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

Approved for public release; distribution is unlimited.

92-23426

92-821099
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.
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
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. Shutters, Head
Surveillance Department

ACKNOWLEDGMENTS

The authors gratefully acknowledge the assistance of Mary Marshall and Billie Wheat of Los Alamos National Laboratory and Vince Insxiengmay 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.
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.
Jobname is Cla
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.
Figure 4. Normal velocity, c1b, 90 incidence.

(a) "Persian rug," for kabin = 100

(b) "Standing wave" plot, for phibin = 0
Figure 5. Three-dimensional representation of far-field pressure at several frequencies.
Figure 6. Axial distribution of normal velocity for various frequencies.
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 unwindowed 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(Dim1,...,Dimn)  
  Byte array, each element its subscript.
- BYTARR(Dim1,...,Dimn)  
  Returns a byte vector or array.
- COMPLEXARR(Dim1,...,Dimn)  
  Complex single-precision vector or array.
- FINDGEN(Dimi,...,Dimn)  
  Floating array, each element its subscript.
- FLTARR(Dimi,...,Dimn)  
  Single-precision floating vector or array.
- INDGEN(Dimj,...,Dimn)  
  Integer array, each element its subscript.
- LNTARR(Dimi,...,Dimn)  
  Returns an integer vector or array.
- LONARR(Dimj,...,Dimn)  
  Returns longword integer vector or array.
- REPLICATE(Value,Dim1,...,Dmn)  
  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,Dim1,...,Dimn)  
  Resamples array to given dimensions.
- REFORM(Array,Dimi,...,Dimn)  
  Reformats array without changing contents.
- SHIFT(Array,Si,...,Sn)  
  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,...,Dimn]]])  
  Converts parameter to integer type.
- STRING(Expr1,...,Exprn)  
  Converts parameter to string type.

File manipulation routines

- CLOSE[,Unit1,...,Unitn]  
  Closes one or more files.
- FREE_LUN,Unit1,...,Unitn)  
  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]
PLOTS, X[, Y[, Z]]
SET_PLOTS, Device
XYOUTS, X, Y, String

Cycles images stored in 3D array.
Draws lines (vectors) and points.
Specifies graphics device.
Sends text to selected graphics device.

General Mathematical functions

ABS(X)
CONJ(X)
FFT(Array, Direction)

Absolute value.
Complex conjugate.
Fast Fourier transform.

Image display routines

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

Displays an image on the display screen.
Scales and displays an image.

Image processing routines

FFT(Array, Direction)

Fast Fourier transform.

Input and output routines

ASSOC(Unit, Array_structure[, Offset])
PRINT, Expri,..., Exprn)
READ, Vari,..., Varn)
READF, Unit, Vari,..., Varn)
READU, Unit, Vari,..., Varn)

 Associates variable with file structure.
 Prints to the standard output stream.
 Reads from the standard input stream.
 Reads from the specified file unit.
 Reads unformatted input.

Plotting routines

OPlot, X[, Y]
PLOT, X[, Y]

Plots vector arguments over old axis.
Plots vector arguments.

Programming routines

HAK
WAIT, Seconds

"Hit any key to continue" function.
Delays program execution.

String processing routines

STRMID(Expr, First_Character, Length)
STRTRIM(String[, flag])

Extracts substring of string expression.
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
This program takes a binary far-field pressure (.Ff#) file and produces an equivalent PV-Wave (.fw#) file. After this conversion, the .ff# is not needed by PV-Wave.

Select the data set

fname=string(replicate(32b,20))
print,'Input job name - no extension'
read,fname
print,'Input incidence angle'
read,iinc

case iinc of
  0:begin
    extension='.ff0'
  end
  45:begin
    extension='.ff4'
  end
  90:begin
    extension='.ff9'
  end
  135:begin
    extension='.ff3'
  end
  180:begin
    extension='.ff8'
  end
endcase
fullname=fname+extension

get_lun,iunit
openr,iunit,fullname,/f77_unformatted
jobname=string(replicate(32b,8))
comment=string(replicate(32b,80))
intrec=lonarr(10)
rrec1=fltarr(10)
rrec2=fltarr(10)

; Although the header information is read from the data file, it is not stored. The header information from the .bco file is used.
readu,iunit,jobname
print,'Reading the following data file: ',fullname
print,jobname
readu,iunit,comment
print,comment
readu,iunit,intrec
print,intrec
readu, iunit, rrecl
print, rrecl
readu, iunit, rrec2
print, rrec2
print, intrec(0), ' ka values from = ', string(rrecl(0))
print, ' in steps of ', string(rrecl(1))
print, intrec(1), ' different patterns'
print, intrec(2), ' theta incidence angles from ', rrecl(2)
print, ' in steps of ', rrecl(3)

; Next, read the far-field pressure data
; data = complexarr(nka, nharms, nobs, isym)
; assumes coefficients read as a block
vvector = complexarr(intrec(6))
data = complexarr(intrec(0), intrec(5), intrec(6), intrec(7))
for i = 0, (intrec(0) - 1) do begin
  for l = 0, (intrec(5) - 1) do begin
    for m = 0, (intrec(7) - 1) do begin
      readu, iunit, vvector
      data(i, l, *, m) = vvec * or
    endfor
  endfor
endfor
close, iunit
print, 'Finished reading the far-field pressure'

; Next, write the far-field data out to an unformatted
; PV-Wave file

for iinc = 0 to 180 do begin
  extension = '.fw0'
  extension = '.fw4'
  extension = '.fw9'
  extension = '.fw3'
  extension = '.fw8'
  extension = '.fw3'
  extension = '.fw8'
endcase
outname = fname + extension
get lun, junit
openw, junit, outname
al = assoc(junit, complexarr(intrec(0), intrec(5), intrec(3), $)
  intrec(7)))
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
This program takes the binary .bco file and produces two ASCII PV-Wave files:

- .hdr - contains all the parameters of the job
- .wco - has the axial and radial coordinates of the scatterer.

After this conversion, the .bco file is not needed by PV-Wave.

Select the data set

fname=string(replicate(32b,20))
print,'Input job name - no extension'
read,fname

Open the file with header and coordinate data (.bco file)
Then, define some variables and read the header

get -lun~iunit
openr,iunit,fname+' .bco' ,/f77_-unformatted
jobname=string(replicate(32b,8))
comment=string(replicate(32b,80))intrec=lonarr(10)
rrecl=fltarr(10)
rrec2=fltarr(10)
readu,iunit,jobname
print,'Reading the following data file: ',fname
print,jobname
readu,iunit,comment
print,comment
readu,iunit,intrec
print,intrec
readu,iunit,rrecl
print,rrecl
readu,iunit,rrec2
print,rrec2
print,intrec(0),’ ka values from = ’,string(rrecl(0))
print,’ in steps of ’,string(rrecl(1))
print,intrec(1),’ different patterns’
print,intrec(2),’ theta incidence angles from ’,rrecl(2)
print,’ in steps of ’,rrecl(3)

; Now write the header info to a formatted PV-Wave file

get_lun,junit
openw,junit,fname+'.hdr'
printf,junit,jobname
printf,junit,comment
printf,junit,format='(lOi5)',intrec
printf,junit,format='(5g10.4/5gl0.4)',rrecl

A-7
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
This program takes a binary surface data file (either surface pressure, .sp#, or normal velocity, .nv#) and produces an equivalent, binary PV-Wave file (either .pw# for surface pressure or .vw# for normal velocity). After this conversion, the .nv# or .sp# file is not needed by PV-Wave.

; Select the data type and set
fname=string(replicate(32b,20))
surtype=''
print,'Input job name - no extension'
read,fname
print,'Input p for pressure or v for velocity'
read,surtype
case surtype of
  'p':begin
    stype='.sp'
    otype='.pw'
    words='surface pressure'
  end
  'v':begin
    stype='.nv'
    otype='.vw'
    words='normal velocity'
  end
case iinc of
  0:begin
    extension=stype+'0'
    oextension=otype+'0'
  end
  45:begin
    extension=stype+'4'
    oextension=otype+'4'
  end
  90:begin
    extension=stype+'9'
    oextension=otype+'9'
  end
  135:begin
    extension=stype+'3'
    oextension=otype+'3'
  end
  180:begin
    extension=stype+'8'
    oextension=otype+'8'
endcase
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      open,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,fullname
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,isymn)
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
outname=fname+oextension

get_lun,junit
openw,junit,outname
al=assoc(junit,complexarr(intrec(0),intrec(5),intrec(3),$
intrec(7)))
al(0)=surf
free_lun,junit

print,'Finished writing the '+words
print,'Finished everything'
end
! pro rdhdr, filename=fname
; Last modification: 20 April 92

; This program reads header from a data set with the same
; job name

;-------------------------------------------------------

; common header, jobname, comment, $

nka, npatts, nthetainc, nareas, ncoords, nharms, nobs, isymm, $

kastart, kainc, thetaincstart, thetaincinc, phiincstart, $

phiincinc, thetaobsstart, thetaobsinc, $

rhof, cf, rhom, young, nu, eta

; First, some definitions

jobname = string(replicate(32b, 8))
comment = string(replicate(32b, 80))
intrec = lonarr(10)
rrecl = fltarr(10)
rrec2 = fltarr(10)

; Now, read the header file

get lun, junit
openr, junit, fname+'.hdr'
readf, junit, jobname ; 8 character name of run
readf, junit, comment ; 80 characters (20 words) for comments
; description, etc.
readf, junit, format='(10i5)', intrec
readf, junit, format='(5g10.4/5g10.4)', rrecl
readf, junit, format='(5g10.4/5g10.4)', rrec2
free lun, junit

; Give the header info some more meaningful names
; First the integer record

nka = fix(intrec(0)) ; number of ka values
npatts = fix(intrec(1)) ; number of ff patterns
nthetainc = fix(intrec(2)) ; number of theta incident angles
nareas = fix(intrec(3)) ; number of areas (rings)
ncoords = fix(intrec(4)) ; number of coordinates
nharms = fix(intrec(5)) ; number of azimuthal harmonics
nobs = fix(intrec(6)) ; number of observation angles in ff patterns
isymm = fix(intrec(7)) ; symmetry flag, 1=Yes, 2=No

; Now, the first real record

kastart = rrecl(0)
kainc = rrecl(1)
thetaincstart = rrecl(2)
thetaincinc = rrecl(3)
phiincstart = rrecl(4)
phiincinc = rrecl(5)
thetaobsstart=rrec1(6)
thetaobsinc=rrec1(7)

; the rest of rrec1 is currently not used

Finally, the second real record

rho=rrec2(0) ; fluid density
cf=rrec2(1) ; sound speed in fluid
rhom=rrec2(2) ; density of shell
young=rrec2(3) ; Young’s modulus of shell
nu=rrec2(4) ; Poisson’s ratio of shell
eta=rrec2(5) ; damping factor

; the rest of rrec2 is currently not used

Print the basic information about this data set

print,’Jobname is ‘,jobname
print,’Comments: ‘,comment
print,’There are ‘,nka,’ ka values from ‘,string(kastart)$
print,’in steps of ‘,string(kainc)
print,’The spectral variable is ka’
print,’There are ‘,npatts,’ ff patterns’
print,’ ‘,nareas,’ areas or rings’
print,’ ‘,ncoords,’ coordinate values’
print,’ ‘,nthetainc,’ theta incident angles’
print,’There are ‘,nthetainc,’ theta incidence angles from ‘$
print,’in steps of ‘,thetaincinc
print,’ ‘,nobs,$
print,’ theta observation angles npff pattern from ‘,$
print,’in steps of ‘,thetaobsinc
print,’The fluid density is ‘,rhof
print,’The sound speed in the fluid is ‘,cf
print,’The density of shell is ‘,rhom
print,’The Young’s modulus of shell is ‘,young
print,’Poisson’s ratio for the shell is ‘,nu
print,’The damping factor is ‘,eta

end
This program reads in PV-Wave binary far-field file (.fw#) placing all far-field data in memory for use as data.

common header, jobname, comment,$
nka, npatts, nthetainc, nareas, ncoords, nharms, nobs, isymm, kastart,$
kinc, thetaincstart, thetaincinc, phiincstart, phiincinc,$
thetaincstart, thetaobsinc, rhof, cf, rhom, young, nu, eta

jobname=''
pathname=''
print,'input jobname (no extension)'
read, jobname
print,'input pathname including trailing /'
read, pathname
fullname=pathname+jobname

; read vital parameters from .hdr file
rdhdr, filename=fullname
print,'Input incidence angle'
read, iinc
case iinc of
  0: begin
    extension='.fw0'
  end
  45: begin
    extension='.fw4'
  end
  90: begin
    extension='.fw9'
  end
  135: begin
    extension='.fw3'
  end
  180: begin
    extension='.fw8'
  end
  else: print,'Invalid angle'
endcase
fullname=fullname+extension

; Open the file, then read in far-field data
get_lun, iunit
openr, iunit, fullname
aa=assoc(iunit, complexarr(nka, nharms, nobs, isymm))
data=aa(0)
This section of the code recombines the harmonic components for each ka. A default increment in phi of 30 degrees is used. Recombined data are held in the variable npff.

\[ \text{deltaphi} = 30 \]
\[ \text{nnumphi} = 1 + \text{fix}(360./\text{deltaphi}) \]
\[ \text{angles} = \text{deltaphi} \cdot (\pi/180.) \cdot \text{findgen(nnumphi)} \]
\[ \text{ivect} = \text{indgen(nharms)} \]
\[ \text{cosangle} = \text{transpose(cos(ivect#angles))} \]
\[ \text{sinangle} = \text{transpose(sin(ivect#angles))} \]
\[ \text{npff} = \text{complexarr(nnumphi,nobs,nka)} \]

for kabin = 0, nka-1 do begin
  \[ \text{ctemp} = \text{complexarr(nnumphi,nobs)} \]
  if isymm eq 1 then ctemp = \text{cosangle#reform(data(kabin,*,*,0))}
  if isymm eq 2 then ctemp = \text{cosangle#reform(data(kabin,*,*,0)) + sinangle#reform(data(kabin,*,*,1))} 
  \[ \text{npff}(*,*,kabin) = \text{ctemp} \]
endfor

This section of code produces three color images of the npff data; target strength in db, magnitude, and phase. The phi observation angle is fixed at 0 degrees. The range for target strength is \([-30,30]\]. The range for magnitude is \([0, \text{max(mag)}]\). The range for phase is \([-180,180]\]. In each case, the entire theta range has been reconstructed \([0,360]\). The x-axis corresponds to the theta observation angle; the y-axis to the ka range.

\[ \text{xsz} = 2 \cdot \text{nobs} - 1 \]
\[ \text{ysz} = \text{nka} \]
\[ \text{halfway} = (\text{nnumphi} - 1)/2 \]

Recombine the full theta observation range

\[ \text{var} = \text{fltarr(xsz,ysz)} \]
\[ \text{var}(0:\text{nobs}-1,*) = \text{abs(npff}(0,*,*)) \]
\[ \text{var}(\text{nobs}:\text{xsz}-1,*) = \text{abs(\text{rever}\cdot\text{reform(npff(halfway,1:nobs-1,*)),1}))} \]

Calculate the target strength in dB

\[ \text{ts} = 20 \cdot \text{alog10(var)} \]
\[ \text{tsimg} = \text{bytscl(ts,min=-30,max=30)} \]

Calculate the magnitude and phase

\[ \text{magimg} = \text{bytscl(var,min=0)} \]
\[ \text{phase} = \text{fltarr(xsz,ysz)} \]
\[ \text{phase}(0:\text{nobs}-1,*) = (180./\pi) \cdot \text{atan(\text{imaginary(npff}(0,*,*))},1\)
109  phase(nobs:xsz-1,*)=reverse(reform((180./1pi)*$
110  atan(imaginary(npff(halfway,1:nobs-1,*)),float(npff(halfway,$
111  l:nobs-1,*)))),1)
112  phaseimg=bytscl(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,magimg,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 ;------------------------------------------------------------
9 common header,jobname,comment,$
10 nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,lsym,$
11 kastart,kainc,thetaincstart,thetaincinc,phiincstart,$
12 phiincinc,thetaobsstart,thetaobsinc,rhof,cf,rhom,young,$
13 nu,eta
14
15 fname='
16 pathname='
17 print,'input jobname (no extension)'
18 read,fname
19 print,'input pathname including trailing /'
20 read,pathname
21 fname=pathname+fname
22
23 ; Read essential parameters from .hdr
24 25 rdhdr,filename=fname
26 print,'input incidence angle'
27 read,iinc
28 surtype=''
29 print,'Do you want surface pressure(p) or velocity(v)'
30 read,surtype
31 if surtype eq 'v' then begin
32     title='NORMAL VELOCITY MAGNITUDE'
33     stype='.vw'
34   endif
35 if surtype eq 'p' then begin
36     title='SURFACE PRESSURE MAGNITUDE'
37     stype='.pw'
38   endif
39 case iinc of
40   0:begin
41     extension=stype+'0'
42   end
43 45:begin
44     extension=stype+'4'
45   end
46 90:begin
47     extension=stype+'9'
48   end
49 135:begin
50     extension=stype+'3'
51   end
52 180:begin
53     extension=stype+'8'
54   end
55 A-17
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(nnumphi,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,byt scl(congrid(abs(reform(sur(binno,*,*))),20*numphi,288),$
107     /interp),min=0,max=maxval),90,70
108     y=[0,1.]
\texttt{\textbackslash{}109 \texttt{x2=[0,360]}}

\texttt{\textbackslash{}110 plot,x2,y,xstyle=1,ystyle=1,xticks=6,yticks=4,/nodata,}$

\texttt{\textbackslash{}111 /noerase,/dev,pos=[90,70,20*numphi+90,358]}

\texttt{\textbackslash{}112 xyouts,/device,210,500,title,size=2}

\texttt{\textbackslash{}113 xyouts,/device,300,480,'minval='+string(min(abs(sur)))}

\texttt{\textbackslash{}114 xyouts,/device,300,465,'maxval='+string(max(abs(sur)))}

\texttt{\textbackslash{}115 tv,bytescl(\text{transpose}(\text{congrid}(\text{reform}(\text{abs(sur(*,phicut,*))))}}$

\texttt{\textbackslash{}116 nka,288,/interp)),min=0,max=maxval),425,70}$

\texttt{\textbackslash{}117 y=[0,1.]}

\texttt{\textbackslash{}118 x3=[min(x),max(x)]}

\texttt{\textbackslash{}119 plot,y,x3,xstyle=1,ystyle=1,xticks=4,yticks=11,/dev,}$

\texttt{\textbackslash{}120 /nodata,/noerase,pos=[425,70,713,nka+70]}

\texttt{\textbackslash{}121 xyouts,/device,240,570,'incidence angle = '+string(iinc)}

\texttt{\textbackslash{}122 xyouts,/device,240,620,fullname,size=2}

\texttt{\textbackslash{}123 xyouts,/device,120,425,'For ka = '+strtrim(string(x(binno)),2)}

\texttt{\textbackslash{}124 xyouts,/device,450,425,'For phibin = '+strtrim(string(phicut),2)}

\texttt{\textbackslash{}125 xyouts,/device,150,25,'\text{Phi}'}

\texttt{\textbackslash{}126 xyouts,/device,45,125,'Normalized axial distance',orientation=90}

\texttt{\textbackslash{}127 xyouts,/device,445,25,'Normalized axial distance'}

\texttt{\textbackslash{}128 xyouts,/device,380,200,'ka',orientation=90}

\texttt{\textbackslash{}129}

\texttt{\textbackslash{}130 end}

\texttt{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
pro closeit
;Last modification: 21 April 92

;------------------------------------------------------------------------
;This program closes the hardcopy file and sets the display
;back to 'x' window, replace 'x' with default driver on system
;------------------------------------------------------------------------

device,/close_file ; close the file, so it can be sent to
; the printer
set_plot,'x' ; redirect the output to the 'x' window driver

end
This program reads in PV-Wave binary far-field file (.fw#), generates a 3-d display of far-field pressure for user-defined increments in ka, and displays the result as a movie.

---

common header, jobname, comment, $

nka, npatts, ntheta-inc, nareas, ncoords, nharms, nob, isymm,$

kastart, kainc, thetaincstart, thetaincinc, phiincstart, $

phiincinc, thetaobsstart, thetaobsinc, rhof, cf, rhom, young, nu, $

eta

jobname='.pathname='

print,'input jobname (no extension)' read, jobname

print,'input pathname including trailing /' read, pathname

fullname=pathname+jobname

rdhdr, filename=fullname

print,'Input incidence angle' read, iinc

case iinc of

0:begin

extension='.fw0'

end

45:begin

extension='.fw4'

end

90:begin

extension='.fw9'

end

135:begin

extension='.fw3'

end

180:begin

extension='.fw8'

end

endcase

fullname=fullname+extension

get_lun,iunit

openr,iunit,fullname

aa=assoc(iunit, complexarr(nka, nharms, nob, isymm))

data=aa(0)

free_lun,iunit

; This section of the code recombines harmonics for a default phi increment of 30 degrees. Note: a finer
; increment may be desired to eliminate plotting
; artifacts.
; define some variables

deltaphi=30
numphi=1+fix(360./deltaphi)
angles=deltaphi*(pi/180.)*findgen(numphi)
ivect=findgen(nharms)
cosangle=transpose(cos(ivect#angles))
sinangle=transpose(sin(ivect#angles))
npff=complexarr(numphi,nobs,nka)

; form npff for each phi angle
for kabin=0,nka-1 do begin
  ctemp=complexarr(numphi,nobs)
  if isym eq 1 then ctemp=cosangle#$
  reform(data(kabin,*,*,0))
  if isym eq 2 then ctemp=cosangle#$
  reform(data(kabin,*,*,0)) +sinangle#$
  reform(data(kabin,*,*,1))
  npff(*,*,kabin)=ctemp
endfor

; generate frames (default of 5)
nframes=5
step=fix(nka/nframes)
xp=fltarr(nobs,numphi)
yp=fltarr(nobs,numphi)
zp=fltarr(nobs,numphi)
indtheta=indgen(nobs)
indphi=indgen(numphi)
unity=replicate(1,numphi)

; the selection of the radius, a, is arbitrary, but
; it does affect the size of the display
a=5.
tang=pi*indtheta/(nobs-1) ; generate a vector of theta's
pang=2*pi*indphi/(numphi-1) ; same for phi's
zp=a*cos(tang)#unity
factor=a*sin(tang)
xp=factor#cos(pang)
yp=factor#sin(pang)

sh=bytarr(nobs,numphi) ; this holds the (bytscaled) target
strength=bytarr(370,365,nframes+1)

window,0,xpos=350,ypos=300,xsize=370,ysize=365
loadct,5
kaval=findgen(nka)*kainc+kastart
for j=0,nframes do begin ; frequency loop
  k=j*step
sh=bytscl(20*aalog10(transpose(abs(npff(*,*)))).$,min=-40,max=40)

; the next routine tends to choke on a number of degenerate
; polygons, which may cause some artifacts, but the
; routine does not bomb and the results are useful as is

shade_surf_irr,sh*zp,sh*xp,sh*yp,shades=sh,$

xrang=[-1000,1000],yrang=[-1000,1000],zrang=[-1000,1000]

xyouts,0.4,0.8,'ka = '+strtrim(string(kaval(k))).$,size=1,/normal

wait,0.0001

newimg(*,*,j)=tvrd(0,0,370,365)

endfor

; redisplay the frames as a movie

movie,newimg,order=0

; make a hard copy of any frame of interest

hardcopy,file='clb93d#18.eps',xsize=3.70,ysize=3.65

tv,newimg(*,*,18)
closeit

end
This program makes an animated multi-color plot showing how surface pressure or normal velocity vary with axial node number and frequency. It assumes that grabsur.pro has been run and sur(nka,numphi,ncoords) is in memory.

```fortran
common header, jobname, comment,$
nka, npatts, nthetainc, nareas, ncoords, nharms, nobs, isymm,$
kastart, kainc, thetaincstart, thetaincinc, phiincstart, phiincinc,$
thetaobsstart, thetaobsinc, rhof, cf, rhom, young, nu, eta
window, /free
plot, sur(0,0,*), ystyle=1, xstyle=1, yrange=[0,maxval]/nodata, $
xtile='Axial bin number', ytitle='Magnitude'
for i=0, nka-1 do begin
  oplot, abs(sur(i,0,*)), color=i
  wait,.02
endfor
end
```
This program generates a set of overlaid spectral plots from supposedly equivalent solutions generated by different codes, allows the user to select a subset of the solutions, and produces a mean and envelope that bound the variations within the data sets selected. Note: This program assumes the data to be compared has been read in prior to the execution of envelope.pro. The data for the Benchmark exercise was stored in variables named amag0 through wmag0, all of which were of the size (nobs,nkas). The data sets were reduced by the selecting the phi observation angle prior to execution. The example given was produced with modified versions of this routine. 

First set some default plotting styles.

I first set some default plotting styles

!x.style=1
!y.style=1
!x.ticks=6
!y.ticks=6

id=''

print,'Enter index of theta angle desired'
if bmname ne '4a' and bmname ne '4b' then begin
  print,'0=0obs, 18=45obs, 36=90obs, 54=135obs, 72=180obs'
fact=2.5
endif else begin
  print,'0=0obs, 15=45obs, 30=90obs, 45=135obs, 60=180obs'
fact=3.0
endelse

read,tangle

print,'Enter number of data sets available'
read,nsets
print,nsets
print,'Enter number of ka values'
read,nkas

Set up x-axis scale
x=findgen(nkas)*.01+.2
if (bmname eq '4a' or bmname eq '4b') then x=findgen(nkas)*.0094+.397

;Determine max(x) as legend positioning is tied to this value
maxx=max(x)
if(bmname eq '1a') then nk=330 ; special case

;The spectral plots are saved in the array results.
results=fltarr(nsets,nkas)
results(0,*)=amag0(tangle,0:nkas-1)
55 results(l,*)=cmagO(tangle,0:nkas-1)
56 results(2,*)=smagO(tangle,0:nkas-1)
57 results(3,*)=mmagO(tangle,0:nkas-1)
58 if nsets ge 5 then results(4,*)=nmagO(tangle,0:nkas-1)
59 if nsets ge 6 then results(5,*)=fmagO(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)=1l-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)=16-fist'
86 endif
87 ;These set up the same for the envelope plot.
88 ltyp=0
89 clr=[164,200]
90
type='average','min / max'
91 ltyp=0
92 colors=[164,200]
93 loadct,12
94 lx.title='ka'
95 ly.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. BMIB 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=' "

A-27
results=resultsave

;User picks how many and which data sets to include in
;the envelope. The user can choose to reprocess the envelope.

;'how many data sets do you want to include'
read,numin

if numin eq nsets then setid='ALL DATA SETS'
if numin ne nsets then begin

    sets=intarr(numin)
    print,'which data sets do you want to include',$
    '1 - nsets possible'
    read,sets
    print,'you have asked to include sets',sets
    for i=0,numin-1 do setid=setid+string(sets(i))
    for i=0,numin-1 do sets(i)=sets(i)-1
    results=results(sets, *)
    endif

average=avg(results, 0)

window,\free,xsiz=625,ysiz=600,xpos=625,ypos=250
mini=min(results)
maxdum=fltarr(nkas)
mindum=fltarr(nkas)

for i=0,nkas-1 do begin
    maxdum(i)=max(results(*,i))
    mindum(i)=min(results(*,i))
endfor
maxd=max(1.2*max(maxdum)
plot,x,(average),/nodata,yrange=[0,maxdata]
oplots,x,(average),color=164
oplots,x,(maxdum)
oplots,x,(mindum)
legend,label,clr,ltyp,psym,.7*maxx,.8*maxdata,.3
xyouts,.6,.85,/normal,strcompress(setid),color=200,siz=1.2
xyouts,.6,.90,/normal,id,color=200,siz=1.2
maxdiff=max(maxdum-mindum)
print,'Maximum difference = ',maxdiff,' at ',

x(where(maxdum-mindum eq maxdiff))
;oplots,x,ntruth(0,*),color=100
print,'Do you want to reprocess, y or n'
read,ans

endwhile
This procedure plots the target strength image for two codes and plots the difference between the images. The difference is taken between dB values which have first been subjected to a user-specified threshold. The user is allowed interactive control over the dynamic range of the images and the difference image.

Note: The procedure assumes npff_m.pro has been compiled.

common header, jobname, comment, 
$nka,npatts,nthetainc,nareas,ncoords,nharms,nobs,isymm,$
$kastart,kainc,thetaincstart,thetaincinc,phiincstart,phiincinc,$
$thetaobsstart,thetaobsinc,rhof,cf,rhom,young,nu,eta$
$common sharel, npff, numphi, diff$

; Get the first set of data

print,'Prompting for the threshold value in db'
read,threshdb
print,'Prompting for dynamic range desired for difference'
read,dmin,dmax
print,'Prompting for the first far-field data file.'
npff_m
npfl=abs(npff)
nkfl(where(abs(npfl) le thresh))=thresh
nkal=nka

print,'Prompting for the second far-field data file.'
npff_m
npff2=abs(npff)
npff2(where(abs(npff2) le thresh))=thresh

xs=2*nobs-1
ys=nka

ysl=nkal

;Calculate the target strengths

ansd='n'
anst='n'
ans='y'
repeat begin
print,'Input the index of the phi angle (0 to 6).'
read,phiinc
halfway=(numphi-1)/2 + phiinc
varl=fltarr(xs,ysl)
varl(0:nobs-1,*)=(npfl(0,phiinc,*))
\begin{verbatim}
55  var1(nobs:xsz-1,*)=(reverse(reform(npff1(halfway,$
56  1:nobs-1,*)),1))
57  ts1=20*alog10(var1)
58  var2=setarr(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,rebin(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,rebin(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,rebin(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'
\end{verbatim}
read, dmin, dmax
endif
endrep until ansd eq 'n'

print,'Do you want another threshold value? (y/n)'
read, anst
if (anst eq 'y') then begin
print,'Input new threshold value'
read, thresh_db
endif
endrep until anst eq 'n'

print,'Do you want another phi angle? (y/n)'
read, ans
endrep until ans eq 'n'
end
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, nob, isym, $
12 kastart, kaic, 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, isym))
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)
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 isyym 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))-complex(0,1)*reform(vf(h,*,l))
79 endif
80 if h lt 0 then begin
81 vh(h+nharms-1,*)=0.5*reform(vf(-h,*,0))
82 endif
83 endfor
84 endif else begin ; to allow cosine only representation
85 vf=reform(vel(f,*,nendnodes:nendnodes+npts-1))
86 vh=complexarr(2*nharms-1,npts)
87 for h=-(nharms-1),nharms-1 do begin
88 if h eq 0 then begin
89 vh(h+nharms-1,*)=reform(vf(h,*))
90 endif
91 if h gt 0 then begin
92 vh(h+nharms-1,*)=0.5*reform(vf(h,*))
93 endif
94 if h lt 0 then begin
95 vh(h+nharms-1,*)=0.5*reform(vf(-h,*))
96 endif
97 endfor
98 endelse
99
100 npts=47 ; ### to force npts to be odd, not general!
101 vh=vh(*,0:46) ; same
102
103 kmat=complexarr(2*nharms-1,npts)
104 sym=complexarr(2*nharms-1,npts)
105 middle=(npts-1)/2
for i=0,2*(nharms-1) do begin
  vect0=reform(vh(i,*))
  kmat(i,*)=fft(vect0,1)
  sym(i,middle)=kmat(i,0)
  unitv=complexarr(middle+1)
  unitv(0)=complex(1,0)
  unitv(1)=complex(cos(lpi*(npts-1)/npts),$
  sin(lpi*(npts-1)/npts))
  for j=1,middle do begin
    unitv(j)=unitv(1)*unitv(j-1)
    sym(i,middle+j)=unitv(j)*kmat(i,npts-j)
    sym(i,middle-j)=conj(unitv(j))*kmat(i,j)
  endfor
  kmat=sym
  new=transpose(kmat)
  sign=float(new)/abs(float(new))
  newer=sign*abs(float(new))
  tv, rebin(bytscl(newer, min=-25, max=25, top=254), mfact*npts, $
  nfact*(2*nharms-1))
  xyouts, 0.05, 0.05,'freq='+strtrim(string((f+1)/2.),2),$
  /normal,color=254
  print,'Frequency bin = ',f
  wait,0.5
  char=get_kbrd(0)
  if char ne '' then begin
    print,'Hit any key to continue'
    hak
  endif
end for
end
This program takes model data in the frequency domain and performs the windowing prescribed by NRL, then presents a comparison of the model results with the experimental data in the frequency domain.

Select the data set Change pathname data file names as required

pid=''
print,'input pid, eg 4a 4b'
read,pid
pathname='/data/schenck/dset2/
print,'input pathname'
read,pathname
jobname='pr.nt,'Input jobname (no extension)
read,jobname
print,'Input incidence angle in degrees'
read,iinc
case pid of
  '4a':begin
    refname='bm4aref.daw' ; reference time series
    case iinc of
      0:begin
        extension='.fw0' ; computational data
cdat='cleana0.dat' ; cleaning window data
        shifty=-0 ; to account for orientation of target
        in NRL tank
        fname='bm4a0.daw' ; experimental echo time series
        end

      45:begin
        extension='.fw4'
cdat='cleana4.dat'
        shifty=-15
        fname='bm4a45.daw'
        end

      90:begin
        extension='.fw9'
cdat='cleana9.dat'
        shifty=-30
        fname='bm4a90.dat'
        end
A-35
endcase
end
'4b':begin
  refname='bm4bref.daw'
case iinc of
  0:begin
    extension='fw0'
    cdat='cleanb0.dat'
    shifty=-0
    fname='bm4b0.daw'
    end
  45:begin
    extension='fw4'
    cdat='cleanb4.dat'
    shifty=-15
    fname='bm4b45.daw'
    end
  90:begin
    extension='fw9'
    cdat='cleanb9.dat'
    shifty=-30
    fname='bm4b90.daw'
    end
  endcase
  end
else:print,'Incorrect jobname'
endcase

fullname=pathname+jobname

; Read the header and open the file
rdhdr, filename=fullname
fullname=fullname+extension
get_lun,iunit
openr,iunit,fullname
aa=assoc(iunit,complexarr(nka,nharms,nobs,iasymm))
data=aa(0)
free_lun,iunit
print,'Finished reading model data set'

; this next part could be made simpler, since we only want phicut=0

deltaphi=180
numphi=1+fix(360./deltaphi)
angles=deltaphi*(1pi/180.)*findgen(numphi)
ivect=indgen(nharms)
cosangle=transpose(cos(ivect#angles))
sinangle=transpose(sin(ivect#angles))
npff=complexarr(numphi,nobs,nka)
for kabinm0,nka-1 do begin
  ctemp=complexarr(numphi,nobs)
109 if isyim eq 1 then ctemp=cosangle#reform(data(kabin,*,*,0))
110 if isyim 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=bytscl(20*alog10(abs(var)),min=-30,max=30,$ 
130 top=1d_n_colors-1)
131 tv,rebiny(bytc,2*121,331),600,35
132 xynouts,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 ; Read in the reference (incident) time series
139 refname=' /data/schenck/expt4/ '+refname
140 get_lun,iunit
141 openr,iunit,refname, /f77_unformatted
142 bb-assoc(iunit,fltarr(4096))
143 ref=bb(0) ;reference (incident) time series
144 free_lun,iunit
145 tape_info=ref(4080:4095)
146 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=ftt(rpad,-1)
160 magpinc-avg(abs(cpad(add:add+nka-1)))
161
162 ;;; change pathnames if required
fullname = '/data/schenck/expt4' + fname  
get_lun, iunit  
openr, iunit, fullname, /f77_unformatted  
aa = assoc(iunit, fliarr(121, 4096))  
echo = aa(0)  
free_lun, iunit  
print,'Finished reading the experimental data'

; The correction factor accounts for the source,  
; target, receiver geometry
geocorrect = 2 * 2 * sqrt(2.18) / (2.239 * 2.18 * 0.0254 * sqrt(1.96))
echo = geocorrect * echo

window, 2, xsize = 400, ysize = 200, xpos = 150, ypos = 470,$
title = '2. Experimental echo'
loadct, 5
window, 3, xsize = 400, ysize = 200, xpos = 150, ypos = 10,$
title = '3. Computed time series'
window, 4, xsize = 400, ysize = 200, xpos = 150, ypos = 240,$
title = '4. Windowed time series'
ffsurf = complexarr(121, 331)
modspec = complexarr(121, 331)

; Now develop cosine squared window for each observation angle  
; this information comes from Brian Houston's memo re from  
; 3600 to 3700 data

cdat = '-/prob4' + cdat
get_lun, kunit
openr, kunit, cdat
cleanarr = intarr(4, 121)
readf, kunit, cleanarr
free_lun, kunit
print, cleanarr

; Now we'll start the large loop over observation angle
for j=0, 120 do begin

k = -(j + shifty)
if k lt 0 then k = k + 121

; Next, let's process the experimental echo
print, echo(k, 4094)  ; print angles of observation
echo(k, 4080:4095) = 0.  ; zero the last sixteen bins
longecho = replicate(0., tlength)
longecho(0:4095) = reform(ech(k, *))
backspec = fft(longecho, -1)  ; convert to frequency domain
ffsurf(j, *) = backspec(add: add + nka - 1)
wset,2
plot,ech(k,0:2500),yrange=[-10.,10.],ystyle=1

; Generate and overplot the cleaning window
frontstart=cleanarr(0,k)
frontend=cleanarr(1,k)
backstart=cleanarr(2,k)
backend=cleanarr(3,k)
flength=frontend-frontstart+1
blength=backend-backstart+1
piover2=pi/2.
frontw=(sin(piover2*findgen(flength)/float(flength-1)))^2
backw=(cos(piover2*findgen(blength)/float(blength-1)))^2
cleanw=replicate(0.0,4096)
cleanw(frontstart:frontend)=frontw
cleanw(frontend+1:backstart-1)=1.0
cleanw(backstart:backend)=backw
longclean=replicate(0.,tlength)
longclean(0:4095)=cleanw
oplot,5.*longclean(0:2500),color=120
wait,0.02

; Next, we will process the computed results
bsff=reform(var(j,*)*cpad(add:add+nka-1)); positive freqs
part=replicate(0.0,tlength)
padded=complex(part,part); a complex array of zeroes
padded(0:add:add+nka-1)=bsff; fill in the negative frequencies
rev=conj(reverse(bsff)); ensures a real time series

; fill in the negative freqs
padded(tlength-nka+1-add:tlength-add)=rev(0:nka-1);
tseries=fft(padded,1); take the FFT
modt0=float(tseries)

wset,3
plot,modt0,yrange=[-10.0,10.0],ystyle=1
oplot,imaginary(tseries),color=120; this should be zero
wait,0.02

; Next, apply the cleaning window to the computed results
newc=longclean*modt0; apply the window in time domain
newspec=fft(newc,-1); compute the windowed model spectrum
modspect(j,*)=newspec(add:add+nka-1)

wset,4
plot,newc(0:2500),yrange=[-10.0,10.0],ystyle=1
oplot,5.*longclean(0:2500),color=120
wait,0.02
print,'j=',strtrim(string(j),2),'
k=',strtrim(string(k),2)
endfor

; nrlimg=reverse(shift(20*alog10(abs(ffsurf)/magpinc),
; shifty,0)); the exptrl data

nrlimg=20*alog10(abs(ffsurf)/magpinc) ; the exptrl data

; the shift and reverse are needed to make my NRL's
; angles the same as for Benchmark problems

bytx=bytscl(nrlimg,min=-30,max=30,top=td.n_colors-1)

wset,5

tv, rebin(bytx,2*121,331),300,35
xyouts,0.4,0.95,'Experimental data',/normal

bytw=bytscl(20*alog10(abs(modspec)/magpinc),min=-30,$
max=30,top=td.n_colors-1)

wset,5
loadct,5

tv, rebin(bytw,2*121,331),20,35
xyouts,0.08,0.95,'Windowed model',/normal

phasem=atan(imaginary(var),float(var))
phasex=atan(imaginary(ffsurf),float(ffsurf))
phases=atan(imaginary(modspec),float(modspec))

window, /free

tvsc1,phasem,50,50

tvsc1,phasex,200,50

tvsc1,phases,350,50

goto,jump

goto, jump

hardcopy, filename=strtrim(jobname,2)+$
strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.31

tv,bytc

closeit

hardcopy, filename='win'+strmid(jobname,1,2)+$
strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.3

tv, rebin(bytw,121,330)

closeit

hardcopy, filename='ex'+strmid(jobname,1,2)+$
strtrim(string(fix(iinc)),2)+'.eps',xsize=1.21,ysize=3.3

tv, rebin(bytex,121,330)

closeit

jump:

end

A-40
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.
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0012</td>
<td>Patent Counsel</td>
<td>1</td>
</tr>
<tr>
<td>144</td>
<td>V. Ware</td>
<td>1</td>
</tr>
<tr>
<td>541</td>
<td>B. Sotirin</td>
<td>1</td>
</tr>
<tr>
<td>701</td>
<td>H. A. Schenck</td>
<td>20</td>
</tr>
<tr>
<td>705</td>
<td>M. Morrison</td>
<td>1</td>
</tr>
<tr>
<td>7103</td>
<td>R. Smith</td>
<td>1</td>
</tr>
<tr>
<td>712</td>
<td>G. W. Benthien</td>
<td>1</td>
</tr>
<tr>
<td>713</td>
<td>J. Price</td>
<td>1</td>
</tr>
<tr>
<td>731</td>
<td>G. Thomson</td>
<td>1</td>
</tr>
<tr>
<td>733</td>
<td>E. P. McDaid</td>
<td>1</td>
</tr>
<tr>
<td>734</td>
<td>J. C. Lockwood</td>
<td>1</td>
</tr>
<tr>
<td>743</td>
<td>S. Sullivan</td>
<td>1</td>
</tr>
<tr>
<td>9103</td>
<td>J. M. Baird</td>
<td>1</td>
</tr>
<tr>
<td>952B</td>
<td>J. Puleo</td>
<td>1</td>
</tr>
<tr>
<td>961</td>
<td>Archive/Stock</td>
<td>6</td>
</tr>
<tr>
<td>964B</td>
<td>Library</td>
<td>2</td>
</tr>
</tbody>
</table>

Initiative Distribution

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 & Development Information Center (NARDIC)  
Washington, DC 20360-5000

Office of the Chief of Naval Operations  
Washington, DC 20350-2000  
(2)

Space & Naval Warfare Systems Command  
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)
<table>
<thead>
<tr>
<th>Company</th>
<th>Address</th>
<th>City, State, Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT&amp;T</td>
<td>Whippany, NJ 07981-0903</td>
<td>(2)</td>
</tr>
<tr>
<td>AT&amp;T Bell Laboratories</td>
<td>Arlington, VA 22202</td>
<td></td>
</tr>
<tr>
<td>BBN Systems &amp; Technologies</td>
<td>New London, CT 06320-6147</td>
<td>(2)</td>
</tr>
<tr>
<td>BBN Systems &amp; Technologies</td>
<td>Cambridge, MA 02138</td>
<td></td>
</tr>
<tr>
<td>Cambridge Acoustical Associates</td>
<td>Cambridge, MA 02139</td>
<td></td>
</tr>
<tr>
<td>Computer Sciences Corporation</td>
<td>San Diego, CA 92110-5164</td>
<td>(2)</td>
</tr>
<tr>
<td>General Dynamics</td>
<td>Electric Boat Division</td>
<td></td>
</tr>
<tr>
<td>International Business Machines</td>
<td>Corporation</td>
<td></td>
</tr>
<tr>
<td>Groton, CT 06340</td>
<td>Kingston, NY 12401</td>
<td></td>
</tr>
<tr>
<td>Newport News Shipbuilding</td>
<td>Newport News, VA 23607</td>
<td></td>
</tr>
<tr>
<td>Newport News Shipbuilding</td>
<td>Arlington, VA 22202</td>
<td></td>
</tr>
<tr>
<td>Precision Visuals, Inc.</td>
<td>Boulder, CO 80301</td>
<td>(2)</td>
</tr>
<tr>
<td>SRI, Inc.</td>
<td>Arlington, VA 22209</td>
<td></td>
</tr>
<tr>
<td>Weidlinger Associates</td>
<td>New York, NY 10001</td>
<td></td>
</tr>
</tbody>
</table>