

AD-A254 535



Technical Document 2287
July 1992

Visualization of Scattering Strength of Elastic Bodies in a Fluid

DTIC
S **A** **D**
ELECTE
AUG 24 1992

H. A. Schenck
Naval Command, Control and Ocean
Surveillance Center
RDT&E Division

J. L. Fales
Mechanical and Electronic
Engineering Division
Los Alamos National Laboratory

✓ Original contains color
plates: All DTIC reproductions
will be in black and
white.

Approved for public release; distribution is unlimited.



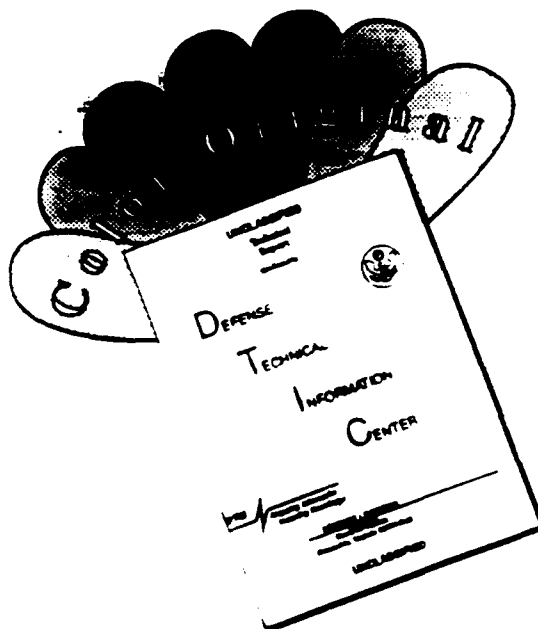
92 8 21 099

92-23426



424521 72 Pg

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF COLOR PAGES WHICH DO NOT REPRODUCE LEGIBLY ON BLACK AND WHITE MICROFICHE.

Technical Document 2287
July 1992

Visualization of Scattering Strength of Elastic Bodies in a Fluid

H. A. Schenck
Naval Command, Control and
Ocean Surveillance Center
RDT&E Division

J. L. Fales
Mechanical and Electronic
Engineering Division
Los Alamos National Laboratory

DTIC QUALITY INSPECTED 5

Accession For	
NTIS CRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution	
Availability	
Dist	Availability of Source
A-1	

**NAVAL COMMAND, CONTROL AND
OCEAN SURVEILLANCE CENTER
RDT&E DIVISION
San Diego, California 92152-5000**

J. D. FONTANA, CAPT, USN
Commanding Officer

R. T. SHEARER
Executive Director

ADMINISTRATIVE INFORMATION

The work documented in this report was sponsored by the Defense Advanced Research Projects Agency (DARPA). The work was a joint effort by Harry A. Schenck of the Naval Command, Control and Ocean Surveillance Center, RDT&E Division and Janine L. Fales of Los Alamos National Laboratory.

Released by
H. A. Schenck
Associate for Undersea
Surveillance

Under authority of
R. E. Shuttters, Head
Surveillance Department

ACKNOWLEDGMENTS

The authors gratefully acknowledge the assistance of Mary Marshall and Billie Wheat of Los Alamos National Laboratory and Vince Insixiengmay of Computer Sciences Corporation in developing the computer code and preparing the examples given in this report.

BACKGROUND

As part of the Submarine Technology Program, the Defense Advanced Research Projects Agency (DARPA) recently sponsored a Low-Frequency Structural Acoustics Benchmark Exercise. The purpose of this exercise was to test and validate several major computational codes that have been developed to solve acoustic scattering problems of elastic objects in a fluid. All of the computations of scattering were done on a Cray YMP at the Los Alamos National Laboratory (LANL). The result of each problem was a large digital data set, which was analyzed and displayed off-line on a workstation using specially developed visualization techniques. The Benchmark Exercise began in May 1991, and a comprehensive report on the results will be issued separately by LANL.

The purpose of this report is to describe some of the visualization techniques and procedures that were developed to review, compare, and analyze the large amount of computational data generated in the Benchmark Exercise. It was felt that the visualization approach and techniques should be described in an independent document because the techniques are generally useful in representing scattering or target strength functions generated by any means, and the approach is applicable to any computational or experimental problem in which there is a need to understand large multidimensional data sets.

THE NEED FOR VISUALIZATION

In the Benchmark Exercise, a series of nine problems was developed to test the accuracy and computational efficiency of six different structural acoustics codes. In each of the problems, the forcing function was a steady-state plane wave at a set of densely spaced frequencies incident upon the scatterer from one of several different directions. In all the problems, it was required that the code generate the complex numbers representing the complete three-dimensional far-field scattering function. In some of the problems, it was also required that the code generate the complex surface pressure and normal velocity on a dense grid over the complete surface of the scatterer. The problems and results were specified in nondimensional terms.

A far-field data set typically consisted of all the information needed to reconstruct the normalized target strength or complex scattering function at a set of frequencies over all space. Each set was given a name associated with the problem and the code that generated it, and an extension of `.ff#`, where the `#` was a single digit identifying the incidence angle. Each set was a file containing a three-dimensional array of complex coefficients in a circumferential harmonic decomposition of the three-dimensional field, as follows:

$$p(\varphi) = \sum_{n=0}^{N-1} a_n \cos(n\varphi) + \sum_{n=0}^{N-1} b_n \sin(n\varphi),$$

where N is the number of harmonics and $b_0 = 0$. The dimensions of the array are the number of frequencies, the number of azimuthal harmonics, and the number of bistatic observation angles. In a typical problem, the computation was made at 331 frequencies, 73 observation angles, and 3 incidence angles, using 7 harmonics. Each complex number required a 16-bit (IEEE) binary representation, so the far-field output from one problem typically comprised three 1.4-megabyte files.

Similarly, a surface pressure (`*.sp#`) or normal velocity (`*.nv#`) data set was also a three-dimensional array of complex numbers, whose dimensions were the number of frequencies by the number of harmonics by the number of grid points describing the generator line of an axially symmetric object. The number of grid points varied with the code, but was typically of order 250 in a large problem, resulting in surface files of approximately 5 megabytes each.

Some of the problems were run more than once for each structural acoustic code, so the total amount of computational data generated and reviewed in the Benchmark Exercise approached 5 gigabytes. Clearly, an efficient and powerful graphical postprocessing method was required.

APPROACH

In addition to the sheer magnitude of the problem, there were many other challenges inherent in trying to display, compare, and understand the results. New formats had to be conceived and implemented to display or compare large amounts of data at one time. The judicious use of color and animation to make the results more understandable was anticipated. In addition to rapid and convenient software development, the need for efficient repetition of tasks for many different data sets was recognized. Reproduction of workstation displays on paper or videotape was also important.

With these criteria in mind, the command language version of PV-WAVE* software was chosen as the means for accomplishing the visualization tasks on available workstations (Sun and Silicon Graphics). The key features of this software that were of importance to the task at hand were as follows:

- Availability of a large number of basic mathematical and graphical operations as simple callable procedures and functions
- Availability of a high-level programming language, with all the usual logical constructs
- Ability to run the software interactively, retaining data and intermediate variables in memory
- Ability to generate new user-defined procedures and functions for efficient development of a family of visualization algorithms
- Ability to develop compiled user-friendly routines for production use by junior programmers
- Speed and efficiency of operation on a workstation, especially with matrix or array operations

The example programs in this document are all written in the PV-WAVE command language, but should be understandable by anyone familiar with a high-level structured programming language.

* PV-WAVE is the trademark for a software product of Precision Visuals, Inc., Boulder, CO.

ARCHITECTURE AND DATA FLOW

An overall guide to the approach used is described in figure 1. At the top of the flowchart are shown the IEEE binary data files written to disk by the structural code that was executing on the Cray YMP. All binary files began with a header in which problem identification and parameter descriptions as used at the time of computation were written. In addition to the far-field, surface pressure, and normal velocity files described above, an additional binary coordinate (* .bco) file was written for each problem, containing the set of cylindrical coordinates describing the shape of the scatterer in discretized form. These coordinates were also the locations at which the surface pressure and normal velocities were calculated.

Each shaded rounded rectangle in figure 1 represents one or more PV-WAVE command language routines written to perform a specific function. These routines are small ASCII files having a .pro extension, created with any convenient word processor. On the top level of the flowchart, there are three conversion routines (convffb.pro, convbco.pro, and convsurb.pro). These and other .pro routines described in this section are listed in their entirety in appendix A. The conversion routines are intended to be used only once on a given data set. The purpose of each conversion routine is to read an IEEE binary file generated by a FORTRAN program on the Cray YMP, and rewrite it as a structured binary file in a format optimized for input to PV-WAVE. One of the built-in functions in the PV-WAVE language is the ASSOC function, whose purpose is to map an array structure onto a named file, thereby permitting the entire file to be read efficiently into memory and to be associated with a structured variable in one statement. Once the structured binary files (* .fw# for far-field PV-WAVE files, * .pw# for surface pressure, and * .vw# for normal velocity) are created, the original IEEE binary files can be discarded or at least archived because they are not needed for subsequent processing of the data.

The convbco.pro routine extracts the coordinate information from the * .bco file, writes it to a * .wco file, and also extracts the header information and writes it to an ASCII * .hdr file. This header file is the key to generalizing the remainder of the visualization routines so that a single simple * .pro routine will work for any problem, with any choice of modeling variables, independent of the particular structural acoustics code that generated it. The information in the header is made available by calling the procedure rdhdr.pro and passing the information as named variables through a common block in rdhdr.pro and the routine that calls it. Calling rdhdr.pro also causes the header information to be displayed on the screen, as shown in figure 2. The five types of PV-WAVE files on the third level of figure 1 contain all the information present in a solution to one of the benchmark problems.

For the purpose of quickly checking the data set or for developing some new means for visualizing it, there are two workhorse routines called grabff.pro and grabsur.pro. The purpose of these routines is to quickly read (i.e., "grab") a far-field or surface variable data set and its associated header, and make the information available in high-speed memory for use on the workstation. Each routine also displays a simple representative portion of the

data on the screen. In the case of `grabff.pro`, this representation is a set of three two-dimensional color images, as shown in figure 3. One of these (figure 3a) shows target strength in dB encoded in color as a function of polar angle and nondimensional frequency for a fixed azimuthal angle. The other two surfaces (figures 3b and 3c) show the linear magnitude of the scattering function and its phase in the same layout. A color table is included with each figure to show how the tabulated values are translated into a color image. Horizontal cuts of these surfaces represent directivity patterns at some fixed frequency, and vertical cuts represent spectral variations in target strength at some (possibly bistatic) angle. In the case of `grabsur.pro`, two color images are given in figure 4: a "Persian rug" (figure 4a), in which surface pressure or normal velocity is plotted as a function of axial and azimuthal coordinates for a fixed frequency, and a "standing wave" plot (figure 4b), in which the axial variation of the surface quantity is displayed as a function of axial coordinate and frequency for a fixed azimuthal angle. If there are any data format problems, or if a fundamental error has been made in computing the results, this initial visual check is often sufficient to identify the problem. Using the built-in procedures and functions of PV-WAVE, the data in memory can be easily manipulated and presented for visual analysis on the workstation screen in a variety of one-, two-, and three- dimensional formats. This is a fast and convenient way to explore the data. In addition, PV-WAVE provides both a "command history" and a "journaling" feature that make this interactive mode very efficient for the development of new algorithms. Instructions can be issued, recalled and modified, and then saved to form the core of a new `*.pro` routine, because the syntax of the language is virtually identical whether used interactively or read and compiled from a program file.

The rectangle labeled `other.pro` is a generic descriptor on the chart for all the other compiled routines that are described in this document or that were developed during the Benchmark Exercise. A number of such application-oriented routines are listed in the next section, along with a brief statement of the scope or purpose of the routine. Examples of the output are also given as appropriate, although these outputs are not a direct representation of what would be displayed on the screen by the PV-WAVE routine. The complete PV-WAVE code for these routines given in appendix A; however, it is not intended that the code could be used as is, since in many cases it is dependent on the particular data sets available for analysis.

Finally, in reference to figure 1, in order to redirect the output of the various routines to a printer or other hardcopy device, two simple routines called `hardcopy.pro` and `closeit.pro` were written. These routines can be used interactively or inserted into compiled visualization code surrounding one or more graphical display statements. They reformat and redirect the output to an encapsulated PostScript file for inclusion in a document or viewgraph using some convenient page-formatting software. For production purposes, some routines were rewritten to make the hardcopy the normal output, or to do both so that the process could be monitored as it was generating the files for producing hardcopy output. Videotapes were also generated using some routines (especially for those cases where animation in time was useful for analysis of the effects of changes in frequency or angle). The simplest way to generate the videotape is to connect appropriate scan converting and recording equipment directly to the workstation.

EXAMPLE APPLICATIONS

Selected examples of visualization and analysis routines are presented in this section. The first two examples are routines used simply to display or visualize far-field and surface results. The next two examples were developed to aid in the comparison of multiple data sets. The last two routines actually process the results in such a way as to reveal the underlying physical mechanisms.

ff3d.pro - The purpose of this routine is to fully use the graphic display device to show the three-dimensional character of the far-field scattering function, using both color and distance from the origin to indicate target strength at a given frequency. This gives a more complete geometrical picture of the scattering function than the color images produced by **grabff.pro** which show the variation in polar angle only, for a fixed azimuthal angle. This visualization technique is particularly useful for understanding whether the results have the correct symmetry in a particular problem. Because each three-dimensional picture is for a fixed frequency, the variation in frequency is shown by animating the display; that is, by showing a sequence of frames at some chosen frequency increment. Figure 5 shows several frames in such an animation.

plotsur.pro - The purpose of this routine is to overlay, by means of animation, line plots of either surface pressure or normal velocity as a function of axial node number. This is useful to determine the spatial variation in relation to the surface or internal features of the scattering object, and to observe and understand the difference between pressure and velocity on the surface. Figure 6 indicates the final frame of the animation, which is built up over time by overlaying many different velocity distributions in a selected frequency range.

envel.pro - The purposes of this routine are to generate a set of overlays of supposedly equivalent solutions generated by different structural acoustic codes, to allow the user to select a subset of the solutions (possibly having rejected one or more outliers), and then to produce the mean and an envelope that bounds the variations among the data sets. The results of this process are illustrated in figure 7.

compff_m.pro - The purpose of this routine is to take two different far-field data sets, display both of them, and then display the difference between them in a similar format. As can be seen in figure 8, the difference plot is rendered using a specially designed color table which highlights the differences; the sign of the difference is retained, thereby providing information about which data set is higher than the other. The routine also allows the user to select or experiment with the dynamic range and with a threshold value below which differences are not considered important and can be blended into the background. This routine calls a secondary procedure named **npff_m.pro**; the PV-WAVE code for this subordinate procedure is not included but is functionally equivalent to **grabff.pro** and is used to read in and display the selected data sets.

fig8.pro - The purpose of this routine is to use wave-vector processing to perform spatial transforms of the computed normal velocity in order to reveal the underlying behavior of elastic waves on the surface of the scatterer. It produces the same kind of helical spectra that have been used in holographic work with experimental data to associate the theoretical dispersion curves of free waves on an infinite shell with the actual motion on finite closed bodies. In its present form, it is only appropriate in problems for which there is a cylindrical section of significant length compared to the overall dimensions of the scatterer. The result of this routine is illustrated in figure 9, in which the bright spots indicate waves traveling in one axial direction with a particular helical wave number. Each wave number is an (m,n) pair, where m is the number of axial half-wavelengths along the length of the cylinder and n is the number of the wavelengths that fit around the circumference of the cylinder. The pattern of these spots in the m,n plane has the appearance of a "figure 8," which gives rise to the name of this routine.

window.pro - This is a sophisticated use of the PV-WAVE software. The purpose of this routine is to simulate a process that was not carried out in the computational problem, but which controlled the accuracy and resolution of the comparison of computed results with experimental data. Although the computation in the Benchmark Exercise was made for a free-field steady-state situation, the experimental data had been collected in a small tank requiring a short time window to be applied. This routine accepts the computational data in the frequency domain, performs a series of fast Fourier transforms to achieve a real time series with the appropriate sampling, aligns the computed time series, and applies the windows actually used in the experiment. The modified time series is then transformed back to the frequency domain for comparison with the experimental data. PV-WAVE includes the fast Fourier transform and other signal processing functions as built-in procedures/functions. Figure 10 illustrates the end result of using **window.pro**. In the center (figure 10b) are experimental results in the normal far-field format, showing target strength in color as a function of polar angle and frequency. On the right (figure 10c) the computed target strength is displayed without modification, and on the left (figure 10a) the windowed computational results that are the result of simulating the experiment are shown. As a quality check, intermediate time-domain results are displayed on the screen while the routine is running through the data by angle, as shown in figure 11.

SUMMARY

The scattering strength of elastic objects can best be understood by displaying its variations in creative ways. Furthermore, visualization of such large multidimensional data sets provides an efficient means for checking as well as understanding or analyzing such a function. When implemented on modern graphical workstations, the software architecture and routines described here provide a powerful environment for dealing with the results of computer-intensive modeling of this nature. These techniques could easily be modified and employed in the analysis and visualization of computational or experimental data from a wide variety of physical problems.

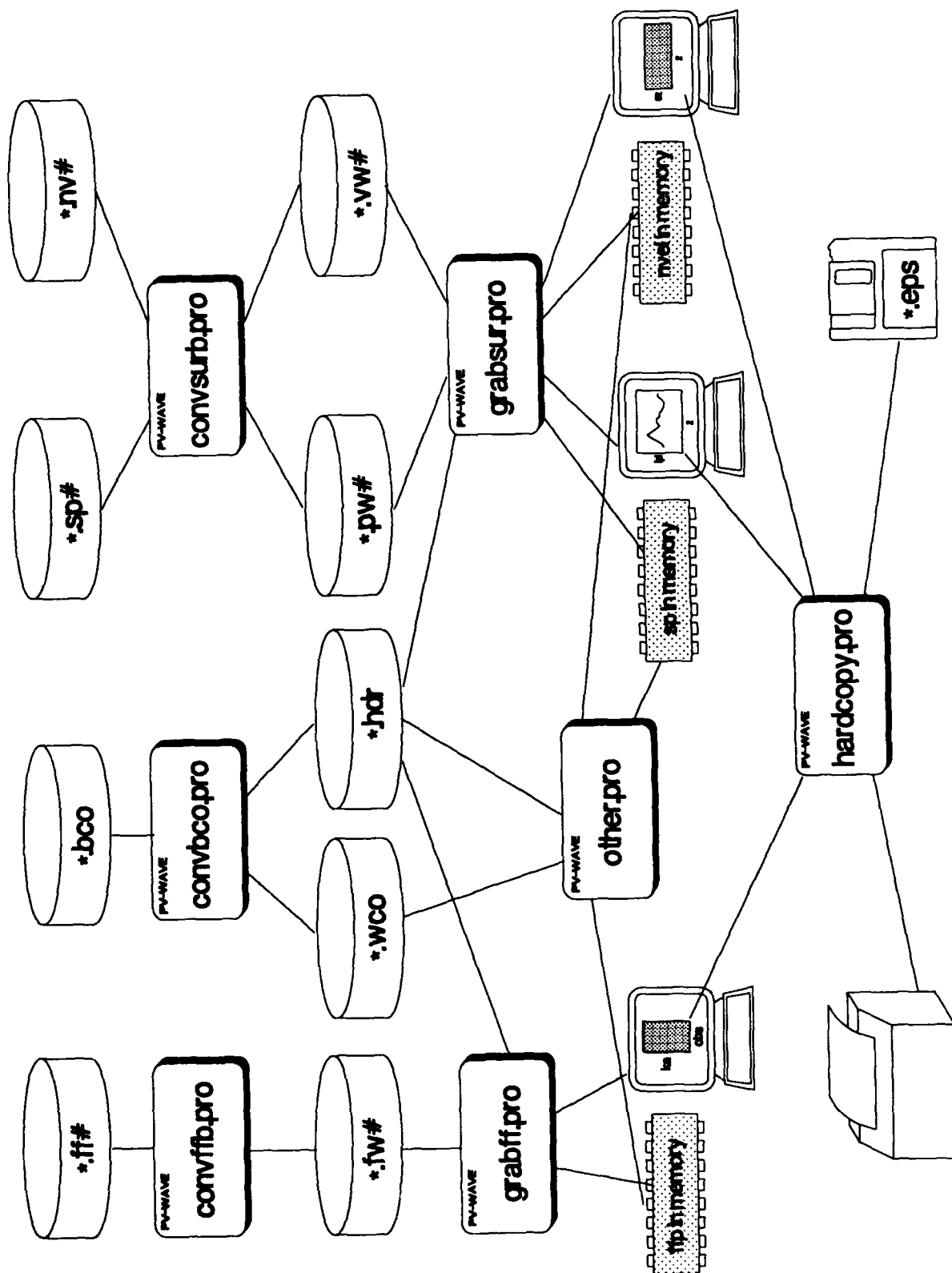


Figure 1. Visualization of Benchmark data sets.

Jobname is c1a
Comments: CHIEF with 2% loss
There are 331 ka values from 0.200000 in steps of 0.0100000
The spectral variable is ka
There are 1 ff patterns
 24 areas or rings
 24 coordinate values
 3 theta incident angles
There are 3 theta incidence angles from 0.00000 in steps of 45.0000
73 theta observation angles in ff pattern from 0.0 in steps of 2.50
The fluid density is 1000.00
The sound speed in the fluid is 1500.00
The density of shell is 7850.00
The Youngs modulus of shell is 2.00000e+11
Poissons ratio for the shell is 0.300000
The damping factor is 0.0200000

Figure 2. Example output from rdhdr.pro.

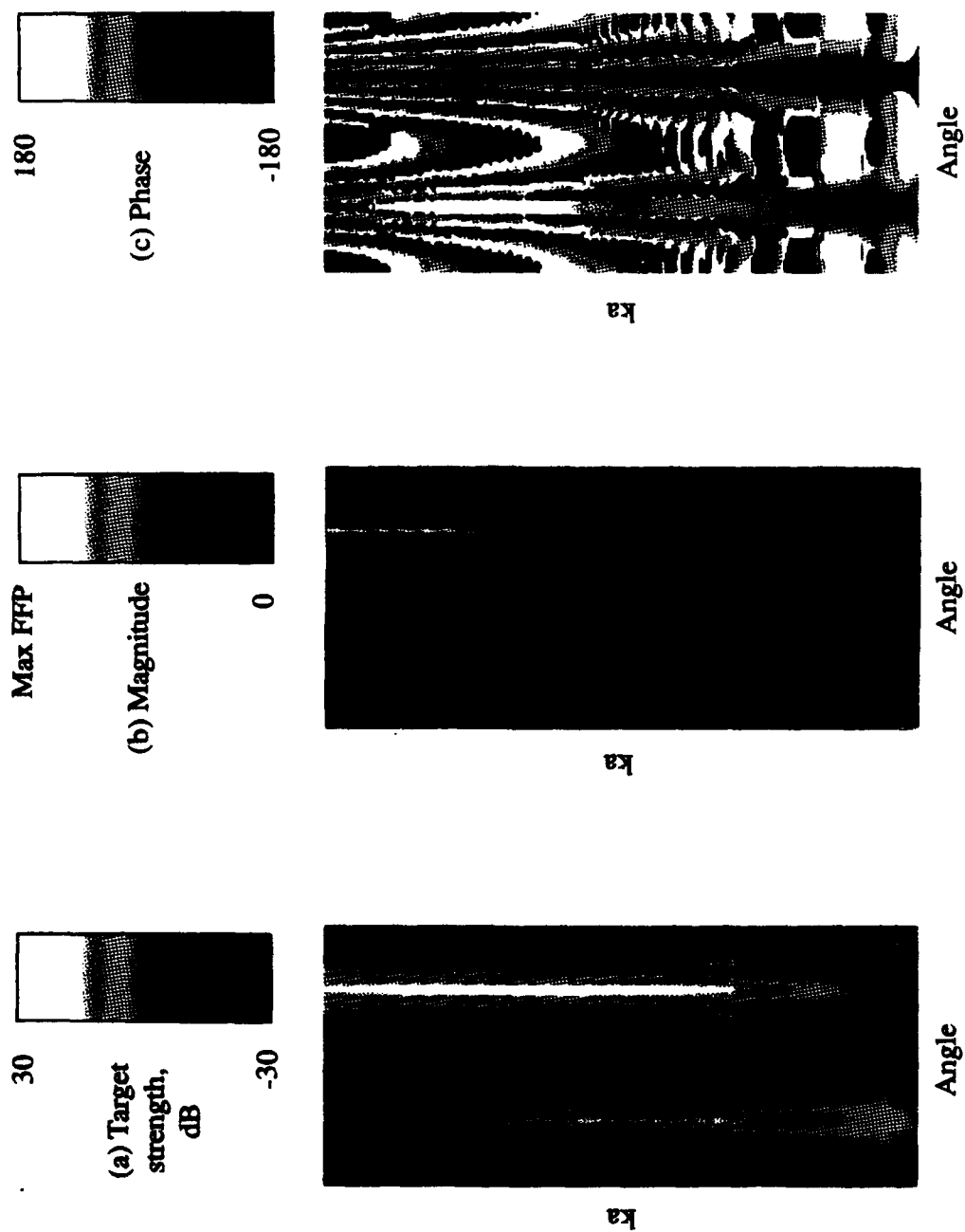


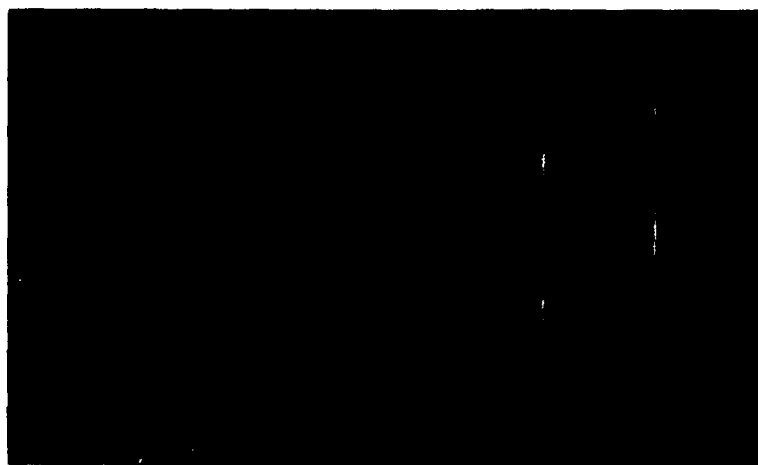
Figure 3. Complex far-field pressure.



Axial node number

Phi

(a) "Persian rug," for $k_{abin} = 100$



k_a

Axial node number

(b) "Standing wave" plot, for $\text{phibin} = 0$

Figure 4. Normal velocity, c1b, 90 incidence.



$ka = 3.03$

$ka = 1.15$

$ka = 0.35$

Figure 5. Three-dimensional representation of far-field pressure at several frequencies.

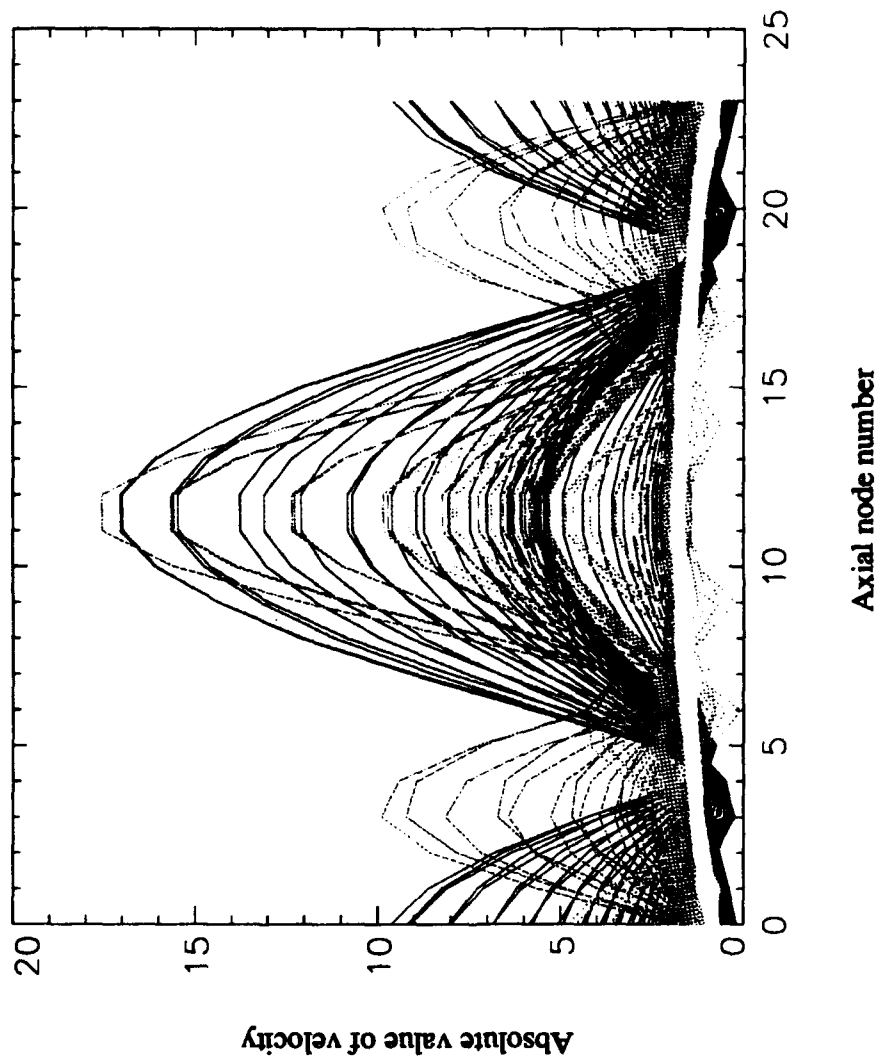
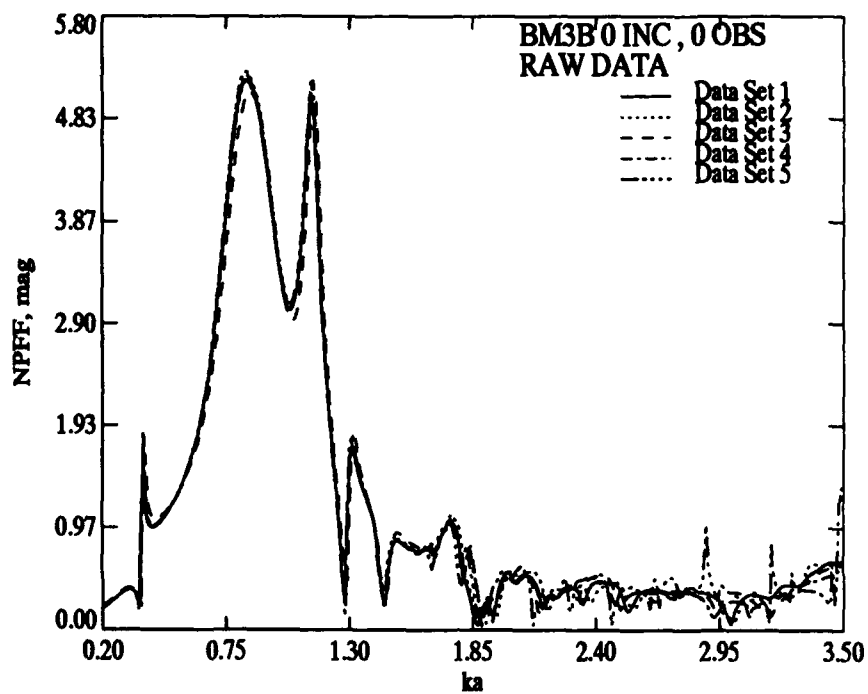
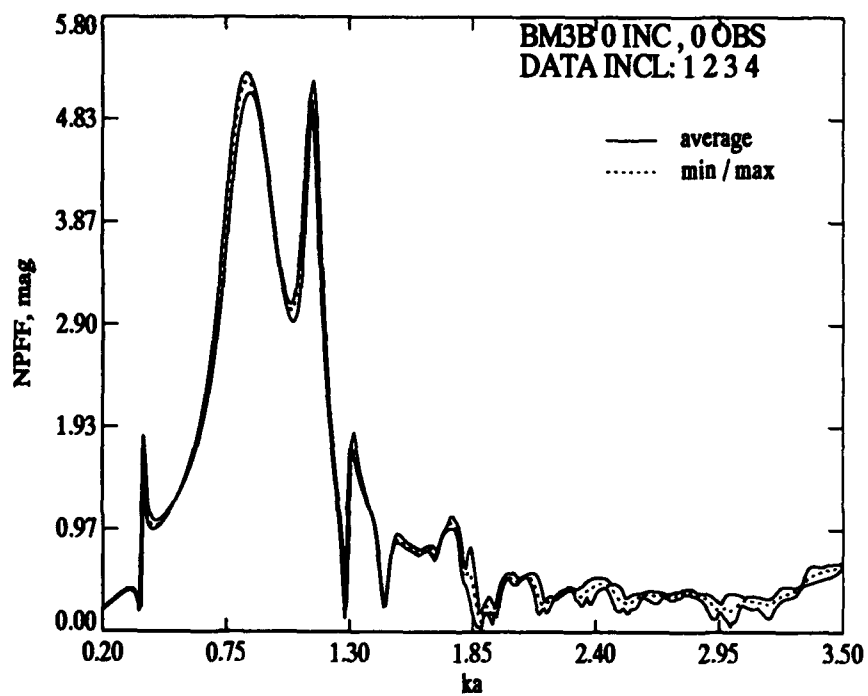


Figure 6. Axial distribution of normal velocity for various frequencies.



(a) Raw data



(b) Envelope of selected data

Figure 7. Data set variation and envelope of selected data sets.

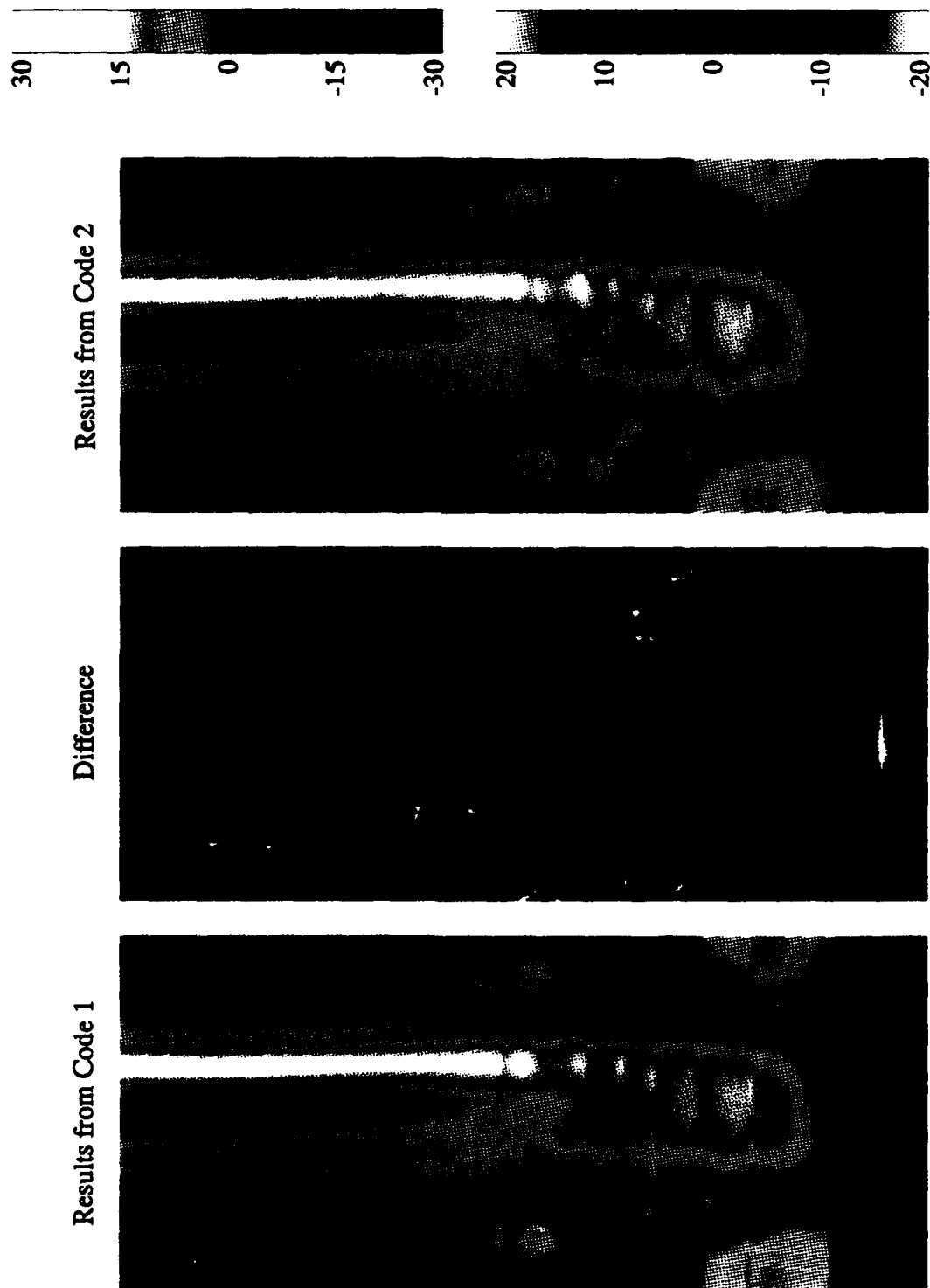


Figure 8. Comparison and differences in target strength.

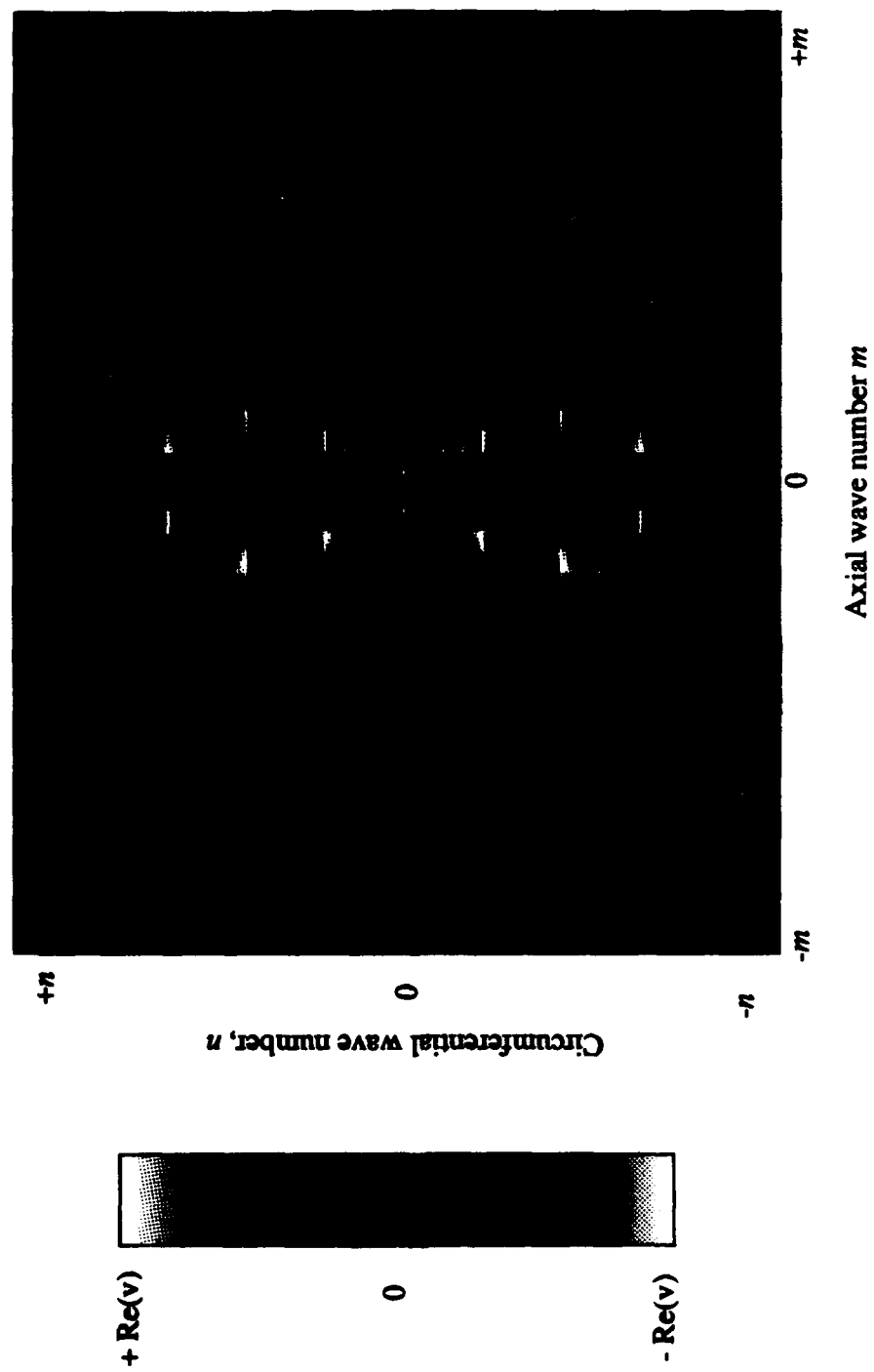


Figure 9. Transformed surface velocity, as a function of m and n .

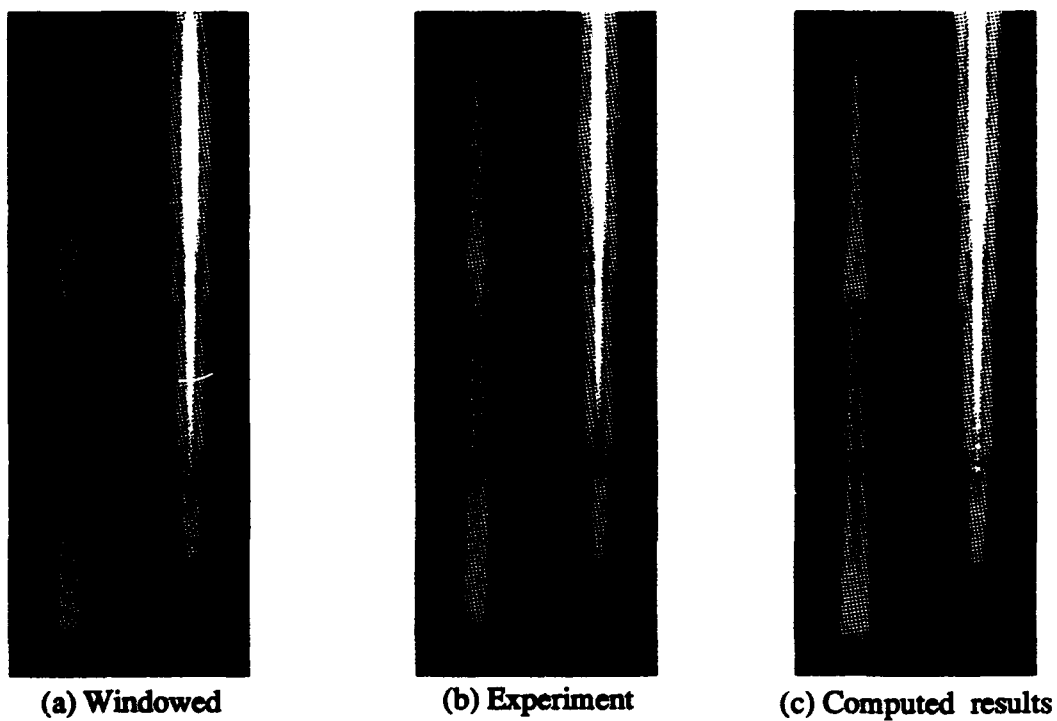


Figure 10. Comparison of experimental results with windowed and unwinded computational results.

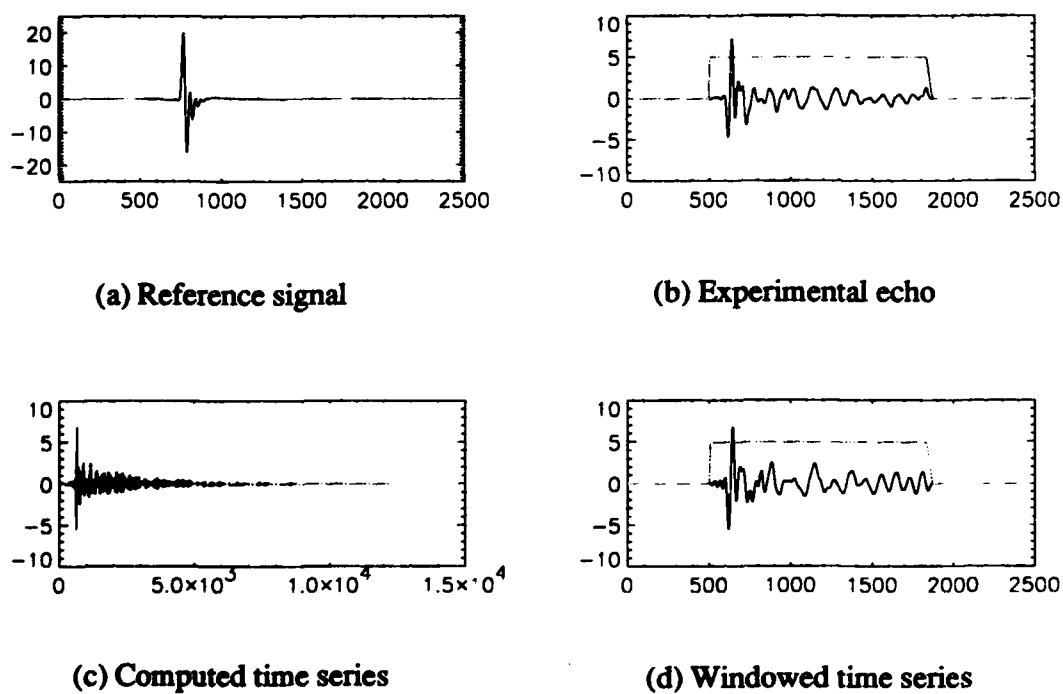


Figure 11. Time domain processing of computational data.

APPENDIX A: LISTINGS OF PROGRAMS

The program code makes use of a number of built-in PV-WAVE functions or procedures. A brief functional summary of the PV-WAVE procedures used in our visualizations is given below. These summaries are extracted with permission from the PV-WAVE Technical Reference Manual. In each case, the syntax of the function or procedure is given, with a brief explanation of the purpose of the routine.

Array creation routines

BINDGEN (Dim ₁ ,...,Dim _n)	Byte array, each element its subscript.
BYTARR (Dim ₁ ,...,Dim _n)	Returns a byte vector or array.
COMPLEXARR (Dim ₁ ,...,Dim _n)	Complex single-precision vector or array.
FINDGEN (Dim ₁ ,...,Dim _n)	Floating array, each element its subscript.
FLTARR (Dim ₁ ,...,Dim _n)	Single-precision floating vector or array.
INDGEN (Dim ₁ ,...,Dim _n)	Integer array, each element its subscript.
INTARR (Dim ₁ ,...,Dim _n)	Returns a integer vector or array.
LONARR (Dim ₁ ,...,Dim _n)	Returns longword integer vector or array.
REPLICATE (Value,Dim ₁ ,...,Dm _n)	Forms new array filled with Value.

Array manipulation routines

MAX (Array[,Max_Subscript])	Finds the maximum element of an array.
MIN (Array[,Min_Subscript])	Finds the minimum element of an array.
REBIN (Array,Dim ₁ ,...,Dim _n)	Resamples array to given dimensions.
REFORM (Array,Dim ₁ ,...,Dim _n)	Reformats array without changing contents.
SHIFT (Array,S ₁ ,...,S _n)	Shifts the elements of an array.
TRANSPOSE (Array)	Transposes an array.

Data conversion routines

BYTSCL (Array)	Scales and converts the array to the byte type.
FIX (Expr[,Offset[,Dim ₁ ,...,Dim _n]])	Converts parameter to integer type.
STRING (Expr ₁ ,...,Expr _n)	Converts parameter to string type.

File manipulation routines

CLOSE [,Unit ₁ ,...,Unit _n]	Closes one or more files.
FREE_LUN ,Unit ₁ ,...,Unit _n)	Deallocates one or more files.
GET_LUN ,Unit	Reserves a file unit.
OPENR ,Unit,File	Opens a file for reading access only.
OPENW ,Unit,File	Opens a new file for writing access.

General graphics routines

MOVIE,Images[,Rate]

Cycles images stored in 3D array.

PLOTS,X[,Y[,Z]]

Draws lines (vectors) and points.

SET_PLOTS,Device

Specifies graphics device.

XYOUTS,X,Y,String

Sends text to selected graphics device.

General Mathematical functions

ABS(X)

Absolute value.

CONJ(X)

Complex conjugate.

FFT(Array,Direction)

Fast Fourier transform.

Image display routines

TV,Image[,X,Y[,Channel]]

Displays an image on the display screen.

TVSCL,Image[,X,Y[,Channel]]

Scales and displays an image.

Image processing routines

FFT(Array,Direction)

Fast Fourier transform.

Input and output routines

ASSOC(Unit,Array_structure[,Offset])

Associates variable with file structure.

PRINT,Expr1,...,Expr_n)

Prints to the standard output stream.

READ,Var1,...,Var_n)

Reads from the standard input stream.

READF,Unit,Var1,...,Var_n)

Reads from the specified file unit.

READU,Unit,Var1,...,Var_n)

Reads unformatted input.

Plotting routines

OPLOT,X[,Y]

Plots vector arguments over old axis.

PLOT,X[,Y]

Plots vector arguments.

Programming routines

HAK

"Hit any key to continue" function.

WAIT,Seconds

Delays program execution.

String processing routines

STRMID(Expr,First_Character,Length)

Extracts substring of string expression.

STRTRIM(String,[flag])

Removes leading and/or trailing blanks.

Transcendental mathematical functions

ALOG10(X)	Base 10 logarithm.
COS(X)	Cosine.
SIN(X)	Sine.

Window routines.

WINDOW[,Window_Index]	Creates a workstation window.
WSET[,Window_Index]	Selects the current window.

A list of the Utility and Application *.pro files follows, after which the actual programs and example output, where appropriate, are given. In the programs, any line or portion of a line beginning with a semicolon is treated as a comment.

Utility routines: Commented program code included

convffb.pro
convbco.pro
convsurb.pro
rdhdr.pro
grabff.pro
grabsur.pro
hardcopy.pro
closeit.pro

Application routines: Commented program code and example output included

ff3d.pro
plotsur.pro
envel pro
compff_m.pro
fig8.pro
window.pro

```

1 ;convffb.pro
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program takes a binary far-field pressure (.Ff#) file
6 ;and produces an equivalent PV-Wave (.fw#) file.
7 ;After this conversion, the .ff# is not needed by PV-Wave.
8 ;-----
9
10 ; Select the data set
11
12 fname=string(replicate(32b,20))
13 print,'Input job name - no extension'
14 read,fname
15 print,'Input incidence angle'
16 read,iinc
17 case iinc of
18     0:begin
19         extension='.ff0'
20     end
21     45:begin
22         extension='.ff4'
23     end
24     90:begin
25         extension='.ff9'
26     end
27     135:begin
28         extension='.ff3'
29     end
30     180:begin
31         extension='.ff8'
32     end
33 endcase
34 fullname=fname+extension
35
36 get_lun,iunit
37 openr,iunit,fullname,/f77_unformatted
38 jobname=string(replicate(32b,8))
39 comment=string(replicate(32b,80))
40 intrec=lonarr(10)
41 rrec1=fltarr(10)
42 rrec2=fltarr(10)
43
44 ; Although the header information is read from the data file,
45 ; it is not stored. The header information from the .bco file
46 ; is used.
47
48 readu,iunit,jobname
49 print,'Reading the following data file: ',fullname
50 print,jobname
51 readu,iunit,comment
52 print,comment
53 readu,iunit,intrec
54 print,intrec

```

```

55 readu,iunit,rrec1
56 print,rrec1
57 readu,iunit,rrec2
58 print,rrec2
59 print,intrec(0),' ka values from = ',string(rrec1(0))
60 print,' in steps of ',string(rrec1(1))
61 print,intrec(1),' different patterns'
62 print,intrec(2),' theta incidence angles from ',rrec1(2)
63 print,' in steps of ',rrec1(3)
64
65 ; Next, read the far-field pressure data
66 ; data = complexarr(nka,nharm,nobs,isymm)
67 ; assumes coefficients read as a block
68
69 vvector=complexarr(intrec(6))
70 data=complexarr(intrec(0),intrec(5),intrec(6),intrec(7))
71 for i=0,(intrec(0)-1) do begin
72     for l=0,(intrec(5)-1) do begin
73         for m=0,(intrec(7)-1) do begin
74             readu,iunit,vvector
75             data(i,l,m)=vvector
76         endfor
77     endfor
78 endfor
79 close,iunit
80 print,'Finished reading the far-field pressure'
81
82 ; Next, write the far-field data out to an unformatted
83 ; PV-Wave file
84
85 case iinc of
86     0:begin
87         extension='.fw0'
88     end
89     45:begin
90         extension='.fw4'
91     end
92     90:begin
93         extension='.fw9'
94     end
95     135:begin
96         extension='.fw3'
97     end
98     180:begin
99         extension='.fw8'
100     end
101 endcase
102 outname=fname+extension
103
104 get_lun,junit
105 openw,junit,outname
106 al=assoc(junit,complexarr(intrec(0),intrec(5),intrec(3)),
107 intrec(7)))
108 al(0)=data

```

```
109 free_lun,iunit
110 free_lun,junit
111
112 print,'Finished writing the far-field data'
113 print,'Finished everything'
114 end
1
```

```

1 ;convbco.pro
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program takes the binary .bco file and produces two
6 ;ASCII PV-Wave files:
7 ; .hdr - contains all the parameters of the job
8 ; .wco - has the axial and radial coordinates of
9 ;         the scatterer.
10 ;After this conversion, the .bco file is not
11 ;needed by PV-Wave.
12 ;-----
13
14 ; Select the data set
15
16 fname=string(replicate(32b,20))
17 print,'Input job name - no extension'
18 read,fname
19
20 ; Open the file with header and coordinate data (.bco file)
21 ; Then, define some variables and read the header
22
23 get_lun,iunit
24 openr,iunit,fname+'.bco',/f77_unformatted
25 jobname=string(replicate(32b,8))
26 comment=string(replicate(32b,80))
27 intrec=lonarr(10)
28 rrec1=fltarr(10)
29 rrec2=fltarr(10)
30 readu,iunit,jobname
31 print,'Reading the following data file: ',fname
32 print,jobname
33 readu,iunit,comment
34 print,comment
35 readu,iunit,intrec
36 print,intrec
37 readu,iunit,rrec1
38 print,rrec1
39 readu,iunit,rrec2
40 print,rrec2
41 print,intrec(0),' ka values from = ',string(rrec1(0))
42 print,' in steps of ',string(rrec1(1))
43 print,intrec(1),' different patterns'
44 print,intrec(2),' theta incidence angles from ',rrec1(2)
45 print,' in steps of ',rrec1(3)
46
47 ; Now write the header info to a formatted PV-Wave file
48
49 get_lun,junit
50 openw,junit,fname+'.hdr'
51 printf,junit,jobname
52 printf,junit,comment
53 printf,junit,format='(10i5)',intrec
54 printf,junit,format='(5g10.4/5g10.4)',rrec1

```

```

55 printf,junit,format='(5g10.4/5g10.4)',rrec2
56 free_lun,junit
57 print, 'Finished with the header info'
58
59 ; Next, read the coordinates
60
61 coord=fltarr(2,intrec(4))
62 junk=fltarr(intrec(4))
63 for i=0,1 do begin
64     readu,iunit,junk
65     coord(i,*)=junk
66 endfor
67 print,coord
68 free_lun,iunit
69
70 ; Now write the coords to a formatted PV-Wave file
71
72 get_lun,junit
73 openw,junit,fname+'.wco'
74 printf,junit,coord
75 free_lun,junit
76 print,'Finished with the coordinates'
77
78 print,'Finished everything'
79 end
1

```

```

1 ;convsub.pro
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program takes a binary surface data file
6 ;(either surface pressure, .sp#, or normal velocity, .nv#)
7 ;and produces an equivalent, binary PV-Wave file
8 ;(either .pw# for surface pressure or .vw# for normal velocity).
9 ;After this conversion, the .nv# or .sp# file is not needed
10 ;by PV-Wave.
11 ;-----
12
13 ; Select the data type and set
14
15 fname=string(replicate(32b,20))
16 surtype=' '
17 print,'Input job name - no extension'
18 read,fname
19 print,'Input p for pressure or v for velocity'
20 read,surtype
21 case surtype of
22     'p':begin
23         stype='.sp'
24         otype='.pw'
25         words='surface pressure'
26     end
27     'v':begin
28         stype='.nv'
29         otype='.vw'
30         words='normal velocity'
31     end
32 endcase
33 print,'Input incidence angle'
34 read,iinc
35 case iinc of
36     0:begin
37         extension=stype+'0'
38         oextension=otype+'0'
39     end
40     45:begin
41         extension=stype+'4'
42         oextension=otype+'4'
43     end
44     90:begin
45         extension=stype+'9'
46         oextension=otype+'9'
47     end
48     135:begin
49         extension=stype+'3'
50         oextension=otype+'3'
51     end
52     180:begin
53         extension=stype+'8'
54         oextension=otype+'8'

```

```

55         end
56     endcase
57     fullname=fname+extension
58
59 ; Open the file
60 ; Then, define some variables and read the header
61 ; Header is read from the data file, but is not
62 ; written. The .hdr file is generated with
63 ; convbco.pro
64
65 get_lun,iunit
66 openr,iunit,fullname,/f77_unformatted
67 jobname=string(replicate(32b,8))
68 comment=string(replicate(32b,80))
69 intrec=lonarr(10)
70 rrec1=fltarr(10)
71 rrec2=fltarr(10)
72 readu,iunit,jobname
73 print,'Reading the following data file: ',fullname
74 print,jobname
75 readu,iunit,comment
76 print,comment
77 readu,iunit,intrec
78 print,intrec
79 readu,iunit,rrec1
80 print,rrec1
81 readu,iunit,rrec2
82 print,rrec2
83 print,intrec(0),' ka values from = ',string(rrec1(0))
84 print,' in steps of ',string(rrec1(1))
85 print,intrec(1),' different patterns'
86 print,intrec(2),' theta incidence angles from ',rrec1(2)
87 print,' in steps of ',rrec1(3)
88
89 ; Next, read the surface data
90 ; dumvector=complexarr(nka,nharms,nareas,isyman)
91 ; assumes coefficients read as a block
92
93 surf=complexarr(intrec(0),intrec(5),intrec(3),intrec(7))
94 dumvector=complexarr(intrec(3))
95 for i=0,(intrec(0)-1) do begin
96     for l=0,(intrec(5)-1) do begin
97         for m=0,(intrec(7)-1) do begin
98             readu,iunit,dumvector
99             surf(i,l,*,m)=dumvector
100         endfor
101     endfor
102 endfor
103 free_lun,iunit
104 print,'Finished reading the '+words
105
106 ; Next, write the surface data out to an unformatted
107 ; PV-Wave file
108

```

```
109 outname=fname+oextension
110
111 get_lun,junit
112 openw,junit,outname
113 al=assoc(junit,complexarr(intrec(0),intrec(5),intrec(3),$
114 intrec(7)))
115 al(0)=surf
116 free_lun,junit
117
118 print,'Finished writing the '+words
119 print,'Finished everything'
120 end
1
```

```

1 pro rdhdr,filename=fname
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program reads header from a data set with the same
6 ;job name
7 ;-----
8
9 common header,jobname,comment,$
10 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,issymm,$
11 kastart,kainc,thetaincstart,thetaincinc,phiincstart,$
12 phiincinc,thetaobsstart,thetaobsinc,$
13 rhof,cf,rhom,young,nu,eta
14
15 ; First, some definitions
16
17 jobname=string(replicate(32b,8))
18 comment=string(replicate(32b,80))
19 intrec=lonarr(10)
20 rrec1=fltarr(10)
21 rrec2=fltarr(10)
22
23 ; Now, read the header file
24
25 get_lun,junit
26 openr,junit,fname+'.hdr'
27 readf,junit,jobname          ; 8 character name of run
28 readf,junit,comment         ; 80 characters (20 words) for comments
29 ; description, etc.
30 readf,junit,format='(10i5)',intrec
31 readf,junit,format='(5g10.4/5g10.4)',rrec1
32 readf,junit,format='(5g10.4/5g10.4)',rrec2
33 free_lun,junit
34
35 ; Give the header info some more meaningful names
36 ; First the integer record
37
38 nka=fix(intrec(0))           ; number of ka values
39 npatts=fix(intrec(1))        ; number of ff patterns
40 nthetainc=fix(intrec(2))     ; number of theta incident angles
41 nareas=fix(intrec(3))        ; number of areas (rings)
42 ncoords=fix(intrec(4))       ; number of coordinates
43 nharms=fix(intrec(5))        ; number of azimuthal harmonics
44 nobs=fix(intrec(6))          ; number of observation angles in ff patterns
45 issymm=fix(intrec(7))        ; symmetry flag, 1=yes, 2=no
46
47 ; Now, the first real record
48
49 kastart=rrec1(0)
50 kainc=rrec1(1)
51 thetaincstart=rrec1(2)
52 thetaincinc=rrec1(3)
53 phiincstart=rrec1(4)
54 phiincinc=rrec1(5)

```

```

55 thetaobsstart=rrec1(6)
56 thetaobsinc=rrec1(7)
57
58 ; the rest of rrec1 is currently not used
59 ; Finally, the second real record
60
61 rhof=rrec2(0)           ; fluid density
62 cf=rrec2(1)            ; sound speed in fluid
63 rhom=rrec2(2)          ; density of shell
64 young=rrec2(3)         ; Young's modulus of shell
65 nu=rrec2(4)            ; Poisson's ratio of shell
66 eta=rrec2(5)           ; damping factor
67
68 ; the rest of rrec2 is currently not used
69 ; Print the basic information about this data set
70
71 print,'Jobname is ',jobname
72 print,'Comments: ',comment
73 print,'There are ',nka,' ka values from ',string(kastart),$
74 ', ' in steps of ',string(kainc)
75 print,'The spectral variable is ka'
76 print,'There are ',npatts,' ff patterns'
77 print,'          ',nareas,' areas or rings'
78 print,'          ',ncoords,' coordinate values'
79 print,'          ',nthetainc,' theta incident angles'
80 print,'There are ',nthetainc,' theta incidence angles from '$
81 ',thetaincstart,' in steps of ',thetaincinc
82 print,'          ',nobs,$
83 ' theta observation angles npff pattern from ',,$
84 thetaobsstart,' in steps of ',thetaobsinc
85 print,'The fluid density is ',rhof
86 print,'The sound speed in the fluid is ',cf
87 print,'The density of shell is ',rhom
88 print,'The Youngs modulus of shell is ',young
89 print,'Poissons ratio for the shell is ',nu
90 print,'The damping factor is ',eta
91
92 end
1

```

```

1 ;grabff.pro
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program reads in PV-Wave binary far-field file
6 ;(.fw#) placing all far-field data in memory for use as
7 ;data.
8 ;-----
9
10 common header,jobname,comment,$
11 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,kastart,$
12 kainc,thetaincstart,thetaincinc,phiincstart,phiincinc,$
13 thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,eta
14
15 jobname=' '
16 pathname=' '
17 print,'input jobname (no extension) '
18 read,jobname
19 print,'input pathname including trailing /'
20 read,pathname
21 fullname=pathname+jobname
22
23 ; read vital parameters from .hdr file
24
25 rdhdr,filename=fullname
26 print,'Input incidence angle'
27 read,iinc
28 case iinc of
29     0:begin
30         extension='.fw0'
31     end
32     45:begin
33         extension='.fw4'
34     end
35     90:begin
36         extension='.fw9'
37     end
38     135:begin
39         extension='.fw3'
40     end
41     180:begin
42         extension='.fw8'
43     end
44     else:print,'Invalid angle'
45     endcase
46 fullname=fullname+extension
47
48 ; Open the file, then read in far-field data
49
50 get_lun,iunit
51 openr,iunit,fullname
52 aa=assoc(iunit,complexarr(nka,nharms,nobs,isymm))
53 data=aa(0)
54

```

```

55 ; This section of the code recombines the harmonic
56 ; components for each ka. A default increment in
57 ; phi of 30 degrees is used.
58 ; Recombined data are held in the variable npff.
59
60 deltaphi=30
61 numphi=1+fix(360./deltaphi)
62 angles=deltaphi*(pi/180.)*findgen(numphi)
63 ivec=indgen(nharms)
64 cosangle=transpose(cos(ivec#angles))
65 sinangle=transpose(sin(ivec#angles))
66 npff=complexarr(numphi,nobs,nka)
67 for kabin=0,nka-1 do begin
68     ctemp=complexarr(numphi,nobs)
69     if isymm eq 1 then ctemp=cosangle#reform(data(kabin,*,*,0))
70     if isymm eq 2 then ctemp=cosangle#reform(data(kabin,*,*,0)) $
71     +sinangle#reform(data(kabin,*,*,1))
72     npff(*,*,kabin)=ctemp
73 endfor
74 loadct,5
75 free_lun,iunit
76
77 ; This section of code produces three color images
78 ; of the npff data; target strength in db, magnitude,
79 ; and phase. The phi observation angle is fixed at
80 ; 0 degrees. The range for target strength is [-30,30].
81 ; The range for magnitude is [0,max(mag)]. The range
82 ; for phase is [-180,180].
83 ; In each case, the entire theta range has been
84 ; reconstructed [0,360]. The x-axis corresponds to
85 ; the theta observation angle; the y-axis to the
86 ; ka range.
87
88 xsz=2*nobs-1
89 ysz=nka
90 halfway=(numphi-1)/2
91
92 ; Recombine the full theta observation range
93
94 var=fltarr(xsz,ysz)
95 var(0:nobs-1,*)=abs(npff(0,*,*))
96 var(nobs:xsz-1,*)=abs(reverse(reform(npff(halfway,1:nobs-1,*)),1))
97
98 ; Calculate the target strength in dB
99
100 ts=20*alog10(var)
101 tsimg=bytsc1(ts,min=-30,max=30)
102
103 ; Calculate the magnitude and phase
104
105 magimg=bytsc1(var,min=0)
106 phase=fltarr(xsz,ysz)
107 phase(0:nobs-1,*)=(180./pi)*atan(imaginary(npff(0,*,*)), $
108 (float(npff(0,*,*))))

```

```

109 phase(nobs:xsz-1,*)=reverse(reform((180./!pi)*$
110 atan(imaginary(npff(halfway,1:nobs-1,*)),float(npff(halfway,$
111 1:nobs-1,*))))),1)
112 phaseimg=bytsc1(phase,min=-180,max=180)
113
114 ; Set up the plotting window
115
116 window,/free,xpos=200,ypos=200,xsize=700,ysize=700
117 tv,tsimg,100,100
118 tv,maging,270,100
119 tv,phaseimg,440,100
120 xyouts,/device,100,450,'Target Strength'
121 xyouts,/device,270,450,'FF Magnitude'
122 xyouts,/device,440,450,'FF Phase'
123 title='FAR-FIELD DATA: ' + comment
124 xyouts,/device,100,520,title
125 title2='Incidence angle' + string(iinc)
126 xyouts,/device,150,500,title2
127 xyouts,/device,400,650,fullname
128
129 end
1

```

```

1 ;grabsur.pro
2 ;Last modification: 20 April 92
3
4 ;-----
5 ;This program reads in a PV-Wave binary normal velocity (.vw#)
6 ;or surface pressure(.pw#) file, placing all surface data
7 ;in memory for use as sur(nka,numphi,ncoords).
8 ;-----
9
10 common header,jobname,comment,$
11 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,$
12 kastart,kainc,thetaincstart,thetaincinc,phiincstart,$
13 phiincinc,thetaobsstart,thetaobsinc,rhof,cf,rhom,young,$
14 nu,eta
15
16 fname=' '
17 pathname=' '
18 print,'input jobname (no extension)'
19 read,fname
20 print,'input pathname including trailing /'
21 read,pathname
22 fname=pathname+fname
23
24 ; Read essential parameters from .hdr
25
26 rdhdr,filename=fname
27 print,'Input incidence angle'
28 read,iinc
29 surtype=' '
30 print,'Do you want surface pressure(p) or velocity(v)?'
31 read,surtype
32 if surtype eq 'v' then begin
33     title='NORMAL VELOCITY MAGNITUDE'
34     stype='.vw'
35     endif
36 if surtype eq 'p' then begin
37     title='SURFACE PRESSURE MAGNITUDE'
38     stype='.pw'
39     endif
40 case iinc of
41     0:begin
42         extension=stype+'0'
43         end
44     45:begin
45         extension=stype+'4'
46         end
47     90:begin
48         extension=stype+'9'
49         end
50     135:begin
51         extension=stype+'3'
52         end
53     180:begin
54         extension=stype+'8'

```

```

55         end
56     endcase
57     fullname=fname+extension
58
59 ; Open the file
60
61 get_lun,iunit
62 openr,iunit,fullname,/f77_unformatted
63 aa=assoc(iunit,complexarr(nka,nharms,ncoords,isymm))
64 data=aa(0)
65 free_lun,iunit
66
67 ; This section of the code recombines harmonics for each
68 ; ka at a default increment in phi of 30 degrees.
69
70 deltaphi=30
71 numphi=1+fix(360./deltaphi)
72 angles=deltaphi*(!pi/180.)*findgen(numphi)
73 ivect=indgen(nharms)
74 cosangle=transpose(cos(ivect#angles))
75 sinangle=transpose(sin(ivect#angles))
76 sur=complexarr(nka,numphi,ncoords)
77 ctemp=complexarr(numphi,ncoords)
78 for kabin=0,nka-1 do begin
79     if isymm eq 1 then ctemp=cosangle#reform(data(kabin,*,*,0))
80     if isymm eq 2 then ctemp=cosangle#reform(data(kabin,*,*,0)) $
81     +sinangle#reform(data(kabin,*,*,1))
82     sur(kabin,*,*)=ctemp
83 endfor
84
85 ; Draw labeled pictures.
86 ; The 'Persian rug' plot is for a fixed ka (default nka/2).
87 ; The 'standing wave' plot is for a fixed phi obs angle of 0 deg.
88
89 binno=nka/2
90 phicut=0
91 x=findgen(nka)*kainc+kastart
92
93 ; Draw labeled picture
94
95 print,min(abs(sur)),max(abs(sur))
96 print,'input maximum value'
97 read,maxval
98 window,/free,xpos=100,ypos=100,xsize=900,ysize=650
99 loadct,5
100 y_scale=bindgen(1,202)
101 tv,congrid(y_scale,30,331),800,70
102 x4=[0,0]
103 y=[0,maxval]
104 plot,x4,y,xstyle=4,ystyle=1,pos=[800,70,830,401],/dev,$
105 /nodata,/noerase,yticks=7
106 tv,bytsc1(congrid(abs(reform(sur(binno,*,*)))),20*numphi,288,$
107 /interp),min=0,max=maxval,90,70
108 y=[0,1.]

```

```

109 x2=[0,360]
110 plot,x2,y,xstyle=1,ystyle=1,xticks=6,yticks=4,/nodata,$
111 /noerase,/dev,pos=[90,70,20*numphi+90,358]
112 xyouts,/device,210,500,title,size=2
113 xyouts,/device,300,480,'minval='+string(min(abs(sur)))
114 xyouts,/device,300,465,'maxval='+string(max(abs(sur)))
115 tv,bytescl(transpose(congrid(reform(abs(sur(*,phicut,*))),$
116 nka,288,/interp)),min=0,max=maxval),425,70
117 y=[0,1.]
118 x3=[min(x),max(x)]
119 plot,y,x3,xstyle=1,ystyle=1,xticks=4,yticks=11,/dev,$
120 /nodata,/noerase,pos=[425,70,713,nka+70]
121 xyouts,/device,240,570,'incidence angle = '+string(iinc)
122 xyouts,/device,240,620,fullname,size=2
123 xyouts,/device,120,425,'For ka = '+strtrim(string(x(binno)),2)
124 xyouts,/device,450,425,'For phibin = '+strtrim(string(phicut),2)
125 xyouts,/device,150,25,'Phi'
126 xyouts,/device,45,125,'Normalized axial distance',orientation=90
127 xyouts,/device,445,25,'Normalized axial distance'
128 xyouts,/device,380,200,'ka',orientation=90
129
130 end
1

```

```

1  pro hardcopy,FILENAME=filename,BITS=bits,XSIZE=xsize,YSIZE=ysize
2  ;Last modification: 24 Mar 92
3
4  ;-----
5  ;This procedure sets the display parameters to generate
6  ;an (encapsulated) Postscript file for producing hardcopy
7  ;output. Sets default values.
8  ;-----
9
10 if n_elements(filename) eq 0 then filename='junk.eps'
11 if n_elements(bits) eq 0 then bits=8
12 if n_elements(xsize) eq 0 then xsize=5.
13 if n_elements(ysize) eq 0 then ysize=5.
14
15 set_plot,'ps'
16 device,xoffset=0,yoffset=0,XSIZE=xsize,YSIZE=ysize,/inches ;sets
17 ;bounding box
18 device,/color ; put it out as color Postscript (delete for b&w)
19 device,bits_per_pixel=bits,filename=filename,/encapsulated
20
21 ; If you want to go directly to the QMS color printer
22 ; use the following settings:
23
24 ;set_plot,'ps'
25 ;device,xoffset=2.,yoffset=3.,xsize=2.,ysize=3.,/inches
26 ;device,/color,bits_per_pixel=8
27 ;device,filename='dcopy.ps'
28
29 end
1

```

```

1 pro closeit
2 ;Last modification: 21 April 92
3
4 ;-----
5 ;This program closes the hardcopy file and sets the display
6 ;back to 'x' window, replace 'x' with default driver on system
7 ;-----
8
9 device,/close_file ; close the file, so it can be sent to
10 ; the printer
11 set_plot,'x'      ; redirect the output to the 'x' window driver
12
13 end
1

```

```

1 ;ff3d.Pro
2 ;Laac modification: 24 Mar 92
3
4 ;-----
5 ; This program reads in PV-Wave binary far-field file
6 ; (.fw#), generates a 3-d display of far-field pressure for
7 ; user-defined increments in ka, and displays the result as a
8 ; movie
9 ;-----
10
11 common header,jobname,comment,$
12 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymp,$
13 kastart,kainc,thetaincstart,thetaincinc,phiincstart,$
14 phiincinc,thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,$
15 eta
16
17
18 jobname=' '
19 pathname=' '
20 print,'input jobname (no extension)'
21 read,jobname
22 print,'input pathname including trailing /'
23 read,pathname
24 fullname=pathname+jobname
25 rdhdr,filename=fullname
26 print,'Input incidence angle'
27 read,iinc
28 case iinc of
29     0:begin
30         extension='.fw0'
31     end
32     45:begin
33         extension='.fw4'
34     end
35     90:begin
36         extension='.fw9'
37     end
38     135:begin
39         extension='.fw3'
40     end
41     180:begin
42         extension='.fw8'
43     end
44 endcase
45 fullname=fullname+extension
46
47 get_lun,iunit
48 openr,iunit,fullname
49 aa=assoc(iunit,complexarr(nka,nharms,nobs,isymp))
50 data=aa(0)
51 free_lun,iunit
52
53 ; This section of the code recombines harmonics for a
54 ; default phi increment of 30 degrees. Note: a finer

```

```

55 ; increment may be desired to eliminate plotting
56 ; artifacts.
57 ; define some variables
58
59 deltaphi=30
60 numphi=1+fix(360./deltaphi)
61 angles=deltaphi*(!pi/180.)*findgen(numphi)
62 ivect=indgen(nharms)
63 cosangle=transpose(cos(ivect#angles))
64 sinangle=transpose(sin(ivect#angles))
65 npff=complexarr(numphi,nobs,nka)
66
67 ; form npff for each phi angle
68
69 for kabin=0,nka-1 do begin
70     ctemp=complexarr(numphi,nobs)
71     if isymm eq 1 then ctemp=cosangle# $
72     reform(data(kabin,*,*,0))
73     if isymm eq 2 then ctemp=cosangle# $
74     reform(data(kabin,*,*,0)) +sinangle# $
75     reform(data(kabin,*,*,1))
76     npff(*,*,kabin)=ctemp
77     endfor
78
79 ; generate frames (default of 5 )
80
81 nframes=5
82 step=fix(nka/nframes)
83 xp=fltarr(nobs,numphi)
84 yp=fltarr(nobs,numphi)
85 zp=fltarr(nobs,numphi)
86 indtheta=indgen(nobs)
87 indphi=indgen(numphi)
88 unity=replicate(1,numphi)
89
90 ; the selection of the radius, a, is arbitrary, but
91 ; it does affect the size of the display
92
93 a=5.
94 tang=!pi*indtheta/(nobs-1) ; generate a vector of theta's
95 pang=2*!pi*indphi/(numphi-1) ; same for phi's
96 zp=a*cos(tang)#unity
97 factor=a*sin(tang)
98 xp=factor#cos(pang)
99 yp=factor#sin(pang)
100 sh=bytarr(nobs,numphi) ; this holds the (bytescaled) target
101 ; strength values
102 newimg=bytarr(370,365,nframes+1)
103
104 window,0,xpos=350,ypos=300,xsize=370,ysize=365
105 loadct,5
106 kaval=findgen(nka)*kainc+kastart
107 for j=0,nframes do begin ; frequency loop
108     k=j*step

```

```

109     sh=bytescl(20*log10(transpose(abs(npff(*,*,k)))),$
110     min=-40,max=40)
111
112     ; the next routine tends to choke on a number of degenerate
113     ; polygons, which may cause some artifacts, but the
114     ; routine does not bomb and the results are useful as is
115
116     shade_surf_irr,sh*zp,sh*xp,sh*yp,shades=sh,$
117     xrange=[-1000,1000],yrange=[-1000,1000],zrange=[-1000,1000]
118     xyouts,0.4,0.8,'ka = '+strtrim(string(kaval(k))),$
119     size=1,/normal
120     wait,0.0001
121     newimg(*,*,j)=tvrd(0,0,370,365)
122     endfor
123
124 ; redisplay the frames as a movie
125
126 movie,newimg,order=0
127
128 ; make a hard copy of any frame of interest
129
130 ;hardcopy,file='clb93d#18.eps',xsize=3.70,ysize=3.65
131 ;tv,newimg(*,*,18)
132 ;closeit
133
134 end
1

```

```

1 ;plotsur.pro
2 ;Last modification: 21 April 92
3
4
5 ;-----
6 ;This program makes an animated multi-color plot showing how
7 ;surface pressure or normal velocity vary with axial node
8 ;number and frequency. It assumes that grabsur.pro has been
9 ;run and sur(nka,numphi,ncoords) is in memory
10 ;-----
11
12 common header,jobname,comment,$
13 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isyms,$
14 kastart,kainc,thetaincstart,thetaincinc,phiincstart,phiincinc,$
15 thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,eta
16
17 window,/free
18 plot,sur(0,0,*),ystyle=1,xstyle=1,yrange=[0,maxval],/nodata, $
19 xtitle='Axial bin number',ytitle='Magnitude'
20 for i=0,nka-1 do begin
21     oplot,abs(sur(i,0,*)),color=i
22     wait,.02
23 endfor
24
25
26 end
1

```

```

1 ;enve` pro
2 ;Last modification: 15 May 92
3
4 ;-----
5 ;This program generates a set of overlaid spectral plots from
6 ;supposedly equivalent solutions generated by different codes,
7 ;allows the user to select a subset of the solutions, and
8 ;produces a mean and envelope that bound the variations within
9 ;the data sets selected.
10 ;Note: This program assumes the data to be compared has been
11 ;read in prior to the execution of envel.pro. The data for
12 ;the Benchmark exercise was stored in variables named amag0
13 ;through wmag0, all of which were of the size (nobs,nkas). The data
14 ;sets were reduced by the selecting the phi observation angle prior
15 ;to execution.
16 ;The example given was produced with modified versions of this
17 ;routine.
18 ;-----
19 ;
20 ;First set some default plotting styles
21 ;
22 ix.style=1
23 iy.style=1
24 ix.ticks=6
25 iy.ticks=6
26 ;
27 id= ' '
28 print,'Enter index of theta angle desired'
29 if bmname ne '4a' and bmname ne '4b' then begin
30     print,'0=0obs, 18=45obs, 36=90obs, 54=135obs, 72=180obs'
31     fact=2.5
32     endif else begin
33         print,'0=0obs,15=45obs,30=90obs,45=135obs, 60=180obs'
34         fact=3.0
35     endelse
36 read,tangle
37 ;
38 print,'Enter number of data sets available'
39 read,nsets
40 print,nsets
41 print,'Enter number of ka values'
42 read,nkas
43 ;
44 ;Set up x-axis scale
45 x=findgen(nkas)*.01+.2
46 if (bmname eq '4a' or bmname eq '4b') then x=findgen(nkas)*.0094+.397
47 ;
48 ;Determine max(x) as legend positioning is tied to this value
49 maxx=max(x)
50 if(bmname eq '1a')then nkas=330 ; special case
51 ;
52 ;The spectral plots are saved in the array results.
53 results=fltarr(nsets,nkas)
54 results(0,*)=amag0(tangle,0:nkas-1)

```

```

55 results(1,*)=cmag0(tangle,0:nkas-1)
56 results(2,*)=smag0(tangle,0:nkas-1)
57 results(3,*)=wmag0(tangle,0:nkas-1)
58 if nsets ge 5 then results(4,*)=nmag0(tangle,0:nkas-1)
59 if nsets ge 6 then results(5,*)=fmag0(tangle,0:nkas-1)
60 resultsave=results
61
62 ; The y-axis is autoscaled to 1.1*max of the data.
63 maxdata=1.1*max(results)
64 ;
65 ; The following statements set up line types &
66 ; colors and the legend on raw data plot.
67
68 ltype=intarr(nsets)
69 ltype(0:3)=[0,0,0,4]
70 colors=intarr(nsets)
71 colors(0:3)=[200,26,164,200]
72 label=strarr(nsets)
73 label(0)='1 - axsar'
74 label(1)='2 - chief'
75 label(2)='3 - sara'
76 label(3)='4 - wascat'
77 if nsets ge 5 then begin
78     label(4)='5 - nashua'
79     ltype(4)=4
80     colors(4)=26
81     endif
82 if nsets ge 6 then begin
83     ltype(5)=4
84     colors(5)=164
85     label(5)='6 - fist'
86     endif
87 ;These set up the same for the envelope plot.
88 lbl=['average','min / max']
89 ltyp=0
90 clr=[164,200]
91 ;
92 window,/free,xsiz=625,ysiz=600,xpos=0,ypos=250
93 loadct,12
94 !x.title='ka'
95 !y.title='NPFF, mag'
96 plot,x,results(0,*),yrange=[0,maxdata],color=colors(0)
97 for i=1,nsets-1 do oplot,x,results(i,*),color=colors(i),$
98 linestyle=ltype(i)
99 legend,label,colors,ltype,psym,.7*maxx,.87*maxdata,maxdata/40.
100 print,'Input problem identifier; e.g. BM1B 45 INC'
101 read,id
102 id=strcompress(strupcase(id+','+'string(fix(fact*tangle))+ ' obs '))
103 xyouts,.6,.90,/normal,id,color=200,size=1.2
104 xyouts,.6,.87,/normal,'RAW DATA',color=200,size=1.2
105 ans='y'
106 while ans eq 'y' do begin
107     maxdiff=0.
108     setid=' '

```

```

109 results=resultsave
110 ;
111 ;User picks how many and which data sets to include in
112 ;the envelope. The user can chose to reprocess the envelope.
113 ;
114 print,'how many data sets do you want to include'
115 read,numin
116 if numin eq nsets then setid='ALL DATA SETS'
117 if numin ne nsets then begin
118     sets=intarr(numin)
119     print,'which data sets do you want to include',$
120     '1 - nsets possible'
121     read,sets
122     print,'you have asked to include sets',sets
123     for i=0,numin-1 do setid=setid+string(sets(i))
124     for i=0,numin-1 do sets(i)=sets(i)-1
125     results=results(sets,*)
126     endif
127 average=avg(results,0)
128 window,/free,xsiz=625,ysiz=600,xpos=625,ypos=250
129 mini=min(results)
130 maxdum=fltarr(nkas)
131 mindum=fltarr(nkas)
132 for i=0,nkas-1 do begin
133     maxdum(i)=max(results(*,i))
134     mindum(i)=min(results(*,i))
135     endfor
136 maxd=1.2*max(maxdum)
137 plot,x,(average),/nodata,yrange=[0,maxdata]
138 oplot,x,(average),color=164
139 oplot,x,(maxdum)
140 oplot,x,(mindum)
141 legend,lbl,clr,ltyp,psym,.7*maxx,.8*maxdata,.3
142 xyouts,.6,.85,/normal,strcompress(setid),color=200,siz=1.2
143 xyouts,.6,.90,/normal,id,color=200,siz=1.2
144 maxdiff=max(maxdum-mindum)
145 print,'Maximum difference = ',maxdiff,' at ',
146 x(where(maxdum-mindum eq maxdiff))
147 ;oplot,x,ntruth(0,*),color=100
148 print,'Do you want to reprocess, y or n'
149 read,ans
150 endwhile
151 end
1

```

```

1 pro compff_m
2 ;
3 ;Last modification: 19 May 1992
4 ;
5 ;-----
6 ; This procedure plots the target strength image for two codes and
7 ; plots the difference between the images.
8 ; The difference is taken between dB values which have first been
9 ; subjected to a user-specified threshold.
10 ; The user is allowed interactive control over the dynamic range
11 ; of the images and the difference image.
12 ;
13 ; Note: The procedure assumes npff_m.pro has been compiled.
14 ;-----
15
16 common header, jobname, comment, $
17 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,$
18 kastart,kainc,thetaincstart,thetaincinc,phiincstart,phiincinc,$
19 thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,eta
20 ;
21 common share1, npff, numphi, diff
22 ;
23 ; Get the first set of data
24 ;
25 print,'Prompting for the threshold value in db'
26 read,thresh_db
27 print,'Prompting for dynamic range desired for difference'
28 read,dmin,dmax
29 print,'Prompting for the first far-field data file.'
30 npff_m
31 npff1=abs(npff)
32 ;npff1(where(abs(npff1) le thresh))=thresh
33 nkal=nka
34 ;
35 print,'Prompting for the second far-field data file.'
36 npff_m
37 npff2=abs(npff)
38 ;npff2(where(abs(npff2) le thresh))=thresh
39 ;
40 xsz=2*nobs-1
41 ysz=nka
42 ;
43 ysz1=nkal
44 ;Calculate the target strengths
45 ;
46 ansd='n'
47 anst='n'
48 ans='y'
49 repeat begin
50     print,'Input the index of the phi angle (0 to 6).'
51     read,phiinc
52     halfway=(numphi-1)/2 + phiinc
53     var1=fltarr(xsz,ysz1)
54     var1(0:nobs-1,*)=(npff1(phiinc,*,*))

```

```

55     var1(nobs:xsz-1,*)=(reverse(reform(npff1(halfway,$
56     1:nobs-1,*)),1))
57     ts1=20*alog10(var1)
58     var2=fltarr(xsz,ysz)
59     var2(0:nobs-1,*)=(npff2(phiinc,*,*))
60     var2(nobs:xsz-1,*)=(reverse(reform(npff2(halfway,$
61     1:nobs-1,*)),1))
62     ts2=20*alog10(var2)
63     loadct,5
64     print,'max values',max(ts1),max(ts2),' min values',$
65     min(ts1),min(ts2)
66     print,'input dynamic range for ts pictures'
67     read,mints,maxts
68     top=202
69     print,'input number of colors on system'
70     read,topval
71     window,0,xsize=2*xsz,ysize=2*ysz,xpos=100,ypos=200,$
72     title='Set 1'
73     tv,rebins(bytscl(ts1,min=mints,max=maxts,top=topval),$
74     2*xsz,2*ysz)
75     window,1,xsize=2*xsz,ysize=2*ysz,xpos=800,ypos=200,$
76     title='Set 2'
77     tv,rebins(bytscl(ts2,min=mints,max=maxts,top=topval),$
78     2*xsz,2*ysz)
79     window,5,xsize=100,ysize=300,xpos=1125,ypos=550
80     y_scale=bindgen(1,topval)
81     tv,congrid(y_scale,30,250),45,30
82     x=[0,0]
83     y=[mints,maxts]
84     plot,x,y,xstyle=4,ystyle=1,pos=[45,30,75,280],/device,$
85     /nodata,/noerase,yticks=4
86     print,'hit when ready'
87     hak
88     repeat begin
89         ts1(where(ts1 le thresh_db))=thresh_db
90         ts2(where(ts2 le thresh_db))=thresh_db
91         diff=ts1-ts2
92         repeat begin
93             loadct,11
94             ;stretch,0,255
95             window,2,xsize=2*xsz,ysize=2*ysz,xpos=450,ypos=200,$
96             title='Difference'
97             tv,rebins(bytscl(diff,min=dmin,max=dmax,top=topval),$
98             2*xsz,2*ysz)
99             window,6,xsize=100,ysize=300,xpos=1125,ypos=200
100            tv,congrid(y_scale,30,250),45,30
101            x=[0,0]
102            y=[dmin,dmax]
103            plot,x,y,xstyle=4,ystyle=1,pos=[45,30,75,280],/device,$
104            /nodata,/noerase,yticks=4,color=101
105            print,'Do you want a new dynamic range for the difference'
106            read,ansd
107            if(ansd eq 'y') then begin
108                print,'Input min and max desired'

```

```

109             read,dmin,dmax
110             endif
111             endrep until ansd eq 'n'
112             print,'Do you want another threshold value? (y/n)'
113             read,anst
114             if(anst eq 'y') then begin
115                 print,'Input new threshold value'
116                 read,thresh_db
117             endif
118             endrep until anst eq 'n'
119             print,'Do you want another phi angle? (y/n)'
120             read,ans
121             endrep until ans eq 'n'
122 end
1

```

```

1 ;fig8.pro
2 ;Last modification: 20 Jul 92
3
4 ;-----
5 ;This program generates and displays color surfaces
6 ;which portray target strength as a function of frequency
7 ;and azimuth
8 ;-----
9
10 common header,jobname,comment,$
11 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,$
12 kastart,kainc,thetaincstart,thetaincinc,phiincstart,$
13 phiincinc,thetaobsstart,thetaobsinc,$
14 rhof,cf,rhom,young,nu,eta
15
16 datapath='/data/schenck/dset2/'
17 jobname=' '
18 print,'input runname (without extension'
19 print,' assumed to be in /data/schenck/dset2)'
20 read,jobname
21
22 fullname=datapath+jobname
23 rdhdr,filename=fullname
24
25 print,'Input incidence angle'
26 read,iinc
27 case iinc of
28     0:begin
29         extension='.vw0'
30     end
31     45:begin
32         extension='.vw4'
33     end
34     90:begin
35         extension='.vw9'
36     end
37     135:begin
38         extension='.vw3'
39     end
40     180:begin
41         extension='.vw8'
42     end
43     endcase
44 fullname=fullname+extension
45
46 get_lun,iunit
47 openr,iunit,fullname
48 aa=assoc(iunit,complexarr(nka,nharms,ncoords,isymm))
49 vel=aa(0)
50 free_lun,iunit
51
52 window,2,xsize=50,ysize=512,xpos=150,ypos=300
53 loadct,11
54 cbar=bytarr(50,512)

```

```

55 for i=0,255 do cbar(*,i*2:i*2+1)=i
56 tv,cbar
57
58 nendnodes=12 ; presently not in header file
59 npts=nareas-2*nendnodes
60 mfact=5 ; multiplication factor for display
61 nfact=5 ; multiplication factor for display
62 window,4,xpos=250,ypos=375,xsize=mfact*npts,$
63 ysize=nfact*(2*nharms-1)
64
65 char='g' ; a junk character
66
67 for l=0,0 do begin
68
69     f=90+10*l
70     if isymm eq 2 then begin
71         vf=reform(vel(f,*,nendnodes:nendnodes+npts-1,*))
72         vh=complexarr(2*nharms-1,npts)
73         for h=-(nharms-1),nharms-1 do begin
74             if h eq 0 then begin
75                 vh(h+nharms-1,*)=reform(vf(h,*,0))
76             endif
77             if h gt 0 then begin
78                 vh(h+nharms-1,*)=0.5*reform(vf(h,*,0))-com-
plex(0,1)*$
79                 reform(vf(h,*,1))
80             endif
81             if h lt 0 then begin
82                 vh(h+nharms-1,*)=0.5*reform(vf(-h,*,0))-com-
plex(0,1)*$
83                 reform(vf(-h,*,1))
84             endif
85         endfor
86     endif else begin ; to allow cosine only representation
87         vf=reform(vel(f,*,nendnodes:nendnodes+npts-1))
88         vh=complexarr(2*nharms-1,npts)
89         for h=-(nharms-1),nharms-1 do begin
90             if h eq 0 then begin
91                 vh(h+nharms-1,*)=reform(vf(h,*))
92             endif
93             if h gt 0 then begin
94                 vh(h+nharms-1,*)=0.5*reform(vf(h,*))
95             endif
96             if h lt 0 then begin
97                 vh(h+nharms-1,*)=0.5*reform(vf(-h,*))
98             endif
99         endfor
100     endelse
101     npts=47 ; ### to force npts to be odd, not general!
102     vh=vh(*,0:46) ; same
103
104     kmat=complexarr(2*nharms-1,npts)
105     sym=complexarr(2*nharms-1,npts)
106     middle=(npts-1)/2

```

```

107     for i=0,2*(nharms-1) do begin
108         vect0=reform(vh(i,*))
109         kmat(i,*)=fft(vect0,1)
110         sym(i,middle)=kmat(i,0)
111         unitv=complexarr(middle+1)
112         unitv(0)=complex(1,0)
113         unitv(1)=complex(cos(1pi*(npts-1)/npts),$
114             sin(1pi*(npts-1)/npts))
115         for j=1,middle do begin
116             unitv(j)=unitv(1)*unitv(j-1)
117             sym(i,middle+j)=unitv(j)*kmat(i,npts-j)
118             sym(i,middle-j)=conj(unitv(j))*kmat(i,j)
119         endfor
120     endfor
121     kmat=sym
122     new=transpose(kmat)
123     sign=float(new)/abs(float(new))
124     newer=sign*abs(float(new))
125     tv,rebin(bytescl(newer,min=-25,max=25,top=254),mfact*npts,$
126         nfact*(2*nharms-1))
127
128     xyouts,0.05,0.05,'freq='+strtrim(string((f+1)/2.),2),$
129         /normal,color=254
130
131     print,'Frequency bin = ',f
132     wait,0.5
133     char=get_kbrd(0)
134     if char ne '' then begin
135         print,'Hit any key to continue'
136         hak
137     endif
138
139     endfor
140 end
1

```

```

1 ; window.pro
2 ; Last modification: 9 Jun 92
3
4 ;-----
5 ; This program takes model data in the frequency domain and performs
6 ; the windowing prescribed by NRL, then presents a comparison of the
7 ; model results with the experimental data in the frequency domain
8 ;-----
9
10 common header,jobname,comment,$
11 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,$
12 kastart,kainc,thetaincstart,thetaincinc,phiincstart,phiincinc,$
13 thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,eta
14
15 ; Select the data set
16
17 ; Change pathname data file names as required
18 pid=' '
19 print,'input pid, eg 4a 4b'
20 read,pid
21 pathname='/data/schenck/dset2/'
22 ;pathname=' '
23 ;print,'input pathname'
24 ;read,pathname
25 jobname=' '
26 print,'Input jobname (no extension) '
27 read,jobname
28 print,'Input incidence angle in degrees'
29 read,iinc
30 case pid of
31     '4a':begin
32         refname='bm4aref.daw' ;reference time series
33         case iinc of
34             0:begin
35                 extension='.fw0' ; computational data
36                 cdat='cleana0.dat' ; cleaning window data
37                 shifty=-0 ; to account for orientation of target
38                 ; in NRL tank
39                 fname='bm4a0.daw' ; experimental echo time series
40             end
41
42             45:begin
43                 extension='.fw4'
44                 cdat='cleana4.dat'
45                 shifty=-15
46                 fname='bm4a45.daw'
47             end
48
49             90:begin
50                 extension='.fw9'
51                 cdat='cleana9.dat'
52                 shifty=-30
53                 fname='bm4a90.daw'
54             end

```

```

55         endcase
56     end
57     '4b':begin
58         refname='bm4bref.daw'
59         case iinc of
60             0:begin
61                 extension='.fw0'
62                 cdat='cleanb0.dat'
63                 shifty=-0
64                 fname='bm4b0.daw'
65             end
66
67             45:begin
68                 extension='.fw4'
69                 cdat='cleanb4.dat'
70                 shifty=-15
71                 fname='bm4b45.daw'
72             end
73
74             90:begin
75                 extension='.fw9'
76                 cdat='cleanb9.dat'
77                 shifty=-30
78                 fname='bm4b90.daw'
79             end
80         endcase
81     end
82     else:print,'Incorrect jobname'
83 endcase
84
85 fullname=pathname+jobname
86
87 ; Read the header and open the file
88 rdhdr,filename=fullname
89 fullname=fullname+extension
90 get_lun,iunit
91 openr,iunit,fullname
92 aa=assoc(iunit,complexarr(nka,nharms,nobs,isymm))
93 data=aa(0)
94 free_lun,iunit
95 print,'Finished reading model data set'
96
97 ; this next part could be made simpler,
98 ; since we only want phicut=0
99
100 deltaphi=180
101 numphi=1+fix(360./deltaphi)
102 angles=deltaphi*(!pi/180.)*findgen(numphi)
103 ivect=indgen(nharms)
104 cosangle=transpose(cos(ivect#angles))
105 sinangle=transpose(sin(ivect#angles))
106 npff=complexarr(numphi,nobs,nka)
107 for kabin=0,nka-1 do begin
108     ctemp=complexarr(numphi,nobs)

```

```

109     if isymm eq 1 then ctemp=cosangle#reform(data(kabin,*,*,0))
110     if isymm eq 2 then ctemp=cosangle#reform(data(kabin,*,*,0)) $
111     +sinangle#reform(data(kabin,*,*,1))
112     npff(*,*,kabin)=ctemp
113     endfor
114
115 ; Calculate the full 360 degree pattern of computed target $
116 ; strength
117
118 xsz=2*nobs-1
119 ysz=nka
120 halfway=(numphi-1)/2
121 var=complexarr(xsz,ysz)
122 var(0:nobs-1,*)=npff(0,*,*) ; phicut = 0
123 var(nobs:xsz-1,*)=reverse(reform(npff(halfway,1:nobs-1,*)),1)
124 print,'Full azimuthal pattern constructed'
125
126 window,5,xsize=900,ysize=400,xpos=200,title=$
127 '5.Target strength vs angle and frequency'
128 loadct,5
129 bytc=bytsc1(20*log10(abs(var)),min=-30,max=30,$
130 top=id.n_colors-1)
131 tv,rebin(bytc,2*121,331),600,35
132 xyouts,0.72,0.95,'Unwindowed model',/normal
133
134 add=42 ; adjust lowest non-zero bin of computed data
135 ; (firstbin*binwidth=1.7kHz)
136
137 ; First, we'll work with the reference signal
138
139 ; Read in the reference (incident) time series
140
141 refname='/data/schenck/expt4/'+refname
142 get_lun,iunit
143 openr,iunit,refname,/f77_unformatted
144 bb=assoc(iunit,fltarr(4096))
145 ref=bb(0) ;reference (incident) time series
146 free_lun,iunit
147 tape_info=ref(4080:4095)
148 print,tape_info
149 ref(4080:4095)=0. ; replace last 16 bins with zero
150
151 ref=shift(ref,380)
152 window,1,xsize=400,ysize=200,xpos=150,ypos=700,$
153 title='1.Reference signal'
154 plot,ref(0:2500),yrange=[-25,25],ystyle=1 ; plot the time series
155 tlength=12423 ; to achieve 2 usec sample period
156 ; in the computed time series
157 rpad=replicate(0.0,tlength)
158 rpad(0:4095)=ref
159 cpad=fft(rpad,-1)
160 magpinc=avg(abs(cpad(add:add+nka-1)))
161
162 ;;; change pathnames if required

```

```

163 fullname='/data/schenck/expt4/'+fname
164 get_lun,iunit
165 openr,iunit,fullname,/f77_unformatted
166 aa=assoc(iunit,fltarr(121,4096))
167 echo=aa(0)
168 free_lun,iunit
169 print,'Finished reading the experimental data'
170
171 ; The correction factor accounts for the source,
172 ; target, receiver geometry
173 geocorrect=2*2*sqrt(2.18)/(2.239*2.18*0.0254*sqrt(1.96))
174 ech=geocorrect*echo
175
176 window,2,xsize=400,ysize=200,xpos=150,ypos=470,$
177 title='2.Experimental echo'
178 loadct,5
179 window,3,xsize=400,ysize=200,xpos=150,ypos=10,$
180 title='3.Computed time series'
181 window,4,xsize=400,ysize=200,xpos=150,ypos=240,$
182 title='4.Windowed time series'
183
184 ffsurf=complexarr(121,331)
185 modspec=complexarr(121,331)
186
187 ; Now develop cosine squared window for each observation angle
188 ; this information comes from Brian Houston's memo re from
189 ; 3600 to 3700 data
190
191 cdat='~/prob4/'+cdat
192 get_lun,kunit
193 openr,kunit,cdat
194 cleanarr=intarr(4,121)
195 readf,kunit,cleanarr
196 free_lun,kunit
197
198 print,cleanarr
199
200 ; Now we'll start the large loop over observation angle
201
202 for j=0,120 do begin
203
204     k=-(j+shifty)
205     if k lt 0 then k=k+121
206
207     ; Next, let's process the experimental echo
208
209     print,echo(k,4094)          ; print angles of observation
210     ech(k,4080:4095)=0.        ; zero the last sixteen bins
211
212     longecho=replicate(0.,tlength)
213     longecho(0:4095)=reform(ech(k,*))
214     backspec=fft(longecho,-1) ; convert to frequency domain
215     ffsurf(j,*)=backspec(add:nka-1)
216

```

```

217 wset,2
218 plot,ech(k,0:2500),yrange=[-10.,10.],ystyle=1
219
220 ; Generate and overplot the cleaning window
221
222 frontstart=cleanarr(0,k)
223 frontend=cleanarr(1,k)
224 backstart=cleanarr(2,k)
225 backend=cleanarr(3,k)
226 flength=frontend-frontstart+1
227 blength=backend-backstart+1
228 piover2=pi/2.
229 frontw=(sin(piover2*findgen(flength)/float(flength-1)))^2
230 backw=(cos(piover2*findgen(blength)/float(blength-1)))^2
231 cleanw=replicate(0.0,4096)
232 cleanw(frontstart:frontend)=frontw
233 cleanw(frontend+1:backstart-1)=1.0
234 cleanw(backstart:backend)=backw
235
236 longclean=replicate(0.,tlength)
237 longclean(0:4095)=cleanw
238 oplot,5.*longclean(0:2500),color=120
239 wait,0.02
240
241 ; Next, we will process the computed results
242
243 bsff=reform(var(j,*))*cpad(add:add+nka-1) ; positive freqs
244 part=replicate(0.0,tlength)
245 padded=complex(part,part) ; a complex array of zeroes
246 padded(0+add:add+nka-1)=bsff ; fill in the negative frequencies
247 rev=conj(reverse(bsff)) ; ensures a real time series
248
249 ; fill in the negative freqs
250 padded(tlength-nka+1-add:tlength-add)=rev(0:nka-1) ;
251 tseries=fft(padded,1) ; take the FFT
252 modt0=float(tseries)
253
254 wset,3
255 plot,modt0,yrange=[-10.0,10.0],ystyle=1
256 oplot,imaginary(tseries),color=120 ; this should be zero
257 wait,0.02
258
259 ; Next, apply the cleaning window to the computed results
260
261 newc=longclean*modt0 ; apply the window in time domain
262 newspec=fft(newc,-1) ; compute the windowed model spectrum
263 modspec(j,*)=newspec(add:add+nka-1)
264
265 wset,4
266 plot,newc(0:2500),yrange=[-10.0,10.0],ystyle=1
267 oplot,5.*longclean(0:2500),color=120
268 wait,0.02
269
270 print,'j=',strtrim(string(j),2),', ' k=',strtrim(string(k),2)

```

```

271     endfor
272
273
274 ;nrling=reverse(shift(20*log10(abs(ffsurf)/magpinc),
275 ;shifty,0)); the exptl data
276 nrling=20*log10(abs(ffsurf)/magpinc)    ; the exptl data
277
278 ; the shift and reverse are needed to make my NRL's
279 ;angles the same as for Benchmark problems
280
281 bytx=bytsc1(nrling,min=-30,max=30,top=!d.n_colors-1)
282
283 wset,5
284 tv,rebin(bytx,2*121,331),300,35
285 xyouts,0.4,0.95,'Experimental data',/normal
286
287 bytw=bytsc1(20*log10(abs(modspec)/magpinc),min=-30,$
288 max=30,top=!d.n_colors-1)
289
290 wset,5
291 loadct,5
292 tv,rebin(bytw,2*121,331),20,35
293 xyouts,0.08,0.95,'Windowed model',/normal
294
295 phasec=atan(imaginary(var),float(var))
296 phasex=atan(imaginary(ffsurf),float(ffsurf))
297 phasew=atan(imaginary(modspec),float(modspec))
298 window,/free
299 tvscl,phasew,50,50
300 tvscl,phasex,200,50
301 tvscl,phasec,350,50
302
303
304 goto,jump
305 hardcopy,filename=strtrim(jobname,2)+$
306 strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.31
307 tv,bytc
308 closeit
309 hardcopy,filename='win'+strmid(jobname,1,2)+$
310 strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.3
311 tv,rebin(bytw,121,330)
312 closeit
313 hardcopy,filename='ex'+strmid(jobname,1,2)+$
314 strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.3
315 tv,rebin(bytex,121,330)
316 closeit
317 jump:
318
319 end
1

```

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE July 1992		3. REPORT TYPE AND DATES COVERED Final	
4. TITLE AND SUBTITLE VISUALIZATION OF SCATTERING STRENGTH OF ELASTIC BODIES IN A FLUID				5. FUNDING NUMBERS AN: DN3/0026 PE: 060569NE	
6. AUTHOR(S) H. A. Schenck (NCCOSC RDT&E Division) J. L. Fales (LANL)					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Command, Control and Ocean Surveillance Center (NCCOSC) RDT&E Division (NRaD) San Diego, CA 92152-5000 Los Alamos National Laboratory (LANL) Los Alamos, NM 87545				8. PERFORMING ORGANIZATION REPORT NUMBER NRaD TD 2287	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203				10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES					
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.				12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) As part of the Submarine Technology Program, the Defense Advanced Research Projects Agency (DARPA) recently sponsored a Low-Frequency Structural Acoustics Benchmark Exercise. The purpose of the exercise was to test and validate several major computational codes that have been developed to solve acoustic scattering problems of elastic objects in a fluid. This report describes some of the visualization techniques and procedures that were developed to review, compare, and analyze the large amount of computational data generated in the exercise.					
14. SUBJECT TERMS visualization structural acoustics target strength acoustic scattering				15. NUMBER OF PAGES 66	
				16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED		18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED		19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	
				20. LIMITATION OF ABSTRACT SAME AS REPORT	

UNCLASSIFIED

21a. NAME OF RESPONSIBLE INDIVIDUAL H. A. Schenk	21b. TELEPHONE (Include Area Code) (619) 553-2012	21c. OFFICE SYMBOL Code 701
---	--	--

INITIAL DISTRIBUTION

Code 0012	Patent Counsel	(1)
Code 144	V. Ware	(1)
Code 541	B. Sotirin	(1)
Code 701	H. A. Schenck	(20)
Code 705	M. Morrison	(1)
Code 7103	R. Smith	(1)
Code 712	G. W. Benthien	(1)
Code 713	J. Price	(1)
Code 731	G. Thomson	(1)
Code 733	E. P. McDaid	(1)
Code 734	J. C. Lockwood	(1)
Code 743	S. Sullivan	(1)
Code 9103	J. M. Baird	(1)
Code 952B	J. Puleo	(1)
Code 961	Archive/Stock	(6)
Code 964B	Library	(2)

Defense Technical Information Center Arlington, VA 22304-6145	(4)	Defense Advanced Research Projects Agency Arlington, VA 22203-1714	(2)
Center for Naval Analyses Alexandria, VA 22302-0268		NCCOSC Washington Liaison Office Washington, DC 20363-5100	
Navy Acquisition, Research & Develop- ment Information Center (NARDIC) Washington, DC 20360-5000		Office of the Chief of Naval Operations Washington, DC 20350-2000	(2)
Space & Naval Warfare Systems Cmd Washington, DC 20363-5100	(2)	Office of Chief of Naval Research Arlington, VA 22217-5000	(3)
Naval Sea Systems Command Washington, DC 20362-5101		Naval Research Laboratory Washington, DC 20375-5000	(4)
Naval Research Laboratory Underwater Sound Ref Detachment Orlando, FL 32856		Naval Surface Warfare Center Carderock Division Bethesda, MD 20084-5000	(7)
Naval Undersea Warfare Center New London, CT 06320-5594	(2)	Naval Postgraduate School Monterey, CA 93943-5000	
Georgia Institute of Technology Atlanta, GA 30332-0405		Massachusetts Institute of Technology Cambridge, MA 02139	(2)
University of Kentucky Lexington, KY 40506-0046		Lawrence Livermore National Laboratory Livermore, CA 94550	
Los Alamos National Laboratory Los Alamos, NM 87545	(3)		

INITIAL DISTRIBUTION (Cont'd)

AT&T Whippany, NJ 07981-0903	(2)	AT&T Bell Laboratories Arlington, VA 22202	
BBN Systems & Technologies New London, CT 06320-6147	(2)	BBN Systems & Technologies Cambridge, MA 02138	
Cambridge Acoustical Associates Cambridge, MA 02139		Computer Sciences Corporation San Diego, CA 92110-5164	(2)
General Dynamics Electric Boat Division Groton, CT 06340		International Business Machines Corporation Kingston, NY 12401	
Newport News Shipbuilding Newport News, VA 23607		Newport News Shipbuilding Arlington, VA 22202	
Precision Visuals, Inc. Boulder, CO 80301	(2)	SRI, Inc. Arlington, VA 22209	
Weidlinger Associates New York, NY 10001			