=======================

% --------------------------------------------------------------------
```

```
function save_Callback(hObject, eventdata, handles)
% hObject    handle to save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function save_array_pattern_Callback(hObject, eventdata, handles)
% hObject    handle to save_array_pattern (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function save_array_setup_Callback(hObject, eventdata, handles)
% hObject    handle to save_array_setup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
axes(handles.axes2)
print -dmeta simulation_setup

% --------------------------------------------------------------------
function save_movie_Callback(hObject, eventdata, handles)
% hObject    handle to save_movie (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function print_Callback(hObject, eventdata, handles)
% hObject    handle to print (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function print_array_setup_Callback(hObject, eventdata, handles)
% hObject    handle to print_array_setup (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
print -f handles.axes2

% --------------------------------------------------------------------
function print_array_pattern_Callback(hObject, eventdata, handles)
% hObject    handle to print_array_pattern (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
print  -f handles.axes3

% --------------------------------------------------------------------
function close_Callback(hObject, eventdata, handles)
% hObject    handle to close (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)

% --------------------------------------------------------------------
function varargout = edit11_Callback(h, eventdata, handles, varargin)

% --------------------------------------------------------------------
function varargout = slider1_Callback(h, eventdata, handles, varargin)
frames1=handles.frames1;
frames2=handles.frames2;
slides=[frames1 frames2];
slider_value = get(handles.slider1,'Value');
axes(handles.axes1)
image(frame2im(slides(slider_value)));
axis image
set(handles.edit11,'string',num2str(slider_value))
%--------------------------------------------------------------------
```

Bibliography

[1] Theodore S. Rappaport. **Wireless Communications; Principles and Practices, 2nd Ed.** Prentice Hall, New Jersey, 2002.

[2] Bernard Sklar. **Digital Communications; Fundamentals and Applications, 2nd Ed.** Prentice Hall, New Jersey, 2001.

[3] Richard Plumb, Mark Fowler, and Edward Li. *"Electromagnetic time reversal methods in imaging and communications."* White Paper, March 13th, 2003.

[4] Simon Haykin. **Communications Systems, 4th Ed.** John Wiley & Sons, Inc., New York, 2001.

[5] M. Fink. *"Time reversal mirror."* Acoustic Imaging, B.F. Jones, Ed., New York: Plenum, 1995, vol. 25, pp. 1-15.

[6] M. Stojanovic, J.A. Capitovic, and J.G. Proakis. *"Phase-coherent digital communications for underwater acoustic channels."* IEEE Journal of Oceanic Engineering, vol. 19, pp 100-11, Jan. 1994.

[7] M. Stojanovic, J.A. Capitovic, and J.G. Proakis. *"Adaptive multichannel combining and equalization for underwater acoustic communications."* Journal of the Acoustic Society of America, vol. 94, no.3, pp. 1621-31, 1993.

[8] Geoffrey F. Edelmann, T. Akal, William S. Hodgkiss, Seongil Kim, William A. Kuperman, and Hee Chun Song. *"An initial demonstration of underwater acoustic communication using time reversal."* IEEE Journal of Oceanic Engineering, vol. 27, no. 3. pp 602-9, July 2002.

[9] Carlton J. Leuschen and Richard G. Plumb. *"A matched filter based reverse-time migration algorithm for ground-penetrating radar data."* IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 5, pp 929-36, May 2001.

[10] M. Born and E. Wolf. **Principles of Optics**. Academic Press, New York, 1970.

[11] B. Steinberg. **Microwave Imaging with Large Antenna Arrays**. Wiley, New York, 1983.

[12] Liliana Borcea, George Papanicolaou, ChrysoulaTsogka, and James Berryman. *"Imaging and time reversal in random media."* 2002. ftp://math.stanford.edu /pub/papers/papanicolaou/ip-paper5.pdf Veiwed April 20th, 2004.

[13] João Gomes and Victor Barroso. *"Time-reversed communications over Doppler-spread underwater channels."* 2002-IEEE-International-Conference-on-Acoustics,-Speech,-and-Signal-Processing.-Proceedings-Cat.-No.02CH37334. 2002: III-2849-52 vol.3.

[14] Karl S. Kunz, Raymond J. Luebbers. **Finite Difference Time Domain Method for Electromagnetics**. CRC Press, Boca Raton, 1993.

[15] Allen Taflove. **Computational Electrodynamics: The Finite-Difference Time-Domain Method**. Artech House, Boston, 1995.

[16] Michael W. Buksas. *"The Perfectly Matched Layer Absorbing Boundary Condition for Maxwell's Equations"* Slides from presentation in T-7 Numerical Anlysis Seminar. May 31, 2000.

[17] João Gomes and Victor Barroso. *"The performance of sparse time-reversal mirrors in the context of underwater communications."* Proceedings-of-the-Tenth-IEEE-Workshop-on-Statistical-Signal-and-Array-Processing-Cat.-No.00TH8496. 2000: 727-31.

[18] A.J. Devaney. *"Super-resolution processing of multi-static data using time reversal and MUSIC."* Preprint, 1999. http://www.ece.neu.edu/faculty/devaney/preprints/paper02n_00.pdf, Viewed April 20th, 2004.

[19] Stewart A. Levin. *"Principle of reverse time migration."* http://sepwww.stanford.edu /oldreports/sep37/37_05.pdf Viewed April 20th, 2004.