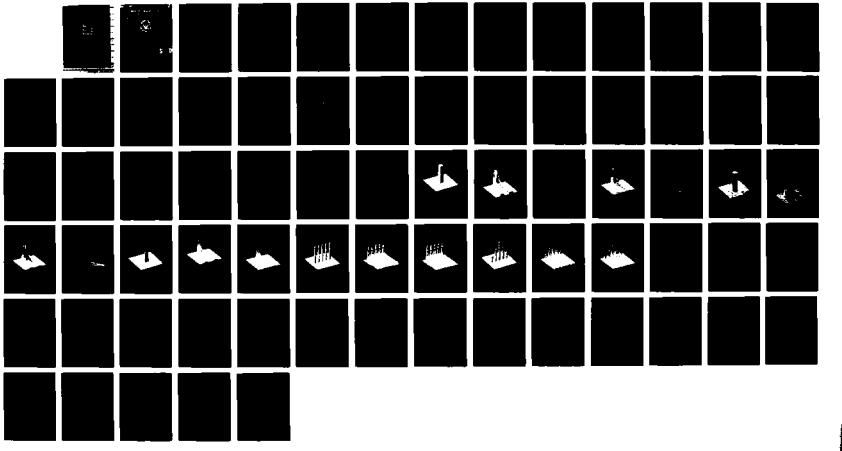
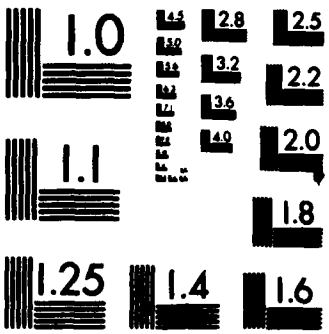
AD-A181 801 A TRANSFER FUNCTION APPROACH TO SCALAR WAVE PROPAGATION IN LOSSY AND LOSSLESS MEDIA(U) NAVAL POSTGRADUATE SCHOOL MONTEREY CA T D MERRILL MAR 87 1/1

UNCLASSIFIED F/G 20/1 NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A TRANSFER FUNCTION APPROACH TO SCALAR WAVE PROPAGATION IN LOSSY AND LOSSLESS MEDIA

by

Timothy D. Merrill

March 1987

Thesis Advisor: John P. Powers

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release;distribution is unlimited
2b DECLASSIFICATION/DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b OFFICE SYMBOL (If applicable) 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification) A TRANSFER FUNCTION APPROACH TO SCALAR WAVE PROPAGATION IN LOSSY AND LOSSLESS MEDIA

12 PERSONAL AUTHOR(S)
MERRILL, Timothy D.

13a TYPE OF REPORT	13b TIME COVERED FROM ____ TO ____	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis		1987 March	72

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Acoustic Imaging, Lossy Media, Ultrasonic Propagation, Transfer Functions, Linear Systems,

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis investigates the feasibility of micro-computer based simulation of scalar wave propagation in various media. Models for lossless media and media with a loss coefficient which is linear in frequency have been coded in FORTRAN and simulated successfully on a commercially available micro-computer, with simulation times less than 30 minutes. The spatial impulse responses for classical problems using square and circular-piston excitation are presented graphically, along with new, innovative, spatial excitation source shapes.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION	
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a NAME OF RESPONSIBLE INDIVIDUAL John P. Powers	22b TELEPHONE (Include Area Code) 408-646-2679	22c OFFICE SYMBOL 62Po

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE

Approved for public release; distribution is unlimited.

A Transfer Function Approach to Scalar Wave Propagation in Lossy and Lossless Media

by

Timothy D. Merrill
Lieutenant, United States Navy
B.S., Florida Southern College, 1980

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1987

Author: Timothy D. Merrill
Timothy D. Merrill

Approved by:
John P. Powers, Thesis Advisor

Robert D. Strum
Robert D. Strum, Second Reader

Harriett B. Rigas, Chairman, Department of Electrical and Computer Engineering

Gordon E. Schacher, Dean of Science and Engineering

ABSTRACT

This thesis investigates the feasibility of micro-computer based simulation of scalar wave propagation in various media. Models for lossless media and media with a loss coefficient which is linear in frequency have been coded in FORTRAN and simulated successfully on a commercially available micro-computer, with simulation times less than thirty minutes. The spatial impulse responses for classical problems using square and circular-piston excitation are presented graphically, along with new, innovative, spatial excitation source shapes.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

Due to diffraction effects, acoustic imaging and tissue characterization techniques require information about the insonifying wave at the object's location. The propagation of a pulsed ultrasound wave with arbitrary temporal and spatial shape is not well understood, and requires development of computer-aided analysis and simulation techniques.

A computationally efficient method to model ultrasound propagation from a planar source within a medium has been developed by Guyomar and Powers [Refs. 1,2]. The method is applicable to lossless media, media with a linear frequency loss coefficient such as biological tissue, and media with a quadratic frequency loss coefficient such as liquids and gases. Relying on Fast Fourier Transforms (FFTs) instead of integral solutions that require geometric interpretation makes the method very suitable for computer implementation.

Models for all three cases have been previously developed, coded in FORTRAN, and simulated on an IBM 3033 mainframe computer [Ref. 3]. The purpose of this thesis was to investigate the feasibility of generating micro-computer solutions of the models within a realistic amount of time. As the average execution time was thirty seconds on the mainframe for a 64 x 64 array size, a time limit of thirty

minutes was set as a realistic goal. While the difference between thirty seconds and thirty minutes may appear unacceptable, rapid advances in micro-computer technology is narrowing this margin appreciably.

Since the previously written FORTRAN programs were unavailable, the project required starting from the beginning. A first attempt was made using a commercially available program called MATLAB. MATLAB is an array-oriented program containing built-in functions such as Bessel and Fast Fourier Transforms, as well as the ability to program new functions. Limitations later discovered in MATLAB required starting over, programming the entire project in Microsoft Fortran Version 3.31. The result is a stand-alone, user-interactive program allowing customized computer runs tailored to a specific set of input variables. Only the lossless and linear loss coefficient cases have been implemented, the third case with quadratic loss coefficient is left for future expansion. The goal of thirty minutes was obtained by code optimization and exploiting array symmetry where it existed.

The program was tested and results verified using classical problems such as a square or circular-shaped piston spatial excitation. Once verified, the program was used to compute the spatial impulse response of new excitation shapes. Examples of these include pyramid-shaped pulsed sources and pulsed linear array elements.

II. PROBLEM DESCRIPTION

The problem solved by Guyomar and Powers [Refs. 1,2] uses the geometry of Figure 1.

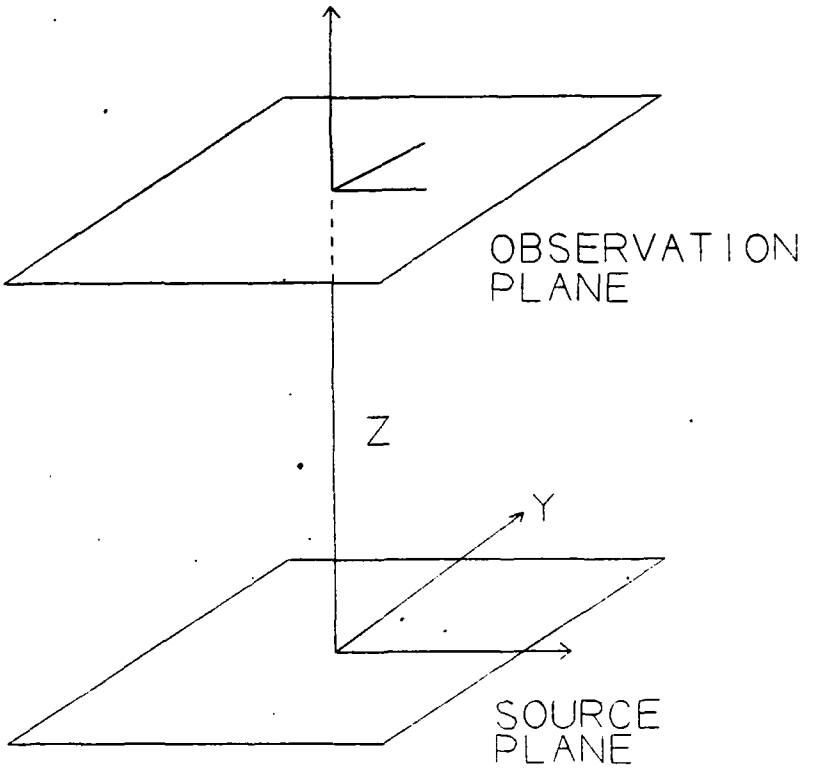


Figure 1. Source and Observation Planes.

Given a separable source of excitation

$$v(x,y,0,t) = s(x,y)T(t), \tag{1}$$

the problem is to find the acoustic potential $\phi(x,y,z,t)$ at an observation point located a distance z above the source plane. A rigidly baffled source plane and linear,

homogeneous media are assumed. It has been shown [Ref. 2] that the potential is given by

$$\phi(x,y,z,t) = s(x,y)T(t) \overset{*}{_x} \overset{*}{_y} \overset{*}{_t} g(x,y,z,t). \tag{2}$$

The * indicates convolution over the variable appearing directly below it and g(x,y,z,t) is the impulse response or Green's function that solves the wave equation and meets the applicable boundary conditions, as shown in the block diagram of Figure 2.

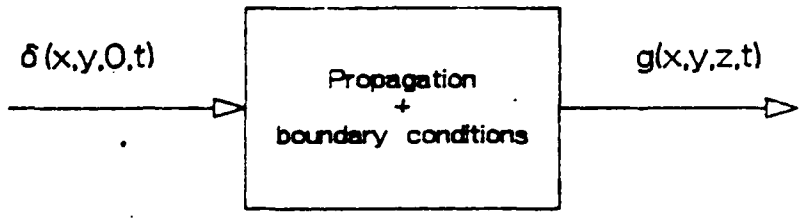


Figure 2. Block diagram of impulse response.

In the spatial domain, ϕ(x,y,z,t) is also given [Refs. 4-8] as

$$\phi(x,y,z,t) = T(t) \overset{*}{_t} h(x,y,z,t) \tag{3}$$

where h(x,y,z,t) is the "spatial impulse response" and is equal to

$$h(x,y,z,t) = s(x,y) \overset{*}{_x} \overset{*}{_y} g(x,y,z,t) \tag{4}$$

where g(x,y,z,t) is the Green's function (or impulse response). In a linear system, the relationship between the

spatial impulse response, h(x,y,z,t), and the Green's function, g(x,y,z,t), is illustrated by the block diagram of Figure 3. Using the two-dimensional spatial Fourier trans-

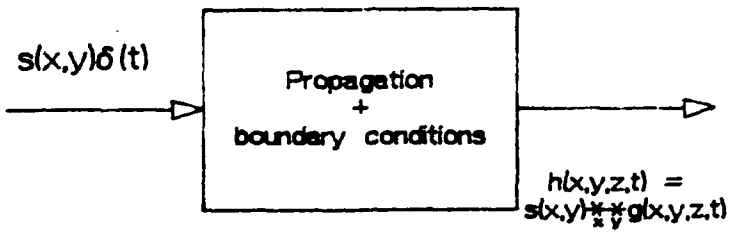


Figure 3. Block diagram of spatial impulse response.

form of Equation 4, $\phi(x,y,z,t)$ can be written as

$$\phi(x,y,z,t) = T(t) \overset{*}{t} \mathcal{F}^{-1}\{\tilde{s}\tilde{g}\} \tag{5}$$

where the tilde notation indicates the transform of the spatial function.

In the spatial frequency domain, the transform of the spatial impulse response is

$$h(x,y,z,t) = \mathcal{F}^{-1}\{\tilde{s}\tilde{g}\}\,. \tag{6}$$

The convolutions in Equations 2 and 4 are difficult to implement on a computer. The technique presented by Equations 3, 5, and 6, and shown graphically in the block diagram of Figure 4, reduce two of the convolutions to multiplication.

From Equation 6 we see that if we interpret $\tilde{s}$ as the angular spectrum of s, then $\tilde{g}$ can be thought to be the "propagation transfer function" associated with propagation

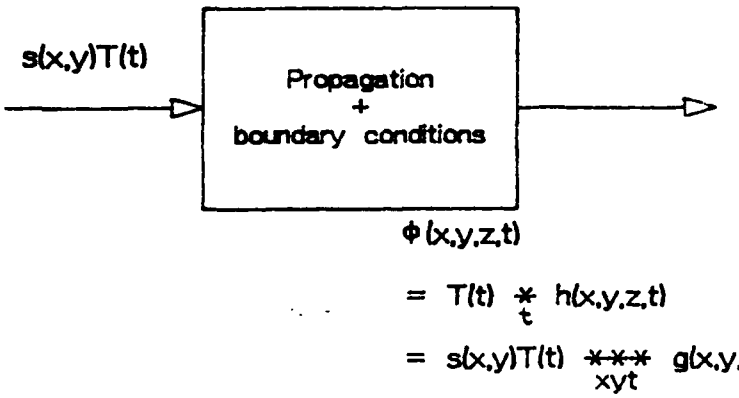


Figure 4. Block diagram of general solution.

in the medium. This propagation transfer function behaves as a time-varying spatial filter that increasingly attenuates the higher spatial frequencies of the source as time increases.

Three propagation models have been developed, one for each type of media. The general technique followed for each model begins with the representative wave equation for the specific medium and finding the Green's function (or the two dimensional spatial transform of the Green's function) which solves the wave equation. Using this Green's function or its spatial transform, the spatial impulse response, h(x,y,z,t), is found from Equation 6. From this, the acoustic potential $\phi(x,y,z,t)$ can be calculated for an arbitrary plane in the positive-z half space using Equation 5.

A. CASE 1: LOSSLESS MEDIA

The results of the transfer function approach for lossless media follow, as published by Guyomar and Powers [Ref. 1].

The wave equation is the Helmholtz equation

$$\nabla^2\phi - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = 0. \tag{7}$$

The Green's function is

$$g(x,y,z,t) = \frac{\delta(ct-R)}{2\pi R}, \tag{8}$$

where

$$R = \sqrt{x^2+y^2+z^2}. \tag{9}$$

For this problem the spatial impulse response is

$$h(x,y,z,t) = \frac{2s(x,y)\overset{*}{}_{x}\overset{*}{}_{y}\delta[t-(R/c)]}{2\pi R} \tag{10}$$

Taking the spatial transform yields

$$\tilde{g}_{p1} = \frac{1}{\pi}\tilde{s}J_0\left(\rho\sqrt{c^2t^2-z^2}\right)H(ct-z), \tag{11}$$

where the tilde indicates the two-dimensional spatial transform and

$$\rho = \sqrt{f_x^2+f_y^2}. \tag{12}$$

The multiplicative constants can be dropped since the results will be normalized to maximum values. The transfer function for propagation in lossless media is then

$$\tilde{g}_{p1} = J_0\left(\rho\sqrt{c^2t^2 - z^2}\right) H(ct - z), \tag{13}$$

where J_0 is the Bessel function of the first kind for order zero. For programming considerations the term

$$z' = \sqrt{c^2t^2 - z^2} \tag{14}$$

is calculated as a unit and identified as ZPRIME in both this paper and the source code.

For a given value of z, one can calculate the value of $\tilde{g}_{p1}$ for each value of time, then take the inverse spatial transform of the product. The result is the spatial impulse response of Equation 8.

B. CASE 2: LOSS COEFFICIENT LINEAR IN FREQUENCY

For media which exhibits a loss coefficient that increases linearly with frequency, the results of Guyomar and Powers [Refs. 1,2] follow.

The wave equation for media with a linear loss coefficient is the telegrapher's equation

$$\nabla^2\phi - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} - A\frac{\partial\phi}{\partial t} = 0. \tag{15}$$

For this case it is the spatial transform of the Green's function that is required. It has been found by an alternate method [Ref. 9] to be

$$\tilde{g}(f_x, f_y, z, t) = \begin{cases} I_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho \le Ac/2 \\ J_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho > Ac/2, \end{cases} \tag{16}$$

where I_0 is the Modified Bessel function of the first kind of order zero. By definition, the propagation transfer function is equal to the transform of the Green's function, so

$$\tilde{g}_{p2}(f_x, f_y, z, t) = \begin{cases} I_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho \le Ac/2 \\ J_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho > Ac/2. \end{cases} \tag{17}$$

The transform of the spatial impulse response is

$$\tilde{h}(f_x, f_y, z, t) = \begin{cases} \tilde{s}(f_x, f_y) I_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho \le Ac/2 \\ \tilde{s}(f_x, f_y) J_0\left[\sqrt{\rho^2 - (A^2c^2/4)}\sqrt{c^2t^2 - z^2}\right] e^{-Ac^2t/2} H(ct - z) & \text{for } \rho > Ac/2. \end{cases} \tag{18}$$

The algorithm for finding the spatial impulse response is similar to the lossless case. The transform is found of the driving function s(x,y) and multiplied by $\tilde{g}_{p2}$ evaluated at each spatial frequency. Taking the inverse transform of this product yields the required spatial impulse response.

III. PROGRAM DESCRIPTION

A. PROGRAMMED IN FORTRAN

Fortran was the language chosen for the program for three reasons: 1) the availability of International Mathematical and Statistical Library (IMSL) [Ref. 10] routines, specifically those used to calculate Bessel functions and one- or two-dimensional Fast Fourier Transforms, 2) the portability of the source code to other brands of microcomputers and compilers, and 3) familiarity of the source code for other students and professionals. Microsoft Fortran Version 3.31 was the specific brand of compiler used.

B. ADVANTAGE OF SYMMETRY

Most of the arrays generated by the program are symmetrical. This was used as an advantage in reducing the number of calculations required and therefore the execution time of the program. As operation of the program is described, the areas where symmetry has been used are identified. Figure 5 is provided to aid in these explanations by providing a "map" of an array base. Figure 5 shows a 64 x 64 array base situated on the X,Y plane and divided into four quadrants. Note that the quadrants are unequal in

size, i.e., quadrant II includes rows 1 through 33 and columns 1 through 33 for a 33 x 33 "sub-array" while quad-

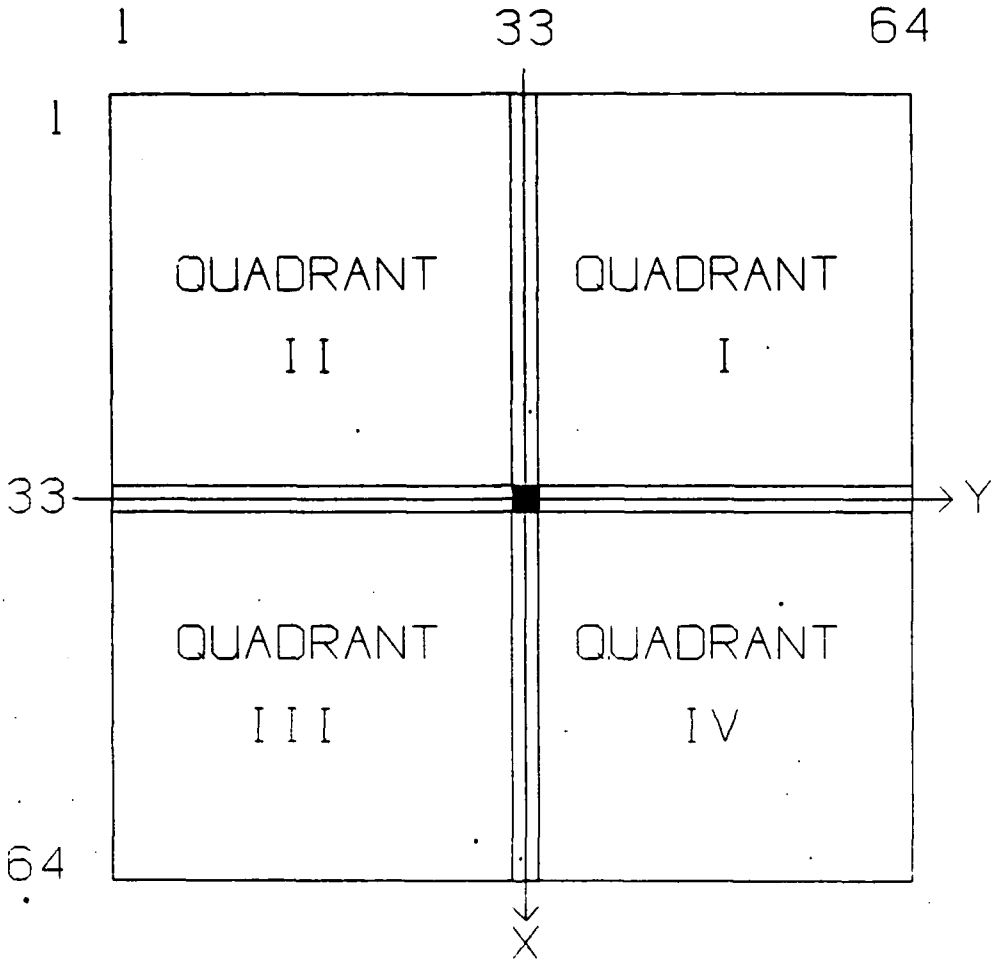


Figure 5. Base array configuration.

rant I includes rows 1 through 33 and columns 33 through 64 for a 32 x 32 "sub-array". Similarly, when the left side (quadrants II and III) is to be calculated as a whole, the calculations cover all 64 rows and columns 1 through 33. The different sizes are required to ensure that only column 33 contains "peak" information and also that element (33,33) is the physical center of the array. The importance of this

and the reason why it is required will be explained later. In-line code provides the required "flipping" actions to expand one or two quadrants into a full 64 x 64 array of information.

C. BLOCK DIAGRAM

Figure 6 is a block diagram illustrating the information flow through the program. The program is explained block by block in the following text.

1. User input

The program starts by obtaining information specific for the problem to be run. Using on-screen prompts, the user selects a problem name, filter type, excitation shape and size, and values for Z, RHO, and MAXTIME. If filter g_{p2} is selected for lossy media, the user is further prompted for a value of the loss coefficient A.

The problem name assigned must be eight characters or less and conform to MS-DOS filename specifications. Once entered, the program concatenates three different extensions to the filename. This creates the names of the three output files generated by the program.

The first file created is <filename>.IMP, where <filename> is the filename entered by the user. The program stores the 64 x 64 spatial excitation array, s(x,y), in the default directory under this name prior to taking its two

dimensional FFT. It is provided should the user want to graphically show the spatial excitation shape.

The second file created is <filename>.TXT. This is a printable text file which summarizes the user's inputs for a particular problem.

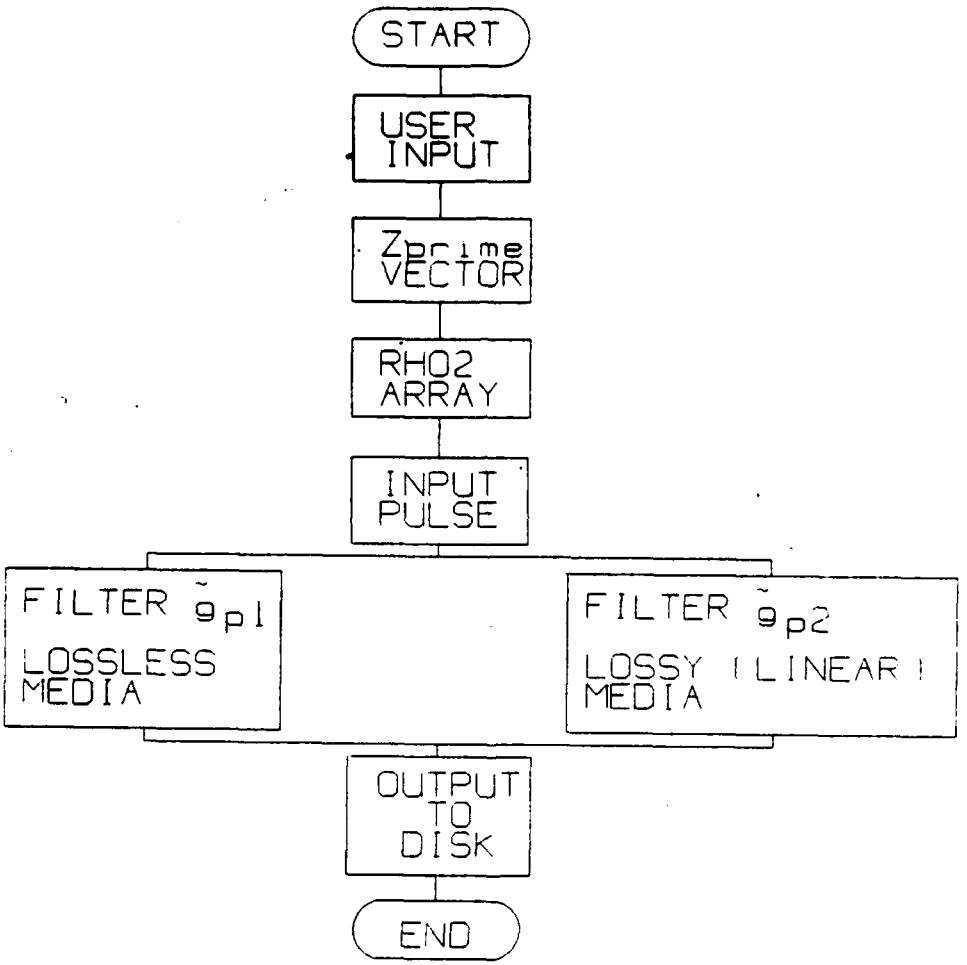


Figure 6. Program block diagram.

The third and last file created is <filename>.DAT. This file contains the final 64 x 64 array which is the spatial impulse response for the given problem.

All three files are in ASCII text format. They can be printed using the DOS PRINT command, put into a word processing program for printing, or put into a graphics program for conversion into a picture. All figures generated for this thesis were produced by translating the <filename>.IMP or <filename>.DAT file to MATLAB format using the translation program supplied with MATLAB. The converted files were then read into MATLAB and plotted using the MESH command. As new, improved, graphic routines for microcomputers are introduced, a future improvement would be incorporation of graphic routines into the program.

As the program is written, fourteen different source excitation configurations are allowed, nine of which have been implemented. The choice of fourteen different configurations was arbitrary, providing a good selection of choices and ease of future expansion. Each of the remaining five configurations can be implemented by adding its name to the screen formatting code and writing the FORTRAN code to generate the excitation shape. The desired source excitation shape is selected by entering the selection number found next to the shape's description.

The variable RHO represents the radial distance in the spatial frequency domain as shown in Equation 12. It is identified by the Greek character ρ in Equations 11 through 13 and 16 through 18 but will be referred to as RHO to match the source code. RHO is used to create a 1 x 64 vector

called RHO1 which in turn is used to generate a 64 x 64 array called RHO2. The vector RHO1 is unique since it must start approximately at -RHO and increment until it is equal to 0.0 in RHO1(33). From here it continues to increase until it equals +RHO in RHO1(64). Since the vector is of even length, it cannot be formed simply by dividing the entire range of RHO (-RHO to +RHO) by 64 and using this value as the increment. To do so would create equal values in RHO1(32) and RHO1(33) which defeats the requirement that RHO1(33) contain singular information, in this case the value 0.0.

The largest value of time, in seconds, for the time dimension is represented by the variable MAXTIME. For each problem, time starts at the instant that the wave front first arrives at the observation plane. This is when time = Z/C, where C is the sound velocity (1500 meters/second), and increments linearly until it equals MAXTIME. To prevent a run-time error, the user is prompted for a value of MAXTIME which is greater than Z/C seconds.

The height of the observation plane above the X,Y plane is represented by Z (in meters). It is typically a small number, several hundredths of a meter, or can be set to zero. A value of 0.1 meters (10 cm) was used for debugging the program and producing the results found in the Numerical Simulations section.

2. Generate ZPRIME vector

At the completion of user input, the program starts to generate the required vectors and arrays. The first of these, the TIME vector, is initialized and then used to generate the ZPRIME vector. Both vectors are 1 x 64 in size and represent a series of time steps starting at time = Z/C and increasing to MAXTIME.

The program contains a variable called STEP. Its default value of 4 (which can be changed) sets rows 1 through 4 of the final RESULT array to zero after all calculations have finished. This simulates the step function H(ct-z). Since any information contained in the first four rows is lost, these values need not be calculated. For this reason, TIME starts at TIME(STEP+1), or TIME(5), with a value of Z/C seconds, and increments linearly until it reaches MAXTIME in TIME(64). The smooth linear progression of time from Z/C to MAXTIME is obtained using the following code segment:

```
INC = (MAXTIME - TS)/REAL(NROWS-STEP-1)
DO XX I = (STEP+1),NROWS
   TIME(I) = TS + (I-STEP-1) * INC
CONTINUE
```

where TS = Time Start = Z/C, NROWS = 64, and STEP = 4.

As an example let MAXTIME = 1.0E-4 seconds and Z = 0.1 meters as determined by the user. The quantity TS equals 6.666667E-5 seconds and this value is stored in TIME(5). Each element of TIME is increased by INC =

5.649718E-7 until it equals MAXTIME in TIME(64). The elapsed time for this particular simulation is MAXTIME - TS = 3.333333E-5 seconds.

The ZPRIME vector is created using values found in the TIME vector as shown in Equation 14. Each value of TIME is squared and multiplied by C^2. Subtracting Z^2 from this product and taking the square root produces a ZPRIME element. Continuing with the previous example, two sample ZPRIME elements are

$$ZPRIME(6) = SQRT(TIME(6)^2 * C^2 - Z^2) = 1.304644E-2$$

$$ZPRIME(7) = SQRT(TIME(7)^2 * C^2 - Z^2) = 1.848934E-2$$

The algorithm loops until ZPRIME(STEP+1) through ZPRIME(64) have been calculated.

3. Generation of RHO2 array

The next significant step is formation of the RHO2 array. First the RHO1 vector is created by calculating an increment and base value using the code

```
INC = REAL(RHO)/REAL(NCOLS/2-1)
BASE = REAL(-RHO1) - INC
```

Assuming a value of 200 for RHO, INC = 6.451613 and BASE = -206.451613. This results in a RHO1 vector starting with -206.451613 in RHO1(1), with each successive element incremented by INC. This method ensures a value of 0.0 in RHO1(33), -200.0 (-RHO) in RHO1(2), and +200.0 (+RHO) in RHO1(64). As with the TIME vector, simply setting the RHO1 vector to range from -RHO in RHO1(1) to +RHO in RHO1(64)

would result in identical values in RHO1(32) and RHO1(33). This would not meet the requirement that RHO1(33) contain singular information.

After completing the RHO1 vector, its values are used to generate the RHO2 array. First the RHO1 vector values are placed in row 33 of RHO2 and its transpose in column 33. The program then fills in the blank array elements in quadrants II and III using the Pythagorean theorem with the values found in the respective row 33 and column 33 positions. Continuing with the example using RHO = 200, the value -206.451613 would be found in elements RHO2(33,1) and RHO2(1,33) (column major format). Using the Pythagorean theorem, element RHO2(1,1) would receive the value 291.966671. This represents the value of RHO as calculated radially from the central position of RHO2(33,33).

Since the RHO2 array is symmetrical, only quadrants II and III (left side) must be calculated. Although the time saved here is small, the algorithm using these values later in the program needs only the values found in the left half. Therefore the right side values are not calculated.

4. Spatial excitation generation

Generating the spatial excitation array is the next step. The first five selections represent single piston spatial excitations centered about position (33,33) on a 64 x 64 base array called PULSE. The five shapes available are

the square piston, circle piston, gaussian, pyramid, and truncated-pyramid. All five shapes require the user to specify the width (or diameter) of the excitation. The value of width must be an odd integer to ensure symmetry about PULSE(33,33), and less than or equal to 63 since the base array size is 64 x 64. Small values between 9 and 15 provide excellent results.

The square and circle selections create a simple spatial excitation with amplitude = 1.0. A resolution problem exists with the circular excitation. Simply, it is impossible to represent a circular shape using a small number of square elements. The larger the diameter, the smoother the outer surface becomes, but with a 64 x 64 array size the range of available diameters is limited. The only solution is an increase in array size to 128 x 128 or larger. This is an excellent area for future expansion as micro-computers become faster and less memory limited.

The Gaussian selection creates a circular-shaped excitation with amplitude following a gaussian distribution. The spatial excitation is truncated at the user specified diameter, and its width measured at the 1/e point. This excitation, as well as the pyramid and truncated-pyramid, represent non-piston sources with spatial variations in amplitude.

The pyramid creates a four-sided tapered spatial excitation with a peak amplitude of 1.0 at the center and

tapering off until it equals 0.0 at the base. The size of the base is determined by the input value of width. For example, a pyramid excitation of width 11 would have its center at PULSE(33,33) and occupy the region bordered by rows 29 through 37 and columns 29 through 37. The amplitude would decrease linearly and equal zero at the outer edges of the above region. The truncated-pyramid is similar except that the amplitude tapers off at a slower rate until it equals 0.0 at the array's outer edges. At the specified value of width, the pyramid is truncated, resulting in a "square house" shape with tapered roof and square sides.

The remaining selections all represent linear arrays of five elements each. An individual element size of 5 x 5 located on 10 unit wide center points was selected for convenience. Shape selections for the individual excitations are square, circular, and gaussian. Two additional arrays also use 5 square elements, but the amplitude of the elements are stepped or parabolic in shape. The stepped array has a center element of amplitude = 1.0, two adjacent elements of amplitude = 2/3, and two outer elements of amplitude = 1/3. The five elements thus create a stepped appearance: 1/3, 2/3, 1.0, 2/3, 1/3. The parabolic array is similar to the stepped array, except the amplitudes follow a parabolic shape.

After the spatial excitation is generated, the two dimensional Fast Fourier Transform is taken using the IMSL

routines FFT3D and FFTCC. The routine FFT3D computes the FFT of a complex-valued two-dimensional array by passing first the columns and then the rows as individual vectors to the routine FFTCC which computes the FFT of the vector argument. The array returned by FFT3D is complex and has its peak information located at the outer four corners of the array, with the largest value in PULSE(1,1). The array is shifted to bring this peak information to the center using the subroutine SHIFT. This subroutine swaps quadrant I with quadrant III and quadrant II with quadrant IV, thus transferring the largest value from element PULSE(1,1) to element PULSE(33,33). This is why it is so important to arrange all arrays so their central value is in element (33,33).

5. Filter $\tilde{g}_{p1}$ for lossless media

From this point the program branches based on the filter type selected. The algorithm for the lossless case, using filter $\tilde{g}_{p1}$, is to take a value of ZPRIME, multiply the RHO2 array by this value and pass each product as the argument to the IMSL subroutine MMBSJN, where MMBSJN calculates the Bessel function of the first kind of order zero for a real number argument. The result returned from the Bessel function is multiplied by the respective element in array PULSE and stored in array WORK. The next value of ZPRIME is selected and the calculations again performed.

As this portion of the program is the most time consuming, two specific actions were taken to optimize the

code. As previously explained, the number of rows identified by STEP are zeroed in the final array. Because of this, calculations start at row (STEP+1), thus saving the calculation of 4 rows for each iteration. Also, since the arrays RHO2 and PULSE are symmetrical in all four quadrants about element (33,33), only quadrant II needs to be calculated. With the default 64 x 64 array size and STEP = 4, only 59 (64-STEP-1) subarrays of size 33 x 33 must be calculated for a total of 64,251 array elements. This compares favorably with the 262,144 array elements that would require calculation if the entire 64 x 64 array was calculated 64 times. It is important to note that the single most time-consuming process is the calculation of the Bessel function.

After each 33 x 33 subarray is calculated, inline code is used to fill the entire WORK array. Quadrant II is flipped to fill quadrant III, then the entire left side is flipped to fill the right side.

At this point the inverse two-dimensional Fast Fourier Transform of the WORK array needs to be taken. But instead of passing the entire array as an argument to a two dimensional inverse FFT subroutine, the array is processed one column at a time. Using the IMSL subroutine FFT2C, which computes the inverse FFT (or FFT) of a complex-valued vector, each of the 64 columns are passed individually as a vector. After all the columns are transformed, only one

row, row 33, is passed as an argument. Assuming a symmetrical transducer, row 33 contains information from the central line across one axis of the observation plane. The inverse transform of this row represents the field values along this line for the specific value of ZPRIME, and it is this information that is used to sequentially build the final array called RESULT. The 59 iterations completed by the above algorithm will each supply one row, corresponding to an instant of time, in array RESULT. Specifically, with the default value of STEP, rows 5 (STEP+1) through 64 (NROWS) are filled.

After completing row 64, the number of rows identified by STEP are set to zero as they do not contain any information. The entire RESULT array is then saved to disk, and the program terminates normally.

6. Filter $\tilde{g}_{p2}$ for lossy media

In the second case, using filter $\tilde{g}_{p2}$ for media with linear loss coefficient, the algorithm has two additional steps. The program implements Equation 17 and therefore must test each argument for RHO greater than AC/2. For RHO > AC/2, the argument is passed to the IMSL Bessel function subroutine MMBSJN as in the lossless case. But, if RHO $\leq$ AC/2, the argument is instead passed to the IMSL subroutine MMBSIN which calculates the modified Bessel function of the first kind of order zero. The result returned from either Bessel function is multiplied by the

respective element of PULSE. Then, as shown in Equation 17, the product is multiplied by an exponential term prior to storing in array WORK. This is the second difference in the lossy case.

The algorithm loops, as in the lossless case, to fill the entire RESULT array. After saving the RESULT array on disk, the program terminates normally.

IV. NUMERICAL SIMULATION

After careful debugging and testing on a point by point basis, the program was used to calculate the spatial impulse response for various source excitations. For each excitation shape, both the lossless and lossy case were calculated and the results plotted. The plots show the amplitude of the wave plotted against cross-direction and time at an observation plane located above the X,Y plane. In all calculations the height of the observation plane, Z, is 10 cm above the source plane. The value of A used for the lossy cases was 8.0E-3. This value was selected as a compromise and arrived at using trial and error. Smaller values of A caused only minor variations in shape while larger values caused gross distortion. All problems, unless noted otherwise, are calculated for a MAXTIME of 1.5E-4 seconds. Thus, with a Z value of 10 cm, each result runs from a starting time = Z/C = 6.666667E-5 seconds to 1.5E-4 seconds, a duration of 8.333333E-5 seconds.

As previously explained, all plots were obtained using the MATLAB MESH command. Two limitations exist with the MESH routine that require further explanation. The first limitation is the way MATLAB sizes the plot. MATLAB uses input array values to fill a window of pre-determined size. This fixes the height of the plot so an excitation with

amplitude equal to 1.0 will plot identically to an excitation with amplitude other than 1.0. The second limitation builds on the first: MATLAB does not show numerically scaled axes when plotting an array of data. If it did, the different amplitudes for identically shaped plots would be apparent.

An alternative way to obtain numerical information was found. Using the PLOT command, MATLAB has the ability to plot a vector in the X,Y domain with numerically scaled axes. Putting an array of data into PLOT creates a side view, plotting successive "slices" of the array with time increasing in the positive X direction. Using this technique, amplitude values could be obtained for a given plot. Since shape is the primary interest, amplitude information obtained in this manner was adequate. Three plotted arrays (Figures 9, 11, and 15) are included for illustration.

Execution times were all less than the stated goal of thirty minutes, with an average of twenty-five minutes. These times were obtained using an AT&T 6300 micro-computer operating at 8 MHz. An 8087 numerical co-processor chip was installed and used to increase performance. Similar performance can be obtained with fast IBM XT compatibles or AT class micro-computers. The next generation of micro-computers utilizing the Intel 80386 microprocessor are the logical choice for continued program development.

A. CLASSICAL EXAMPLES

The first two source shapes presented are the square piston and circular piston.

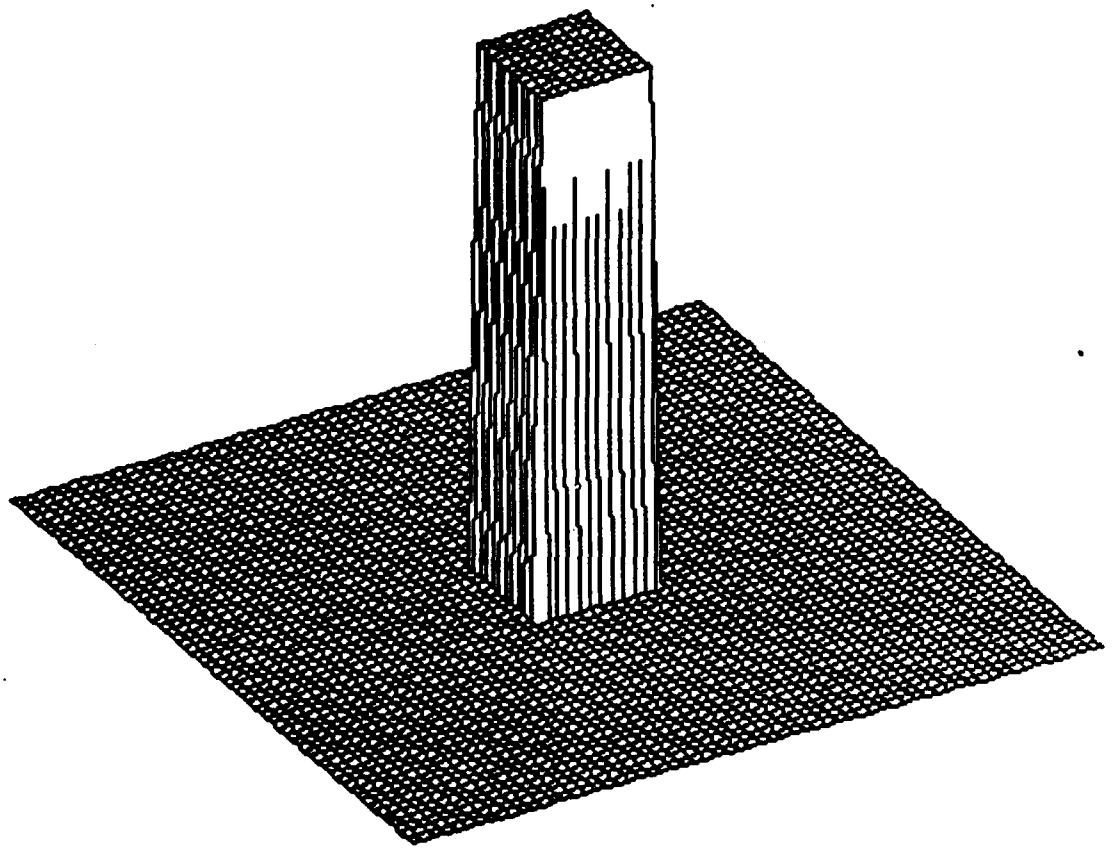


Figure 7. Square piston spatial excitation.

Figure 7 shows graphically the source excitation shape for a square transducer of width 11 and amplitude 1.0. The excitation is centered over the base array's central element, PULSE(33,33). In Figure 8 the calculated spatial impulse response for the square transducer in lossless media is shown, as is the array's orientation with time and space.

This orientation of time and space applies to all spatial impulse response plots. The origin of the time axis starts at Z/C where the potential is known to be a replica of the excitation source shape for lossless media. As time progresses, the potential is reduced until it forms two outward traveling "tails". As expected, the "tails" are square

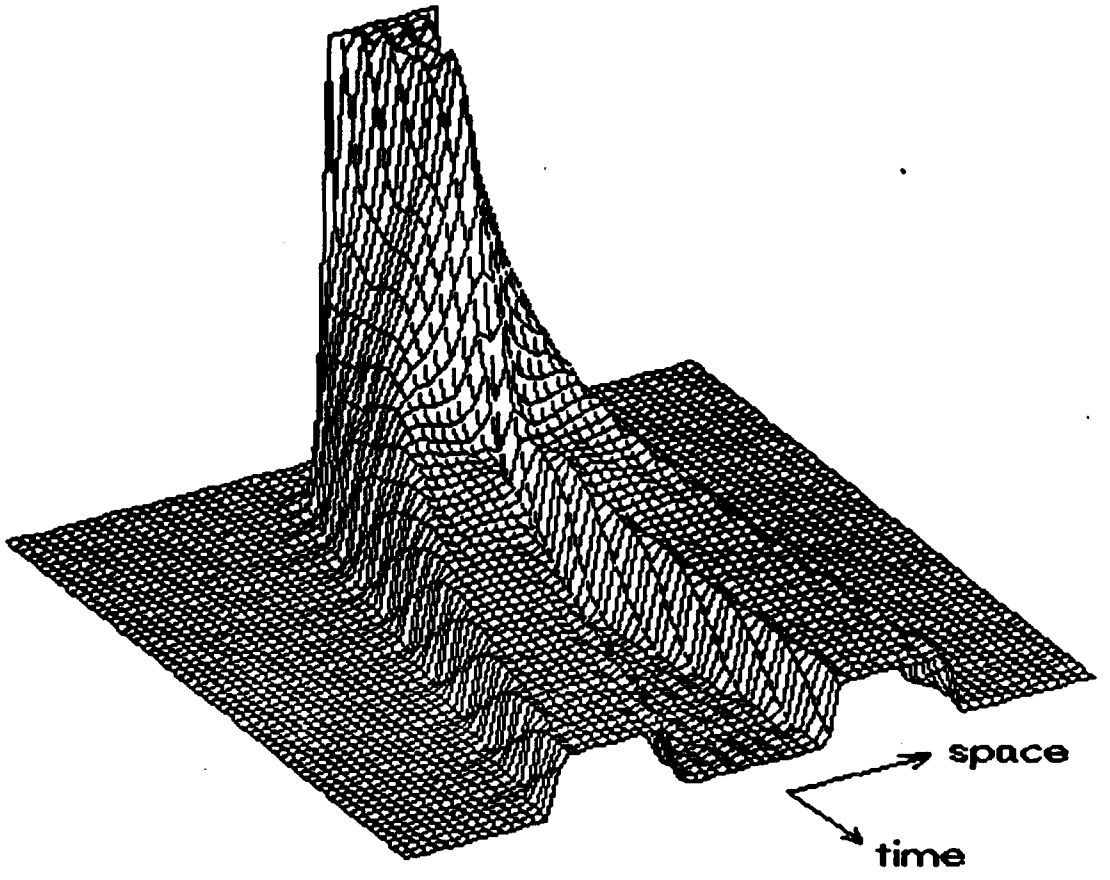


Figure 8. Field for square transducer (lossless case).

to match the spatial excitation shape, and slowly attenuate with time in the lossless case. Numerical information can be obtained from Figure 9 which is the side view of the

plot in Figure 8. The step function is clearly visible as is the jump in potential to 1.0 in row 5. From this point the potential peaks at a value slightly greater than 1.0 and then drops rapidly. At infinity the tails should reduce to zero.

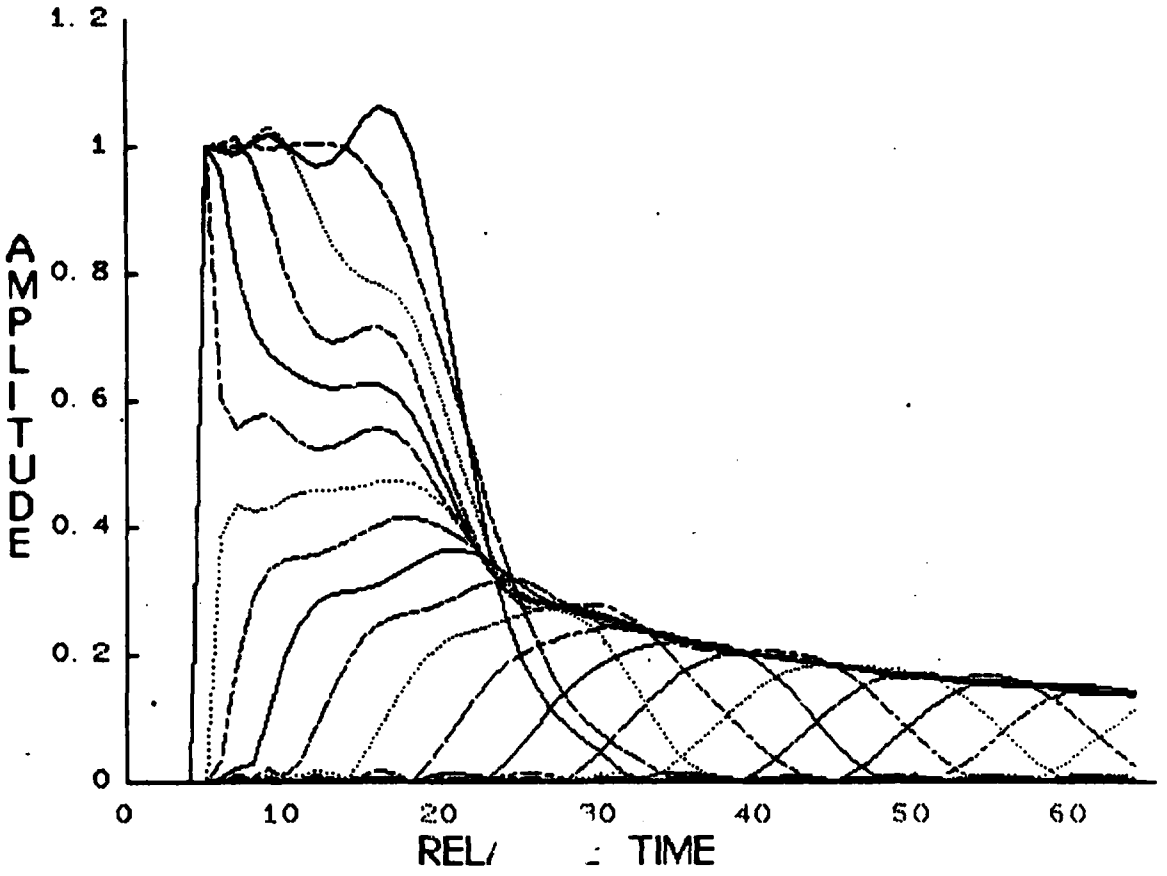


Figure 9. Side view for lossless case.

In Figure 10 the same excitation has been analyzed for the lossy case. Comparing the shapes in Figures 8 and 10 reveals two distinct differences. The "tails" are no longer square but have taken on a triangular appearance. Larger

values of A will produce a sharper triangular shape. The second major difference concerns the region between the tails. In the lossless case this region was at zero potential. In the lossy case, the region does not approach zero but instead fills in, developing a shape of its own.

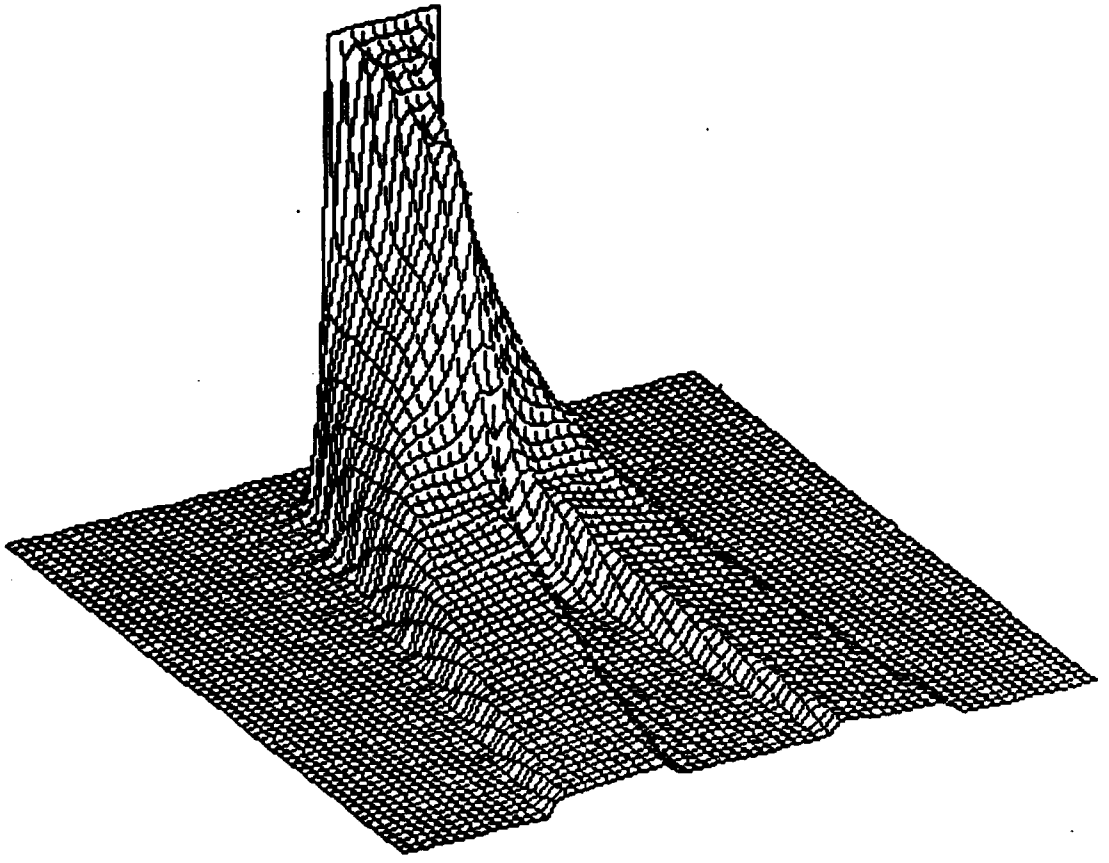


Figure 10. Field for square transducer (lossy case).

Another major difference is discovered when comparing Figures 9 and 11. Where the initial amplitude for the lossless case was 1.0, the amplitude of the spatial excitation, the initial amplitude for the lossy case has been reduced to

approximately 0.30. This reduction in amplitude was observed for all test cases with a linear loss coefficient.

The results obtained using a circular transducer are similar to those obtained with the square transducer. The spatial excitation for a circular transducer of width 15 and amplitude 1.0 is shown in Figure 12. Here a width of 15 was

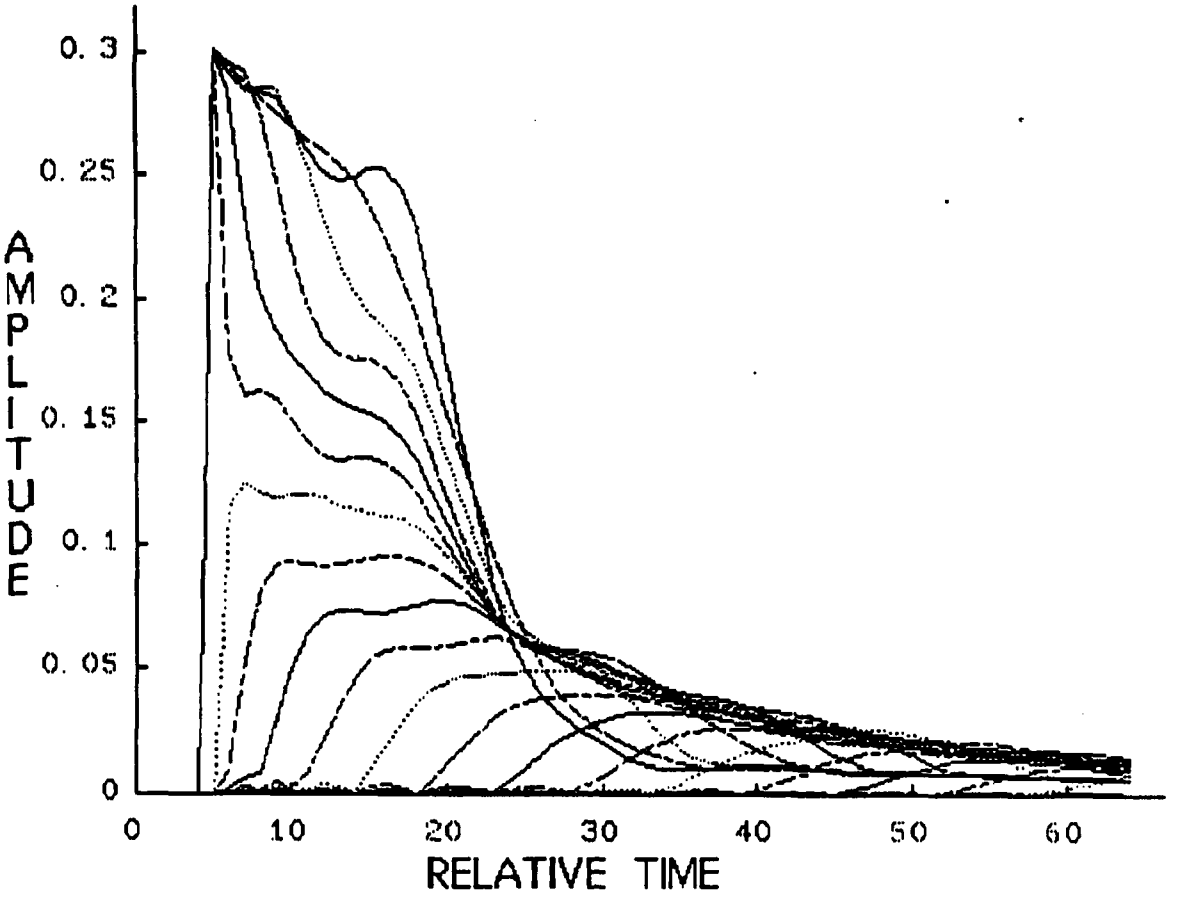


Figure 11. Side view for lossy case.

used to improve the resolution of the circular excitation. The spatial impulse response using this source is shown in Figure 13. Comparison of Figures 8 and 13 illustrates the

differences between the spatial excitation shapes. The major difference is the tails, which now have a circular cross-sectional shape. Some minor variations can also be observed in the large initial response early in time. Figure 14 shows the spatial impulse response, using the circular excitation of Figure 12, for the lossy case. The previous comments for the square transducer concerning the

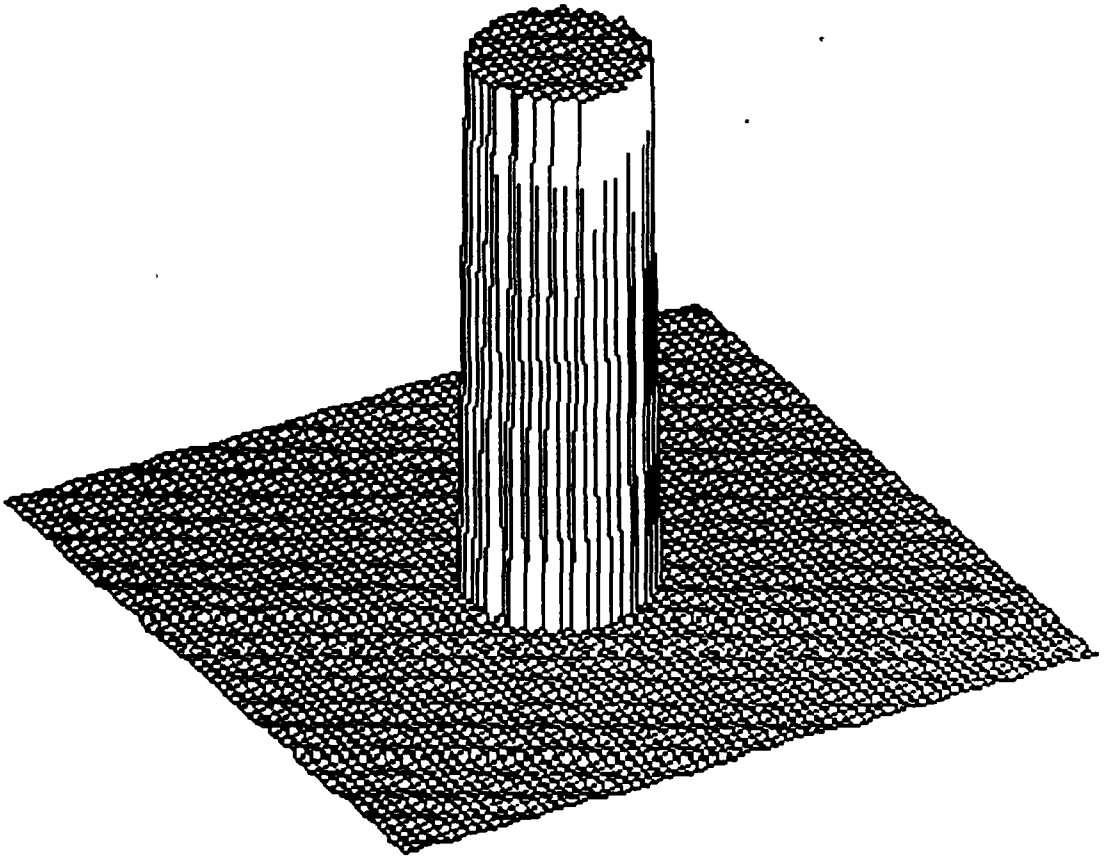


Figure 12. Circular piston spatial excitation.

difference in amplitudes and shape of the "tails" for the lossless and lossy cases also apply to the circular

transducer. Figure 15 is a side view plot of Figure 14 showing the reduction in amplitude is also present with the circular transducer.

The results obtained when using the square-piston and circular-piston spatial excitation for both lossless and lossy media compare favorably with those obtained by Guyomar and Powers [Refs. 1-3,9]. Based on these results, it was determined that the program correctly implemented the propagation models, and therefore could be used to compute the spatial impulse response using new types of excitation.

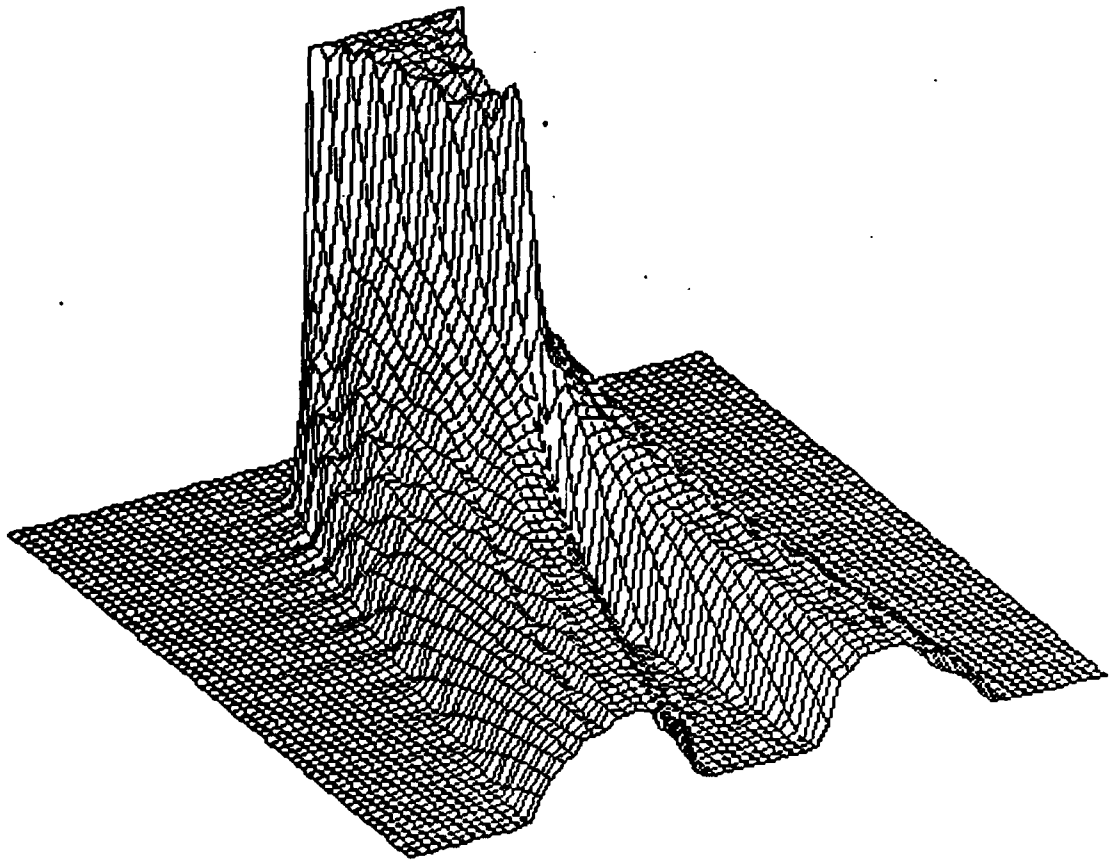


Figure 13. Field for circular transducer (lossless case).

B. NEW EXCITATION SHAPES

The program has to ability to analyze any excitation source shape that will fit on the 64 x 64 base array. Two new single excitation shapes and four multiple element excitation shapes are in the program. The results for three of

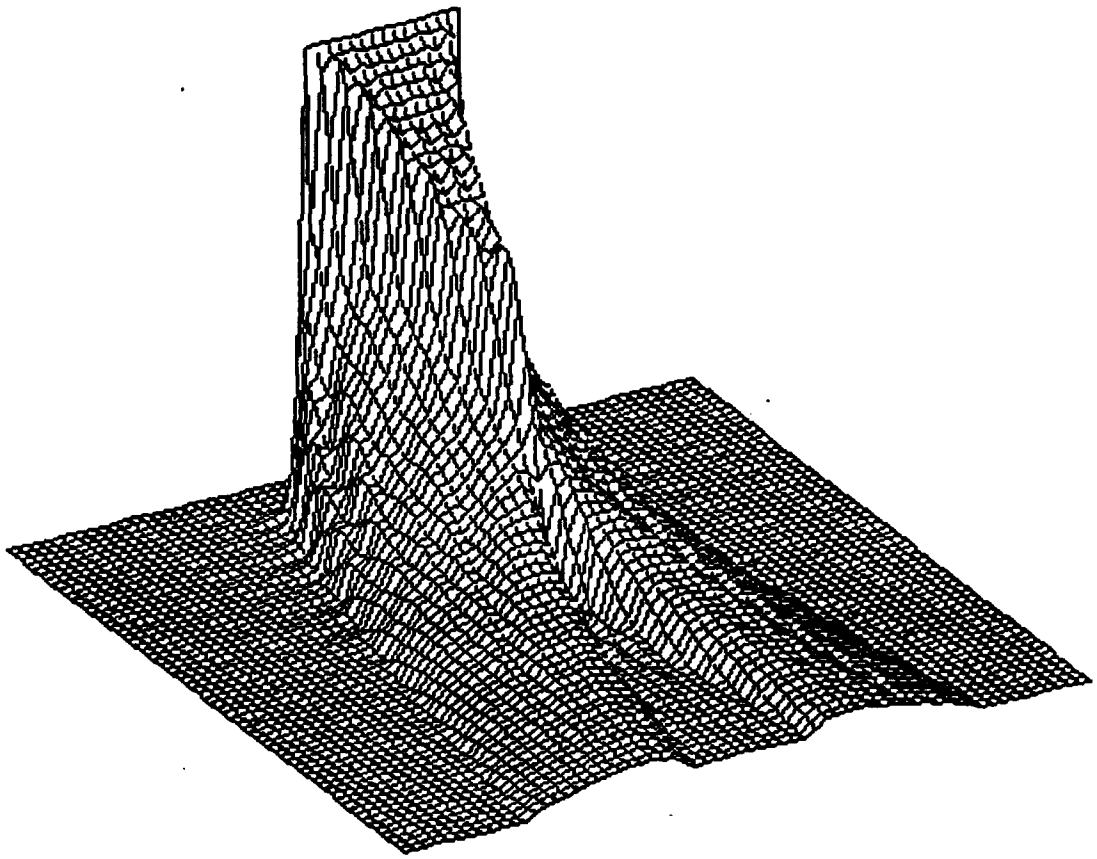


Figure 14. Field for circular transducer (lossy case).

these, the pyramid, the 5 element linear array with constant amplitude, and the 5 element linear array with stepped amplitude are included.

A pyramid shaped spatial excitation of width 11 and amplitude 1.0 is shown in Figure 16. The computed spatial impulse response for this excitation in lossless media is shown in Figure 17. Since this particular waveshape has a

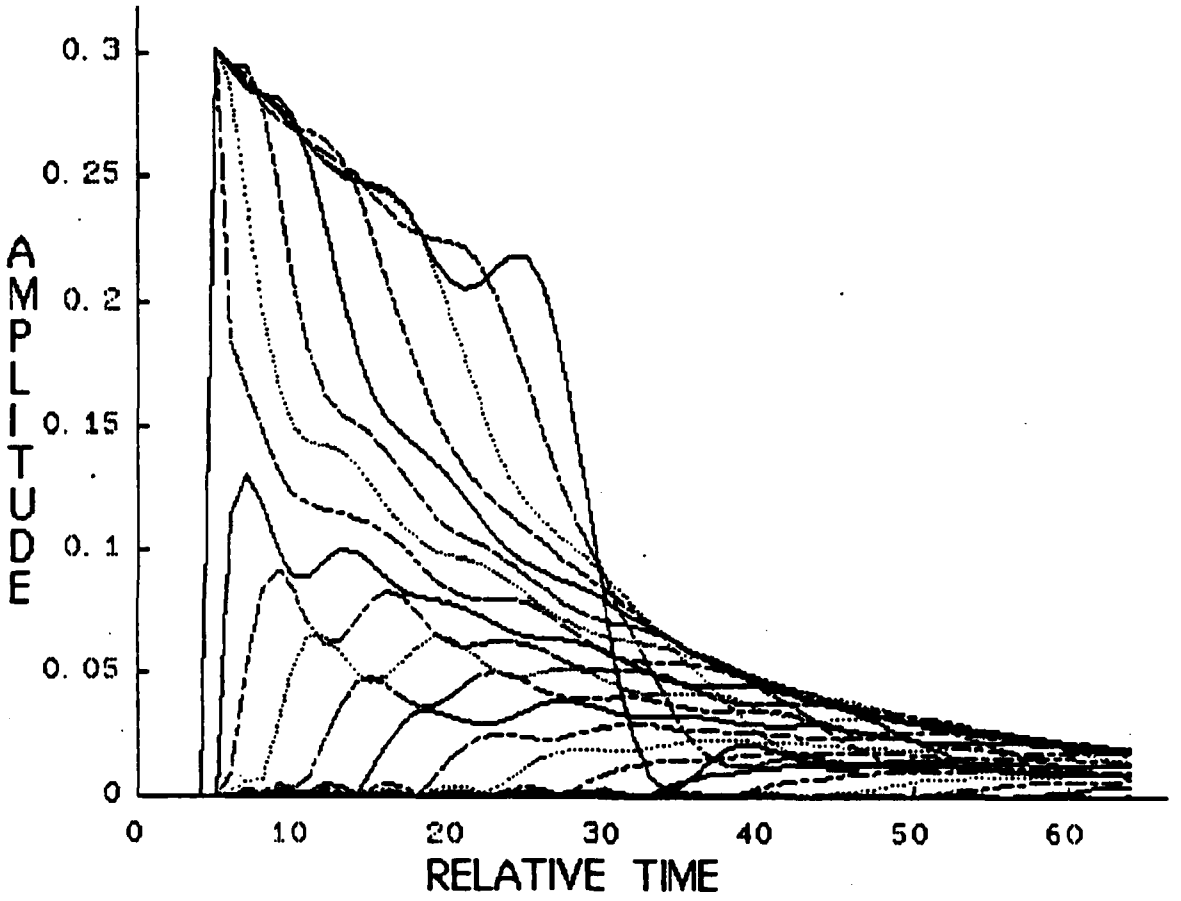


Figure 15. Side view for lossy case.

low spatial frequency content, the shape of the wave stays much the same for small values of time. Figure 18 shows the spatial impulse response for the same pyramid excitation in lossy media.

Figure 19 shows the 5-element linear array configuration. For this array the elements are each 5 x 5 with their centers positioned 10 units apart. This produces a space of 5 units between each element. The calculated spatial impulse response for this excitation is shown in Figure 20.

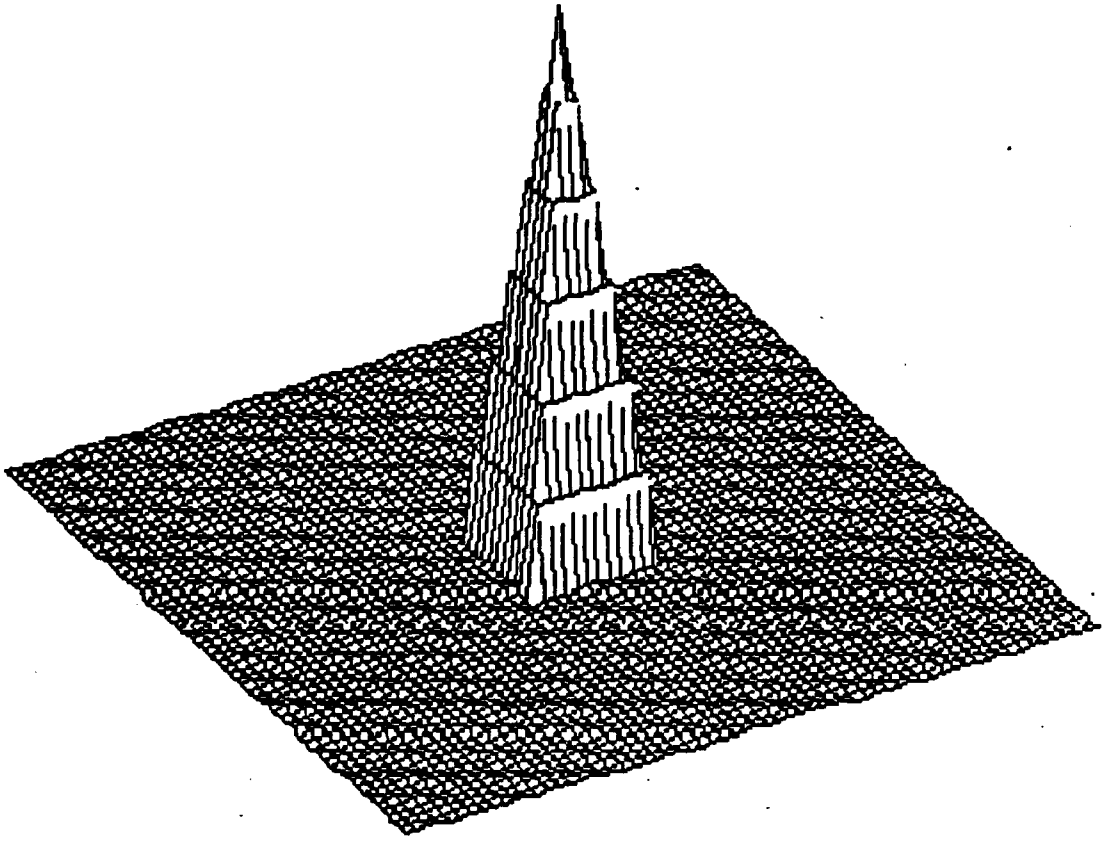


Figure 16. Pyramid shaped spatial excitation.

The shape for each individual excitation is similar to the single excitation response in Figure 8 until the tails start to spread out and interfere with each other. In the lossy

case, Figure 21, the pattern is similar but with increased interference in the region between the individual responses.

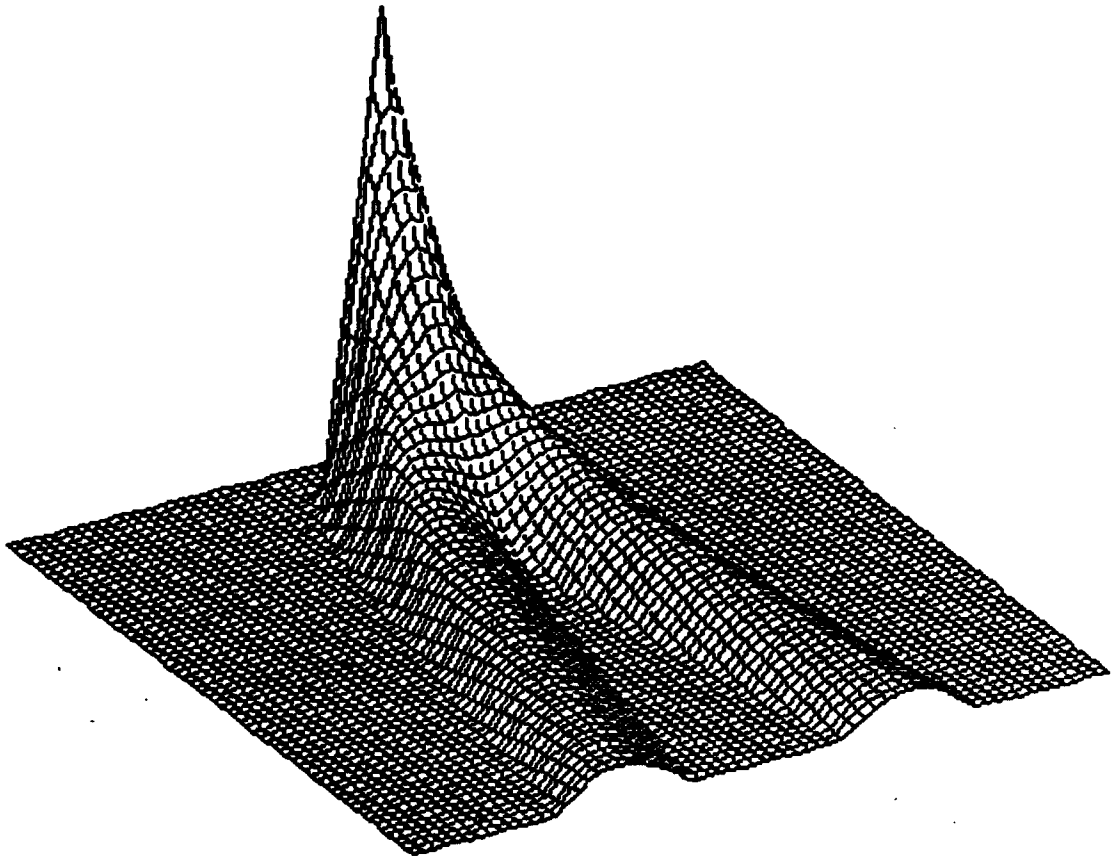


Figure 17. Field for pyramid excitation (lossless case).

The spatial excitation for the last test case presented is shown in Figure 22. This is a 5-element linear array similar to Figure 19 but with stepped amplitude. The center element has amplitude of 1.0 as before. The two elements adjacent to the center have amplitudes of 2/3, and the outer two elements have amplitudes of 1/3. This particular case is presented to show the flexibility of the program. Figure

23 shows the spatial impulse response in lossless media. The results are similar to those of Figure 20 except for the dominance of the center element. Figure 24 shows the spatial impulse response for the same excitation in lossy media.

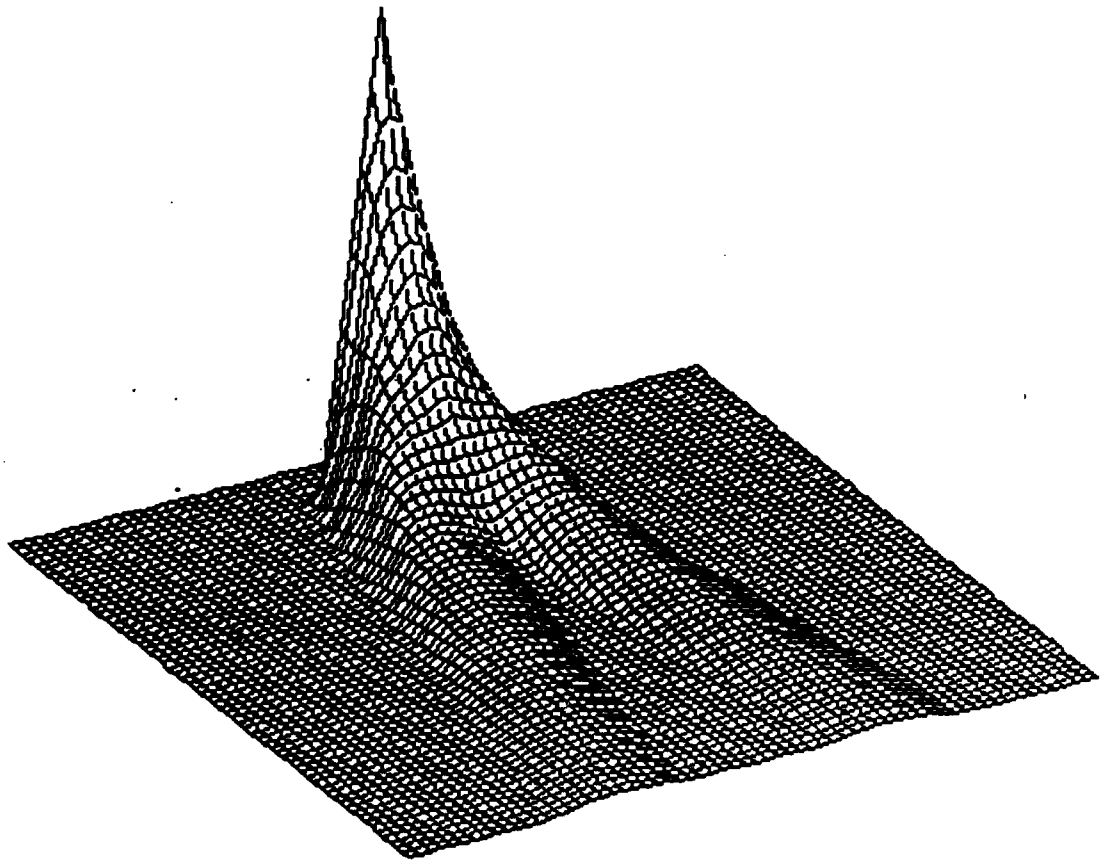


Figure 18. Field for pyramid excitation (lossy case).

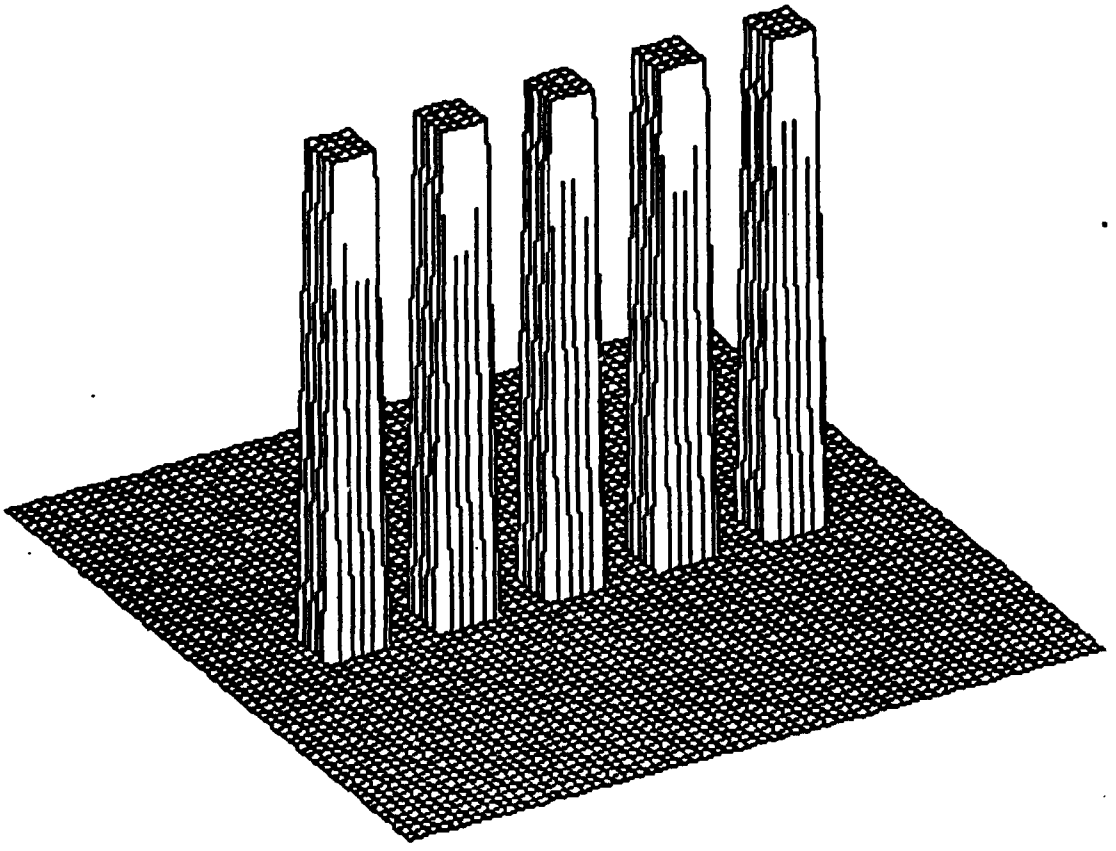


Figure 19. Excitation for five element linear array.

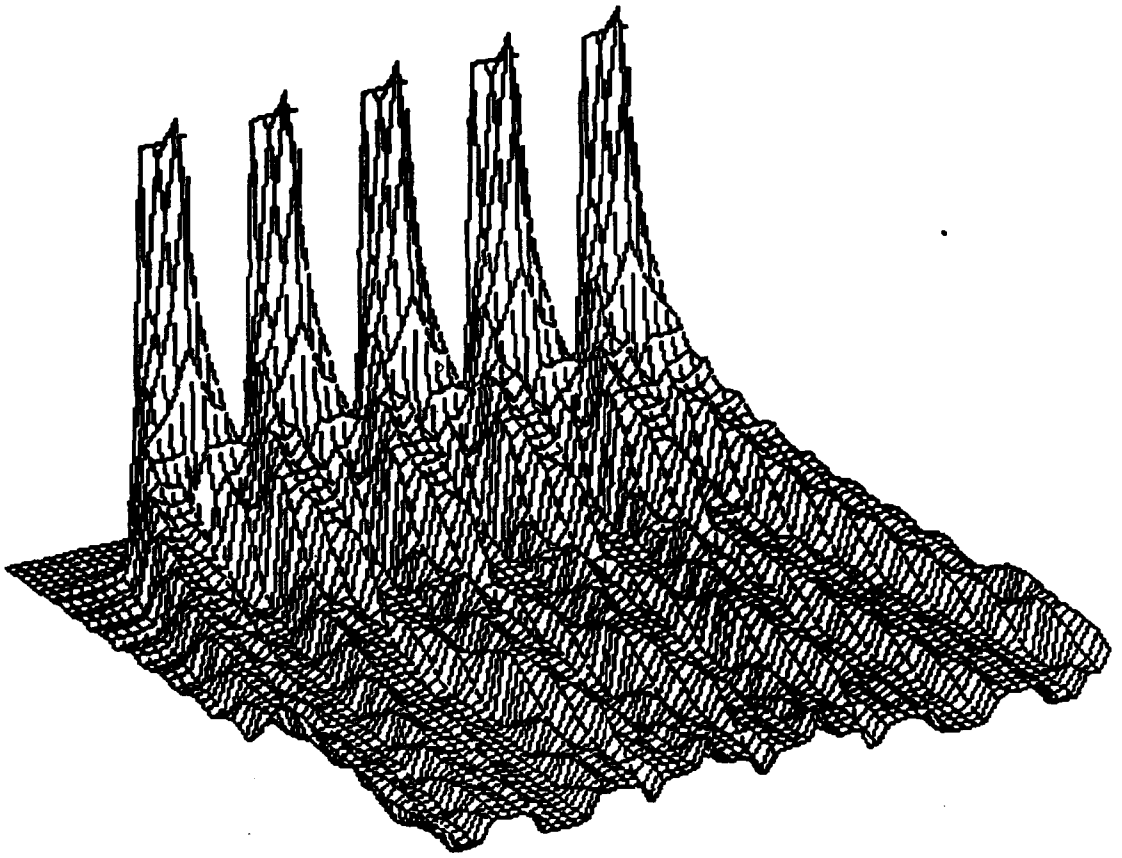


Figure 20. Field for five element linear array (lossless case).

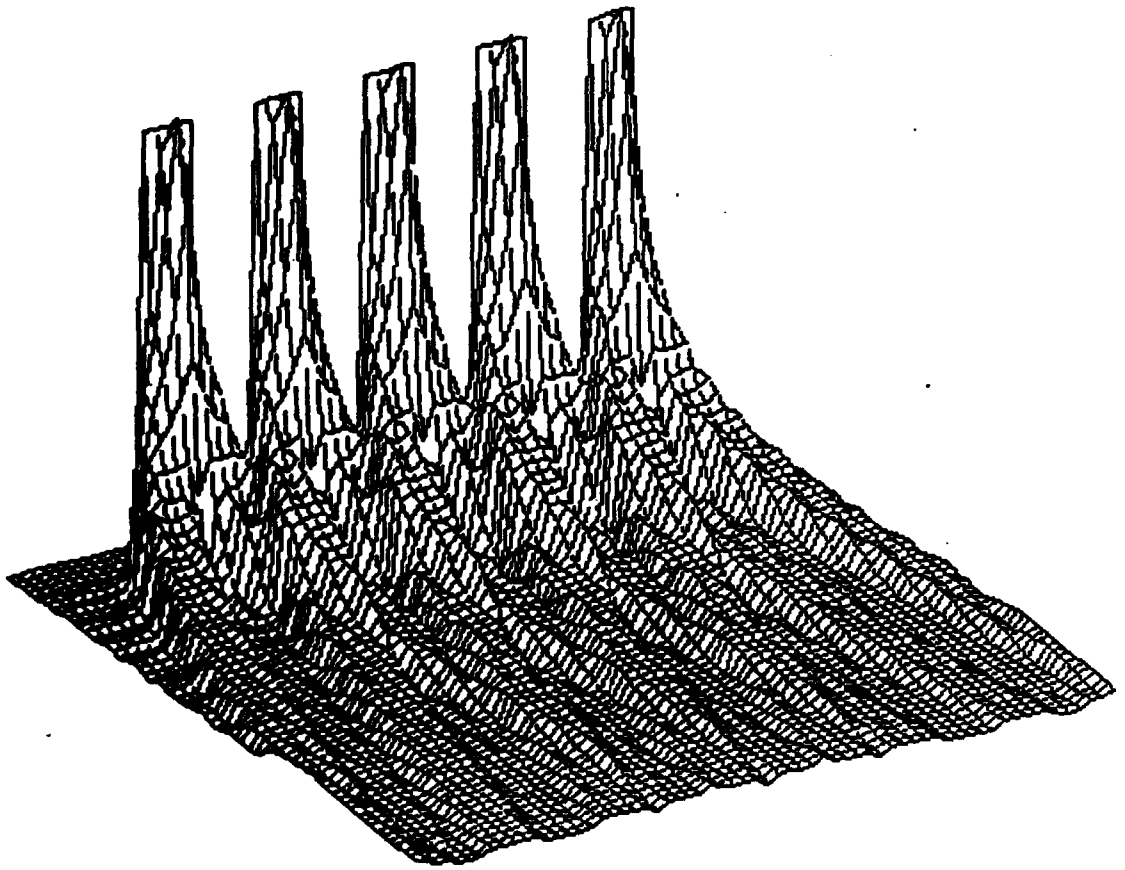


Figure 21. Field for five element linear array (lossy case).

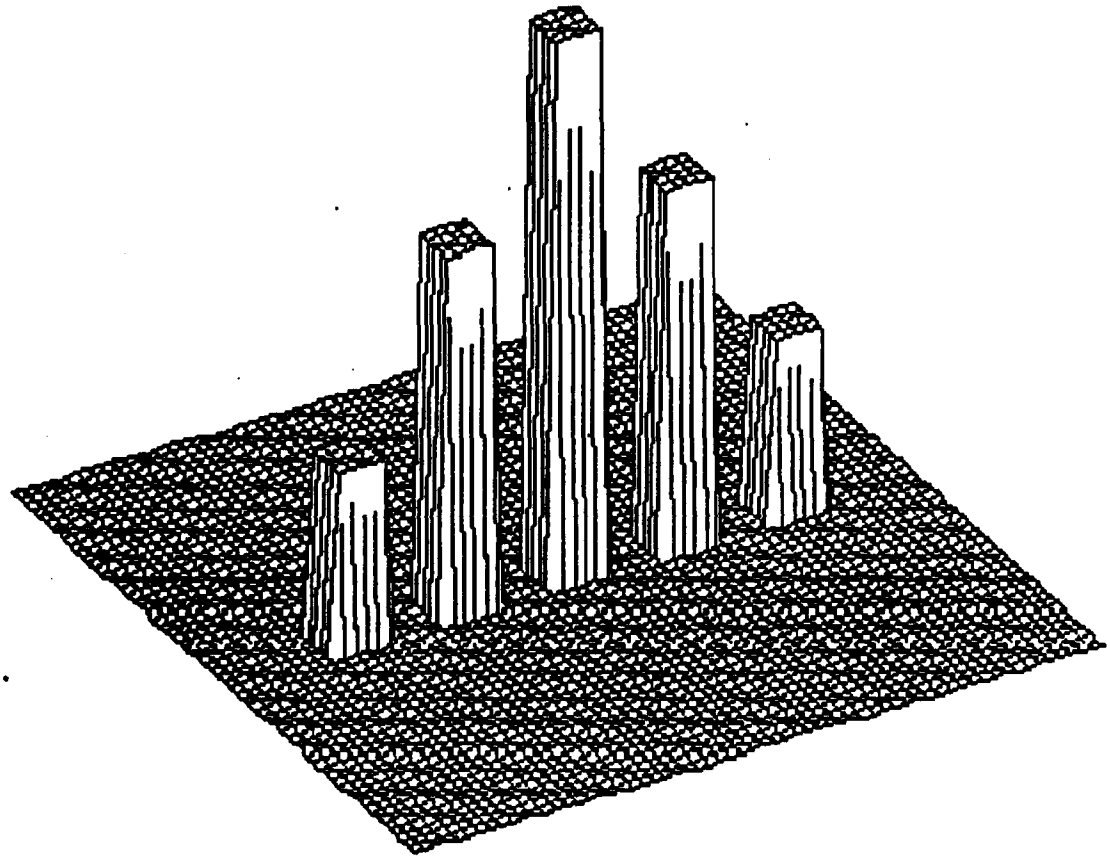


Figure 22. Excitation for five element linear array with stepped amplitude.

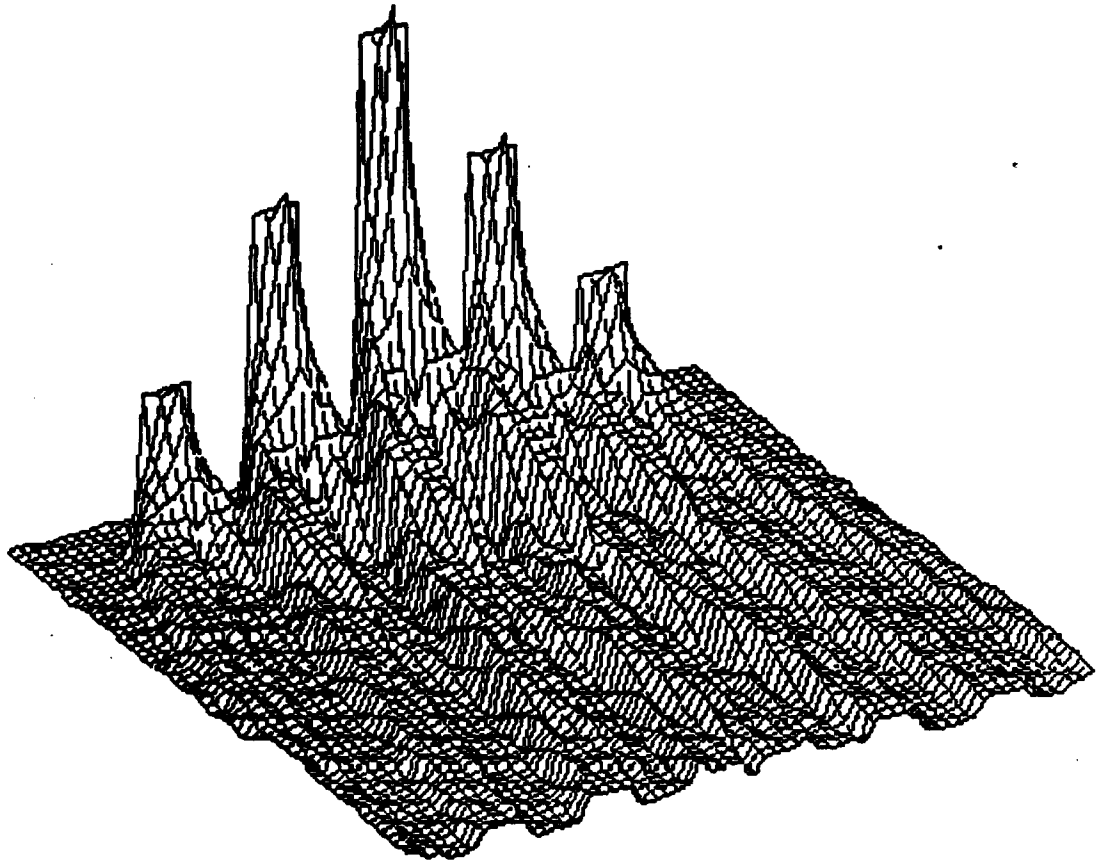


Figure 23. Field of five element linear array with stepped amplitude (lossless case).

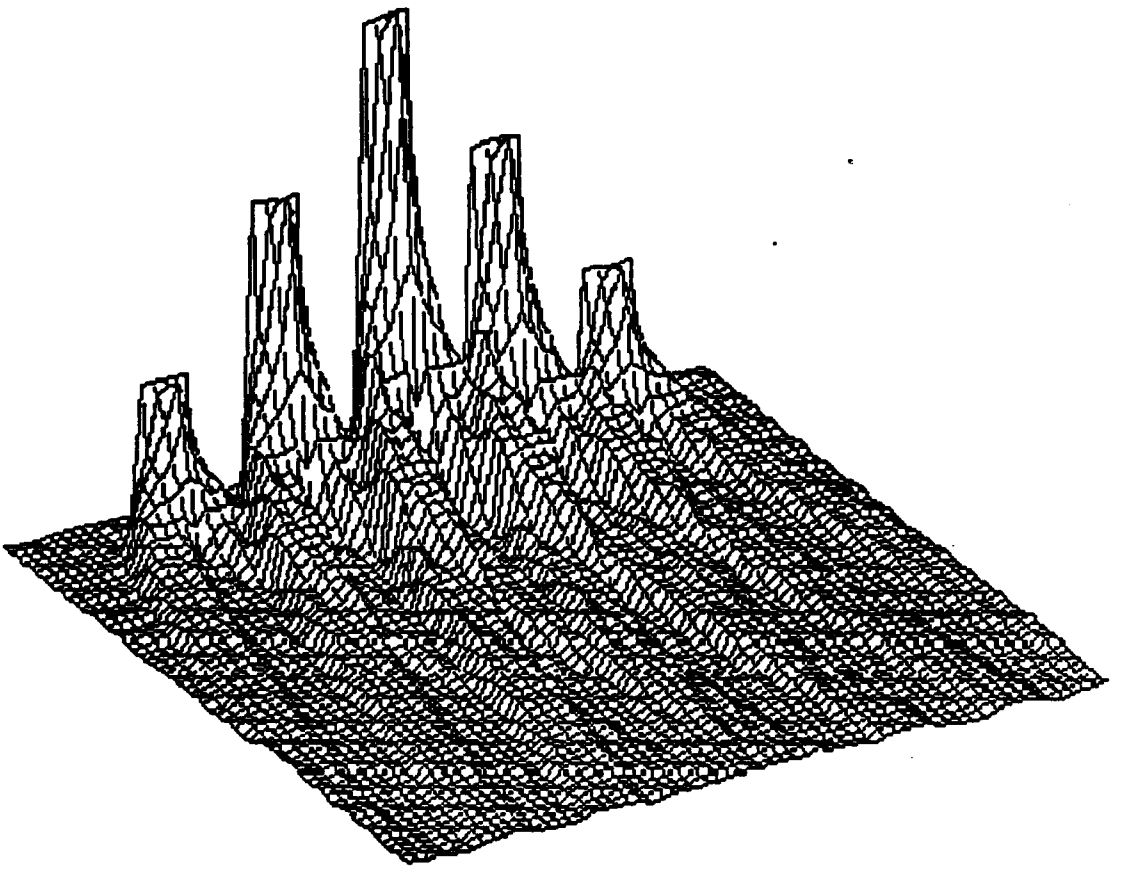


Figure 24. Field for five element linear array with stepped amplitude (lossy case).

V. SUMMARY

This thesis has investigated the ability to perform complex simulations of scalar wave propagation through loss-less media and media with a loss coefficient linear in frequency on a micro-computer. A program was written in Microsoft Fortran V3.31 which allows the user to select or specify the initial conditions and media type, then calculates the resulting spatial impulse response. The set goal of thirty minutes for each simulation was achieved through code optimization and partial array calculation permitted by symmetry.

After initial program verification, the spatial impulse responses for both square-piston and circular-piston spatial excitation were computed. The results obtained agreed with previously accepted results. The spatial impulse response was then calculated for three new spatial excitation sources, a pyramid-shaped excitation, a five element linear array excitation with constant amplitude, and a five element linear array excitation with stepped amplitude. The computed spatial impulse response for all simulations have been plotted and are presented for visual interpretation.

Future development should concentrate in two areas. First, the third model for media with a loss coefficient which is quadratic in frequency must be added to the

program. This will provide all the tools necessary for analysis of wave propagation through air, liquid, gas, and biological tissue. Second, the base array size must be increased beyond the present 64 x 64. An array size of 64 x 64 proved adequate for program development and testing but increased size, and therefore increased resolution, is necessary for serious analysis and evaluation. The limitation in all presently available micro-computers which restricts array size is processing speed. For example, increasing array size to 128 x 128 would increase execution time by a factor of four, requiring almost two hours for each simulation run. Transferring program development to one of the newer micro-computers utilizing the Intel 80386 microprocessor will allow larger array sizes while maintaining reasonable execution times.

APPENDIX: FORTRAN SOURCE CODE

This appendix contains the FORTRAN source code for the program developed in the thesis. This program was compiled and run using Microsoft Fortran Version 3.31.

```
$LARGE
$NOFLOATCALLS
C
C     ************************************************************************
C     COMMENTS:
C
C     THIS PROGRAM WAS WRITTEN BY LT. TIMOTHY MERRILL, USN
C
C     IT IS WRITTEN IN MICROSOFT FORTRAN V3.31 AND IS CAPABLE OF RUNNING
C     ON ANY IBM OR COMPATIBLE MICRO-COMPUTER WITH LITTLE OR NO
C     MODIFICATION.  THE USE OF A 8087 NUMERICAL CO-PROCESSOR IS HIGHLY
C     RECOMMENDED TO REDUCE EXECUTION TIME.
C
C     TO REDUCE EXECUTION TIME THE PROGRAM IS WRITTEN TO USE A 64 X 64
C     BASE ARRAY SIZE, AND IS WRITTEN AS SINGLE PRECISISON.  FUTURE
C     MODIFICATION TO A 128 X 128 (OR LARGER) ARRAY SIZE AND/OR DOUBLE
C     PRECISION IS VERY EASY.
C
C     THE PROGRAM CALLS THE FOLLOWING SUBROUTINES WHOSE SOURCE CODE IS
C     NOT INCLUDED HERE:
C          FFT2C        IMSL ROUTINE TO COMPUTE FFT OF A COMPLEX
C                       VALUED SEQUENCE OF LENGTH EQUAL TO POWER
C                       OF TWO
C          FFT3D        IMSL ROUTINE TO COMPUTE THE FFT OF A COMPLEX
C                       VALUED 1,2, OR 3 DIMENSIONAL ARRAY
C          FFTCC        IMSL ROUTINE CALLED BY FFT3D
C          MMBSJN       IMSL ROUTINE TO CALCULATE BESSEL FUNCTION OF
C                       THE FIRST KIND OF ZERO ORDER WITH REAL ARGUMENT
C          MMBSIN       IMSL ROUTINE TO CALCULATE MODIFIED BESSEL
C                       FUNCTION OF THE FIRST KIND OF ZERO ORDER WITH
C                       REAL ARGUMENT
C
C     T.D.M.  20 JANUARY 1987
C     ************************************************************************
C
      REAL          A,A2,B,B1,BASE,BI,BR,C,C2,CENTER
```

```
      REAL            BWIDTH,INC,MAXTIME,MEANX,PI
      REAL            R123,RESULT,RHO1,RHO2,RWK,SIGMA,SIGMA2
      REAL            TEMP1,TEMP2,TEMP3,TEMP4,TEMP5
      REAL            TIME,TS,TWOPI,X,Y,Z,Z2,ZPRIME,TVECTR
C
      COMPLEX         C123,CWK,CTEMP1,CTEMP2,CTEMP3
      COMPLEX         PULSE,WORK
C
      INTEGER         COL,FTYPE,H,I,IER,IJOB,IWK,IWK1,J
      INTEGER         M,N,NAMLEN,NCOLS,NROWS,R1,R2
      INTEGER         ROW,RBO,SHAPE,STEP,WIDTH,XPOS,YPOS
C
      DIMENSION       CWK(64),IWK(534),IWK1(7),RHO1(64),RWK(534)
      DIMENSION       PULSE(64,64),RESULT(64,64),RHO2(64,64)
      DIMENSION       TIME(64),WORK(64,64),ZPRIME(64)
C
      CHARACTER       FN*8,FNAME*8,T1,T2
      CHARACTER       EXT1*4,EXT2*4,EXT3*4,EXT4*4
      CHARACTER       FNAME1*12, FNAME2*12, FNAME3*12, FNAME4*12
C
C     ************************************************************************
C                                 CONSTANTS
C     ************************************************************************
C        C       SPEED OF SOUND IN VACUUM (1500 METERS/SEC)
C        C2      SPEED OF SOUND SQUARED (METERS**2/SEC**2)
C        H       HALF OF NROWS (OR NCOLS) PLUS 1
C        M       INTEGER USED TO SPECIFY SIZE OF VECTOR SENT TO
C                   FFT2C (REFER TO IMSL SOURCE CODE FOR FFT2C)
C        MEANX   USED FOR GAUSSIAN IMPULSE FUNCTION
C        N       USED ON BESSEL FUNCTION CALLS TO SIGNIFY ZERO ORDER
C        NCOLS   DEFAULT NUMBER OF COLUMNS
C        NROWS   DEFAULT NUMBER OF ROWS
C        SIGMA   USED FOR GAUSSIAN IMPULSE FUNCTION
C        SIGMA2  SIGMA SQUARED
C        STEP    NUMBER OF ROWS SET TO ZERO - SIMULATES STEP FUNCTION
C        TWOPI   2 * PI
C     ************************************************************************
      C = 1500.0
      N = 1
      M = 6
      STEP = 4
      NROWS = 64
      NCOLS = 64
      PI = 3.14159265
      SIGMA = 1.0
      MEANX = 0.0
      C2 = C * C
      H = NROWS/2 + 1
      TWOPI = 2.0 * PI
      SIGMA2 = SIGMA * SIGMA
C
```

```
C     ************************************************************************
C                                GET USER INPUT
C     ************************************************************************
      XPOS = 1
      YPOS = 2
      CALL CLRSCR
      CALL GOTOXY(1,1)
      WRITE(*,*) 'ENTER REQUESTED VALUES.  RECOMMENDED VALUES IN ( )'
      CALL GOTOXY(YPOS,XPOS)
      WRITE(*,*) 'ENTER FILENAME (8 CHAR M/Y):  '
      CALL GOTOXY(YPOS,XPOS+32)
      READ(*,'(A)') FN
      CALL GOTOXY(YPOS+1,XPOS)
      WRITE(*,*) 'SELECT FILTER TYPE'
      CALL GOTOXY(YPOS+2,XPOS+8)
      WRITE(*,*) '<1>  GP1 (LOSSLESS MEDIA)'
      CALL GOTOXY(YPOS+3,XPOS+8)
      WRITE(*,*) '<2>  GP2 (LOSSY MEDIA)'
      CALL GOTOXY(YPOS+4,XPOS)
      WRITE(*,9) ' ENTER 1 OR 2:  '
      READ(*,*) FTYPE
      CALL GOTOXY(YPOS+5,XPOS)
      WRITE(*,*) 'SELECT SOURCE SHAPE'
      CALL GOTOXY(YPOS+6,XPOS+4)
      WRITE(*,*) '<1>  SQUARE'
      CALL GOTOXY(YPOS+7,XPOS+4)
      WRITE(*,*) '<2>  CIRCLE'
      CALL GOTOXY(YPOS+8,XPOS+4)
      WRITE(*,*) '<3>  GAUSSIAN'
      CALL GOTOXY(YPOS+9,XPOS+4)
      WRITE(*,*) '<4>  PYRAMID'
      CALL GOTOXY(YPOS+10,XPOS+4)
      WRITE(*,*) '<5>  TRUNCATED PYRAMID'
      CALL GOTOXY(YPOS+11,XPOS+4)
      WRITE(*,*) '<6>  5 PULSE LINEAR ARRAY (SQ.)'
      CALL GOTOXY(YPOS+12,XPOS+4)
      WRITE(*,*) '<7>  5 PULSE LINEAR ARRAY (GAUSS.)'
      CALL GOTOXY(YPOS+6,XPOS+40)
      WRITE(*,*) ' <8>  5 PULSE LINEAR ARRAY (STEPPED)'
      CALL GOTOXY(YPOS+7,XPOS+40)
      WRITE(*,*) ' <9>  5 PULSE LINEAR ARRAY (PARABOLA)'
      CALL GOTOXY(YPOS+8,XPOS+40)
      WRITE(*,*) '<10>  RESERVED FOR FUTURE USE'
      CALL GOTOXY(YPOS+9,XPOS+40)
      WRITE(*,*) '<11>  RESERVED FOR FUTURE USE'
      CALL GOTOXY(YPOS+10,XPOS+40)
      WRITE(*,*) '<12>  RESERVED FOR FUTURE USE'
      CALL GOTOXY(YPOS+11,XPOS+40)
      WRITE(*,*) '<13>  RESERVED FOR FUTURE USE'
      CALL GOTOXY(YPOS+12,XPOS+40)
      WRITE(*,*) '<14>  RESERVED FOR FUTURE USE'
```

```
      CALL GOTOXY(YPOS+13,XPOS)
      WRITE(*,9) ' ENTER SELECTION:  '
      READ(*,*) SHAPE
      IF (SHAPE .LT. 6) THEN
         CALL GOTOXY(YPOS+14,XPOS)
         WRITE(*,9) ' ENTER SOURCE SIZE (ODD INTEGER <= 63) (11):  '
         READ(*,*) WIDTH
      ENDIF
      CALL GOTOXY(YPOS+15,XPOS)
      WRITE(*,*) 'ENTER RHO  (200): '
      CALL GOTOXY(YPOS+15,XPOS+26)
      READ(*,*) RHO
      CALL GOTOXY(YPOS+16,XPOS)
      WRITE(*,*) 'ENTER Z (METERS)  (0.1): '
      CALL GOTOXY(YPOS+16,XPOS+26)
      READ(*,*) Z
      CALL GOTOXY(YPOS+17,XPOS)
      WRITE(*,7) Z/C
      CALL GOTOXY(YPOS+17,XPOS+44)
      READ(*,*) MAXTIME
      IF (FTYPE .EQ. 2) THEN
         CALL GOTOXY(YPOS+18,XPOS)
         WRITE(*,*) 'ENTER VALUE FOR A  (0.008):  '
         CALL GOTOXY(YPOS+18,XPOS+26)
         READ(*,*) A
         A2 = A * A
      ENDIF
C
7     FORMAT(' ENTER MAXTIME (REAL > ',1PE14.6,'):')
9     FORMAT(A\)
      HWIDTH = REAL(WIDTH)/2
      Z2 = Z * Z
      TS = Z/C
      CALL CLRSCR
      CALL GOTOXY(1,1)
      WRITE(*,*) 'CALCULATING...'
C
C     ************************************************************************
C                  INITIALIZE ARRAYS AND VECTORS TO ZERO
C     ************************************************************************
      DO 5 ROW = 1,NROWS
         TIME(ROW) = 0.0
         RHO1(ROW) = 0.0
         DO 5 COL = 1,NCOLS
            PULSE(COL,ROW) = (0.0,0.0)
            WORK(COL,ROW) = (0.0,0.0)
            RESULT(COL,ROW) = 0.0
            RHO2(COL,ROW) = 0.0
5     CONTINUE
C
```

```
C     ********************************************************************
C                          OPEN OUTPUT FILES
C     ********************************************************************
C        PROGRAM GETS NAME OF PROBLEM <FN> FROM USER AND FORMS:
C            FILENAME.EXT            FILE DESCRIPTION
C            ----------------------------------------------
C                <FN>.IMP     ARRAY OF INPUT IMPULSE FUNCTION
C                <FN>.TXT     SUMMARY INFORMATION FOR THIS PROBLEM
C                <FN>.DAT     FINAL RESULTS ARRAY
C
C        NOTE 1:  ALL FILES ARE IN ASCII TEXT FORMAT
C     ********************************************************************
      EXT1 = '.IMP'
      EXT2 = '.TXT'
      EXT3 = '.DAT'
      NAMLEN = 0
      T1 = ' '
      DO 10 I=1,8
         T2 = FN(I:I)
         IF (T2 .NE. T1) THEN
            FNAME(I:I) = FN(I:I)
            NAMLEN = NAMLEN + 1
         ENDIF
10    CONTINUE
      FNAME1(1:NAMLEN) = FNAME
      FNAME1(NAMLEN+1:NAMLEN+4) = EXT1
      FNAME2(1:NAMLEN) = FNAME
      FNAME2(NAMLEN+1:NAMLEN+4) = EXT2
      FNAME3(1:NAMLEN) = FNAME
      FNAME3(NAMLEN+1:NAMLEN+4) = EXT3
      IF (NAMLEN .LT. 8) THEN
         FNAME1(NAMLEN+5:12) = '        '
         FNAME2(NAMLEN+5:12) = '        '
         FNAME3(NAMLEN+5:12) = '        '
      ENDIF
      OPEN(1,FILE=FNAME1,STATUS='NEW')
      OPEN(2,FILE=FNAME2,STATUS='NEW')
      OPEN(3,FILE=FNAME3,STATUS='NEW')
C
C     ********************************************************************
C                  WRITE HEADER INFORMATION TO <FN>.TXT
C     ********************************************************************
      WRITE(2,11) FN
      IF (FTYPE .EQ. 1) THEN
         WRITE(2,*) '  FILTER IS GP1 FOR LOSSLESS MEDIA'
      ELSEIF (FTYPE .EQ. 2) THEN
         WRITE(2,*) '  FILTER IS GP2 FOR LOSSY (LINEAR) MEDIA'
      ENDIF
      IF (SHAPE .EQ. 1) THEN
         WRITE(2,12) WIDTH,NCOLS,NROWS
```

```
      ELSEIF (SHAPE .EQ. 2) THEN
         WRITE(2,13) WIDTH,NCOLS,NROWS
      ELSEIF (SHAPE .EQ. 3) THEN
         WRITE(2,14) WIDTH,NCOLS,NROWS
      ELSEIF (SHAPE .EQ. 4) THEN
         WRITE(2,111) WIDTH,NCOLS,NROWS
      ELSEIF (SHAPE .EQ. 5) THEN
         WRITE(2,112) WIDTH,NCOLS,NROWS
      ELSEIF (SHAPE .EQ. 6) THEN
         WRITE(2,113)
      ELSEIF (SHAPE .EQ. 7) THEN
         WRITE(2,114)
      ELSEIF (SHAPE .EQ. 8) THEN
         WRITE(2,115)
      ELSEIF (SHAPE .EQ. 9) THEN
         WRITE(2,116)
      ENDIF
      WRITE(2,15) STEP
      WRITE(2,16) Z
      WRITE(2,17) RHO
      WRITE(2,18) MAXTIME
      IF (FTYPE .EQ. 2) THEN
         WRITE(2,19) A
      ENDIF
 11   FORMAT('SUMMARY INFORMATION FOR ',A/)
 12   FORMAT('   IMPULSE FUNCTION IS A SQUARE WITH WIDTH ',I2,' ON A BAS
     +E OF',I3,' X',I3)
 13   FORMAT('   IMPULSE FUNCTION IS A CIRCLE WITH DIAMETER ',I2,' ON A
     +BASE OF',I3,' X',I3)
 14   FORMAT('   IMPULSE FUNCTION IS GAUSSIAN WITH DIAMETER ',I2,' ON A
     +BASE OF',I3,' X',I3)
 111  FORMAT('   IMPULSE FUNCTION IS PYRAMID WITH WIDTH ',I2,' ON A BAS
     +E OF',I3,' X',I3)
 112  FORMAT('   IMPULSE FUNCTION IS TRUNCATED PYRAMID WITH WIDTH ',I2,'
     +ON A BASE OF',I3,' X',I3)
 113  FORMAT('   IMPULSE FUNCTION IS 5 PULSE LINEAR ARRAY WITH SQUARE PU
     +LSES')
 114  FORMAT('   IMPULSE FUNCTION IS 5 PULSE LINEAR ARRAY WITH GAUSSIAN
     +PULSES')
 115  FORMAT('   IMPULSE FUNCTION IS 5 PULSE LINEAR ARRAY WITH STEPPED A
     +MPLITUDE')
 116  FORMAT('   IMPULSE FUNCTION IS 5 PULSE LINEAR ARRAY WITH PARABOLIC
     + AMPLITUDE')
 15   FORMAT('   STEPSIZE = ',I2)
 16   FORMAT('   Z = ',1PE14.7,' METERS')
 17   FORMAT('   RHO = ',I3)
 18   FORMAT('   MAXTIME = ',1PE14.7,' SECONDS')
 19   FORMAT('   A = ',1PE14.7)
C
```

```
C     ************************************************************************
C                          INITIALIZE TIME VECTOR
C     ************************************************************************
      INC = (MAXTIME-TS)/REAL(NROWS-STEP-1)
      DO 20 I = (STEP+1),NROWS
         TIME(I) = TS + (I-STEP-1) * INC
20    CONTINUE
C
C     ************************************************************************
C                      OUTPUT TIME SUMMARY TO <FN>.TXT
C     ************************************************************************
      WRITE(2,*)
      WRITE(2,9) 'TIME SUMMARY:'
      WRITE(2,21) TS,STEP+1
      WRITE(2,22) INC
21    FORMAT('   ZERO TIME STARTS AT T=Z/C =',1PE14.7,' SECONDS IN ROW('
     +,I2,')')
22    FORMAT('      AND INCREMENTS BY ',1PE14.7,' SECONDS PER ROW')
C
C     ************************************************************************
C          CREATE Z-PRIME VECTOR:  SQUARE EACH VALUE OF TIME,
C          MULTIPLY BY C SQUARED AND SUBTRACT Z SQUARED. THEN
C          TAKE SQUARE ROOT.
C     ************************************************************************
      DO 30 I = (STEP+2),NROWS
         ZPRIME(I) = SQRT((TIME(I) * TIME(I) * C2) - Z2)
30    CONTINUE
C
C     ************************************************************************
C          INITIALIZE RHO VECTOR AND GENERATE 33 X 64 RHO MATRIX
C          ONLY ONE HALF OF THE MATRIX HAS TO BE CALCULATED SINCE
C          THE OTHER SIDE IS SYMMETRICAL
C     ************************************************************************
      INC = REAL(RHO)/REAL(NCOLS/2-1)
      BASE = REAL(-RHO) - INC
      RHO1(1) = BASE
      DO 50 I = 1,NCOLS-1
         RHO1(I+1) = BASE + I * INC
50    CONTINUE
C
      DO 60 I = 1,NROWS
         RHO2(H,I) = RHO1(I)
         RHO2(I,H) = RHO1(I)
60    CONTINUE
C
      DO 70 ROW = 1,NROWS
         DO 70 COL = 1,H
            RHO2(COL,ROW) = SQRT(RHO2(COL,H)**2 + RHO2(H,ROW)**2)
70    CONTINUE
C
```

```
C     ********************************************************************
C                       OUTPUT RHO SUMMARY TO <FN>.TXT
C     ********************************************************************
      WRITE(2,9) 'RHO SUMMARY:'
      WRITE(2,44) H,NCOLS/2+1
      WRITE(2,41) BASE
      WRITE(2,42) INC
      WRITE(2,43) RHO2(1,1)
41    FORMAT('      BASE VALUE = (-RHO - INC)     = ',1PE14.7)
42    FORMAT('      INCREMENT = RHO/(64/2 - 1)    = ',1PE14.7)
43    FORMAT('      MAXIMUM VALUE IS AT RHO2(1,1) = ',1PE14.7)
44    FORMAT('   ARRAY IS CENTERED ABOUT (',I3,',',I3,')')
C
C     ********************************************************************
C          GENERATE EXCITATION  -  CENTER POINT MUST BE AT (33,33)
C     ********************************************************************
C
C     ***   GENERATE SQUARE EXCITATION   ***
      IF (SHAPE .EQ. 1) THEN
         DO 1200 ROW = (NROWS/2+1-WIDTH/2),(NROWS/2+1+WIDTH/2)
            DO 1200 COL = (NCOLS/2+1-WIDTH/2),(NCOLS/2+1+WIDTH/2)
               PULSE(COL,ROW) = (1.0,0.0)
1200     CONTINUE
C
C     ***   GENERATE CIRCULAR PULSE   ***
      ELSEIF (SHAPE .EQ. 2) THEN
         DO 1300 ROW = (NROWS/2+1-WIDTH/2),(NROWS/2+1+WIDTH/2)
            DO 1300 COL = (NCOLS/2+1-WIDTH/2),(NCOLS/2+1+WIDTH/2)
               IF (SQRT(REAL(COL-H)**2+REAL(ROW-H)**2) .LT. HWIDTH) THEN
                  PULSE(COL,ROW) = (1.0,0.0)
               END IF
1300     CONTINUE
C
C     ***   GENERATE CIRCULAR EXCITATION WITH GAUSSIAN AMPLITUDE   ***
      ELSEIF (SHAPE .EQ. 3) THEN
         TEMP1 = 1/SQRT(TWOPI*SIGMA2)
         TEMP2 = 1/(2*SIGMA2)
         DO 1400 ROW = (NROWS/2+1-WIDTH/2),(NROWS/2+1+WIDTH/2)
            DO 1400 COL = (NCOLS/2+1-WIDTH/2),(NCOLS/2+1+WIDTH/2)
               TEMP3 = SQRT(REAL(COL-H)**2+REAL(ROW-H)**2)
               IF (TEMP3 .LT. HWIDTH) THEN
                  PULSE(COL,ROW) = TEMP1*EXP(-TEMP2*(TEMP3-MEANX)**2)
               ENDIF
1400     CONTINUE
C
C     ***   GENERATE PYRAMID EXCITATION   ***
      ELSEIF (SHAPE .EQ. 4) THEN
         R1 = (NROWS/2) - (WIDTH/2)
         R2 = (NROWS/2 + 2) + (WIDTH/2)
         INC = 1.0/(WIDTH/2 + 1)
```

```
        DO 1500 I = 1,WIDTH/2
           DO 1500 ROW = R1+I,R2-I
              DO 1500 COL = R1+I,R2-I
                 PULSE(COL,ROW) = INC * I
1500    CONTINUE
        PULSE(H,H) = 1.0
C
C     ***   GENERATE TRUNCATED PYRAMID EXCITATION   ***
      ELSEIF (SHAPE .EQ. 5) THEN
        R1 = (NROWS/2) - (WIDTH/2)
        R2 = (NROWS/2 + 2) + (WIDTH/2)
        INC = 1.0/(NCOLS/2)
        BASE = (NCOLS/2 - WIDTH/2 + 1) * INC
        INC = 1.0/H
        DO 1600 I = 1,WIDTH/2
           DO 1600 ROW = R1+I,R2-I
              DO 1600 COL = R1+I,R2-I
                 PULSE(COL,ROW) = BASE + INC * I
1600    CONTINUE
        PULSE(H,H) = 1.0
C
C     ***   GENERATE 5 ELEMENT LINEAR ARRAY - SQUARE   ***
      ELSEIF (SHAPE .EQ. 6) THEN
        WIDTH = 5
        DO 1700 I = 1,WIDTH
           COL = I * 10
           DO 1700 J = 1,WIDTH
              COL = COL + 1
              DO 1700 ROW = (H - WIDTH/2),(H + WIDTH/2)
                 PULSE(COL,ROW) = (1.0,0.0)
1700    CONTINUE
C
C     ***   GENERATE 5 ELEMENT LINEAR ARRAY - GAUSSIAN   ***
      ELSEIF (SHAPE .EQ. 7) THEN
        WIDTH = 5
        TEMP1 = 1/SQRT(TWOPI*SIGMA2)
        TEMP2 = 1/(2*SIGMA2)
        DO 1800 ROW = (NROWS/2+1-WIDTH/2),(NROWS/2+1+WIDTH/2)
           DO 1800 COL = (NCOLS/2+1-WIDTH/2),(NCOLS/2+1+WIDTH/2)
              TEMP3 = SQRT(REAL(COL-H)**2+REAL(ROW-H)**2)
              IF (TEMP .LT. REAL(WIDTH/2)) THEN
                 TEMP4 = TEMP1*EXP(-TEMP2*(TEMP3-MEANX)**2)
                 PULSE(COL-20,ROW) = TEMP4
                 PULSE(COL-10,ROW) = TEMP4
                 PULSE(COL,ROW)    = TEMP4
                 PULSE(COL+10,ROW) = TEMP4
                 PULSE(COL+20,ROW) = TEMP4
              ENDIF
1800    CONTINUE
C
```

```
C     ***   GENERATE 5 ELEMENT LINEAR ARRAY - STEPPED   ***
      ELSEIF (SHAPE .EQ. 8) THEN
         WIDTH = 5
         TEMP1 = 1.0/3.0
         TEMP2 = 2.0/3.0
         TEMP3 = 1.0
         DO 1900 COL = 1,WIDTH
            DO 1900 ROW = (H-WIDTH/2),(H+WIDTH/2)
               PULSE(COL+10,ROW) = TEMP1
               PULSE(COL+20,ROW) = TEMP2
               PULSE(COL+30,ROW) = TEMP3
               PULSE(COL+40,ROW) = TEMP2
               PULSE(COL+50,ROW) = TEMP1
1900     CONTINUE
C
C     *** LINEAR ARRAY OF 5 ELEMENTS - PARABOLIC AMPLITUDE   ***
C         IMPLEMENTS THE FOLLOWING FORMULA:
C                 (X - H)**2 = -4 * P * (Y - K)
C         WITH
C                H = 0
C                P = 256
C                K = 256
C         TO GIVE A DOWNWARD OPENING PARABOLA THAT HAS A PEAK OF 1.0
C         AND IS EQUAL TO ZERO AT X = -32, 32 (ROWS 2 AND 64).
C         EQUATION IS EVALUATED AT X = 0, 10 AND 20.
C
      ELSEIF (SHAPE .EQ. 9) THEN
         WIDTH = 5
         TEMP1 = (4.*256. - 20**2)/(4.*256.)
         TEMP2 = (4.*256. - 10**2)/(4.*256.)
         TEMP3 = (4.*256. -  0**2)/(4.*256.)
         DO 2000 COL = 1,WIDTH
            DO 2000 ROW = (H-WIDTH/2),(H+WIDTH/2)
               PULSE(COL+10,ROW) = TEMP1
               PULSE(COL+20,ROW) = TEMP2
               PULSE(COL+30,ROW) = TEMP3
               PULSE(COL+40,ROW) = TEMP2
               PULSE(COL+50,ROW) = TEMP1
2000     CONTINUE
C
      ENDIF
C
C     ********************************************************************
C                 OUTPUT IMPULSE FUNCTION ARRAY TO <FN>.IMP
C     ********************************************************************
      DO 3000 I = 1,NROWS
         WRITE(1,102) (REAL(PULSE(J,I)),J = 1,NCOLS)
3000  CONTINUE
      CLOSE(1)
C
```

```
C     ***********************************************************************
C        PASS PULSE TO 2-D FFT SUBROUTINE.  IJOB = 1 MEANS TAKE FFT
C     ***********************************************************************
      IJOB = 1
      CALL FFT3D(PULSE,NCOLS,NROWS,NCOLS,NROWS,1,IJOB,IWK,RWK,CWK)
C
C     ***********************************************************************
C       SHIFT RESULTS - IE: SWITCH QUADRANTS 1 AND 3, 2 AND 4 IN THE
C                           COL,ROW PLANE
C     ***********************************************************************
      CALL SHIFT (PULSE,NROWS,NCOLS)
C
C     ***********************************************************************
C     MAIN PROGRAM STARTS HERE
C        CALCULATIONS START AT ROW = STEP+1.
C        NOTE THAT ONLY ONE QUARTER OF THE WORK MATRIX HAS TO
C        BE CALCULATED.  THE REMAINING THREE QUARTERS IS FORMED BY
C        FOLDING AND FLIPPING.  THIS IS POSSIBLE DUE TO SYMMETRY.
C     ***********************************************************************
      CALL GOTOXY(1,1)
      WRITE(*,*) 'CALCULATING ROW  '
      IF (FTYPE .EQ. 1) THEN
         DO 200 I = STEP+1,NROWS
         CALL GOTOXY(1,18)
         WRITE(*,101) I
            DO 210 ROW = 1,H
               DO 210 COL = 1,H
                  B1 = ZPRIME(I) * RHO2(COL,ROW)
                  CALL MMBSJN(B1,N,TEMP1,IER)
                  WORK(COL,ROW) = TEMP1 * PULSE(COL,ROW)
210         CONTINUE
C
C     ***********************************************************************
C                       FLIP QUADRANT 2 TO QUADRANT 3
C     ***********************************************************************
            J = 0
            DO 220 ROW = H+1,NROWS
               J = J + 2
               DO 220 COL = 1,H
                  WORK(COL,ROW) = WORK(COL,ROW-J)
220         CONTINUE
C
C     ***********************************************************************
C                     FLIP LEFT SIDE TO RIGHT SIDE
C     ***********************************************************************
            J = 0
            DO 215 COL = H+1,NCOLS
               J = J + 2
               DO 215 ROW = 1,NROWS
                  WORK(COL,ROW) = WORK(COL-J,ROW)
215         CONTINUE
```

```
C
C     ******************************************************************
C        TAKE INVERSE FFT OF EACH COLUMN.  THEN TAKE INVERSE FFT OF
C        ROW 33 ONLY, SINCE THIS IS THE ROW CONTAINING CENTRAL
C        INFORMATION.  ROW 33 BECOMES THE NEXT ROW OF THE FINAL
C        OUTPUT MATRIX.
C     ******************************************************************
           DO 415 COL = 1,NCOLS
              DO 405 ROW = 1,NROWS
                 CWK(ROW) = CONJG(WORK(COL,ROW))
405           CONTINUE
              CALL FFT2C (CWK,M,IWK1)
              DO 410 ROW = 1,NROWS
                 WORK(COL,ROW) = CWK(ROW)
410           CONTINUE
415        CONTINUE
C
           DO 440 COL = 1,NCOLS
              CWK(COL) = WORK(COL,H)
440        CONTINUE
           CALL FFT2C (CWK,M,IWK1)
C
           R123 = NROWS * NCOLS
           C123 = CMPLX(R123,0.0)
           DO 450 COL = 1,NCOLS
              RESULT(COL,I) = ABS(REAL((CONJG(CWK(COL))/C123)))
450        CONTINUE
C
200     CONTINUE
C
      ELSEIF(FTYPE .EQ. 2) THEN
         TEMP1 = A * C / 2
         TEMP2 = A2 * C2 / 4
         CALL GOTOXY(1,1)
         WRITE(*,*) 'CALCULATING ROW  '
         DO 300 I = STEP+1,NROWS
         CALL GOTOXY(1,18)
         WRITE(*,101) I
            DO 310 ROW = 1,H
               DO 310 COL = 1,H
                  TEMP3 = ZPRIME(I) * SQRT(ABS(RHO2(COL,ROW)**2-TEMP2))
                  IF (ABS(RHO2(COL,ROW)) .GT. TEMP1) THEN
                     CALL MMBSJN(TEMP3,N,TEMP4,IER)
                  ELSE
                     CALL MMBSIN(TEMP3,N,TEMP4,IER)
                  ENDIF
                  WORK(COL,ROW) = TEMP4 * EXP(-A*C2*TIME(I)) *
     &                            PULSE(COL,ROW)
310         CONTINUE
C
```

```
C     ******************************************************************
C                       FLIP QUADRANT 2 TO QUADRANT 3
C     ******************************************************************
            J = 0
            DO 320 ROW = H+1,NROWS
               J = J + 2
               DO 320 COL = 1,H
                  WORK(COL,ROW) = WORK(COL,ROW-J)
320         CONTINUE
C
C     ******************************************************************
C                       FLIP LEFT SIDE TO RIGHT SIDE
C     ******************************************************************
            J = 0
            DO 315 COL = H+1,NCOLS
               J = J + 2
               DO 315 ROW = 1,NROWS
                  WORK(COL,ROW) = WORK(COL-J,ROW)
315         CONTINUE
C
C     ******************************************************************
C        TAKE INVERSE FFT OF EACH COLUMN.  THEN TAKE INVERSE FFT OF
C        ROW 33 ONLY, SINCE THIS IS THE ROW CONTAINING CENTRAL
C        INFORMATION.  ROW 33 BECOMES THE NEXT ROW OF THE FINAL
C        OUTPUT MATRIX.
C     ******************************************************************
            DO 615 COL = 1,NCOLS
               DO 605 ROW = 1,NROWS
                  CWK(ROW) = CONJG(WORK(COL,ROW))
605            CONTINUE
               CALL FFT2C (CWK,M,IWK1)
               DO 610 ROW = 1,NROWS
                  WORK(COL,ROW) = CWK(ROW)
610            CONTINUE
615         CONTINUE
C
            DO 640 COL = 1,NCOLS
               CWK(COL) = WORK(COL,H)
640         CONTINUE
            CALL FFT2C (CWK,M,IWK1)
C
            R123 = NROWS * NCOLS
            C123 = CMPLX(R123,0.0)
            DO 650 COL = 1,NCOLS
               RESULT(COL,I) = ABS(REAL((CONJG(CWK(COL))/C123)))
650         CONTINUE
C
300      CONTINUE
C
         ENDIF
```

```
C     ******************************************************************
C           SET ROWS 1 TO STEP TO ZERO TO SIMULATE STEP FUNCTION
C     ******************************************************************
      DO 500 ROW = 1,STEP
         DO 500 COL = 1,NCOLS
            RESULT(COL,ROW) = 0.0
500   CONTINUE
C
C     ******************************************************************
C                        OUTPUT FINAL ARRAY TO DISK
C     ******************************************************************
      DO 510 I = 1,NROWS
         WRITE(3,102) (RESULT(J,I),J = 1,NCOLS)
510   CONTINUE
C
101   FORMAT (I2)
102   FORMAT (64F16.7)
C
C
C     ******************************************************************
C                         CLOSE REMAINING OPEN FILES
C     ******************************************************************
C
      CLOSE(2)
      CLOSE(3)
C
C     ******************************************************************
C                      REMIND USER OF DATA FILES CREATED
C     ******************************************************************
      CALL CLRSCR
      WRITE(*,*) 'FINISHED.'
      WRITE(*,*)
      WRITE(*,*) '   THE FOLLOWING ASCII TEXT FILES ARE IN THE DEFAULT'
      WRITE(*,*) '   DIRECTORY:'
      WRITE(*,*)
      WRITE(*,*) '          FILENAME          DESCRIPTION'
      WRITE(*,*) '  ------------------------------------------------'
      WRITE(*,*) '          ',FNAME1,'   INPUT IMPULSE FUNCTION'
      WRITE(*,*) '          ',FNAME2,'   SUMMARY INFORMATION'
      WRITE(*,*) '          ',FNAME3,'   OUTPUT ARRAY'
      WRITE(*,*)
      WRITE(*,*)
C
      STOP
      END
C
```

```
C     ************************************************************************
C     *                                                                      *
C     *  SUBROUTINE GOTOXY                                                   *
C     *     MOVES CURSOR TO LINE X, COLUMN Y                                 *
C     *     CALL:  CALL GOTOXY(X,Y)                                          *
C     *                                                                      *
C     ************************************************************************
C
      SUBROUTINE GOTOXY(X,Y)
      CHARACTER*1 C1,C2,C5,C8,LC(5)
      CHARACTER*5 CBUFF
      INTEGER*2 IC(4),L,X,Y
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C5,IC(3)),(C8,IC(4)),
     +(CBUFF,LC(1))
      DATA IC/16#1B,16#5B,16#3B,16#66/
C
      L = 10000+100*X+Y
C
C *** WRITE ESCAPE CODES TO CHARACTER BUFFER
      WRITE(CBUFF,2) L
2     FORMAT(I5)
C
C *** WRITE ESCAPE CODES TO THE DISPLAY
      WRITE(*,1) C1,C2,LC(2),LC(3),C5,LC(4),LC(5),C8
1     FORMAT(1X,8A1,\)
C
C *** END OF SUBROUTINE
      RETURN
      END
C
C     ************************************************************************
C     *                                                                      *
C     *   SUBROUTINE CLRSCR                                                  *
C     *      SUBROUTINE TO CLEAR THE DISPLAY                                 *
C     *      CALL:  CALL CLRSCR                                              *
C     *                                                                      *
C     ************************************************************************
C
      SUBROUTINE CLRSCR
      CHARACTER*1 C1,C2,C3,C4
      INTEGER*2 IC(4)
      EQUIVALENCE (C1,IC(1)),(C2,IC(2)),(C3,IC(3)),(C4,IC(4))
      DATA IC/16#1B,16#5B,16#32,16#4A/
C
C *** WRITE ESCAPE CODE TO DISPLAY
      WRITE(*,1) C1,C2,C3,C4
1     FORMAT(1X,4A1)
C
C *** END OF SUBROUTINE
      RETURN
      END
```

```
C
C     ***************************************************************
C     * SUBROUTINE TO SHIFT A SQUARE MATRIX - EXCHANGES QUADRANTS ONE *
C     * AND THREE, AND TWO AND FOUR.                                  *
C     * USE:  CALL SHIFT(X,NROWS,NCOLS)                               *
C     *       WHERE X IS A SQUARE ARRAY - REAL OR COMPLEX             *
C     *       NROWS,NCOLS IS THE ARRAY SIZE                           *
C     *                                                               *
C     ***************************************************************
      SUBROUTINE SHIFT (X,NROWS,NCOLS)
C
C
      INTEGER NROWS,NCOLS,ROW,COL,R2,C2
      COMPLEX X(NCOLS,NROWS),T1,T2
C
      R2 = NROWS/2
      C2 = NCOLS/2
C
      DO 10 ROW = 1,R2
         DO 10 COL = 1,C2
            T1 = X(COL,ROW)
            X(COL,ROW) = X(COL+C2,ROW+R2)
            X(COL+C2,ROW+R2) = T1
            T2 = X(COL+C2,ROW)
            X(COL+C2,ROW) = X(COL,ROW+R2)
            X(COL,ROW+R2) = T2
10    CONTINUE
      RETURN
      END
```

LIST OF REFERENCES

1. D. Guyomar and J. Powers, "Propagation of transient acoustic waves in lossy and lossless media," in Acoustical Imaging Volume 14, A. J. Berkhout, J. Ridder, and L. F. van der'Wal, Eds. New York, NY: Plenum Press, 1985, pp. 521-531.

2. D. Guyomar and J. Powers, "A transfer function model for propagation in homogeneous media," presented at the 112th Meeting of the Acoustical Society of America, Anaheim, CA, 8-12 December 1986.

3. D. Guyomar and J. Powers, "Boundary effects on transient radiation fields from vibrating surfaces," J. Acous. Soc. Am., vol. 77(3), pp. 907-915, 1985.

4. P. R. Stepanishen, "Transient radiation from pistons in an infinite planar baffle," J. Acous. Soc. Am., vol. 49(5), pp. 1629-1637, 1971.

5. P. R. Stepanishen, "Acoustic transients in the far-field of a baffled circular piston using the impulse response approach," J. Sound Vib., vol. 32(3), pp. 295-310, 1974.

6. P. R. Stepanishen, "Acoustic transients from planar axisymmetric vibrators using the impulse response method," J. Acous. Soc. Am., vol. 70(4), pp. 1176-1181, 1981.

7. P. R. Stepanishen and G. Fisher, "Experimental verification of the impulse response method to evaluate transient acoustic fields," J. Acous. Soc. Am., vol. 69(6), pp. 1610-1617, 1981.

8. A. P. Medeiros and P. R. Stepanishen, "The forward and backward projection of acoustic fields from axisymmetric ultrasonic radiators using impulse response and Hankel transform techniques," J. Acous. Soc. Am., vol 75(6), pp. 1732-1740, 1984.

9. D. Guyomar and J. Powers, "Diffraction of pulsed ultrasonic waves in absorbing media with a linear frequency dependence," In preparation.

10. International Mathematical & Statistical Libraries, edition 9.2, 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Department of Electrical and Computer Engineering Code 62 Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor John P. Powers Code 62PO Naval Postgraduate School Monterey, California 93943-5000	4
5.	Lt. Timothy D. Merrill 1057 Halsey Drive Monterey, California 93940	2

END

7-87

DTIC