To The Graduate Council:

I am submitting herewith a thesis written by Luaie L. Al-BaBa entitled "Adaptive Echo Cancellation Using Genetic Algorithm". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Dr. Paul Benjamin Crilly, Major Professor

We have read this thesis and recommend its acceptance:

Accepted for the Council:

Associate Vice Chancellor and
Dean of The Graduate School

ADAPTIVE ECHO CANCELLATION USING A GENETIC ALGORITHM

A Thesis Presented

for the

Master of Science Degree

The University of Tennessee, Knoxville

Luaie A. Al-BaBa

December 1993

ACKNOWLEDGMENTS

In writing acknowledgments, I thank the following people for their time and patience. I thank Dr. Paul Crilly who is my thesis advisor in the Electrical and Computer Engineering Department (ECE) at the University of Tennessee Knoxville. I thank him for his time, patience, and sympathetic ear on numerous occasions.

I thank Dr. Daniel Koch in the ECE Department at UTK for using the equipment in the CISP Lab.

I thank Steven Horner who is a Ph.D student in the ECE Department at UTK. I thank him for his comments, suggestions, and the use of several of his C programs.

I thank Dr. Herbert Neff in the ECE Department for his encouragement in undergraduate education and him being on the committee.

I also thank my spouse Elizabeth Ann and my son Zachariah for their patience.

Finally, I thank the BellSouth Foundation for research support from August 1990 to July 1991.

ABSTRACT

It is well known that processing waveforms containing echoes is a nontrivial problem. This problem arises in such situations as telephone communications, high fidelity recording, and seismic data processing. In such cases, one is interested in detecting echoes and removing their presence. In this thesis, a method is introduced which uses the Hilbert Transform (HT) and Genetic Algorithm (GA) to obtain echo cancellation.

The Hilbert Transform and Genetic Algorithm method (HTGA) is based upon detecting the echo position and eliminating it with a near optimal subtraction. The detection of the echo is done with the Hilbert Transform and the cancellation is implemented via the GA. Simulated data will be presented to demonstrate that this method can suppress an echo or multiple echoes. Finally, an adaptive filter using the least-mean-square (LMS) algorithm will be implemented in place of the GA and a subjective comparison between the two optimization methods will be made.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
	Problem	1
	Thesis Goals	4
II.	LITERATURE REVIEW	6
	Echo Cancellation by Impulse Response	6
	Generalized Linear Filtering	7
	Multimicrophone Signal-Processing Technique to Remove Echo	9
	Adaptive Echo Cancellation by Sondhi	10
III.	HILBERT TRANSFORM, GENETIC ALGORITHM, AND ADAPTIVE FILTERING	11
	Hilbert Transform Theory	11
	Hilbert Transform Properties	12
	Genetic Algorithm Theory	13
	Optimization in Genetic Algorithm	13
	Overview of Genetic Algorithm	15
	Genetic Algorithm Properties	17
	Genetic Algorithm Operators	18
	A Genetic Algorithm Example	22
	Adaptive Filter Theory	25
IV.	PROPOSED SOLUTION	26
	Tone Signal	26
	Hilbert Transform	27
	Genetic Algorithm	28

V. EXPERIMENTAL CONSIDERATIONS 30
Overview 30
Echo Signal 31
Hilbert Transform and Location 31
Genetic Algorithm and Sectioning 31
Results 33

VI. RESULTS AND DISCUSSION 43

VII. COMPARISON WITH LMS ADAPTIVE FILTERING .. 45

VIII. CONCLUSION 53

LIST OF REFERENCES 55

APPENDICES 58

APPENDIX A. HILBERT TRANSFORM TABLE 59

APPENDIX B. ECHO SIGNAL DESIGN 61

APPENDIX C. ECHO LOCATION DESIGN 64

APPENDIX D. GA PROGRAM AND C CODE 67

APPENDIX E. THE MAIN HTGA PROGRAM 84

APPENDIX F. LMS ADAPTIVE FILTERING 87

VITA 90

LIST OF FIGURES

FIGURE		PAGE
1.	Simulated tone signal with a single echo.	2
2.	Simulated tone signal with two echoes.	3
3.	Single-peak decision space.	14
4.	Double-peak decision space.	16
5.	Block diagram for the HTGA method of echo cancellation.	30
6.	The sectioning procedure for echo cancellation.	32
7.	Simulated tone signal. (a) Signal with echo. (b) The Hilbert Transform output	34
8.	Simulated tone signal. (a) Signal with two echoes. (b) The Hilbert Transform output	35
9.	Simulated data for echo cancellation with no	

overlapping. (a) Tone signal. (b) Enhanced echo signal.. 37

10. Simulated tone signal. (a) The original signal. (b) Original signal with overlapping echo. (c) The HTGA output 38

11. Simulated tone signal. (a) The original signal. (b) Original signal with overlapping echo. (c) The HTGA output 39

12. Simulated tone signal. (a) The original signal. (b) Original signal with overlapping echo. (c) The HTGA output 40

13. Simulated tone signal. (a) Signal with two echoes. (b) The HTGA output. 41

14. Simulated tone signal. (a) Two signal with two echoes. (b) The HTGA output. 42

15. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of

the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1. 46

16. Simulated tone signal. (a) Simulated tone signal with partialy overlapping echo. (b) Simulated data of he original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1. 47

17. Simulated tone signal. (a) Simulated data for continuously increasing frequency partially overlapping an echo signal. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1. 48

18. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 5. 49

19. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 0.05..... 50

LIST OF TABLES

TABLE PAGE

1. MSE values for both signal with echo and HTGA enhanced signal. 44

2. MSE values for signal with echo, HTGA enhanced signal, and LMS enhanced signal. 52

CHAPTER I

INTRODUCTION

Problem

When a conversation takes a place, the speech signals can be disrupted by the presence of nearby reflecting objects. Speech signals travel not only in a direct path from source to receiver, but also travel to the receiver after reflecting off one or more objects. In some cases the signal will reflect from one object, causing a single echo as shown in Figure 1. In other cases, the signal will bounce off several objects creating multiple echoes as shown in Figure 2.

The human ear can detect echoes from speech waves when they reflect off walls, floors, and surrounding objects [1]. For a small time delay between the speech and the echo, the echo is not detected by the human ear and is perceived as a distortion. The ear will detect an echo if the delay exceeds approximately ten milliseconds [1]. Processing waveforms containing echoes is a problem that appears in situations such as telephone communications, high fidelity recording, and sometimes even in simple conversations[2]. This can be unpleasant to the listener and under certain conditions can disrupt a conversation.

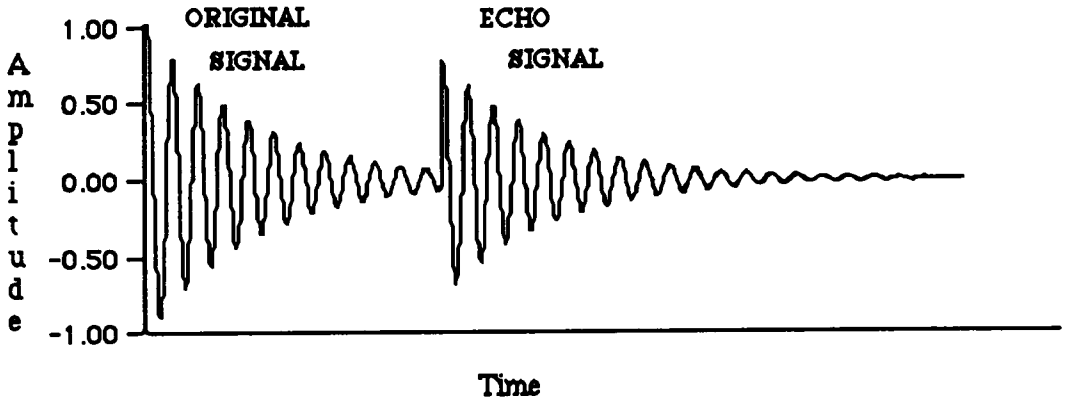


Figure 1. Simulated tone signal with a single echo.

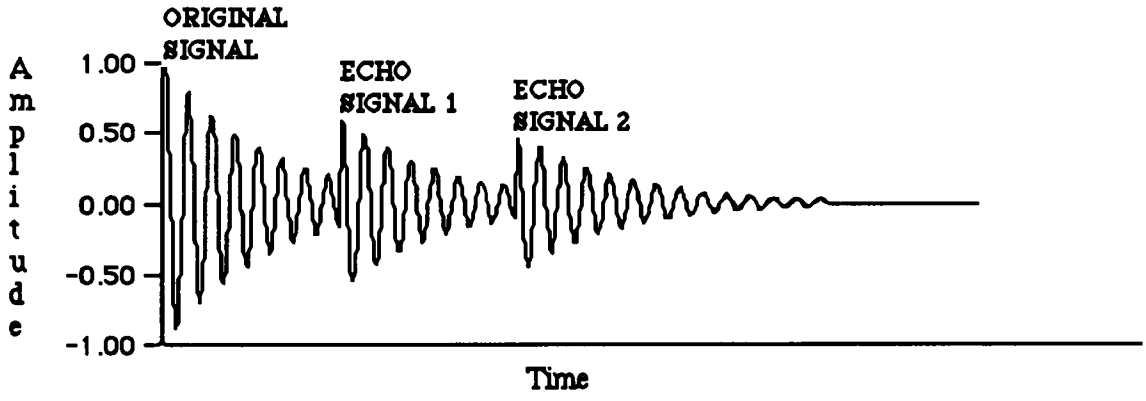


Figure 2. Simulated tone signal with two echoes.

Thesis Goals

There are three main goals for this thesis. The first goal is to use the Hilbert transform (HT) and the genetic algorithm (GA) to suppress an echo signal. The GA is used because of its ability to find the global minimum, even in the case of multiple minima, of a solution space.

The second goal consists of testing the Hilbert transform and genetic algorithm (HTGA) on a simulated tone signal. The simulated data will include a tone signal that has one nonoverlapping echo, a continuously increasing frequency signal with one-quarter overlapping echo signal, a continuously increasing frequency signal with one-half overlapping echo signal, a continuously increasing frequency signal with extreme overlapping echo signal, a tone signal with two echoes and little overlap, and a signal with an echo followed by another signal after the first echo dies out which is followed by its echo.

Finally, the third goal of this thesis is to determine if the solution space has multiple minima. This will be accomplished by using an adaptive filter algorithm called least-mean-square (LMS). If the LMS algorithm produces a similar solution as the genetic algorithm, then it can be concluded that the solution space has a global minimum. The LMS algorithm is based on a gradient technique that finds only the closest minimum to its initial weight factor. To make absolutely sure that the solution space has a global minimum, several

different initial weight factors will be tried. If the LMS algorithm converges to a similar solution as the GA, then it is very likely the solution space has a global minimum. But if the LMS algorithm converges to a different solution than the GA, then the LMS finds a local minima. This indicates that the solution space has multiple minima.

CHAPTER II

LITERATURE REVIEW

Echo signal cancellation is a problem in various signal processing applications. The problem arises in such cases as telephone communications, high fidelity recording, and sometimes even in a room conversation. Many approaches have been used to solve the problem. This chapter will discuss several systems that have been used in room echo cancellation and echo cancellation in long distance telephone communications.

Echo Cancellation by Impulse Response

The impulse response method is used to suppress room effects which can be preceived as an echo. The room effects can be viewed as a convolution between the time domain representation of a speech signal and the room impulse response[3]. For a minimum phase impulse response, this method uses an inverse filter that should be able to remove the room's effect completely. One way to find out if the impulse response is minimum phase or not is by using the Nyquist plot. The above method uses the Nyquist plot to detect the presence of non minimum phase zeros.

In the above, by using a point image source to solve the room effects, synthetic room impulses were created to investigate the

efficiency of the minimum phase inverse filter. Some of the synthetic room impulse responses were found to be minimum phase when the initial delay was removed. If this case is true, then using a minimum phase inverse filter is good enough to remove the room effect from a disrupted speech signal.

This method has resulted in successful outputs in some cases. For a given room size, fixed source, and receiver locations, there was a threshold for the reflectivity. The room impulse had a non minimum phase when the reflectivity was increased beyond the threshold value[3]. The threshold was not constant for a given room size.

Generalized Linear Filtering

The generalized linear filtering technique has been developed by Schafer[4]. This method of processing signals containing echoes does not require a precise knowledge of the signal or the echo. This system obeys a generalized superposition principle. These systems are called homomorphic systems in which the transformations involved are algebraically linear.

The generalized linear filtering method treats the signal and its echo as a convolution problem[1]. The homomorphic deconvolution method is used to remove the echo signal. The homomorphic deconvolution consists of three systems. The first system is called the characteristic system. This system converts a convolution into a sum

of corresponding signals. The characteristic system consists of three transformations. The first transformation consists of taking the Z-transform of the convolved signals. This will result in the product of the two signals. The second transformation consists of taking the complex logarithm of the product of the signals that give the sum of the logarithm of the Z-transforms. The third transformation consists of taking the inverse Z-transforms of the terms. The mathematical development for the characteristic system can be explained by starting with a simple echo signal.

$$Te(n) = T(n) + aT(n-n_i) \tag{1}$$

The first transformation consists of taking the Z-transform of Equation (1) as shown in Equation (2).

$$T_E(Z) = T(Z)(\ 1 + aZ^{-ni}) \tag{2}$$

The second transformation of the characteristic system is obtained by taking the complex logarithm of Equation (2).

$$\hat{T}_E(Z) = log\ [T(Z)] + log\ [1+aZ^{-ni}] \tag{3}$$

For a < 1 Equation (3) can be written as

$$\hat{T}_E(Z) = log\ [T(Z)] + \Sigma(-1)^{n+1}\ (a^n/n)Z^{-nin} \tag{4}$$

The third transformation is obtained from the inverse of the Z-transform of Equation (4) which in most cases produces only the peak of the echo.

The second system is a linear system that in most cases is a frequency invariant linear system. The third system is the inverse of the characteristic system. This system converts the sum into a convolution.

The generalized linear filtering method for removing echo signals is an effective approach. It does not require any information about the size of the echo, but approximate knowledge of the echo delay is needed in the linear system. The peak of the echo shows in most of the cases after enhancement.

Multimicrophone Signal-Processing Technique to Remove Echo

This method is based on using a multimicrophone digital processing strategy for removing the room distortion. For untreated rooms, a distortion is produced by the reflection of a signal. The distortion is characterized by a hollow echo-like quality[4]. The signal of each microphone is divided into frequency bands and the delay differences of the corresponding output signal are compensated and added. From the cross correlation between corresponding microphone signals in the band, the gain for each resulting band is set.

The operation is equivalent to a linear time varying filter having two input ports and one output port.

The above method was tested and it was found to be effective in removing the long-term echo from speech.

Adaptive Echo Canceller by Sondhi

This method is used in long distance telephone communications to remove the echo signal. The procedure used in this method consists mainly of subtracting a synthesized replica of the echo signal from the returning one. Synthesis of the replica is accomplished by using an adaptive filter that is controlled by a feedback loop that is determined by a group of simultaneous nonlinear first-order differential equations[5]. This filter adapts to the transmission characteristic of the echo path and keeps up with the variations of the echo path that may occur during a conversation.

The above method for canceling echoes in telephone connections is feasible and successful. However the process needs a period of time in which only the echo is present in the return signal.

CHAPTER III

HILBERT TRANSFORM, GENETIC ALGORITHM, AND ADAPTIVE FILTERING

Hilbert Transform Theory

The Hilbert transform is simply a filter that phase-shifts all frequency components of its input by -1/2 π radians[6]. There are many applications in communication where it is necessary to delay all the (real) positive frequency components of a signal by 90 degrees[7]. The Hilbert transform transfer function is

$$H(f) = -j\ sgn(f) \tag{5}$$

where the sgn function is

$$sgn = \begin{cases} 1 & , f > 0 \\ 0 & , f = 0 \\ -1 & , f < 0 \end{cases} \tag{6}$$

For an input spectrum of $X(f)$, the output spectrum should be $-j\ sgn(f)\ X(f)$. The corresponding time function $\hat{x}(t)$ is

$$\hat{x}(t) = F^{-1}\{\ -j\ sgn(f)\ X(f)\ \} \tag{7}$$

$$= [-j\, F^{-1} \{ sgn(f) \}] * x(t) \tag{8}$$

The impulse response h(t) of the filter is

$$h(t) = -j\, F^{-1} \{ sgn(f) \} \tag{9}$$

The Hilbert transform $\hat{x}(t)$ after few steps[6] is

$$\hat{x}(t) = \int [x(t')/\pi(t-t')]dt' \tag{10}$$

Hilbert Transform Properties

Some properties of the Hilbert transform are

1. Both the signal $x(t)$ and its Hilbert transform $\hat{x}(t)$ have equal energy.
2. A signal and its Hilbert transform are orthogonal, that is, the integral of the product of the signal and its Hilbert transform is zero.

$$\int x(t)\ \hat{x}(t)\ \ dt = 0 \tag{11}$$

3. If $x(t)$ is a low pass signal and $y(t)$ is a high pass signal with nonoverlapping spectra then

$$h[\, x(t)y(t) \,] = x(t)h[\, y(t) \,] \tag{12}$$

A table of Hilbert transform pairs can be found in Appendix A.

Genetic Algorithm Theory

Genetic algorithms were developed by John Holland at the University of Michigan. Genetic algorithms are mainly search algorithms. They are based on genetics to efficiently search for a new optimal solution[8]. GA's achieve the solution in a randomized fashion. The randomized procedure of the GA's are not a simple random procedure. GA's are very effective in various applications such as optimization[8]. This chapter will present a detailed overview of the genetic algorithm and compare the GA with traditional methods.

Optimization in Genetic Algorithm

Optimization is one of the applications where the GA has been very effective. Optimization improves performance toward some optimal point. In many optimization methods, the movement from a single point in a decision space to the next is based on some transition rule to determine the next point. These deterministic rules will work for a single-peak decision space as shown in Figure 3. The

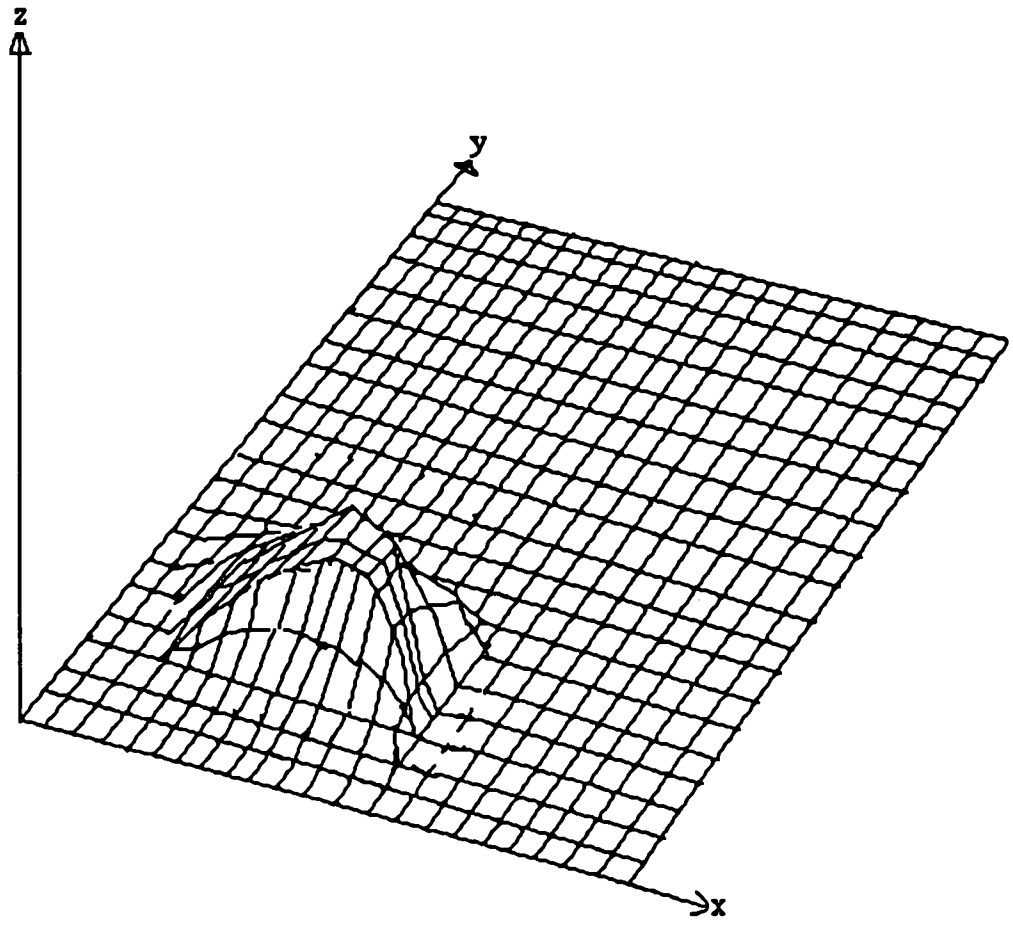


Figure 3. Single-peak decision space.

optimum point achieved in Figure 3 is the best in a neighborhood of points around that one peak. But for a decision space with more than one peak as in Figure 4, the deterministic rules will not work. Starting the search in the neighborhood of the lower peak will find an optimum point in the lower peak and miss the higher peak. In most deterministic procedures, once the lower peak is reached, the search is completed. On the other hand, GA's work from a rich database of points simultaneously climbing many peaks in parallel. By using the randomized technique of parallel hill-climbing, the probability of finding a false peak is reduced.

Overview of Genetic Algorithm

The mechanics of genetic algorithms are surprisingly simple. The procedure is no more complex than copying strings and swapping partial strings of data. The operation of the GA starts by processing a number of strings of data, in which each string is a potential solution. The group of strings is called a population. The generalized strings in the population are randomly created. After the strings are created, they are tested in a designated function that is provided by the creator. The designated function is called a fitness function. A set of genetic operations is designed to take the initial population and create a new one with better strings. Each new string is called a generation. The process goes on until the best population is

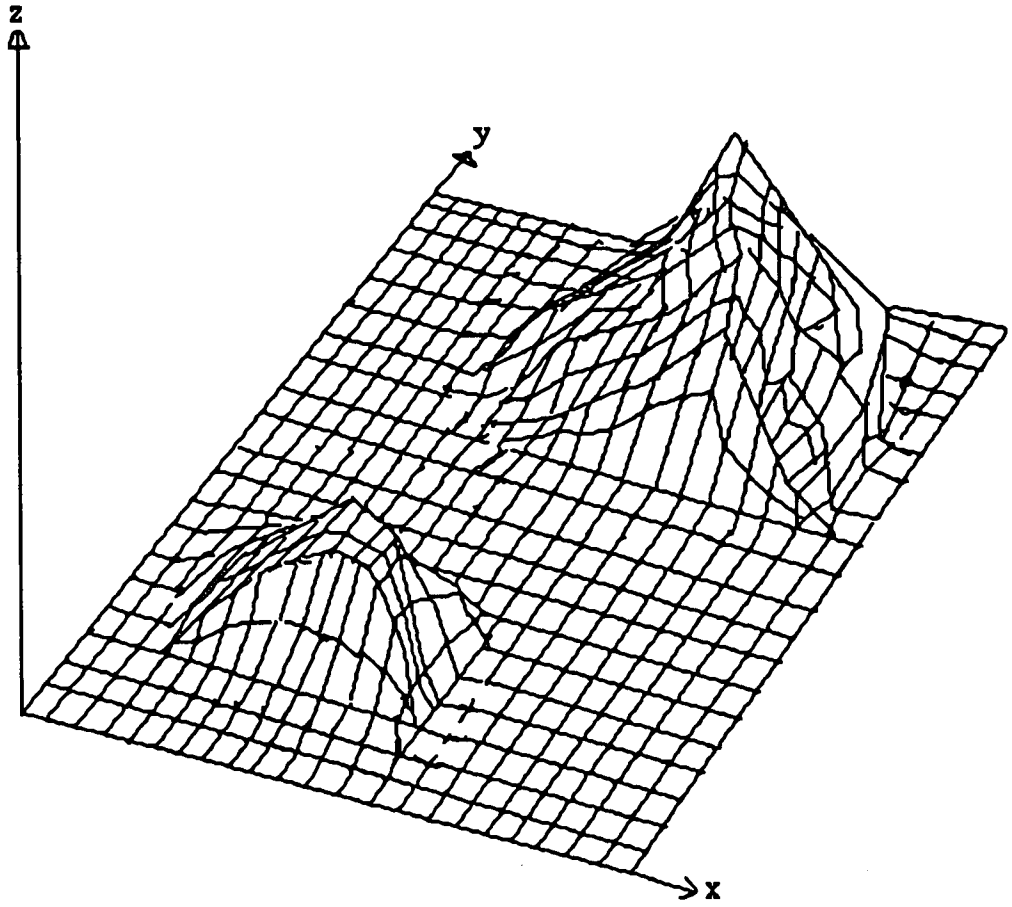


Figure 4. Double-peak decision space.

reached which is associated with an optimum solution according to the designated function.

Genetic Algorithm Properties

GA properties are what make the GA different in very fundamental ways from traditional methods. There are four properties in which GA's are different from normal optimization and searching procedures[8].

1. GA's work with variables coded in strings.
2. GA's search from a population of strings, not a single string.
3. GA's use objective functions, not auxiliary knowledge.
4. GA's use probabilistic transition rules, not deterministic rules.

The operation of the GA requires the variable to be coded as a finite string. All of these coded variables are concatenated to form a string of data. Binary coding is used in the following example that has a population size of four and a string size of five.

Population	
Index	String
1)	10110
2)	01101
3)	00111
4)	00101

Genetic Algorithm Operators

The GA uses operators to find better individual strings within a population. The GA operators consist of reproduction, crossover, and mutation.

Reproduction is the first GA operator. It is a process where the best individual strings in a population are picked to establish a new population. The best strings are picked according to their quality function value or fitness function value. The higher the value a string has, the higher the probability it will contribute to the next generations. The reproduction procedure can be implemented in various ways such as the biased roulette wheel or the pairing procedure. In the biased roulette wheel, each current string in the population has a wheel slot sized in proportion to its fitness. The higher a value the string has, the bigger the wheel slot size it will have, which will make it easier to be picked off the wheel. In the

pairing procedure, numbers of strings are randomly paired in the population. The number of pairs is the same as the population size. A population with size four should have a set of four pairs. The best fitness value of each pair is chosen to be a string in a new population. The following example should give a better picture of the pairing procedure.

The following example has a population size of 4 and a string size of 5 with a fitness function of $f(x) = x^2$. The pairing between strings is done randomly.

	Initial Population				New Population		
Index	String	String Value	Fitness	Pairing	String	String Value	Fitness
1)	01111	15	225	1-2	11001	25	625
2)	11001	25	625	3-4	10111	23	529
3)	00101	05	025	1-3	01111	15	225
4)	10111	23	529	2-4	11001	25	625

Crossover is the second operator. It consists of two steps. The first step randomly divides the population into pairs. The second step is crossover in each pair according to the crossover position assigned randomly to the string. Each position in a string has a crossover probability where in most cases it is set to 0.6. The position is

randomly picked between one and the string size less one as shown below.

S_1 and S_2 are two strings randomly picked from a population. The string size is 5.

S_1 11010

S_2 00111

A crossover position is randomly picked between 1 and 4. 3 is picked to be the crossover position.

S_1 110 | 10

S_2 001 | 11

Crossover yields two new strings in a new generation.

S'_1 11011

S'_2 00110

Where S'_1 and S'_2 are the new strings.

Mutation is the third and last GA operator. It plays a secondary role in the genetic operation. Even though reproduction and crossover are effective in recombining and searching genetic material, the two operators still can lose some of the genetic material. That is where the mutation operator operates against the loss. Mutation

complements the value of a digit in the string. It changes a 0 to a 1 or a 1 to a 0. Each position in a string has a mutation probability. Mutation probability is usually small and most of the time set to 0.1. Thus the frequency of mutation is very small. The expected number of mutations *Em* in one generation can be calculated as

$$Em = (Pm)(Np)(Ns) \tag{13}$$

where *Pm* is the mutation probability, *Np* the population size, and *Ns* the string size.

The following example has a mutation probability of 0.2, a population size of 2, and a string size of 8. The expected number of mutations that will occur in the current generation is

$$Em = (0.2)(2)(8) \approx 3 \tag{14}$$

S_1 01100101

S_2 11010001

S_1 and S_2 are the strings in a population. The underlined digits in the strings are randomly chosen for mutation. The result is new strings that are created as seen in $S'_1$and S'_2.

S'_1 00100101

S'_2 11011000

A Genetic Algorithm Example

Before starting the example, a review of the GA steps are presented: a) the members of the population should be encoded to form a string, b) the fitness function should be applied for each string, c) the GA operators are applied to create a new generation, d) the third step is repeated until a specified number is achieved to find the best solution.

For this example, the string size is 5, the population size is 4, the mutation probability is 0.2, the number of generation is 2, and the fitness function is $f(x) = x^2$.

Generation 1.

Initial Population				Reproduction	
String Index	String	String Value	Fitness	Pairing	String
1)	01001	09	081	1-2	10101
2)	10101	21	441	3-4	11100
3)	00110	06	036	1-3	01001
4)	11100	28	784	2-4	11100

	Crossover Randomly Selected		Mutation Randomly Selected	
String	Crossover Site	Crossover	Mutation Site	Mutation
1010 1	Y/4	10100	2	11100
1110 0	Y/4	11101	3	11001
01 001	Y/2	01100	1	11100
11 100	Y/2	11001	4	11011

String Value	Fitness Function
28	784
25	625
28	784
27	729

Generation 2.

Initial Population				Reproduction	
String Index	String	String Value	Fitness	Pairing	String
1)	11100	28	784	1-2	11100
2)	11001	25	625	3-4	11100
3)	11100	28	784	1-3	11100
4)	11011	27	729	2-4	11011

	Crossover Randomly Selected		Mutation Randomly Selected	
String	Crossover Site	Crossover	Mutation Site	Mutation
11 100	N/2	11100	5	11101
11 100	N/2	11100	4	11110
111 00	Y/3	11111	....	11111
110 11	Y/3	11000	3	11100

String Value	Fitness Function
29	841
30	900
31	961
28	748

The best fitness value in generation 1 is 784. In generation 2 the best answer is the string with code 11111 which has a fitness value of 961. The value has improved in running two generations.

Adaptive Filter Theory

An adaptive filter can be used in place of the GA to obtain echo cancellation for comparison purposes. The least-mean-square (LMS) algorithm is used to determine the best scale factor to cancel the echo. The LMS can be implemented by using the following equation

$$K_{n+1} = K_n + 2\mu Er_n E(n) \tag{15}$$

where K is a weight vector or scale factor, μ is the gain constant that changes the speed and stability of converge, Er_n is the error signal, n is the time index, and $E(n)$ is the vector input[9].

CHAPTER IV

PROPOSED SOLUTION

The proposed solution in this thesis for echo cancellation consists of two steps. The first step consists of detecting and locating the echo signal using the Hilbert Transform. The second step uses the Genetic Algorithm to cancel the echo signal.

Tone Signal

A tone signal has been developed for simulation in the HTGA method for echo cancellation. A tone signal can be modeled as $T(n)$. Let $Te(n)$ represent a signal and an echo as shown in Equation (16).

$$Te(n) = T(n) + E(n) \tag{16}$$

where $T(n)$ is the original tone signal and $E(n)$ is the echo of $T(n)$. $E(n)$ can be represented as in Equation (17).

$$E(n) = \sum_{i=1}^{J} a_i \, T(n - n_i) \tag{17}$$

The summation term represents *1* to *j* echoes delayed in time by n_i.

Hilbert Transform

The Hilbert Transform is an effective tool for detecting and locating an echo signal[10]. The following explains the HT method for detecting an echo location by using the analytic signal $T(t)$.

$$T(t) = Ae(^{-t/T})cos(\omega_O t) \tag{18}$$

The Hilbert Transform of $T(t)$ is

$$T_h(t) = -Ae(^{-t/T})sin(\omega_O t) \tag{19}$$

where A is the amplitude, ω_O is the resonant frequency, and T is the decay constant. Let $T_A(t)$ be the magnitude for $T(t)$ and $T_h(t)$ where $T(t)$ is defined as the real part and $T_h(t)$ as the imaginary part. The natural logarithm of $T_A(t)$ is

$$ln|T_A(t)| = ln|\, T(t) + jTh(t)| \tag{20}$$

$$= ln|\, Ae(^{-t/T})cos(\omega_O t) - jAe(^{-t/T})sin(\omega_O t)| \tag{21}$$

$$= ln|\, Ae(^{-t/T})\,| + ln|\, cos(\omega_O t) - jsin(\omega_O t)| \tag{22}$$

$$= ln\, A - t/T \tag{23}$$

By plotting Equation (23), the graph reveals a discontinuity that indicates the location of the echo as shown in Chapter V.

Genetic Algorithm

The Genetic Algorithm will eliminate an echo signal $E(n)$ by using a near optimal subtraction.

$$Te(n) = T(n) + E(n) \tag{24}$$

The near optimal subtraction occurs by scaling $T(n)$ to minimize the absolute value of the sum of the differences between $Te(n)$ and $T(n)$ for all points in time. The near optimal subtraction is feasible if the original signal and the echo signal are identical in shape.
Let

$$X(t) = Te(n) - K\,T(n - n_i) \tag{25}$$

where K is a scaling factor. Equation (25) can be written as

$$X(t) = T(n) + [\,E(n) - KT(n - n_i)\,] \tag{26}$$

and the fitness function can be written as

$$F = (1/N) \left(\sum_{n=n_i}^{N} | [E(n) - KT(n - n_i)] + T(n) | \right) \qquad (27)$$

where N is the number of samples. The GA can minimize Equation (27) quickly by efficient selection of scale factors. As mentioned in the previous chapter, in order for a GA to have a near optimal solution, the individual has to be coded and a fitness function has to be created. In this case the scaling factor K is coded as an individual string. The fitness function F is created to determine the fitness value for each scale factor K. F for this problem needs to be minimized. When the fitness function becomes as small as possible, the echo signal $E(n)$ will be eliminated as much as possible from the signal $Te(n)$.

CHAPTER V

EXPERIMENTAL CONSIDERATIONS

Overview

This chapter studies the implementation of the HTGA method for echo cancellation. Figure 5 demonstrates the HTGA method for echo cancellation. A signal processing package called LabVIEW from National Instruments was used to implement this project. The first step in the project was to create a tone signal with one or more echo signals. The second step consisted of using the HT to detect the echo signal. By graphing the HT of the signal, the echo can be made visible. The graph will show a discontinuity which indicates the echo signal. A location program was designed to find the location of the discontinuity which is the starting point of the echo signal. After the location of the echo signal is found, if the echo overlaps the signal then the data is broken into sections where each section is treated as a separate signal in the GA in order to cancel the echo.

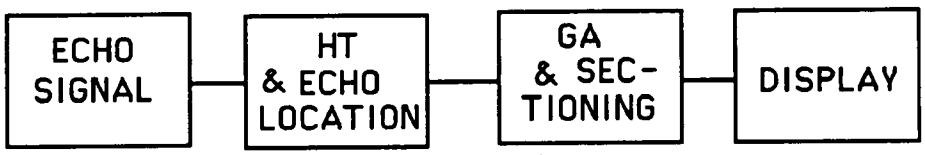


Figure 5. Block diagram for the HTGA method of echo cancellation.

Echo Signal

This project used a tone signal for simulation. The signal was created using the LabVIEW signal processing package. The design is shown in Appendix B. The signal parameters such as amplitude (for signal and echo), frequency, echo delay, and decay value are changeable.

Hilbert Transform and Location

After the echo signal is created, the signal and its echo are input to the HT icon. As explained in Chapter IV in the HT section, after taking the Hilbert transform of both the signal and the echo, the natural logarithm of the magnitude of the signal and echo is taken. By plotting the natural logarithm, a discontinuity will be visible. The discontinuity is the starting point of the echo signal. The location of the discontinuity is found by using the echo location as seen in Appendix C.

Genetic Algorithm and Sectioning

The GA code was written using C software as seen in Appendix D[11]. Figure 6 demonstrates the sectioning procedure. The procedure is used only if the signal and the echo are extremely overlapped. Otherwise, the sectioning procedure is not used.

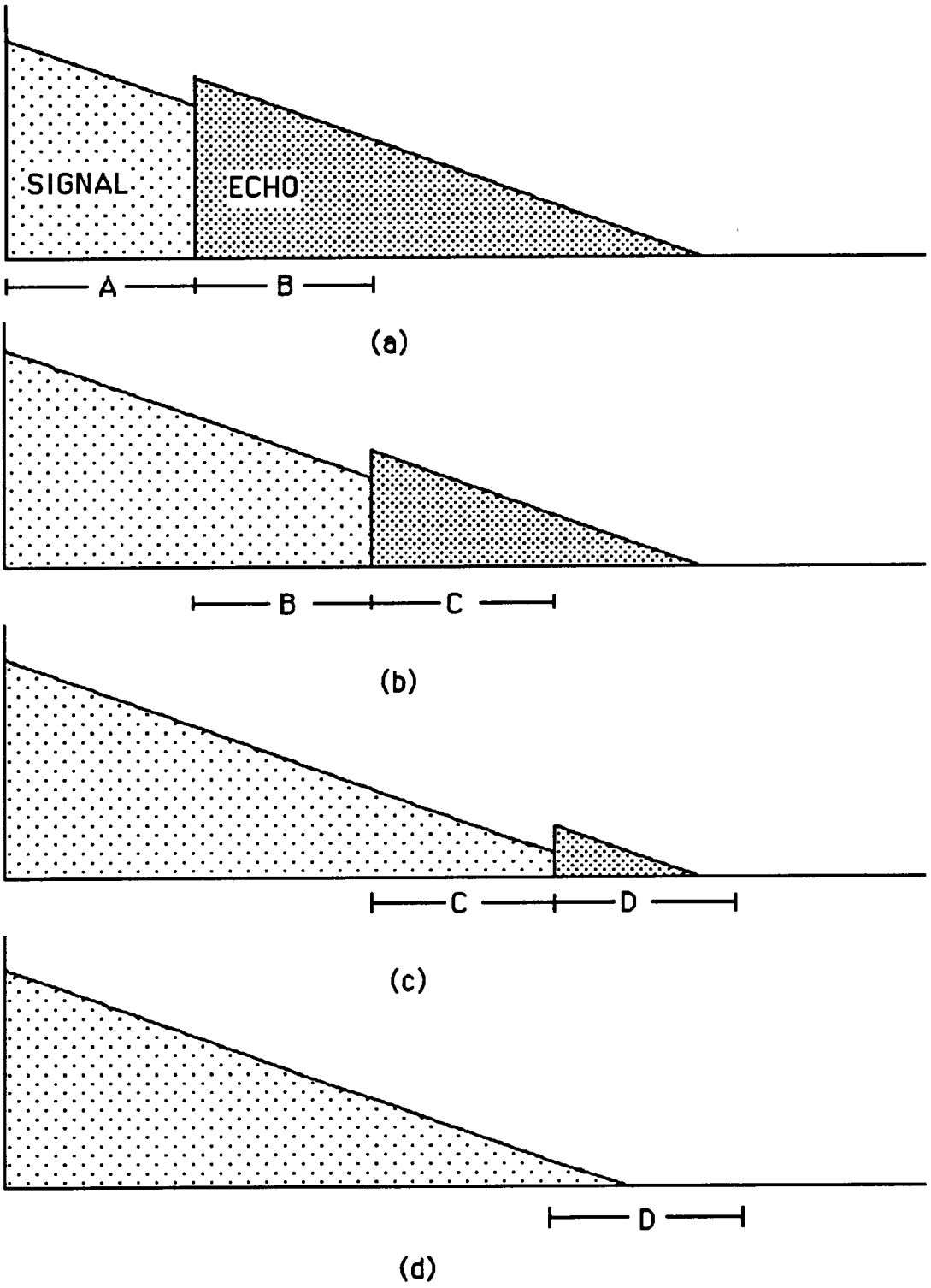


Figure 6. The sectioning procedure for echo cancellation.

Figure 6 (a) represents a pure tone signal (section A) and an echo. The length A is the location of the echo signal. The same length of section A was taken in the echo signal (section B). The GA is used to cancel the echo in section B using the signal in section A as seen in Figure 6(b). The same procedure is done in Figure 6(b). Section B in Figure 6(b) is used as the pure signal to cancel the echo in section C by applying the GA as seen in Figure 6(c). In Figure 6(c), the GA uses the pure signal in section C to cancel the echo in section D. The procedure goes on until the echo signal is canceled.

The HTGA design is shown in Appendix E. The design consists of echo signal generation, HT application, echo location design, and the GA application.

Results

In this section, there are eight figures to demonstrate the simulated data. In most of the figures, there is a signal and an echo signal in the upper part of the figure and the enhanced signal in the lower part of the figure.

Figures 7(a) and 8(a) show a simulated tone signal which has an echo. Figures 7(b) and 8(b) are graphs of the output of the Hilbert Transform where the discontinuity is an indication of the echo position. Figures 7(b) and 8(b) show some distortion at the end of the signal. This is where the HT method diverges because the amplitude

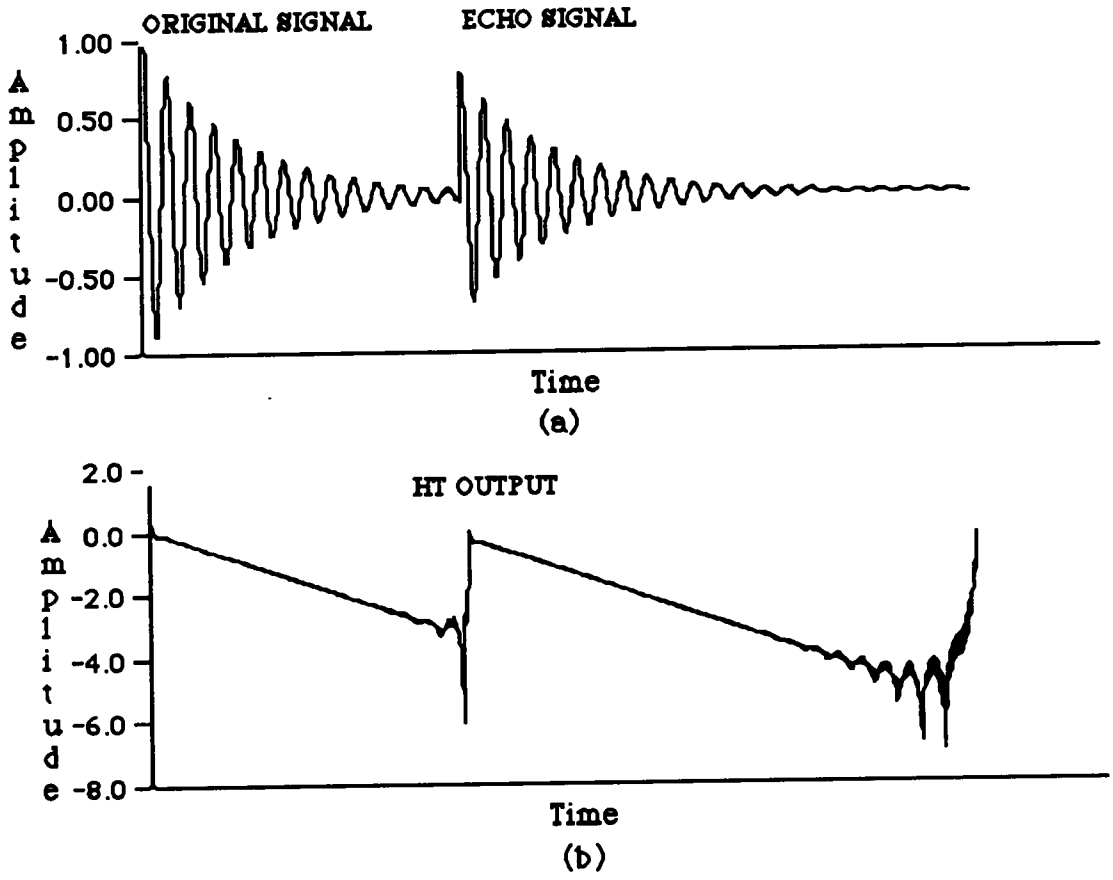


Figure 7. Simulated tone signal. (a) Signal with echo. (b) The Hilbert Transform output.

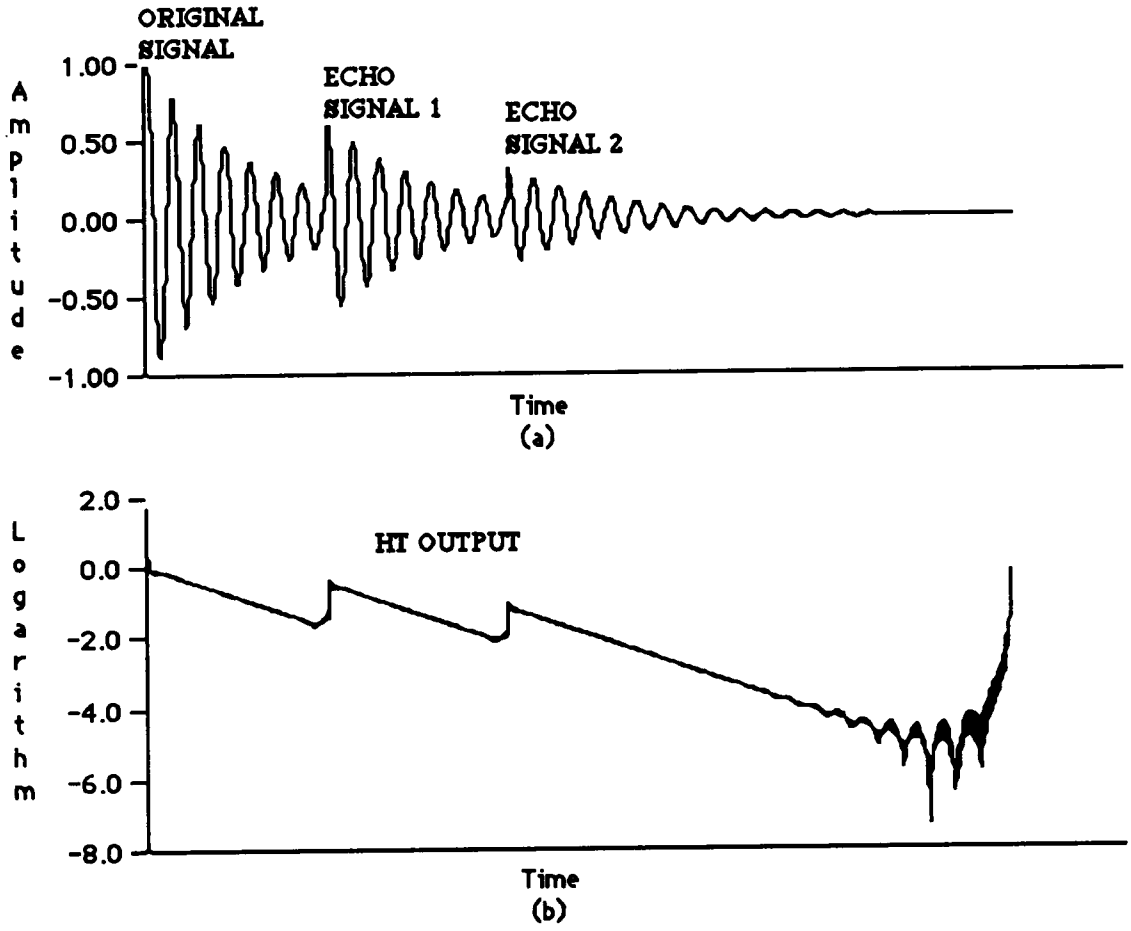


Figure 8. Simulated tone signal. (a) Signal with two echoes. (b) The Hilbert Transform output.

of the signal in Figures 7(a) and 8(a) is decreasing and taking the natural logarithm of a small number like zero results in infinity.

Figures 9 through 14 demonstrate the effectiveness of the HTGA method. Figures 9 through 13 are broken into two cases. The first case is for non-overlapping signals and the second case is for overlapping signals.

Figure 9 demonstrates the HTGA for nonoverlapping signals. Figure 9(a) shows signal plus echo and Figure 9(b) shows signal after the echo has been eliminated.

Figures 10 through 13 demonstrate the HTGA for overlapping echo signals. Figures 10 through 12 demonstrate the HTGA method for a continuously increasing frequency partial overlapping echo signal. Figure 10(a) shows the original signal before the echo signal is added. Figure 10(b) shows the signal plus echo and Figure 10(c) shows the result after the echo has been canceled from the signal appearing in Figure 10(b). Notice how similar Figure 10(a) and Figure 10(c) appear to be.

Figures 11 and 12 are laid out similarly, but the echo delay is small in Figure 11 compared to 10 and even smaller in Figure 12. Figure 13 shows the HTGA for a case where there are multiple echoes. Finally, Figure 14 shows the case where there are two signals and each has overlapping echoes.

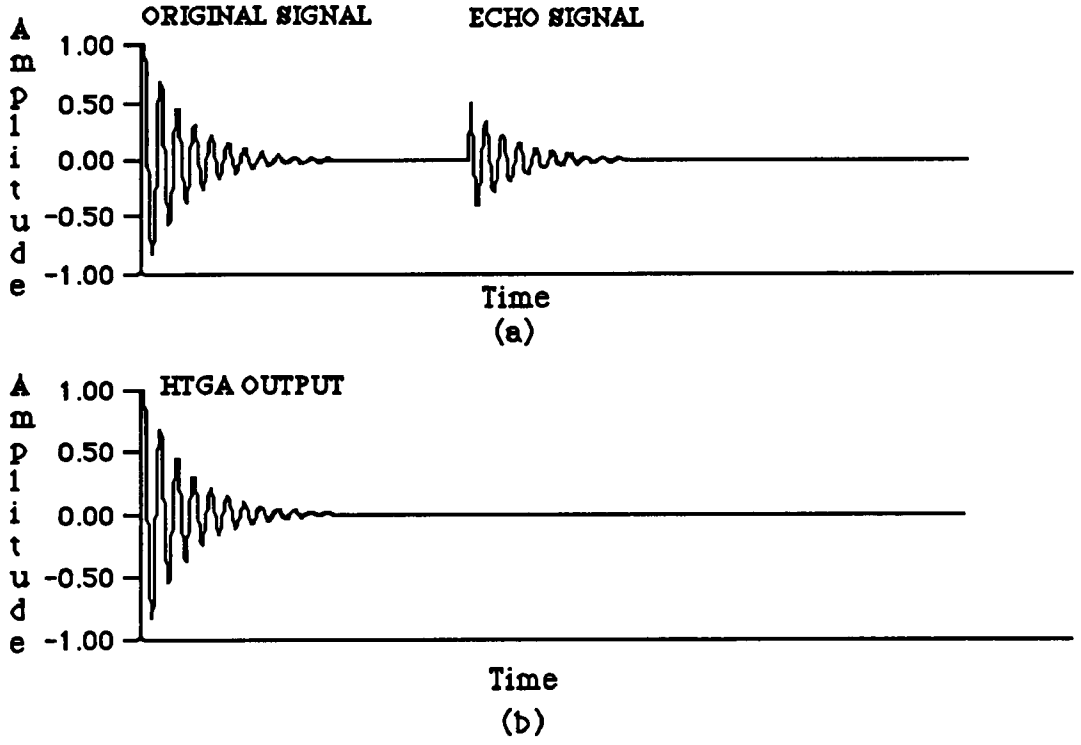


Figure 9. Simulated data for echo cancellation with no overlapping. (a) Tone signal. (b) Enhanced echo signal.

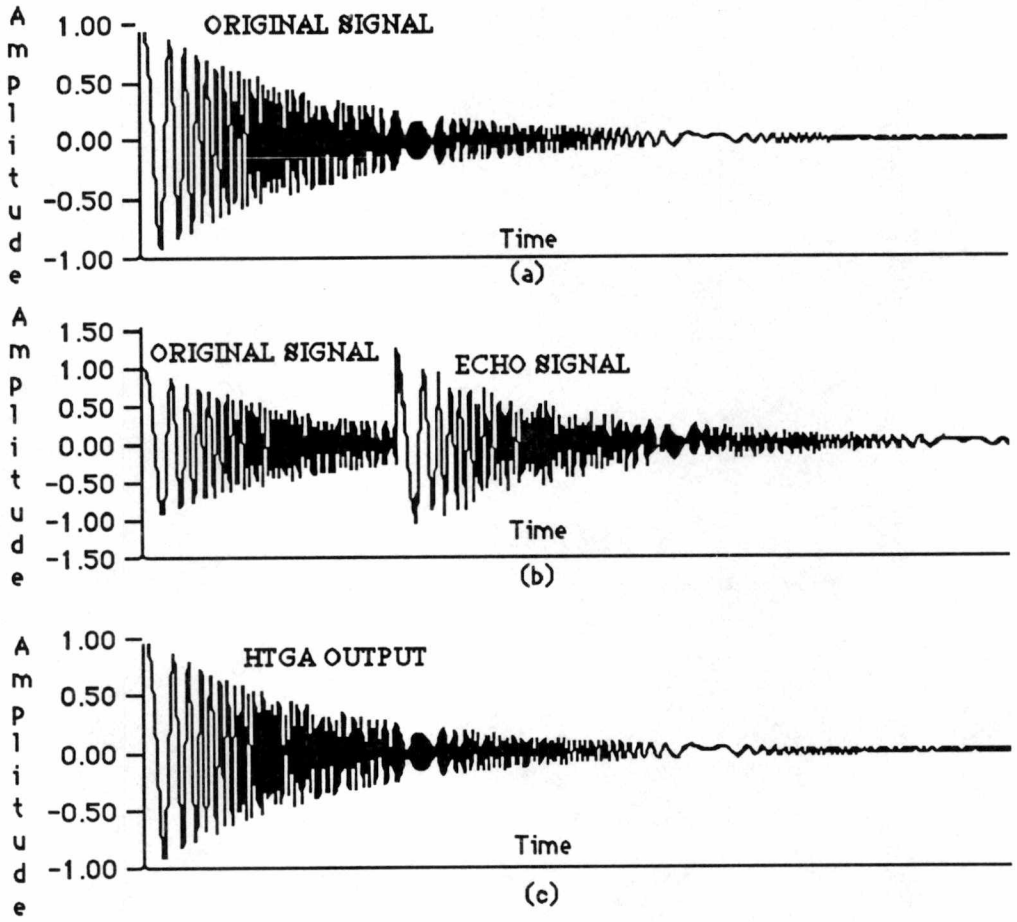


Figure 10. Simulated tone signal. (a) The original signal. (b) Original signal with overlapping echo. (c) The HTGA output.

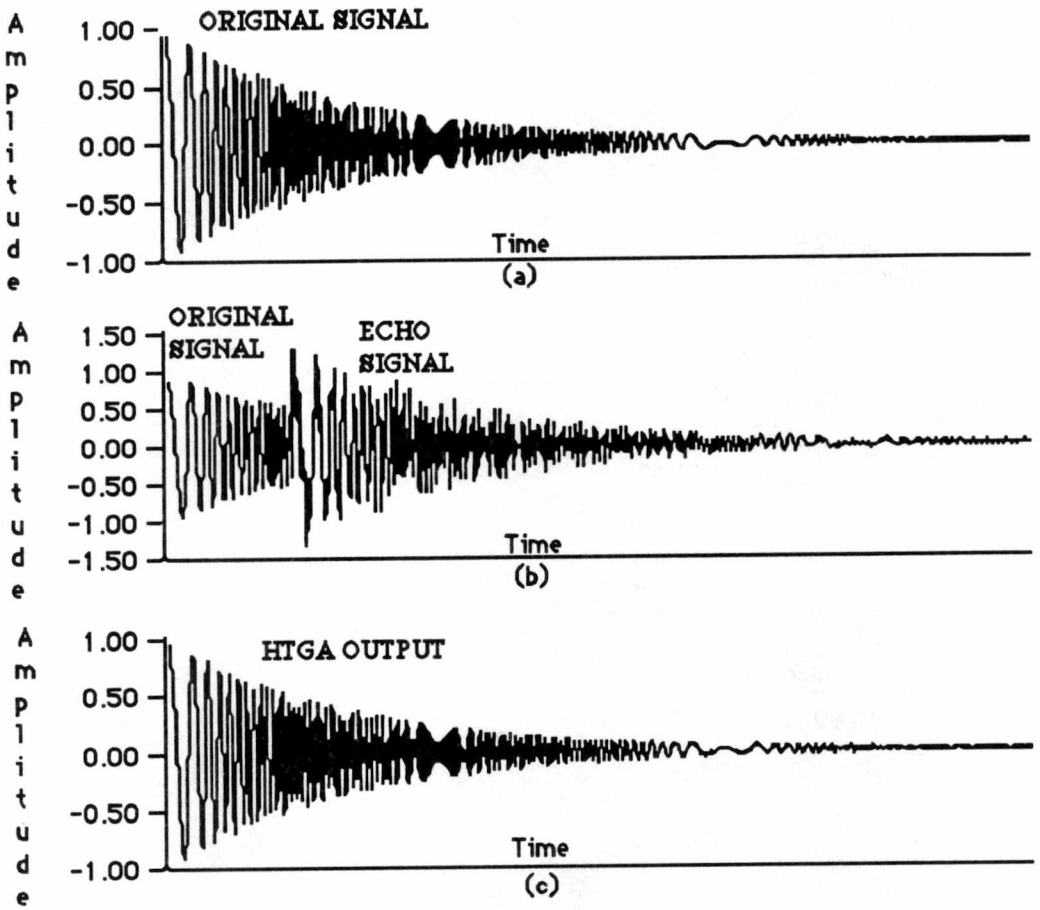


Figure 11. Simulated tone signal. (a) The original signal . (b) Original signal with overlapping echo. (c) The HTGA output.

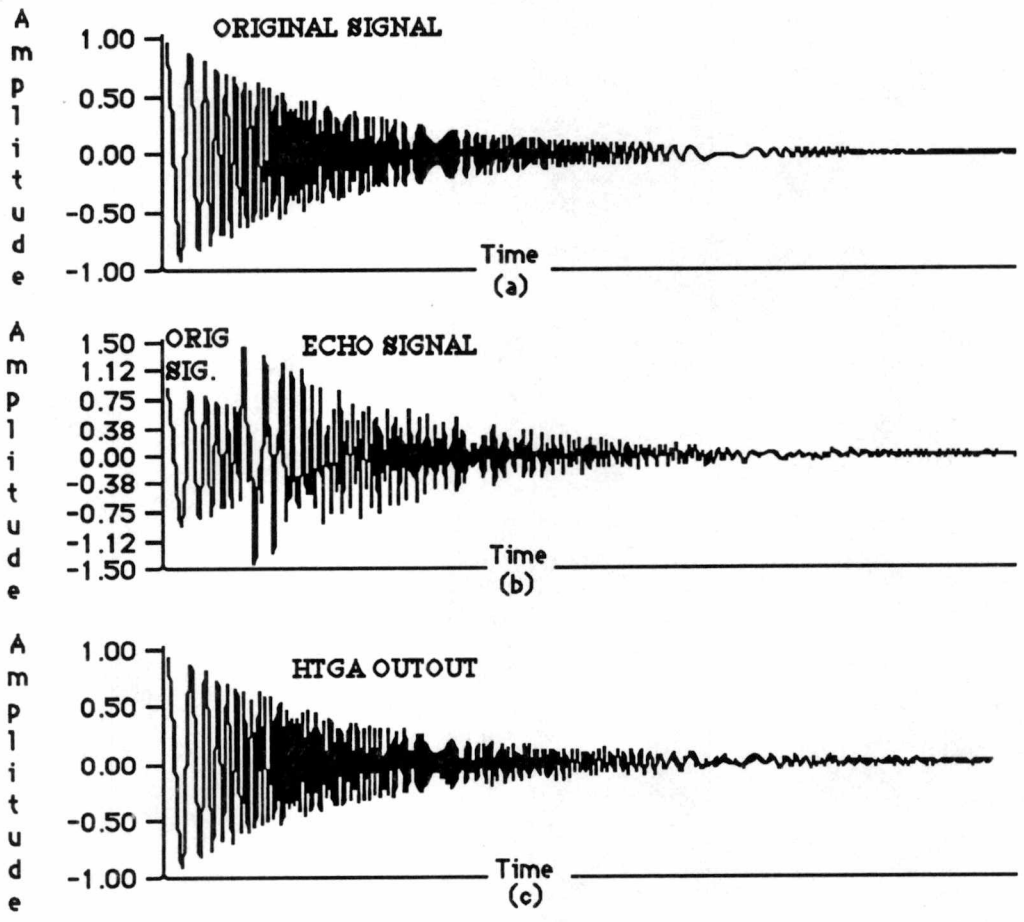


Figure 12. Simulated tone signal. (a) The original signal . (b) Original signal with overlapping echo. (c) The HTGA output.

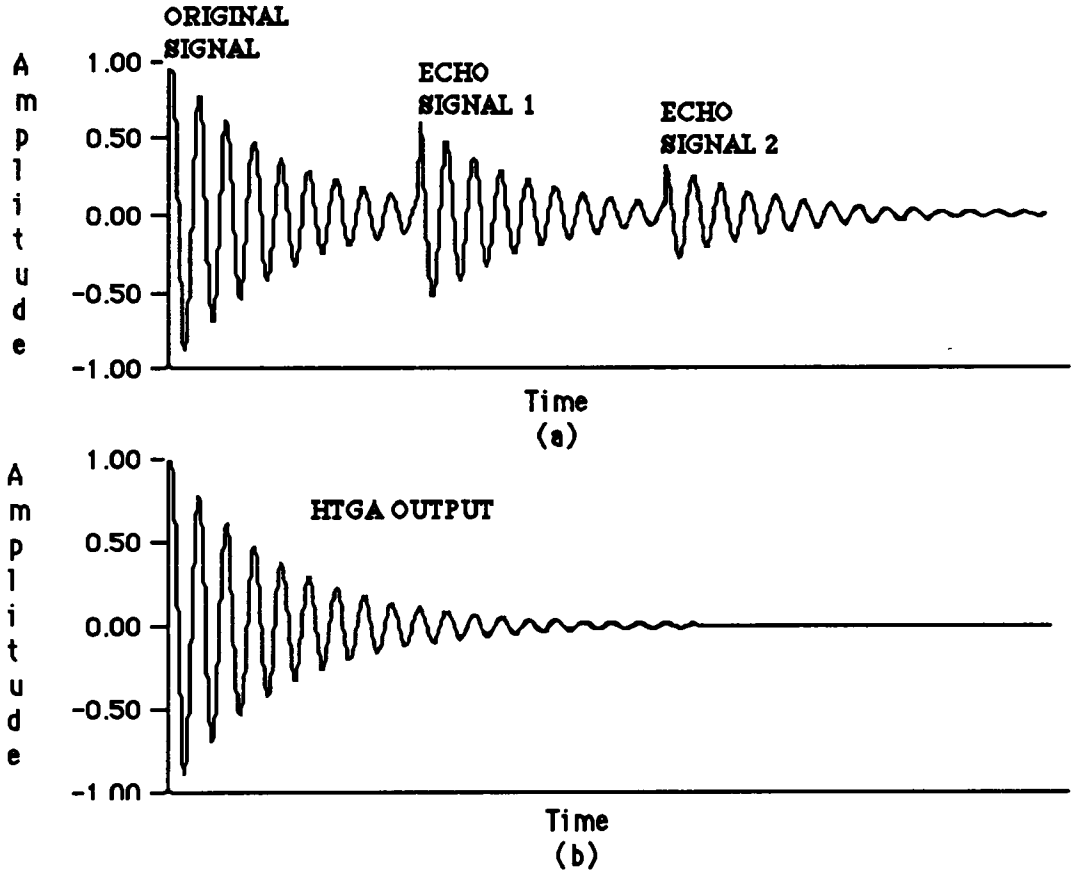


Figure 13. Simulated tone signal. (a) Signal with two echoes. (b) The HTGA output.

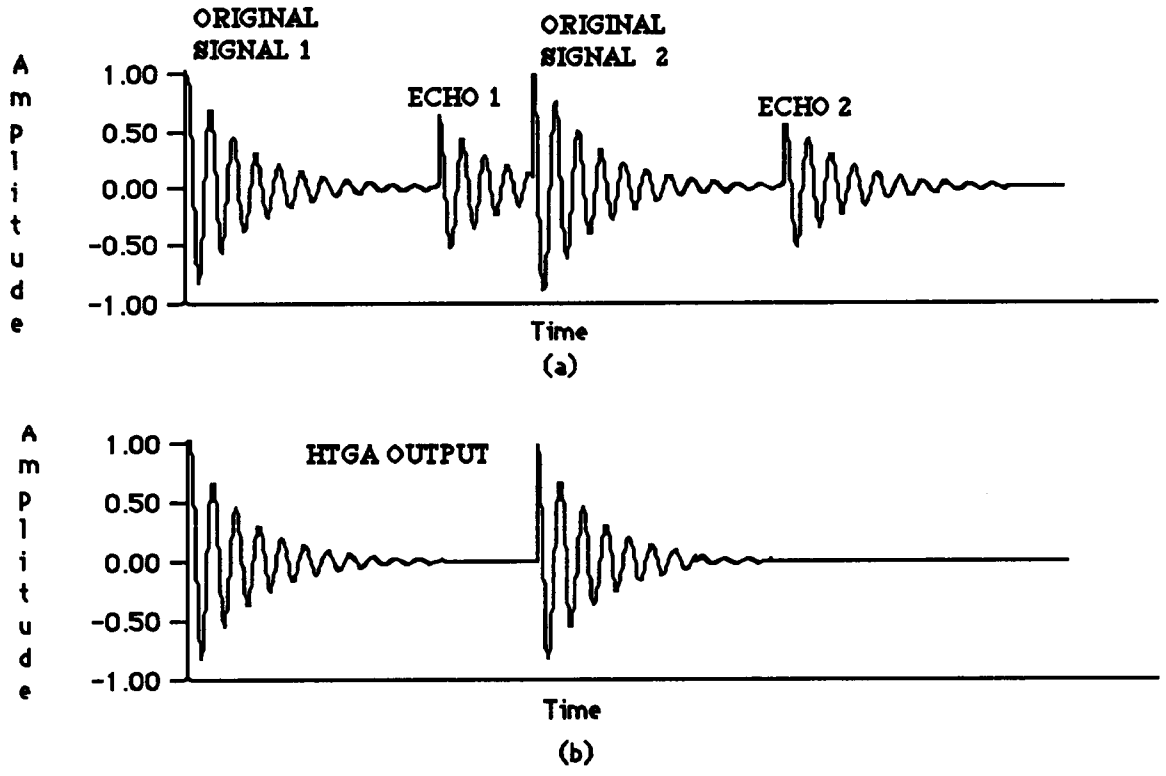


Figure 14. Simulated tone signal. (a) Two signals with two echoes. (b) The HTGA output.

CHAPTER VI

RESULTS AND DISCUSSION

From the previous figures, the HTGA method showed the ability to cancel the echo in a tone signal. The method was effective in several cases of non overlapping and overlapping signals. For the overlapping cases, the method worked well in canceling the echo even for the extreme case of overlapping.

MSE values were calculated for the signal with echo and for the enhanced HTGA signal. As seen in Table 1, MSE value for signal with echo is much higher than the one produced by the HTGA method. The low MSE values for the HTGA enhanced signals indicate the effectiveness of the method.

FIGURE #	MSE S+E	MSE HTGA
09	5.2491E-3	2.73E-8
10	3.97987E-2	8.6E-6
11	3.97989E-2	1.763E-4
12	3.97992E-2	1.794E-4
13	4.031E-3	1.7E-6
14	1.24011E-2	7.92E-5

Table 1. MSE values for both signal with echo and HTGA enhanced signal.

CHAPTER VII

COMPARISON WITH LMS ADAPTIVE FILTERING

This chapter compares the results of using the least-mean-square algorithm (LMS algorithm) in place of the GA with the results of using the GA algorithm. An LMS adaptive filtering design is shown in Appendix F.

Five sets of simulated data are used to demonstrate the comparison between LMS and GA algorithm. The results of LMS adaptive filtering were obtained using different gain constants. One value was used for the gain constant in the first three sets of data. For the fourth set of data a high gain constant was used and in the last set of data a low gain constant was used. The first set, as shown in Figure 15, uses a tone signal with nonoverlapping echo signal. The second set uses a partial overlapping echo signal as shown in Figure 16. The third set uses a continuously increasing frequency tone signal partially overlapping echo signal as shown in Figure 17. Figures 18 and 19 are the same as Figures 15 and 16 but with a different gain constant for each one. In the following Figures, part (a) of each figure is simulated data for signal and echo, part (b) is simulated data for the original signal before the echo, part (c) is the output of the HTGA algorithm, and part (d) is the output of the LMS adaptive filtering.

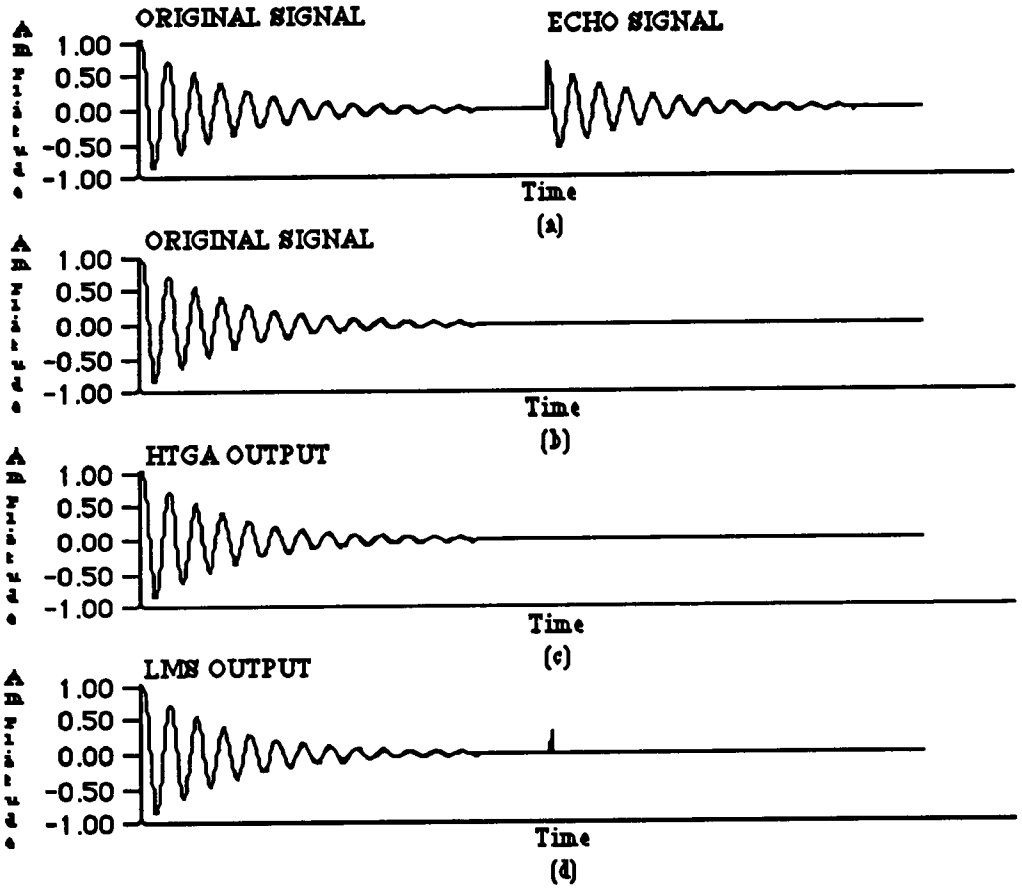


Figure 15. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1.

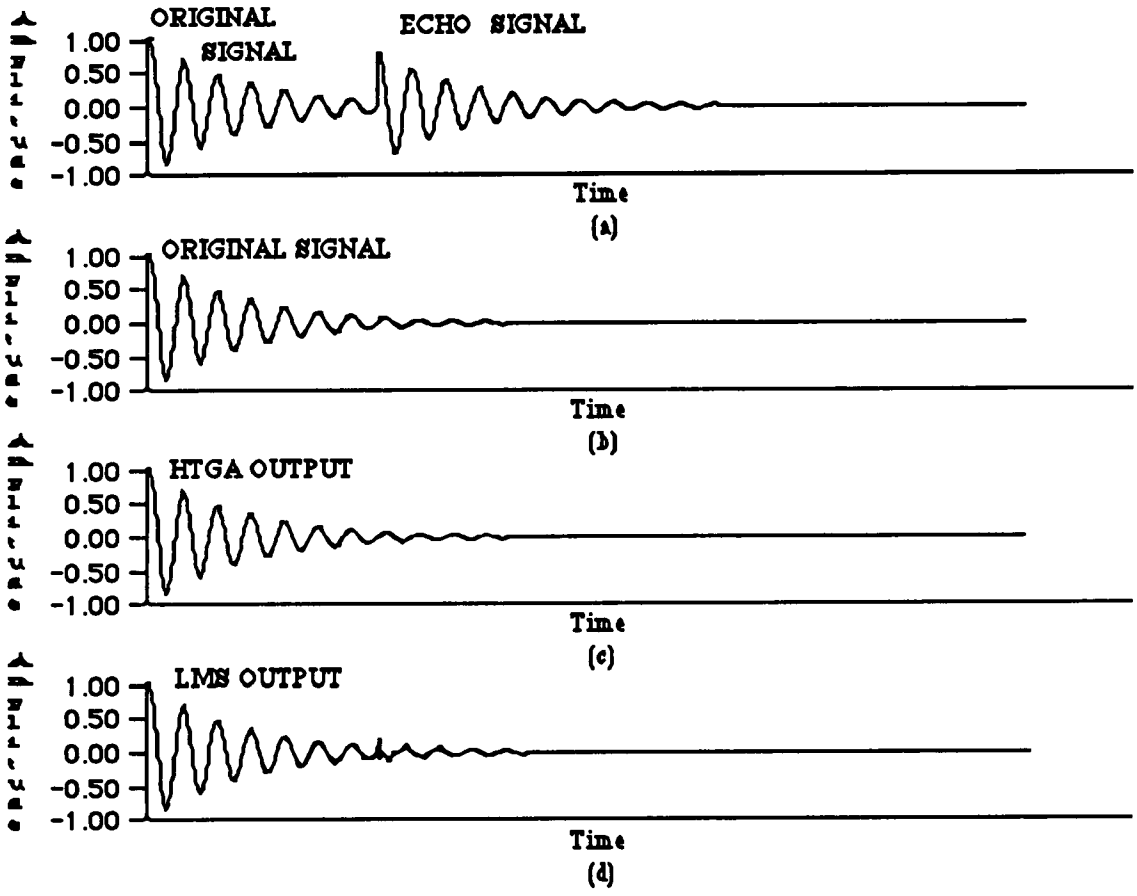


Figure 16. Simulated tone signal. (a) Simulated tone signal with partially overlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1.

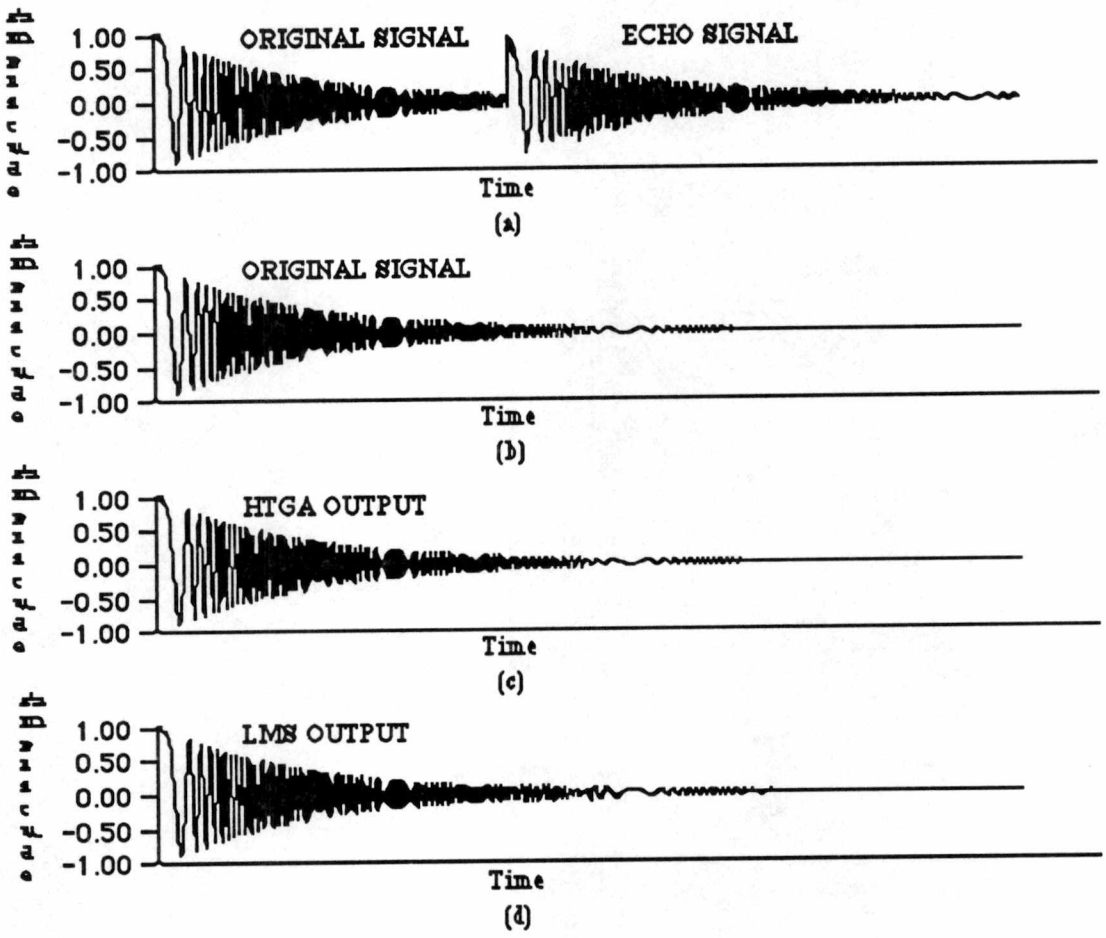


Figure 17. Simulated tone signal. (a) Simulated data for continuously increasing frequency partially overlapping an echo signal. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 1.

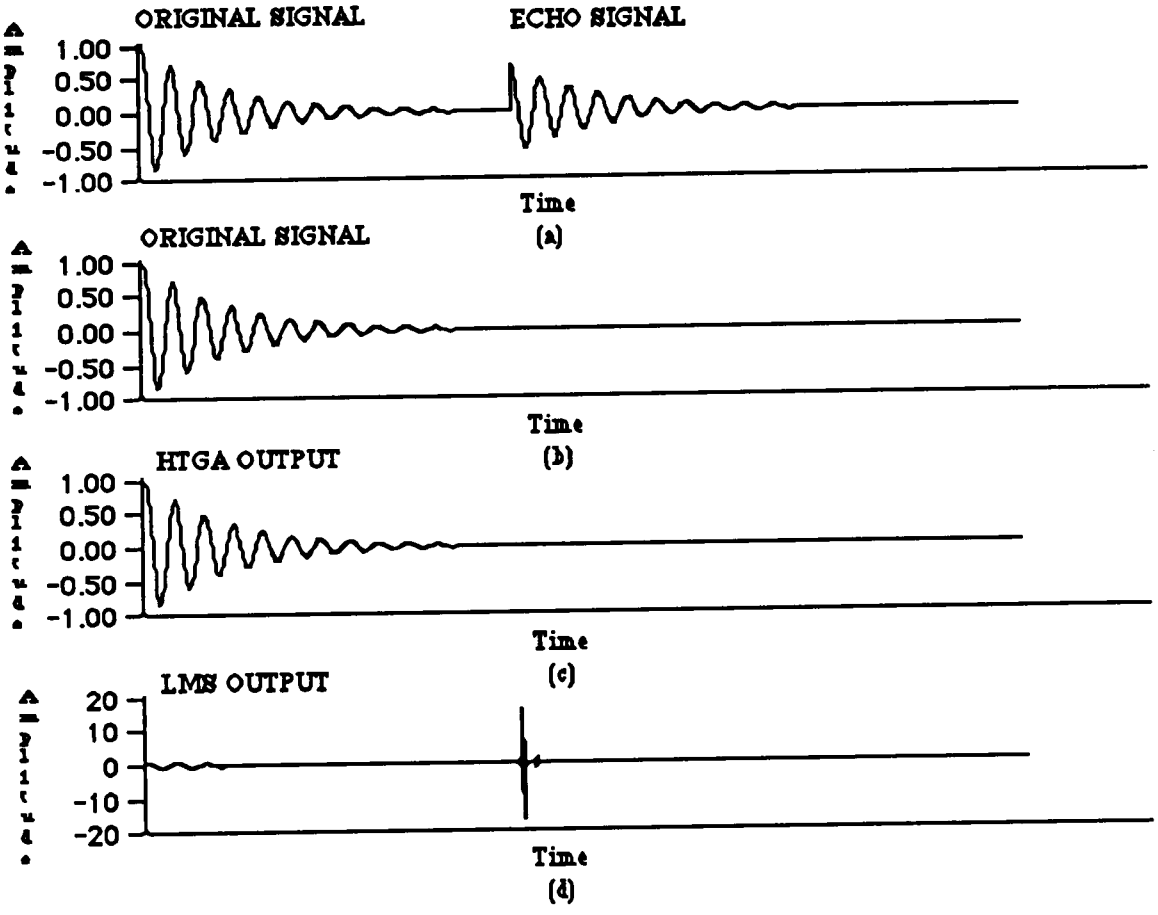


Figure 18. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 5.

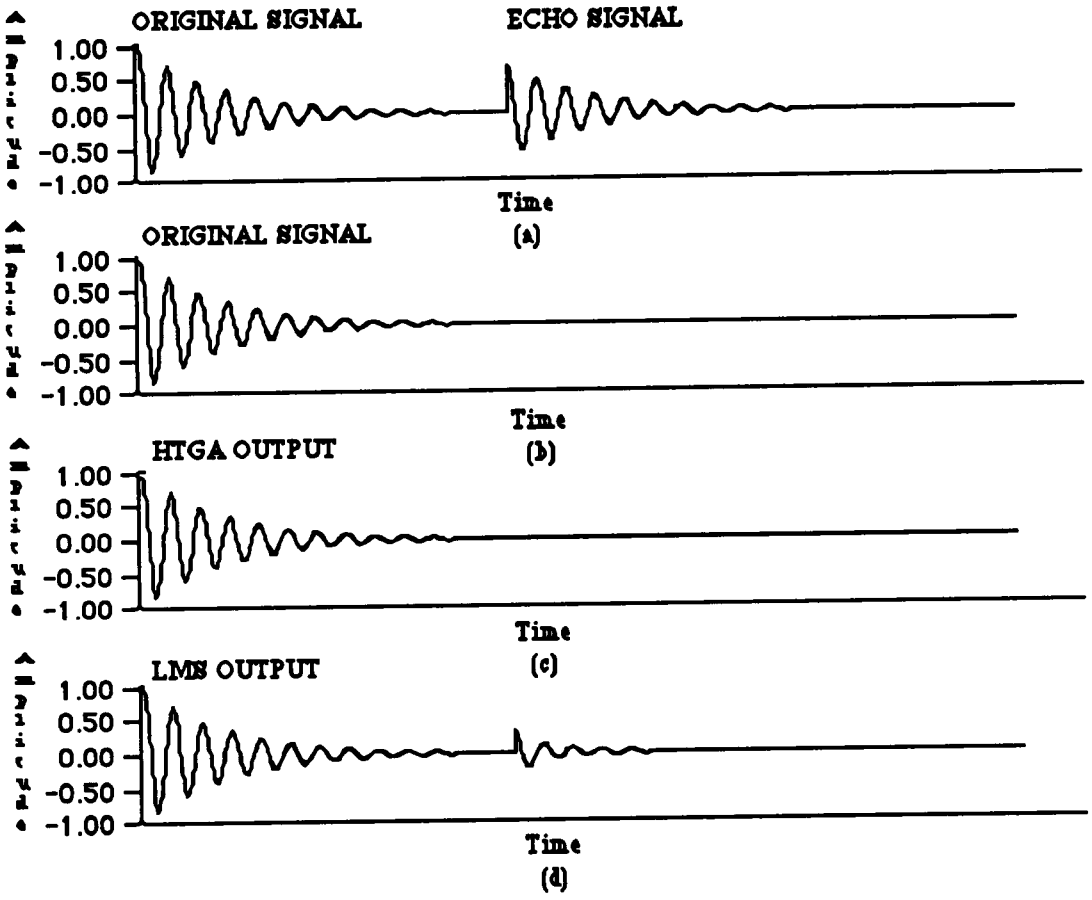


Figure 19. Simulated tone signal. (a) Simulated tone signal with nonoverlapping echo. (b) Simulated data of the original tone signal. (c) The result of GA algorithm. (d) The result of LMS adaptive filtering using a gain constant of 0.05.

As seen in part (c) for all the figures, the HTGA algorithm was very successful in eliminating the echo signal. On the other hand, LMS adaptive filtering gave different results for different gain constants. For a gain of one, the LMS results were very similar to the ones from the GA, but as seen in Figures 15(d) and 16(d) there was a part of the echo that was not canceled. By increasing the gain constant to five the LMS diverges as seen in Figure 18(d). For a low gain of 0.05 the LMS adaptive filtering was not able to eliminate the echo as seen in Figure 19(d). Different weight factors have been used throughout the LMS adaptive filtering procedure. This did not affect the final output. The MSE values for signal with echo, HTGA enhanced signal, and LMS enhanced signal were calculated for Figures 15 through 19 as seen in Table 2.

FIGURE #	MSE S+E .	MSE HTGA .	MSE LMS .
15	1.08903E-2	1.7E-6	1.263E-4
16	1.08904E-2	1.7E-6	3.679E-4
17	1.40943E-2	1.333E-4	1.8252E-3
18	1.08903E-2	1.7E-6	4.2843E-1
19	1.08903E-2	1.7E-6	1.2376E-3

Table 2. MSE values for signal with echo, HTGA enhanced signal, and LMS enhanced signal.

CHAPTER VIII

CONCLUSION

The Hilbert transform and genetic algorithm approach for removing echoes has been found to be very effective in eliminating the echo signal. The method was able to cancel an echo signal without losing or adding misleading information to the original signal. The HTGA method was also very effective for both the nonoverlapping and overlapping echo signal. The method was even effective in the three quarter overlap case.

The LMS results were very similar to the ones from the GA algorithm when a gain constant of one was used. By making the gain constant five, the LMS method was not able to eliminate the echo signal. This is not due to multiple minima in the solution space, but from the fact that the LMS algorithm will diverge if the gain constant is too large. Furthermore, changing the initial weight or scale factor does not change the final result which definitely indicates that there is only one minimum.

The MSE values in both Tables 1 and 2 show the effectiveness of the HTGA method. For example, the MSE value for the enhanced signal in Figure 9 was 2.73E-8. This indicates that the HTGA was able to cancel all of the echo signal and was able to produce a signal identical to the original one. The MSE values for the HTGA enhanced signals in Table 2 are much lower than the ones for the LMS enhanced

signals. This indicates that the HTGA method for echo cancellation is more effective than the LMS method. For future work, the HTGA method could be modified to handle more general signals other than a simple tone.

LIST OF REFERENCES

LIST OF REFERENCES

[1] M. M. Sondhi and D. A. Berkley, "Silencing Echoes on the Telephone Network," Proc. IEEE, Vol. 68, No. 8, pp. 948-963.

[2] R. W. Schafer, "Echo Removal by Generalizing Linear Filtering," IEEE, NEREM Record, pp. 118-119, 1967.

[3] S. T. Neely and J. B. Allen, "Invertibility of A Room Impulse Response," The Journal of the Acoustical Society of America, Vol. 66, No. 1, pp. 165-169, July 1979.

[4] J. B. Allen, D. A. Berkley, and J. Blavent, "Multimicrophone Signal-Processing Technique to Remove Room Reverberation from Speech Signal," The Journal of the Acoustical Society of America, Vol. 62, No. 4, pp. 912-915, October 1979.

[5] M. M. Sondhi , "An Adaptive Echo Canceller," The Bell System Technical Journal, Vol. 46, No. 3, pp. 497-511, March 1967.

[6] R. E. Ziemer and W. H. Tranter, Principle of Communications Systems, Modulation, and Noise. Houghton Mifflin, Lawrenceville, New Jersey: Ch. 2, pp. 85-90.

[7] L. B. Jackson, Digital Filters and Signal Processing. Kluwer Academic Publishers, Boston: Ch. 13, pp. 374-376.

[8] D. E. Goldberg, Genetic Algorithm in Search, Optimization and Machine Learning. Reading, Massachusetts: Addison-Wesley, 1989, Ch. 3, pp. 1-54.

[9] B. Widrow and S. D. Streans, Adaptive Signal Processing. Prentice-Hall, Inc. Englewood Cliffs, N.J: Ch. 6, pp. 99-101.

[10] "Echo Detection", LabVIEW 2 Analysis VI Library Reference Manual, National Instrument Corp. , pp. 6-12, 1990.

[11] S. L. Horner, Genetic Algorithm And Non-Invasive FECG Enhancement. Master of Science Thesis, University of Tennessee, Knoxville, Dec. 1991, pp. 63-71.

APPENDICES

APPENDIX A

HILBERT TRANSFORM TABLE

HILBERT TRANSFORM PAIRS *

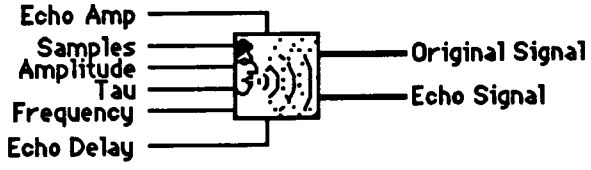
SIGNAL	HILBERT TRANSFORM
1. $x(at + b)$	$\hat{x}(at + b)$
2. $x(t) + y(t)$	$\hat{x}(t) + \hat{y}(t)$
3. $\frac{d^n x(t)}{dt^n}$	$\frac{d^n}{dt^n}\hat{x}(t)$
4. $A, \ -\infty < t < \infty$	0
5. 1	$-\pi\,\delta(t)$
6. $\sin(\omega_0 t + \theta)$	$-\cos(\omega_0 t + \theta)$
7. $\operatorname{sinc}(at)$	$-\frac{1}{2}\,at\,\operatorname{sinc}^2 at$
8. $e^{\pm j\omega_0 t}$	$\mp j e^{\pm j\omega_0 t}$
9. $\delta(t)$	$\frac{1}{\pi t}$
10. $\frac{a}{\pi(t^2 + a^2)}$	$\frac{t}{\pi(t^2 + a^2)}$
11. $\Pi(t)$	$\frac{1}{\pi}\ln\left\lvert\frac{2t - 1}{2t + 1}\right\rvert$

* R. E. Ziemer and W. H. Tranter, Principles of Communications Systems Modulation, and Noise. Second Edition, p. 592.

APPENDIX B

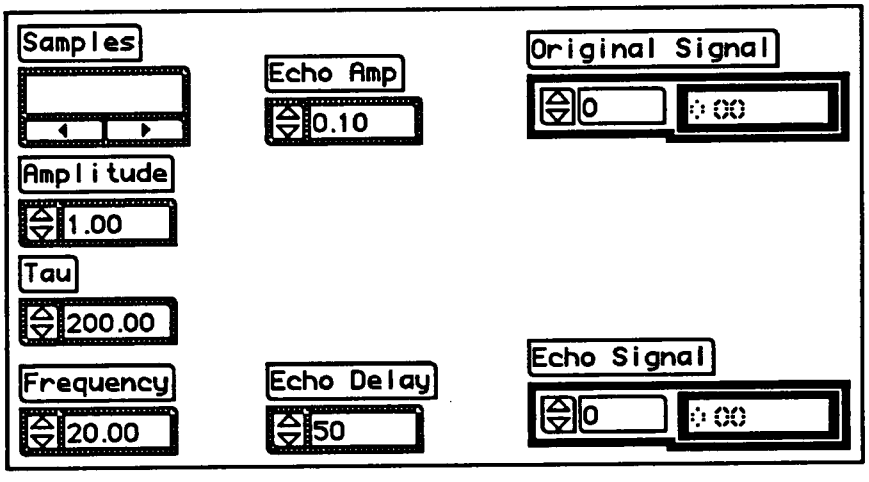
ECHO SIGNAL DESIGN

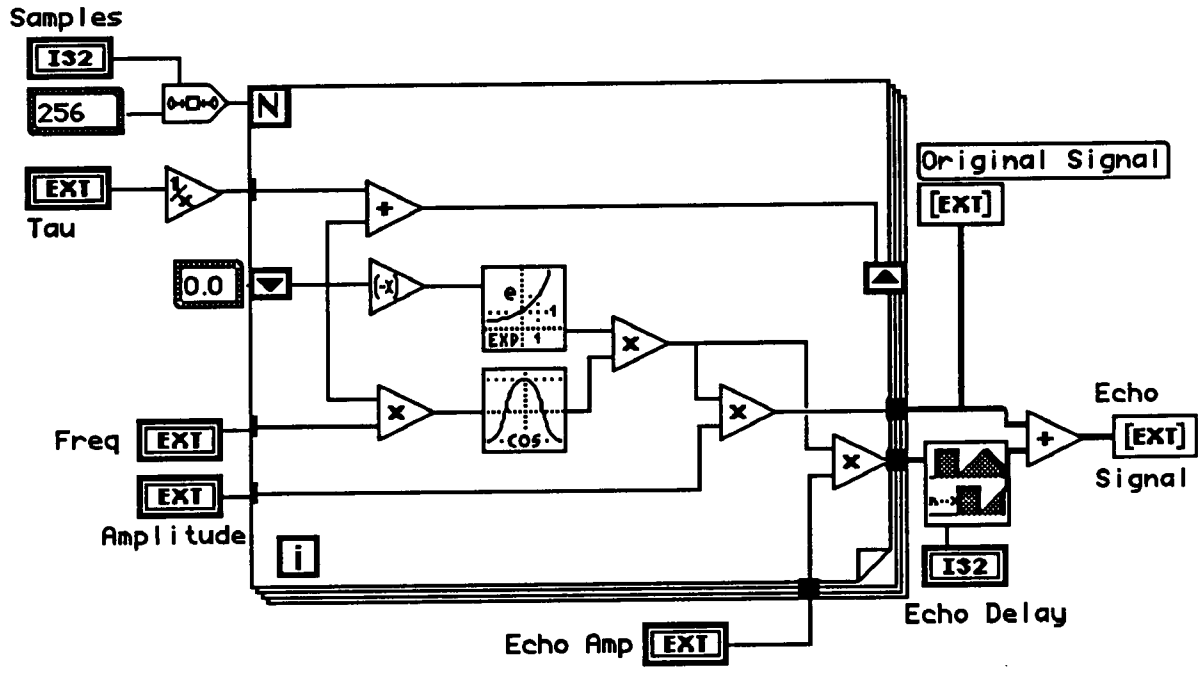
Connector Pane



Echo Maker

Front Panel




Samples
I32
256
N
EXT
Tau
1/x
0.0
(-x)
e
EXP
+
x
cos
Freq
EXT
EXT
Amplitude
i
Original Signal
[EXT]
Echo
[EXT]
Signal
I32
Echo Delay
Echo Amp
EXT

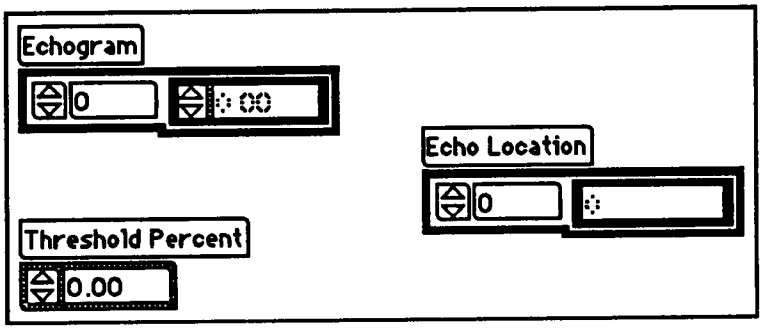
APPENDIX C

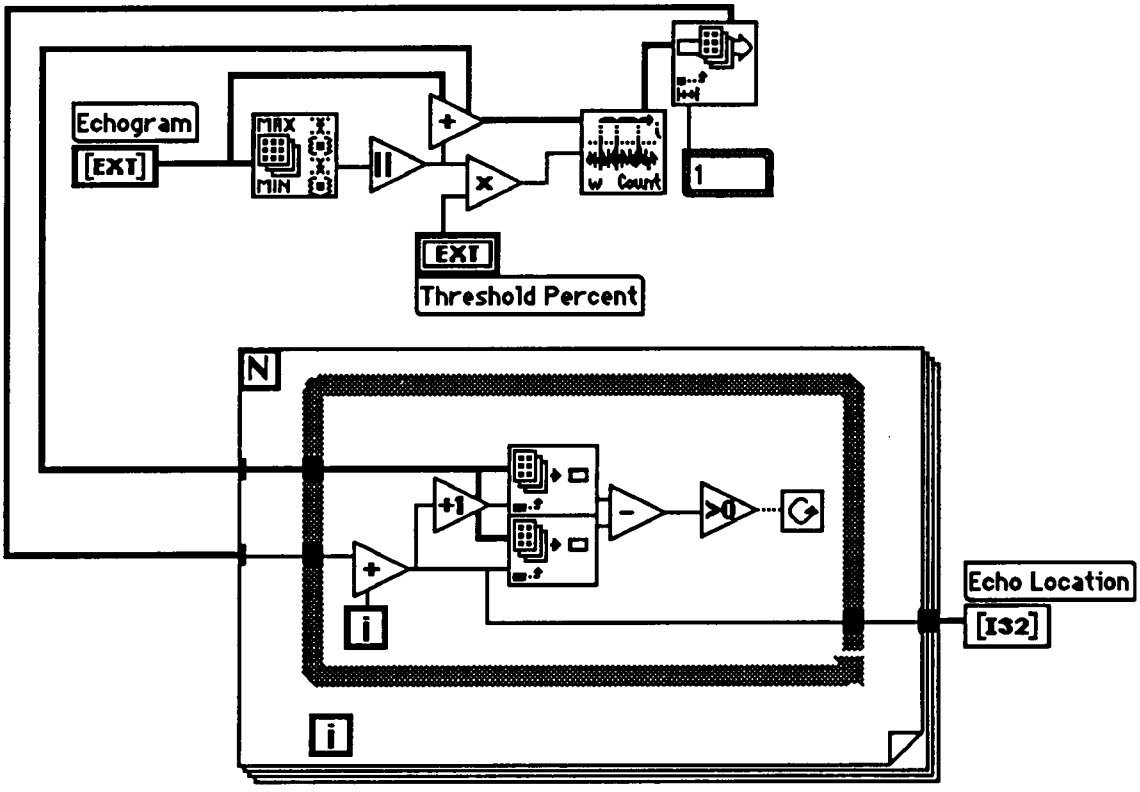
ECHO LOCATION DESIGN



Echo Location

Front Panel

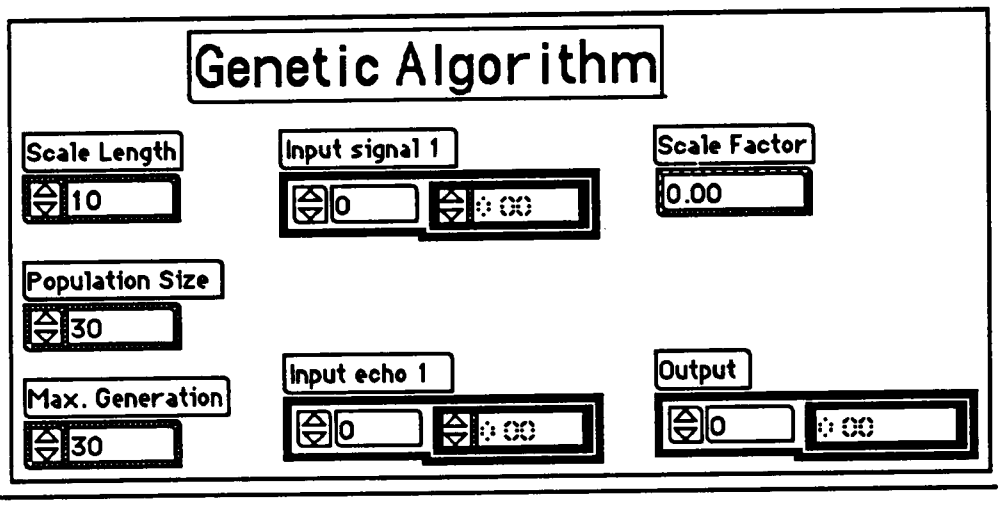



Echogram
EXT
MAX
MIN
w Count
1
EXT
Threshold Percent
N
+1
>0
i
i
Echo Location
[I32]

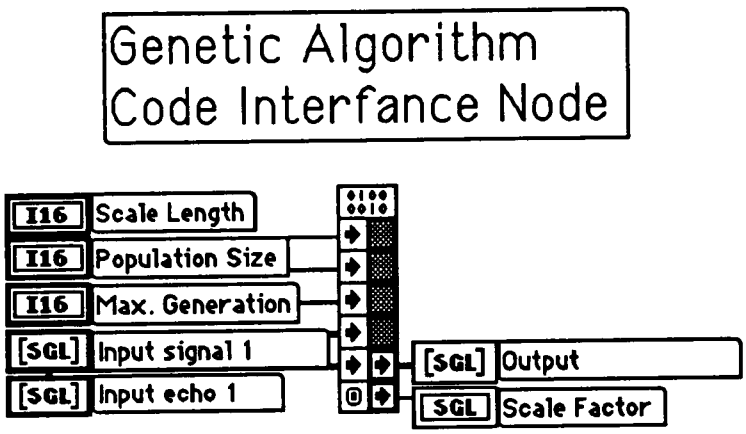
APPENDIX D

GA PROGRAM & C CODE

Front Panel



Block Diagram



```
#include  "define1.h"
#include  "fitness1.h"
#include "GA1.h"
#include "GA2.h"
#include "GA3.h"
#include "GA4.h"
#include "GA5.h"
#include "GA1CIN.h"

pascal void CINInit() { }

pascal void CINDispose() { }

pascal void CINAbort() { }

pascal void CINRun(s, c, ps, mg, echo, sig, bs)
        TD1Hdl echo, sig;
        int16 *s, *c, *ps, *mg;
        float32 *bs;
        {
                int32 i;
                TD1 *p, *q;

                scaleLength = *s;
                scale = 1.0;
                for(i = 0; i < scaleLength; i++){
                        scale *= 2.0;
                }
                chromLength = *c;
                popsize = *ps;
                maxgen = *mg;
                p = *echo;
                q = *sig;
                Len = p->dimSize;
                for(i = 0; i < Len; i++){
                        echo1[i] = p->arg1[i];
                        sig1[i] = q->arg1[i];
                }
```

```
                ga();

                *bs = bscaler;
                for(i = 0; i < Len; i++)
                        q->arg1[i] = sig2[i];
        }

pascal void CINLoad() { }

pascal void CINSave() { }
```

```
/*********************  fitness 1. h  ***************************/
float  fitness(ind)
struct  individual  *ind;
{
        float sum, diff, scaler;
        int i, *c;

        c = (*ind).chrom;
        scaler = 0.0;
        for(i = 0; i < scaleLength; i++){
                scaler *= 2.0;
                scaler += c[i];
        }
        scaler /= (scale - 1.0);
        }
        sum = 0;
        for(i = 0; i < Len; i++){
                diff = sig1[i] - (scaler*echo1[i]);
                if(diff >= 0.0)
                        sum += diff;
                else
                        sum -= diff;
        }
        sum /= (Len);
        return(-sum);
}
/*****************************************************************/
 doStatistics()
{
        float sum, fit, scaler;
        int i, j, *c, temp, offset;
        sum = newPop[0].fitness;
        max = -10000.0;
        for(j = 0; j < popsize; j++){
                fit = newPop[j].fitness;
                sum += fit;
                if(fit > max){
                        max = fit;
                        temp = j;
```

```
        }
}
avg = sum/popsize;
if(max > glmax){
        c = newPop[temp].chrom;
        scaler = 0.0;
        for(i = 0; i < scaleLength; i++){
                scaler *= 2.0;
                scaler += c[i];
        }
        scaler /= (scale - 1.0);
        bscaler = scaler;
        }

        for(i = 0; i < Len; i++){
                sig2[i] = sig1[i] - (scaler*echo1[i]);
        }
        glmax = max;
}
```

```
/********************** define 1. h ****************************/
#define  cardinality  2
#define  pickBestProb  1.0
#define  pcross  .6
#define  chromL  50
#define  popsz  50
#define RAND_MAX    32767
typedef  float  float32;
typedef  long  int32;
typedef  int  int16;
struct  individual{
        int chrom[chromL];
        float fitness;
};
struct  individual  *oldPop;
struct  individual  *newPop;
struct  individual  pop1[popsz];
struct  individual  pop2[popsz];
int  scaleLength,  chromLength,  popsize;
int  gen,  maxgen;
float  scale,bscaler;
float  max,  avg,  glmax;
float  mutProb;
float  echo1[1500],  sig1[1500],  sig2[1500];
int  Len;
int  pick,  shuffle[popsz];
int  childPos;
struct  individual  *child1;
struct  individual  *child2;
struct  individual  *select();
float  random();
unsigned  long  seed  =  1;
int  yrand(void);
                    void  ysrand(unsigned);
```

```
/********************  GA1C1N.h **********************/
/*
 * Code Interface Node header file
 */

/*
 * typedefs
 */

typedef struct {
      int32 dimSize;
      float32 arg1[];
} TD1;
typedef TD1 **TD1Hdl;

/*
 * Code Interface Node prototypes:
 */

pascal void CINInit(void);
pascal void CINDispose(void);
pascal void CINAbort(void);
pascal void CINRun(int16 *input1, int16 *input2, int16 *input3,
int16 *input4, TD1Hdl input5, TD1Hdl inout1, float32 *output1,
int16  *output2);
pascal void CINLoad(void);
pascal void CINSave(void);
```

```
/********************** GA1.h          ************************/

        ga()
        {

                initData();
                gen = 0;
                initPopulation();
                doStatistics();
                agePopulation();

                gen = 1;
                while(gen <= maxgen){
                        doGeneration();
                        doStatistics();
                        agePopulation();
                        gen = gen + 1;
                }


        }
```

```
/*************** GA2.h **********************/

initData()
{
        int i;
        float  t1,t2;
        pick = 0;
        glmax  =  -10000.0;
        shufflePop();
        t1  =  1.0/popsize;
        t2  =  1.0/chromLength;
        if(t1 < t2)
                mutProb  =  t1;
        else
                mutProb  =  t2;
        oldPop  =  pop1;
        newPop  =  pop2;
}

shufflePop()
{
        int num, other, j;
        for(j = 0; j < popsize; j++)
                shuffle[j] = j;
        for(j = 0; j < popsize - 1; j++){
                num = rndFromTo(j, popsize - 1);
                other = shuffle[num];
                shuffle[num] = shuffle[j];
                shuffle[j] = other;
        }
}

initPopulation()
{
        int j, k;
        struct individual *ind;
        int *c;
        for(j = 0; j < popsize; j++){
                ind = (&newPop[j]);
                c = (*ind).chrom;
```

```
            for(k = 0; k < chromLength; k++){
                  c[k] = rndFromTo(0, cardinality - 1);
            }
            eval(ind);
      }
}

eval(ind)
struct individual *ind;
{
      float fit;
      fit = fitness(ind);
      (*ind).fitness = fit;
}
```

```
/***************** GA3.h ********************/

rndFromTo(low,hi)
int low, hi;
{
      float n;
      n = random();
      return((int)((n*(hi - low + 1)) + low));
}

float  random()
{

      double  t;
      t  =  yrand()/32769.0;
      return((float)t);
 }

int  yrand()
{
      seed  =  seed*1103515245  +  12345;
      asm {
            move.w  seed,d0        ;  high word of long
            andi.w #0x7FFF,d0      ;  remove high bit
      }
}

void  ysrand(n)
unsigned  n;
{
      seed = n;
}

  agePopulation()
 {
      struct  individual  *temp;
      temp = oldPop;
      oldPop = newPop;
      newPop = temp;
```

```
        childPos = 0;
        child1 = (&newPop[0]);
        child2 = (&newPop[1]);
}

 doGeneration()
{
        struct individual *mate1, *mate2;
        int j;
        for(j = 0; j < popsize; j = j + 2){
                mate1 = select();
                mate2 = select();
                cross(mate1,mate2);
        }
}
```

```
/********************  GA4.h **********************/

        struct individual *select()
        {
              struct individual *first, *second;
              if(pick >= popsize - 1){
                    pick = 0;
                    shufflePop();
              }
              first = (&oldPop[shuffle[pick]]);
              second = (&oldPop[shuffle[pick +1 ]]);
              pick = pick + 2;
              if((*first).fitness < (*second).fitness)
                    if(pickBest())
                          return(second);
                    else
                          return(first);
               else
                    if(pickBest())
                          return(first);
                    else
                          return(second);
        }

        pickBest()
        {
              if(random() < pickBestProb)
                    return(1);
              else
                    return(0);
        }

        cross(ind1,ind2)
        struct individual *ind1, *ind2;
        {
              int jcross, *c1, *c2, *d1, *d2, j;
              jcross = crossSite();
              c1 = (*ind1).chrom;
              c2 = (*ind2).chrom;
              d1 = (*child1).chrom;
```

```
        d2 = (*child2).chrom;
        for(j = 0; j <= jcross; j++){
                d1[j] = c1[j];
                d2[j] = c2[j];
        }
        if(jcross < chromLength){
                for(j = jcross + 1; j < chromLength; j++){
                        d1[j] = c2[j];
                        d2[j] = c1[j];
                }
        }
        mutation(child1);
        mutation(child2);
        addChildren();
}
```

```
/********************* GA5.h ************************/

crossSite()
{
      if(random() < pcross)
            return(rndFromTo(0, chromLength - 2));
      else
            return(chromLength);
}

mutation(ind)
struct individual *ind;
{
      int *c, j;
      c = (*ind).chrom;
      if(mutProb > 0.0)
            for(j = 0; j < chromLength; j++)
                  if(random() < mutProb)
                        c[j] = mutate(c[j]);
}

mutate(v)
int v;
{
      int r;
      r = rndFromTo(0,(cardinality - 2));
      if(r < v)
            return(r);
      else
            return(r + 1);
}

addChildren()
{
      eval(child1);
      eval(child2);
      positionNextChildren();
}
```

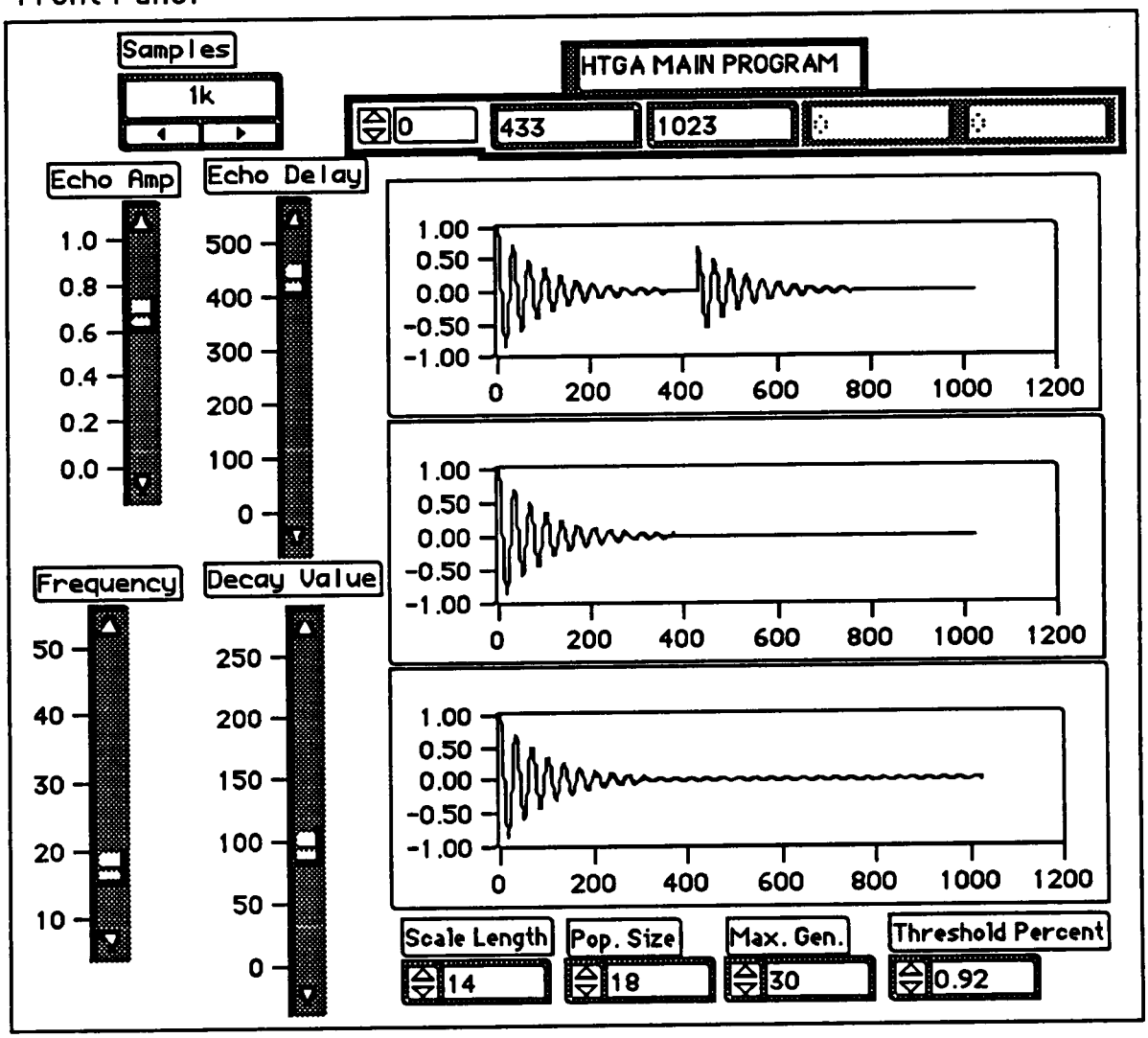
```
 positionNextChildren()
{
        childPos +=2;
        if(childPos <=popsize - 1)
                child1  =(&newPop[childPos]);
        if(childPos < popsize - 1)
                child2 = (&newPop[childPos + 1]);

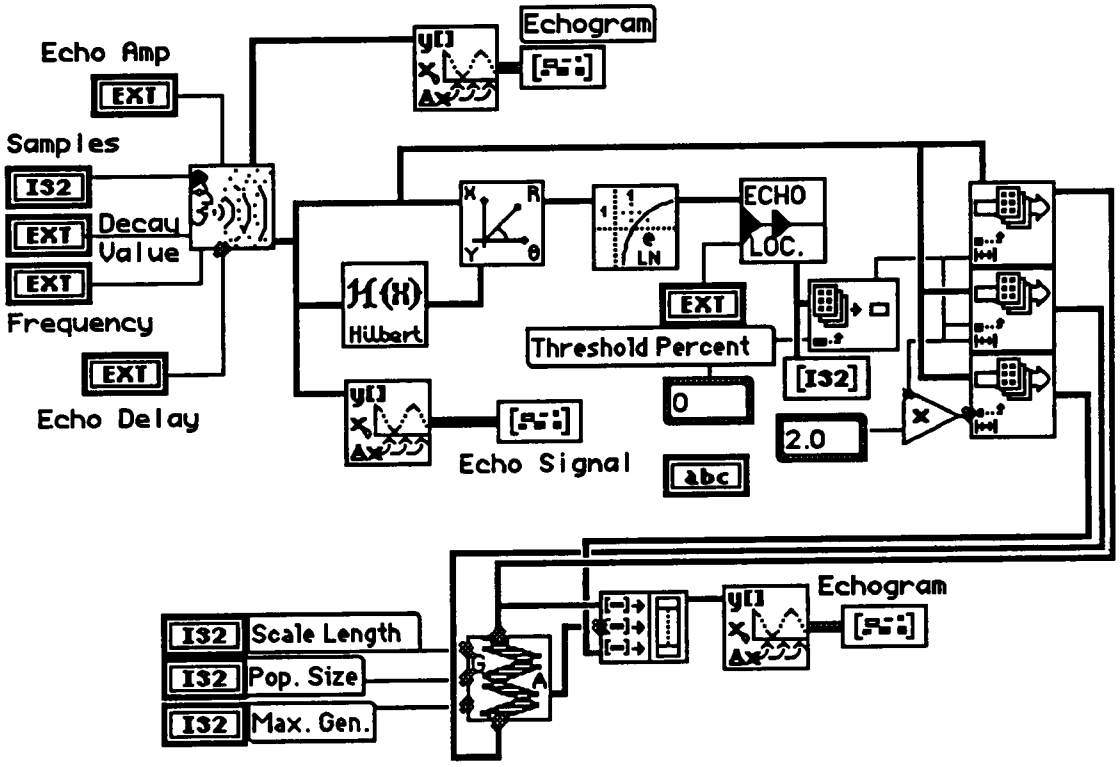
}
```

APPENDIX E

THE MAIN HTGA PROGRAM

Front Panel

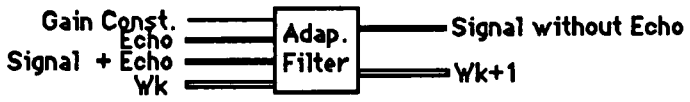

Samples
1k
HTGA MAIN PROGRAM
0
433
1023
Echo Amp
1.0
0.8
0.6
0.4
0.2
0.0
Echo Delay
500
400
300
200
100
0
Frequency
50
40
30
20
10
Decay Value
250
200
150
100
50
0
1.00
0.50
0.00
-0.50
-1.00
0
200
400
600
800
1000
1200
Scale Length
14
Pop. Size
18
Max. Gen.
30
Threshold Percent
0.92


Echo Amp
EXT
Samples
I32
EXT
Decay
Value
EXT
Frequency
EXT
Echo Delay
y[]
Echogram
H(x)
Hilbert
ECHO
LOC.
EXT
Threshold Percent
0
[I32]
2.0
abc
Echo Signal
I32
Scale Length
I32
Pop. Size
I32
Max. Gen.
Echogram

APPENDIX F

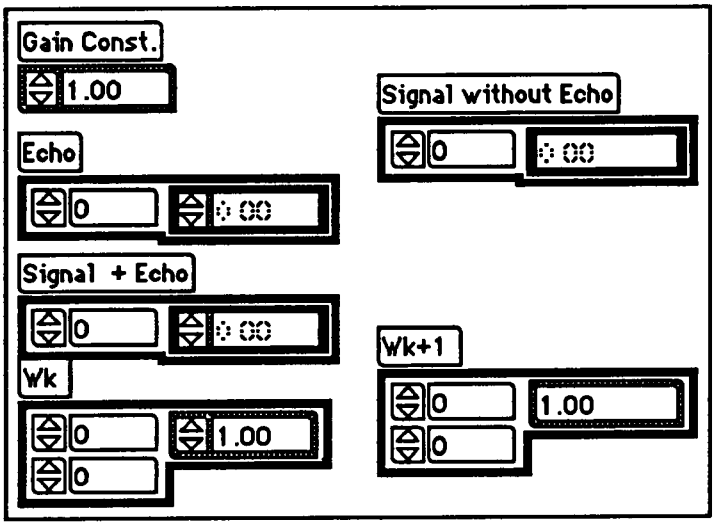
LMS ADAPTIVE FILTERING

Connector Pane

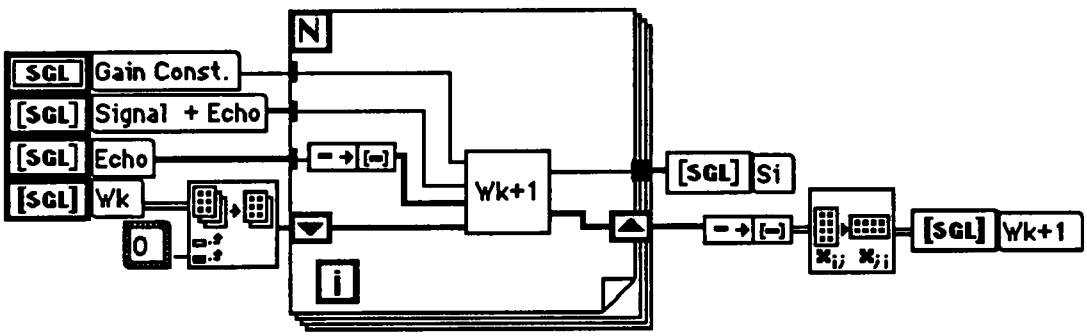


Adap. Filter Echo

Front Panel



Block Diagram

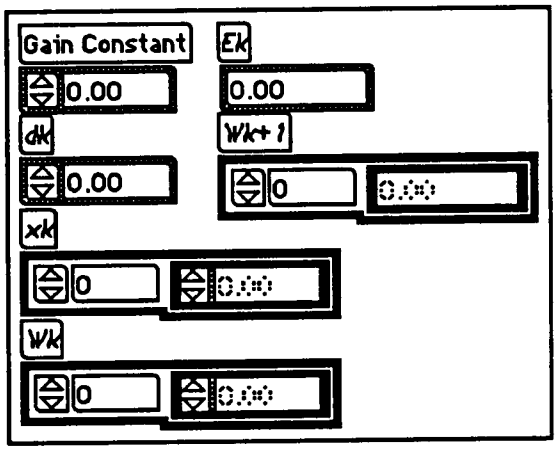


Connector Pane

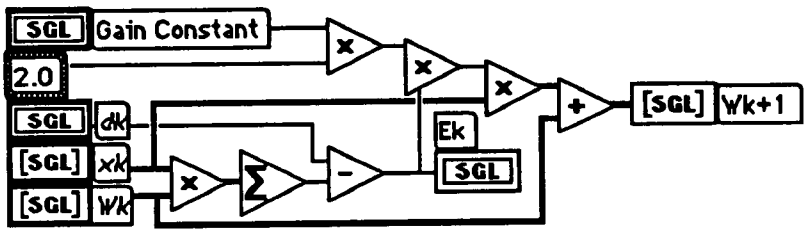


Wk+1 Adap. Filt.

Front Panel



Block Diagram



VITA

Luaie Abdullah Al-BaBa was born in Jericho, Palestine on October 7, 1964. He graduated from High School in Amman, Jordan in 1983. On October 2 of the same year he came to the United States to start higher education. Luaie started his undergraduate education at the University of Tennessee Knoxville in March of 1984. He graduated with a Bachelor of Science in Electrical Engineering in March of 1988. In Fall of 1989, the author started graduate school toward a Master's degree in Electrical Engineering at the University of Tennessee.

The author worked to provide more than 50% of his college expenses. He is a member of Eta Kappa Nu and The Institute of Electrical and Electronics Engineers.