User’s Guide

Acoustic Lens Simulation Software Package (ALSSP)

by

Kevin Fink

Applied Physics Lab, University of Washington

This software runs on PV-WAVE by Precision Visuals

Last Software Revision: 8 February, 1995
Last Documentation Revision: 9 February, 1995

Table of Contents

OVERVIEW

ALSSP is a software set designed to simulate acoustic lenses. It runs in the PV-WAVE environment. It can be run interactively from the PV-WAVE command line or as batch files, or a combination of both. The software implements both geometric and wave acoustics. Geometric acoustics (ray-tracing) is used within the lens system, and wave acoustics is used after the final lens interface. For a more complete description of the algorithms used, see Fink, Kevin. Computer Simulation of Pressure Fields Generated by Acoustic Lens Beamformers, Masters Thesis, University of Washington, 1994.

Capabilities

- Ambient system parameters include system temperature, salinity, depth, attenuation rate, sound speed, and frequency of operation.
- Lens systems may have any number of components.
- System components may be placed arbitrarily in space.
- Lens interfaces may be planar, spherical, or aspherical. Aspherical shapes include hyperboloids, paraboloids, and ellipsoids.
- The density, sound attenuation rate, and sound speed of each lens material can be specified.
- Sound sources include point sources, planar sources, and curved linear sources.
- Receive element types include point, rectangular, curved rectangular, and circular.
- Any number of rays can be traced through the system, either in two dimensions or three.
- Lens systems, including traced rays, may be viewed and printed.
- Calculation types include pressure fields, focal point determination, and beam patterns.
- Beam patterns may be lines, arcs, or piecewise-circular trajectories.

Limitations

- Reflection is ignored, so reflected beams are not propagated.
- Lens interfaces must be listed in the order in which the propagating sound will pass through them.

Notes on PV-WAVE:

PV-WAVE is not case sensitive, so commands can be entered in either upper or lower case. Note, however, that UNIX is case-sensitive, so UNIX commands and filenames must be entered in the correct case. If a PV-WAVE command line is more than one line long, then it can be broken and continued on the next line by ending it with the dollar sign character ('$'). Lines longer than 80 characters will often be split this way in order to preserve readability. For example,

```
CALC_LENS,'test'
```

is the same as

```
CALC_LENS $
 ,'test'
```

Notes on this manual:

In this manual, program and function names will be shown in all capital letters. Parameter names passed to commands will be shown in *italics*, but actual parameters will be in normal text. Keyword parameters will also be in */italics*, but will begin with a forward slash character ('/') to differentiate them from other parameter names. For program and function specifications, [optional parameters] will be shown within square brackets. For example, part of the specification for CALC_LENS is:

CALC_LENS,*filename* [,*/num_rays*]

meaning that *filename* is a required parameter, but */num_rays* is an optional keyword parameter. For example:

```
filename='testfile'
CALC_LENS,filename,num_rays=21
```

would call the program CALC_LENS with the string 'testfile', and would set the keyword parameter */num_rays* to 21.

Installation Instructions

1. If ALSSP is to be installed in a system directory, su to root.
2. Extract Makefile and alssp.tar.Z from a backup tape (tar xf /dev/rst0).
3. If ALSSP is to be installed in /usr/local/lib (the default), type "make". If ALSSP is to be installed in a different directory, type
   "make INSTALLDIR=directory_to_install_in".
4. If desired, delete the extracted backup files (Makefile, alssp.tar.Z, and README).
5. Make sure that WAVE_PATH is set correctly. This is set in each user's .cshrc file. The path which must be included will be given by the make command. It should be added to each user's .cshrc file as a line of the form:
   setenv WAVE_PATH "/usr/local/lib/alssp/wave"
6. Make sure that PV-WAVE is configured with enough space for ALSSP. The best way to do this is to create a file called ".wave_startup" in each user's home directory. This file should include the line ".size 32000 32000". Then the environment variable WAVE_STARTUP should be set to ~/.wave_startup.
7. Before using ALSSP in PV-WAVE, WAVE_PATH and WAVE_STARTUP must be set. Once the above line has been added to the user's .cshrc file, this will happen automatically when they log on. Note that changes to the .cshrc file do not affect the user's environment until they either log out then back in or run the command "source .cshrc".
8. If desired, copy the EXAMPLE directory to your own directory tree so you can run the example lens system described next. Assuming that ALSSP was installed in the default directory, the command "cp -r /usr/local/lib/alssp/EXAMPLE ." will create the EXAMPLE directory as a subdirectory of the current working directory. If ALSSP was installed somewhere other than /usr/local/lib, then that path should be substituted in the cp command. The -r option for cp is required because the EXAMPLE directory contains a subdirectory called template which also must be copied.

Example Installation:

Become the superuser:
```
 su
```
Enter the Root Password:
```
 G1xx&f4a
```
Extract the files from tape (device rst0):
```
 tar xf /dev/rst0
```
Install the software:
```
 make
```
Returns:
```
 Installing ALSSP in /usr/local/lib
 Make sure WAVE_PATH includes /usr/local/lib/alssp/wave
```
Go back to being a normal user:
```
 exit
```
Edit .cshrc to set WAVE_PATH and WAVE_STARTUP. Add the lines:
```
 setenv WAVE_PATH /usr/local/lib/alssp/wave
 setenv WAVE_STARTUP ~/.wave_startup
```
Create the file ~/.wave_startup and include the line:
```
 .size 32000 32000
```
Source the .cshrc file to set WAVE_PATH and WAVE_STARTUP in this session:
```
 source .cshrc
```
Copy the EXAMPLE directory:
```
 cp -r /usr/local/lib/alssp/EXAMPLE .
```
Move to the EXAMPLE directory:
```
 cd EXAMPLE
```
Run PV-WAVE:
```
 wave
```

Coordinate System

The coordinate system used by this package is shown in Figure 1 with an example lens system. All length units used in the programs are in centimeters (cm) with one exception, which is that the speed of sound is given in meters/second (m/sec). The rays are always traced moving in the positive x direction (left to right). The screen plots always display y vs. x, although the lens system may project into the z direction. In such cases, the projection of the entire system into the xy plane is displayed (unless */plot2d* is set).

The position of the origin can be selected arbitrarily. If a BEAM THREE OPTICS file is used to specify the lens system, the origin will be set to the same position as it was in BEAM THREE.

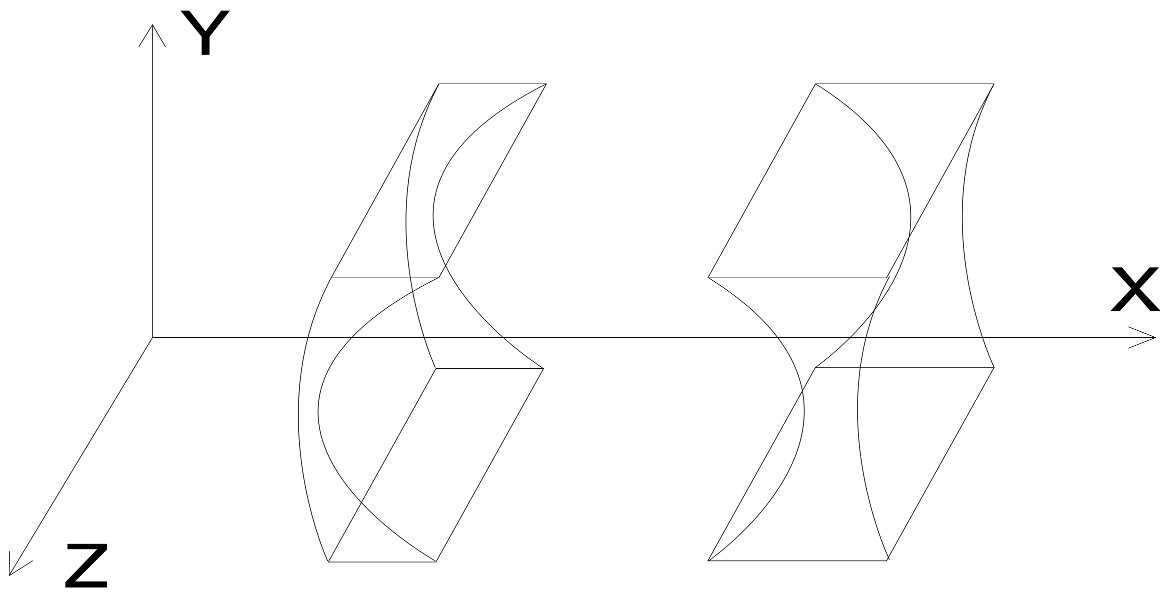

Figure 1 - Coordinate System

EXAMPLE LENS SYSTEM

To illustrate the use of ALSSP in an iterative design project, a thin lens system has been designed that is to operate at 900 kHz in 10 °C fresh water. The beams formed should be less than 0.25° wide with sidelobes more than 40 dB down within a 10° field of view, and less than 0.5° wide with sidelobes more than 20 dB down within a 50° field of view. The element size and positioning need to be determined in order to finish the system design.

The files necessary to follow this example in PV-WAVE are in the EXAMPLE directory which is installed on the system along with ALSSP. See the Installation Instructions for information on how to copy the EXAMPLE directory to your own directory tree. Before running any of the commands in this example, make sure that you are in the EXAMPLE directory.

The first step in setting up a lens system is to build the system parameter file. In this case, the system was designed using BEAM THREE, so the OPTICS file is available. That data is saved in the file 'example.opt':

```
6 surfaces             w25r4c6.opt
Index     Zvx      F  Dia   Curv      Shape      Cx      Mir/Lens
-------- :---------- :-:------ :--------- :----------- :------- :------------- :
1        :.5       :S:30    :         :          :       :iris          :
1        :1        :S:30    :0.0049   :124.1379  :0      :lens          :
0.5613   :1.5      :S:30    :0.0372   :-0.3251   :0      :lens          :
1        :16.62    :S:30    :-0.0107  :18.1977   :0      :lens          :
0.5613   :17.827   :S:30    :0.0230   :-0.9181   :0      :lens          :
1        :53.4944  :S:40    :-0.0169  :1.2129    :0      :film          :
0.5613 : Syntactic Foam : 0.6983 : 2613.5 : -1.25 : 1.0 : 0.0
```

GENERATE_SYSTEM will convert this file to a form that CALC_LENS can use. However, BEAM THREE only does ray tracing, so does not provide information about the system frequency or the ambient conditions. Some of this information is specified in the BEAM THREE OPT file by the BEAM THREE user. BEAM THREE reads the number of surfaces from the first line of the OPT file, then only reads that number of data lines. Lines after that can have anything on them, including additional information on the lens materials which BEAM THREE does not use. GENERATE_SYSTEM requires an extra line for each lens material (except water). These lines contain the refractive index (matching that in the body of the OPT file), the material name, the material density, the coefficients for the speed of sound in the material, and the coefficients for the attenuation rate of sound in the material. The items must be separated by colons or commas. The sound speed coefficients are specified as the constants A and B in the equation Speed=A+B*T, with A in m/sec and B in m/sec/°C. A constant sound speed can be entered by setting B to zero. The attenuation coefficients are specified similarly, as the constants A and B in the equation Attenuation=A+B*f, with the units dB/cm and db/cm/kHz, respectively. Again, a constant attenuation can be specified by setting B to zero.

This information will be stored in a system header file ’system_header’:

```
/logfile            ; Use log file ’batch.log’
/nopressure         ; Do not do pressure calculation, only ray trace.
/trace2d            ; Do ray trace in xy plane only.
/summation          ; Do ray trace for summation method.
/savefile           ; Save results in filename.save
Sal=0               ; Water Salinity, parts per thousand
Depth=1.0           ; Water Depth, meters
freq=900k           ; Frequency, Hz
xrange=[0,65]       ; X range to display
yrange=[-25,25]     ; Y range to display
num_rays=301        ; Number of rays to trace
waterdensity=1.0    ; Water Density, g/cm^3
wateratten=0.0      ; Water Attenuation, dB/cm
```

In order to decide where elements should be placed, the focal point of the system for various source angles must be found. GENERATE_SYSTEM will make parameter files to do this automatically if given the desired source angles. In this case, source angles of 0°, 1°, 2.5°, 5°, 10°, and 25° will provide a good picture of the retina shape. In addition, the point source distance must be defined. For this lens system, the targets will be approximately 25 meters from the lens, so this will be used as the point source distance. The lens material used in this design is syntactic foam, which has a sound attenuation rate of approximately 1.0 dB/cm at 900 kHz. From within PV-WAVE, the command to make the desired parameter files is:

```
GENERATE_SYSTEM,'example.opt',sys_header='template/system_header' $
 ,angles=[0,1,5,10,25],Temperature=10.0,dist=2500,/mouse,/verbose
```

In the process of generating the parameter files, GENERATE_SYSTEM will ask the user to select the regions of the ray-trace plots in which to search for focal points. See the MOUSEBOX instructions for more information. GENERATE_SYSTEM also provides a batch file which contains the commands required to run the generated parameter files, so that should be run next. It will take approximately 20 minutes to run. The PV-WAVE command to run a batch file is the "@" symbol followed by the batch file name. The batch file generated by GENERATE_SYSTEM is named "batch", so the command to run it is:

```
@batch
```

After running the batch file, the results of the focal point determination routines are held in the log file 'batch.logm'. In order to view those results, the program GET_LOG can be run and the results printed and plotted. The output of GET_LOG is described in its description under Utility Programs.

```
points=GET_LOG('batch.logm')
PRINT,points
CALC_LENS,'t0_10',/view,num_rays=5,xrange=[-5,50],/square
OPLOT,points(0,*),points(1,*),psym=-1
OPLOT,points(0,*),-points(1,*),psym=-1
```

Figure 2 shows the resulting plot. The lens system is the five curved surfaces on the left and the retina is the plus symbols connected by line segments. The left-most lens element is a planar surface which doesn't affect the rays except to define the aperture of the system. In BEAM THREE, this is an iris. The next two surfaces represent the first syntactic foam lens. The first is an oblate ellipsoid, the second a hyperboloid. The final two surfaces are a second syntactic foam lens. The front surface is another hyperboloid and the back surface is a prolate ellipsoid. The focal points found for each source angle are shown as plus symbols ('+') in the figure, and

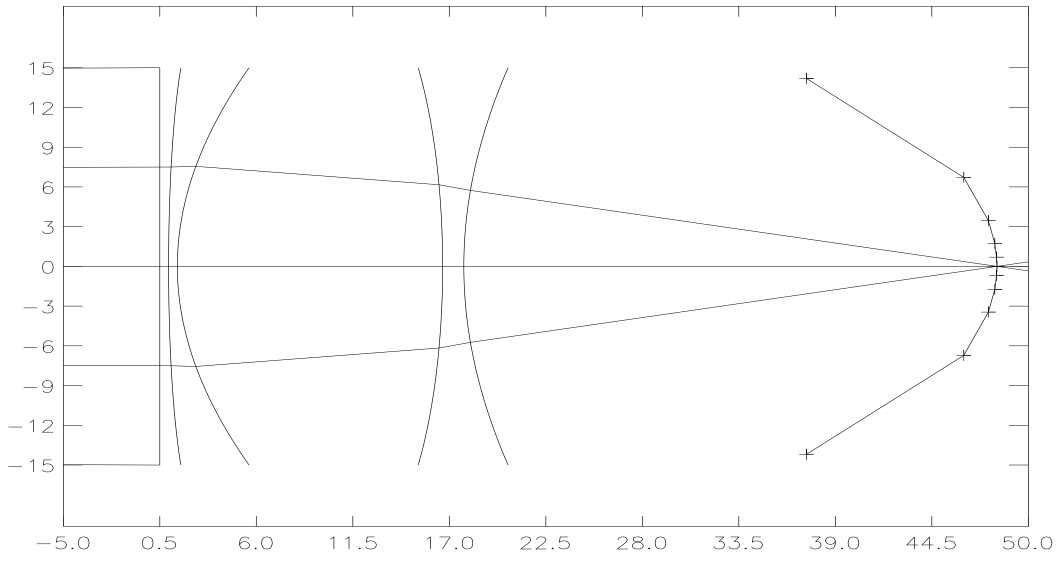

Figure 2 - EXAMPLE Lens System with Focal Trajectory

are joined by line segments to approximate the ideal retina shape for the lens system. Note that this system is symmetric about the x-axis, so the focal points found for positive source angles can be mirrored for the corresponding negative source angles. Although it appears that only three rays were traced instead of the five specified in the CALC_LENS command, all five are shown on the left-most side of the plot. The outer two are clipped by the first lens surface, due to round-off error.

The next step in analyzing this system is to compute beam patterns. The program MAKE_BEAMS2 will be used to generate the parameter files to do this. Since the retina for this system is not an arc, the element must be moved along a trajectory approximating the retina shape. However, the retina can be approximated as a collection of arcs for small changes in the source angle. The center of each arc is referred to as the secondary principal point in optical lens design. The program FOCI_CIRC1 finds the parameters of these arcs, which can then be fed to MAKE_BEAMS2. See the instructions for GET_LOG and FOCI_CIRC1 for more details on the output of these programs. In brief, *foci(5:7,n)* is the x,y,z coordinates of the secondary principal point of the nth source angle and *foci(8,n)* is the radius of the corresponding arc which passes through the focal point.

The first line of the next set of commands removes the batch file from the previous set of instructions so we don't run them again (MAKE_BEAMS2 appends to the end of the file rather than removing it.) If the second and third lines have already been executed in the current PV-WAVE section, the arrays "points" and "foci" will already be defined, and so the lines need not be executed again. They are include here, however, to ensure that "points" and "foci" contain the correct information.

```
$rm batch
points=GET_LOG('batch.logm')
foci=FOCI_CIRC1(points)
MAKE_BEAMS2,foci(5:7,0),foci(8,0),[0,1],'t0_10',numpos=51
MAKE_BEAMS2,foci(5:7,2),foci(8,2),[4,6],'t5_10'
MAKE_BEAMS2,foci(5:7,3),foci(8,3),[9,11],'t10_10'
MAKE_BEAMS2,foci(5:7,4),foci(8,4),[23,27],'t25_10'
@batch
```

This will generate four data sets. Note that the default of 101 positions was used on all but the first MAKE_BEAMS2 call. The first call is generating half the arc length (1 degree instead of 2) and so half as many points are used to give identical spacing. Since the source is on the x-axis for this example, the response will be symmetric and the results can be mirrored, halving the number of computations required. Alternatively, the first MAKE_BEAMS2 line could have specified an arc range of [-1,1] and used the default 101 positions. MAKE_BEAMS2 generates parameter files named by using the system file name plus "a1" appended to the end. These can be plotted using KPLOT:

```
KPLOT,'t0_10a1',/find3db,/mirror,xticks=8,yticki=6,/two_way,origin=foci(5:7,0) $
 ,title='EXAMPLE: On-axis Source, Point Element',/side

KPLOT,'t5_10a1',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,2) $
 ,title='EXAMPLE: 5 degree off-axis Source, Point Element',/side

KPLOT,'t10_10a1',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,3) $
 ,title='EXAMPLE: 10 degree off-axis Source, Point Element',/side

KPLOT,'t25_10a1',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,4) $
```

```
 ,title='EXAMPLE: 25 degree off-axis Source, Point Element',/side
```

The results of the first and last KPLOT commands are shown in Figures 3 and 4. Note that the keyword parameter */two_way* was specified in each command so that the two-way rather than one-way beam patterns are shown. Also, the keyword */mirror* was specified for the first command so that the response, which was calculated from 0 to 1 degrees, would be mirrored to get a plot from -1 to 1 degrees.

The numerical results are tabulated in Table 1. Note that the */find3db* option in KPLOT normalized all the plots so that the peak amplitude was 0 dB. Adding the */verbose* keyword to the KPLOT command will display the maximum absolute magnitude on each plot. In order to compare two beam patterns, */offset* can be set to the same value for each plot. See the end of this section for an example (Pages 14 and 15).

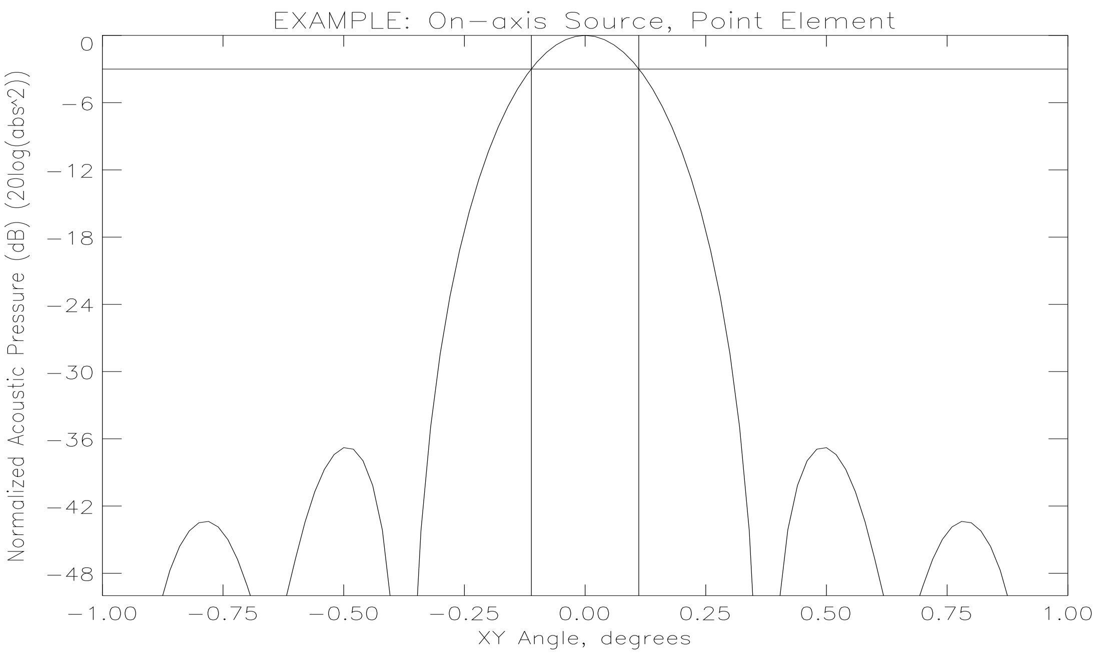

Figure 3 - Beam Pattern for Point Element and On-Axis Source

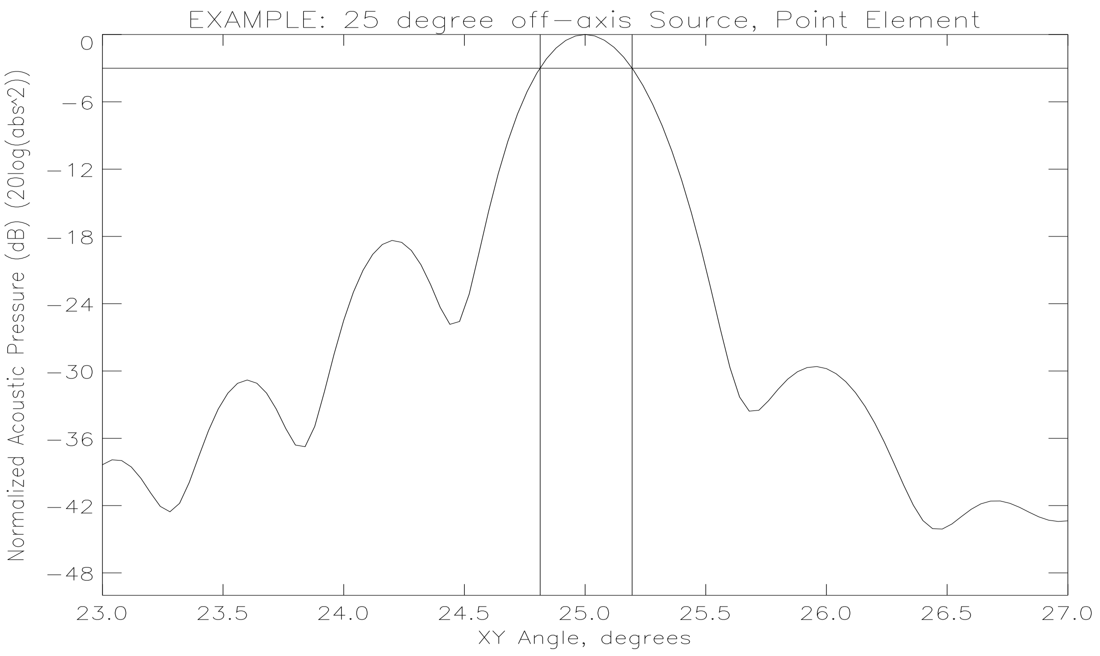

Figure 4 - Beam Pattern for Point Element and 25° Off-Axis Source

Table 1 - Point Element Data		
Source Location	Two-way beam width	Sidelobe Level
0°	0.223°	-36.8 dB
5°	0.224°	-34.8 dB
10°	0.232°	-32.6 dB
25°	0.377°	-18.4 dB

Table 1 shows the beam widths are all within the specifications, but the sidelobe levels are all too high. However, these calculations were made for a point element. Any real element will have a physical size and shape which will affect the beam widths and sidelobe levels. By adding the effect of element size and shape into the calculations, the sidelobe levels can be decreased for a small penalty in beam width. For example, adding a 0.268 cm element is done by modifying the above instructions to:

```
$rm batch
points=GET_LOG('batch.logm')
foci=FOCI_CIRC1(points)
MAKE_BEAMS2,foci(5:7,0),foci(8,0),[0,1],'t0_10',numpos=51 $
 ,elementfile='template/rect0.268',start=2
MAKE_BEAMS2,foci(5:7,2),foci(8,2),[4,6],'t5_10' $
 ,elementfile='template/rect0.268',start=2
MAKE_BEAMS2,foci(5:7,3),foci(8,3),[9,11],'t10_10' $
 ,elementfile='template/rect0.268',start=2
MAKE_BEAMS2,foci(5:7,4),foci(8,4),[23,27],'t25_10' $
 ,elementfile='template/rect0.268',start=2
@batch
```

This is identical to before, but an element template file has been specified which contains the information needed for a 0.268 cm line element:

```
rect                    ; Use a rectangular element.
0.268,0                 ; Element is 0.268 cm high (y direction) and 0 cm wide (x)
direction               ; Element will always face in the same direction.
```

```
-1.0,0,0                ; That direction is (-1,0,0), which is the -x direction.
10                      ; Number of points on element
```

In addition, the keyword parameter */start* has been set to 2, which means the output files will end with a '2' so that they won't overwrite the previous output files, which ended with the default '1'. Plotting these files is done by the commands:

```
KPLOT,'t0_10a2',/find3db,/mirror,xticks=8,yticki=6,/two_way,origin=foci(5:7,0) $
 ,title='EXAMPLE: On-axis Source, 0.268 cm Element',/side

KPLOT,'t5_10a2',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,2) $
 ,title='EXAMPLE: 5 degree off-axis Source, 0.268 cm Element',/side

KPLOT,'t10_10a2',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,3) $
 ,title='EXAMPLE: 10 degree off-axis Source, 0.268 cm Element',/side

KPLOT,'t25_10a2',/find3db,xticks=8,yticki=6,/two_way,origin=foci(5:7,4) $
 ,title='EXAMPLE: 25 degree off-axis Source, 0.268 cm Element',/side
```

These patterns are similar, but exhibit slightly larger beam widths and lower sidelobe levels, as expected. The results are tabulated below:

EXAMPLE: 0.268 cm Element Data		
Source Location	Two-way beam width	Sidelobe Level
0°	0.246°	-47.8 dB
5°	0.248°	-46.8 dB
10°	0.255°	-44.6 dB
25°	0.401°	-22.6 dB

Now the specifications have been met. For the 10° field of view, the beam widths are under 0.25° and the sidelobe levels are below -40 dB. For the 50° field of view, the beam widths are under 0.5° and the sidelobe levels are below -20 dB.

In order to visually inspect the effects of including element size, two beam patterns can be plotted on the same graph. This requires using the */noerase* option on KPLOT. In addition, if the graph is to show relative amplitudes, */offset* must be set to the same value for each plot. The following lines will create the plot shown in Figure 5. The offset of -35.3147*2.0 comes from the maximum absolute value (in dB) returned by the first KPLOT command (due to the */verbose* option.) doubled since the plot shows two-way beam patterns (due to the */two_way* option.)

```
points=GET_LOG('batch.logm')
foci=FOCI_CIRC1(points)
origin=foci(5:7,0)

KPLOT,'t0_10a1',/db,/two_way,/mirror,origin=origin $
 ,xticki=0.25,xminor=2,yrange=[-70,0],yticki=6,yminor=2 $
 ,title='On-Axis Source, 2-Way Beam Pattern' $
 ,/verbose,offset=-35.3147*2.0

KPLOT,'t0_10a2',/db,/mirror,/two_way,origin=origin,/verbose,psym=-1 $
 ,offset=-35.3147*2.0,/noerase

KLEGEND,['Point Element','0.268 cm Square Element'],[0,-1] $
 ,ystart=-1,xstart=-0.95
```

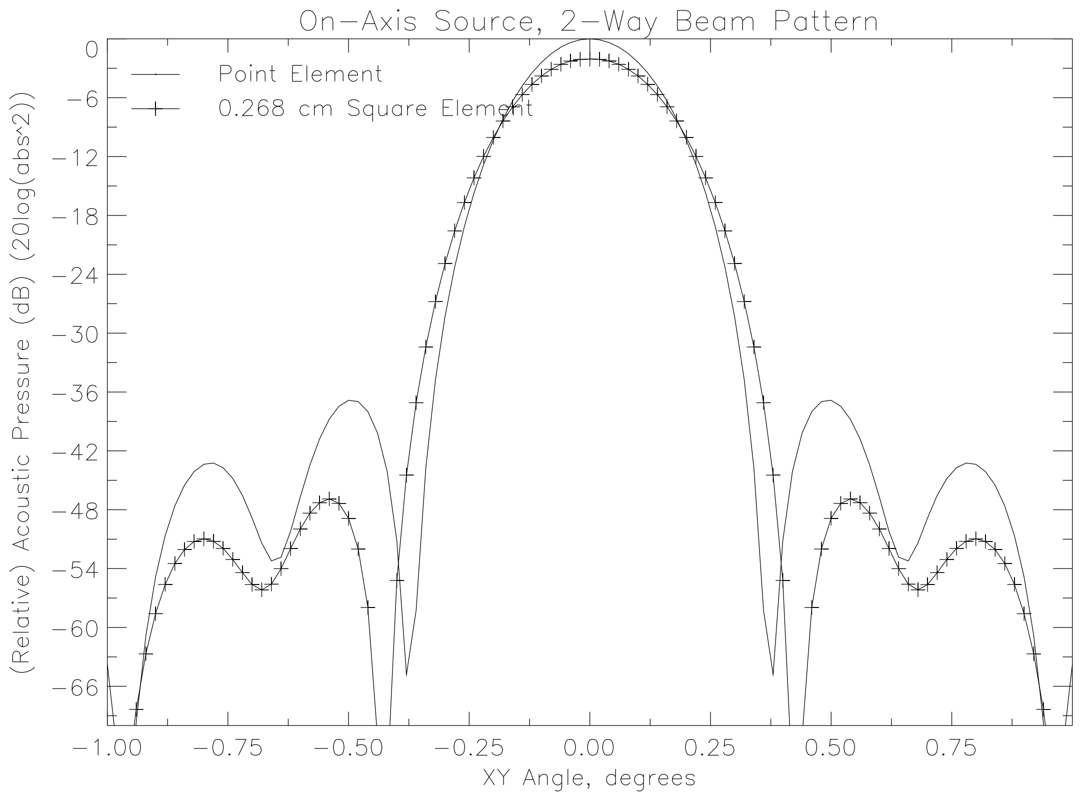

Figure 5 - Comparison of Beam Patterns with and without Element

Although the above information and plots are useful, we might want to create a more visually compelling display of the lens's operation. CALC_LENS can generate pressure fields within a plane or rectangular volume, and KPLOT can display them as contour plots or images. The utility program MAKE_VOLUME creates a parameter file which will instruct CALC_LENS to calculate the pressure field in a plane. The first parameter is the file name of the ray-trace parameter file for the system. This is used to draw a ray-trace plot so that the user can then select the region of image space within which to calculate the pressure field. The second parameter is the name of the parameter file which MAKE_VOLUME will create. The parameters *xpts* and *ypts* specify the number of points in each dimension to sample the pressure field at. The keyword parameter */dec* specifies the number of decimal points to round the pressure field's boundary coordinates to.

After the parameter file has been created, we use CALC_LENS to calculate the pressure field and KPLOT to display it. In this case, we will display an image with a 3 dB contour plot overlayed on top of it (Figure 6). The PV-WAVE commands are:

```
MAKE_VOLUME,'t0_10','t0_10v',xpts=60,ypts=40,dec=0
CALC_LENS,'t0_10v'
KPLOT,'t0_10v',/db,/contour,nlevels=1      ; Create correct axes for image.
KPLOT,'t0_10v',/db,contour=3               ; Display image.
KPLOT,'t0_10v',/db,/contour,/noerase       ; Overlay contour plot.
```

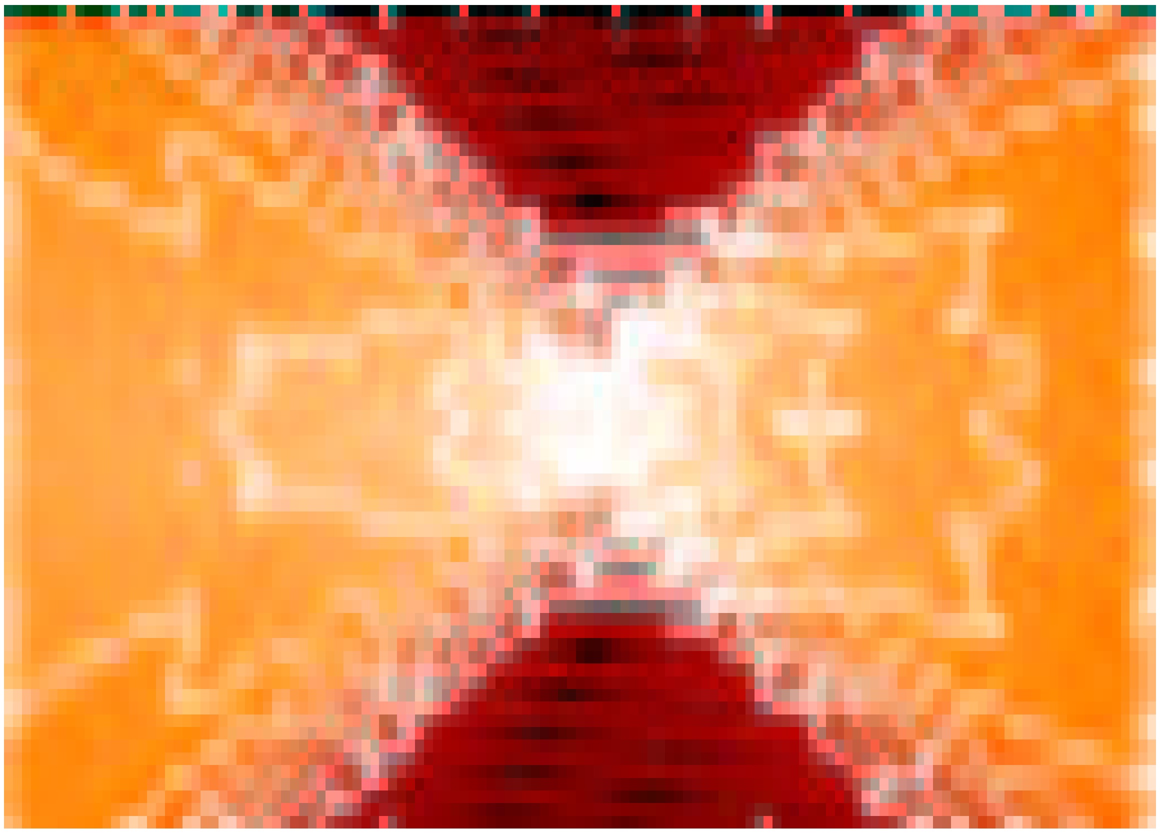
Figure 6 - Pressure Field for On-Axis Point Source

CALC_LENS

The main program is CALC_LENS. It can be run either interactively (from within PV-WAVE) or non-interactively (as a batch file). From within PV-WAVE, the command is:

CALC_LENS,*filename* [,optional keyword parameters]

The program can also be run in a batch file (i.e. non-interactively). This is done by listing lines of the same form as the interactive command shown above. The batch file could look like this:

```
CALC_LENS,’infile1’
CALC_LENS,’infile2’
CALC_LENS,’infile2’,num_rays=1001,outfile=’infile2.1001.out’
```

This would run three different parameter sets. The files ’infile1’ and ’infile2’ would contain the parameters. The third run would be modified from the second by tracing 1001 rays instead of whatever value was in ’infile2’ and then saving the results in ’infile2.1001.out’ instead of ’infile2.out’.

If the above three lines were saved in the file ’batch’, then the command to run the batch file would be: wave < batch >& batch.err &

This tells the computer to run PV-WAVE using the file ’batch’ as input. The trailing ampersand puts the process in the background, so you can continue using the computer for other tasks. Note, however, that if only one license for PV-WAVE is available, then PV-WAVE can’t be run while this batch file is running. The ">& batch.err" stores the output of the program that would normally go to the screen to the file "batch.err". Timing information and errors will be stored in that file. The actual results of the program will be stored in the file named by */outfile*, if specified, or by the input file name with ’.out’ appended to the end if */outfile* isn’t specified.

Keyword Parameters

The *filename* parameter is the file name of a parameter file which contains information on the lens system and the type of calculation desired. The optional keyword parameters supersede the values stored in the parameter file(s). The available keyword parameters are:

/integration	Use Kirchhoff Integration instead of summation method.
/logfile	File to save stderr output in.
/noexit	If set, don't plot exit plane.
/noplot	If set, don't plot anything [on by default].
/nopressure	If set, don't do any pressure calculations.
/norhoc	If set, don't calculate rho-c losses (transmission coefficients).
/num_rays	Number of rays to trace (for 2-D ray tracing).
/num_rings	Number of hexapolar rings (for 3-D ray tracing).
/outfile	Output File Name
/plot2d	If set, only plot rays in z=0 plane.
/plotelement	If set, plot sampling positions on element.
/restorefile	if set, restore ray tracing instead of redoing it.
/savefile	If set, save results of ray tracing for future use.
/skip	Number of rays to skip when plotting.
/square	If set, set */yrange* from */xrange* for square aspect ratio.
/summation	Use summation method [default] instead of Kirchhoff Integration.
/trace2d	Use 2D ray tracing instead of 3D.
/verbose	Amount of feedback during run (0,1,2) (sent to stderr).
/view	If set, don't save or log anything, just plot 21 rays and exit.
/waterspeed	Override speed of sound in water.

/xrange, /yrange, /xticks, /yticks, /ticklen, /xtitle, /ytitle, /title, /subtitle, /color, and */noerase* are the same as for the PV-WAVE plot command.

If */outfile* is set, then the output file name will be the input file name with ".out" appended. It can also be set to a string, in which case that will be used as the output file name. The output file stores the results of pressure calculations, so is only applicable when pressure calculations are performed.

If */savefile* is set, then the save file name will be the input file name with ".save" appended. It can also be set to a string, in which case that will be used as the save file name. The save file stores the results of the ray tracing calculations, not the results of pressure calculations.

If */logfile* is set, then the log file name will be "batch.log". It can also be set to a string, in which case that will be used as the log file name. The log file stores information on the progress of the calculation and the values being used. It is useful for debugging lens parameter files and viewing the progress of batch jobs, as well as storing the results of maxima search.

/num_rays sets or supersedes the number of rays which will be traced in the 2-D ray-tracing calculations. Similarly, */num_rings* sets or supersedes the number of hexapolar rings in 3-D ray tracing. In 3-D ray tracing, the rays are arranged in hexapolar rings. There is 1 ray in the center, 6 rays in the first ring, 12 in the second, 18 in the third, etc. The rays in each ring are evenly distributed around the center of the lens. This results in 1+3**num_rings**(*num_rings*+1) total rays traced.

/skip is used when a large number of rays are to be traced, but only some fraction are to be plotted. For example, *num_rays*=300, *skip*=10 will plot every tenth ray, for a total of 30 rays plotted.

/waterspeed sets the speed of sound in water (or other ambient medium). The sound speed in water can be either a speed in meters/second, or as the number 0.0. If */waterspeed* is not specified or is set to 0, the water speed will be calculated using equation (1.2.1) from Clay & Medwin, <u>Acoustical Oceanography</u>, 1977:

$$WaterSpeed = 1449.2 + 4.6T - 0.055T^2 + 0.00029T^3 + (1.34 - 0.010T)(Sal - 35) + 0.016Depth$$

where T is the water temperature in degrees Celsius, Sal is the water salinity in parts per million, and Depth is the system depth in meters.

The simulation uses the summation method by default. This can also be implicitly forced by the */summation* keyword, or overridden by the */integration* keyword.

If the summation method is selected, then all ray tracing is done in 3 dimensions, unless */trace2d* is set. This allows the simulation of non-radially-symmetric lens systems with off-axis sources. The Kirchhoff Integration method uses 2-D ray tracing, so is only applicable to radially-symmetric lens systems with on-axis sources.

If */plot2d* is set, then only rays which lie in the xy (z=0) plane will be plotted. This parameter is only applicable to */summation* with */trace2d* not set.

For */integration*, the exit plane is plotted unless */noexit* is set.

If */norhoc* is set, then the transmission coefficient at each interface is set to 1.0. The transmission coefficients model the loss of sound energy through each interface due to the impedance mismatch between two materials of different acoustic impedance. This option is provided so that the total rho-c loss can be calculated by running the same lens system with and without */norhoc* set.

If */nopressure* is set, then the simulation will exit after finishing (and possibly saving the results of) the ray tracing, whether or not the parameter file goes on to describe a pressure calculation.

/verbose sets the amount of output given to the user during the simulation. It ranges from 0 to 2, with 0 corresponding to minimal information and 2 to very complete reporting (useful for debugging).

If */view* is set, then no calculations are performed or saved. The ray trace is done with 21 2-D rays or 4 3-D rings (unless overridden by num_rays on the command-line) and plotted, then the program exits. No matter what they were originally set to, logging, saving, and pressure calculations are turned off. Other parameters are still available.

If */noplot* is set, then nothing is plotted. This speeds up calculations and keeps the system from crashing when operating in the background with no windowing system operating. It should always be set when a batch file will be left running after the user has logged off the system. This is set on by default. Use *noplot*=0 or */view* to turn it off.

If */plotelement* is set, then a plot will be made of the sampling positions on the element. This is only applicable when an element is being used.

If */square* is set, then yrange will be set from xrange in order to achieve a square aspect ratio. */square* overrides any yrange set.

Parameter Files

All of the information about a given lens system is stored in parameter file(s). The parameter file can include any of the keyword parameters described above, plus several more. In addition, the parameter file includes information on the lens configuration and the type of calculation desired. The additional parameters (*T*, *freq*, *Sal*, *Depth*, *wateratten*, and *waterdensity*) are described below. The other information specified is described in following sections (**Source Types**, **Lens Surfaces**, and **Calculation Types**).

The first lines of a parameter file may begin with semi-colons, in which case they will be ignored. This is useful for adding comments or descriptions of the parameter file.

The parameter file must specify the ambient temperature and the sonar frequency. The ambient temperature is specified in degrees C by a line in the parameter file such as:

```
T=20.0              ; System temperature in degrees C.
```

The sonar frequency is given in Hertz (Hz), unless it is followed by a ’k’ or an ’M’, in which case it is in kilohertz (kHz) or megahertz (MHz). For example:

```
freq=3000           ; Frequency is 3000 Hz (3 kHz).
freq=3.2k           ; Frequency is 3200 Hz (3.2 kHz).
freq=5.0M           ; Frequency is 5000000 Hz (5.0 MHz).
freq=4.2e6          ; Frequency is 4200000 Hz (4.2 MHz).
```

The parameter file may also specify other system parameters which have default values assigned to them. The water salinity, *Sal*, defaults to 0.0 parts per thousand (i.e. fresh water.) The water depth, *Depth*, defaults to 0.0 meters. The rate of sound attenuation in water, *wateratten*, defaults to 0.0 dB/cm. The density of the water, *waterdensity*, defaults to 1.00 g/cm^3. The speed of sound in water, *waterspeed*, defaults to 0.0, which means that it will be calculated using the system parameters *Temperature*, *Salinity*, and *Depth*.

For an example system with a simple single-component thin lens, the parameter file would look like:

```
; This is an example parameter file.  The program will skip the first lines
; which begin with semicolons, so they may be used for comments.
logfile=batch.log   ; Anything after a semicolon is also ignored.
/trace2d            ; Trace in 2 dimensions rather than 3.
```

```
/summation           ; Use the summation method. (This is the default.)
Sal=0                ; Water Salinity, parts per thousand
Depth=1.0            ; Water Depth, meters
freq=600000          ; Frequency, Hz (600 kHz)
Source=point         ; Source Type
0.0,0.0,0.0          ; Point Source Location, cm
xrange=[4950,5200]   ; x range to display on ray-trace plots.
num_rays=301         ; Number of rays to trace
waterdensity=1.0     ; Water Density, g/cm^3
wateratten=0.0       ; Water Attenuation Rate, dB/cm
T= 20.0              ; System Temperature, degrees C
waterspeed=0.0       ; Calculate Speed of Sound in Water
2                    ; Number of Interfaces
spherical            ; Lens Type
4859.016113,0,0      ; Lens Center, cm
-140.984070          ; Lens Radius of Curvature, cm
20.500000            ; Lens Aperture Radius, cm
2613.5,-1.25         ; Sound Speed Coefficient for Syntactic Foam, m/sec
0.6983               ; Syntactic Foam Density, g/cm^3
2.5                  ; Syntactic Foam Attenuation Rate, dB/cm
spherical            ; Lens Type
5031.522461,0,0      ; Lens Center, cm
30.522234            ; Lens Radius of Curvature, cm
20.500000            ; Lens Aperture Radius, cm
0.0                  ; Use Sound Speed for Water
1.00                 ; Water Density, g/cm^3
0.0                  ; Water Attenuation Rate, dB/cm
axial                ; Calculate pressures along a line parallel to an axis.
y                    ; Line will be parallel to the y axis.
5058.76,0.0          ; The line will be at x=5058.76, z=0.0 cm.
0.0,2.0              ; The line will run from y=0.0 to 2.0 cm.
51                   ; Calculate at 51 element positions along the line.
point                ; The element is a point.
```

This parameter file could also be split into two sections. The first section describes the lens system and the second describes the desired calculation. This is useful when many different beam patterns/area calculations/etc will be performed on one system. The ray tracing is done based on the first section, then saved. Each pressure calculation then uses the saved ray trace results, eliminating unnecessary repetition of the ray tracing process. If this were done, the parameter files for the example system would look like:

```
; Example parameter file number 1.  This section describes the lens system.
logfile=batch.log           ; Use 'batch.log' as the log file.
/nopressure                 ; Don't do any calculations except the ray trace.
/trace2d                    ; Trace in 2 dimensions rather than 3.
/summation                  ; Use the summation method. (Default)
savefile=example.save       ; Save the ray trace results in 'example.save'.
Sal=0                       ; Water Salinity, parts per thousand
Depth=1.0                   ; Water Depth, meters
freq=600000                 ; Frequency, Hz
Source=point                ; Source Type
0.0,0.0,0.0                 ; Point Source Location, cm
xrange=[4950,5200]          ; x range to display
num_rays=301                ; Number of rays to trace
waterdensity=1.0            ; Water Density, g/cm^3
wateratten=0.0              ; Water Attenuation Rate, dB/cm
T= 20.0                     ; System Temperature, degrees C
waterspeed=0.0              ; Calculate Speed of Sound in Water
2                           ; Number of Interfaces
spherical                   ; Lens Type
4859.016113,0,0             ; Lens Center, cm
-140.984070                 ; Lens Radius of Curvature, cm
20.500000                   ; Lens Aperture Radius, cm
2613.5,-1.25                ; Sound Speed Coefficients for Syntactic Foam, m/sec
0.6983                      ; Syntactic Foam Density, g/cm^3
2.5                         ; Syntactic Foam Attenuation Rate, dB/cm
spherical                   ; Lens Type
5031.522461,0,0             ; Lens Center, cm
30.522234                   ; Lens Radius of Curvature, cm
20.500000                   ; Lens Aperture Radius, cm
0.0                         ; Use Sound Speed for Water
1.00                        ; Water Density, g/cm^3
0.0                         ; Water Attenuation Rate, dB/cm
```

```
; This is the second parameter file.  It describes the specific
; calculation to be performed.
restorefile=example.save; Restore the ray trace from 'example.save'
logfile=batch.log       ; Use 'batch.log' as the log file.
/summation              ; Use the summation method.
axial                   ; Calculate pressures along a line parallel to an axis.
y                       ; Line will be parallel to the y axis.
5058.76,0.0             ; The line will be at x=5058.76, z=0.0 cm.
0.0,2.0                 ; The line will run from y=0.0 to 2.0 cm.
51                      ; Calculate at 51 element positions along the line.
point                   ; The element is a point.
```

If CALC_LENS can't find the file specified by */restorefile,* it will attempt to create it by running the parameter file named by deleting the '.save' suffix off of */restorefile.*

Source Types

The source type can be 'point', 'planar', 'curved_line', or three numbers separated by commas. If the source type is 'point', then the next line must contain the xyz coordinates of the point source location. A source type of three numbers separated by commas is the same as a source type of 'point' followed by those three numbers, unless the three numbers are '999,0,0' which is the same as a source type of 'planar'. For example:

```
source=point              ; Point Source
-100.0,0.0,0.0            ; on-axis, 100 cm back from origin
source=-100.0,0.0,0.0     ; Same as last example
source=planar             ; Planar Source
source=999,0,0            ; Same as last example
source=curved_line        ; Curved-line source
-500.0,0.0,0.0            ; Point Source Location, cm
1.0                       ; Source Height, cm
4.0                       ; Source Curvature, cm (0.0 means straight line)
49                        ; Number of Point Sources on Line Source
```

Lens Surfaces

After the system parameters have been set, the lens surfaces are described. Each surface, or interface, is listed separately. There are currently four lens surface types available: planar, spherical, aspherical, and ellipsoid.

In each case, the specified sound speed and density are of the material between this interface and the next. Therefore, the last lens specification describes the outer surface of the lens and whatever is between it and the retina. For a thick lens, this would be the working fluid, whereas for a thin lens it would probably be water.

The sound speed in each material can be specified in one of four forms: as an offset/slope pair, a speed in meters/second, a refractive index (re: water=1.00), or as the number 0.0. If two numbers are given, the first form is assumed, and the sound speed calculated as offset+slope*Temperature. If one number is given and it is greater than 100, then it is used as the sound speed in m/s. If the number is between 0 and 100, then it is assumed to be a refractive index and the sound speed is calculated using the speed of sound in water. If the number is 0, then the speed of sound in water given in or calculated from the first section of the parameter file is used.

The density of each material can be specified as a fixed value in g/cm^3 or as the coefficients A,B of a linear fit of the form density=A+B*T, where T is the system temperature in °C.

The attenuation in each lens material can also be specified as a fixed value in dB/cm or as the coefficients A,B of a linear fit of the form attenuation=A+B*f, where f is the system operating frequency in kHz.

Type ’planar’ describes a planar lens surface. Its parameters are:

```
planar                  ; Type
0.0,0.0                 ; Origin, cm
0.5                     ; Aperture Size (diameter), cm
1400.0                  ; Sound Speed in Lens Material, m/sec
1.026                   ; Density of Lens Material, g/cm^3
0.05                    ; Lens Material Attenuation Rate, dB/cm
```

This lens is a disk in 3-space so looks like a vertical line in the ray plots. It is centered at the specified origin with a diameter of the specified aperture size.

Type ’spherical’ represents spherical lenses. A section of a parameter file which has a spherical lens looks like:

```
spherical               ; Type
-3.0,0.0,0.0            ; Center of Curvature, cm
3.5                     ; Radius of Curvature, cm
4.0                     ; Aperture Size (diameter), cm
2200                    ; Sound Speed in Lens, m/sec
1.3                     ; Density of Lens Material, g/cm^3
0.2                     ; Lens Material Attenuation Rate, dB/cm
```

Note: A positive radius corresponds to a convex lens surface (as viewed by the source), negative to a concave surface.

Type ’aspherical’ describes a lens surface which is spherical with an added shape factor that allows different conic classes. The equation governing the surface is $x = \frac{cr^2}{1+\sqrt{1-sc^2r^2}}$ *where* $r^2 = y^2 + z^2$. The curvature **c** is specified by 1/(radius of curvature at the center). The shape factor **s** can be:

s < 0	hyperboloid
s = 0	paraboloid
0 < **s** < 1	prolate ellipsoid
s = 1	sphere
s > 1	oblate ellipsoid

These parameters are identical to those in BEAM THREE, so can be copied directly over. The equation for "Standard" surfaces in ZEMAX is identical except that it uses the conic constant **k=s-1** instead. Thus **k** is zero for spheres, for example. A parameter file section looks like:

```
aspheric                ; Type
0,0,0                   ; Vertex location, cm
```

```
0.05                ; Lens curvature, 1/cm
200                 ; Lens shape factor
30                  ; Lens aperture, cm
2200                ; Sound Speed in Lens, m/sec
1.3                 ; Density of Lens Material, g/cm^3
0.2                 ; Lens Material Attenuation Rate, dB/cm
```

Type ’ellipsoid’ corresponds to a lens described by a 3-D ellipse:

```
ellipsoid           ; Type
0.0,0.0,0.0         ; Origin, cm
2.5                 ; Ellipsoid A Parameter
3.2                 ; Ellipsoid B Parameter
1.2                 ; Ellipsoid C Parameter
0.5                 ; Aperture Size, cm
1400                ; Sound Speed in Lens
1.5                 ; Density of Lens Material
0.1                 ; Lens Material Attenuation Rate, dB/cm
```

The equation for the ellipsoid is A(x-x0)^2+B(y-y0)^2+C(z-z0)^2=1. Note that the spherical lens is an ellipsoid with A=B=C=1/radius^2.

Calculation Types

This simulation can calculate the complex pressure field along a line, an arc, a calculated trajectory, or in a region of space. It can also find the focal point of a lens system along a line or in a plane. In addition, an element can be placed in the pressure field in all of the above calculations.

Beam Patterns

Beam patterns may be found along a line, an arc, or a trajectory. They may also be found by rotating an element around its (fixed) center. The motion or rotation of a point is specified first, then the element type to be moved or rotated.

A line is specified by an origin, a direction, and two distances. The origin is an (x,y,z) triplet in 3-space. The direction is specified by the angle of the line from the x-axis in the y and z directions. The extent of the line is determined by two distances, each of which are measured from the origin. The beam pattern is calculated from the first distance to the second along the line. The number of points along the line at which to calculate the element response must also be specified.

The parameter file section to move along a line looks like:

```
line                    ; Calculate along a line
0.0,0.0,0.0             ; Origin of line
25.0,100.0              ; Distances of interest along line (start and end)
30,0                    ; Angle of line from x-axis (in y and z directions)
101                     ; Number of points to calculate.
```

The preceding sample defines a line which starts at the origin and travels in the xy plane (because the z angle is 0) at an angle 30 degrees off the x axis. The calculations will be performed between 25 cm and 100 cm from the origin.

For the case in which the line lies parallel to an axis, the calculation could also be specified as follows:

```
axial                   ; Calculate parallel to an axis
x                       ; The axis is the x-axis.
10.0,0.0                ; The line is offset by 10.0 cm in the y direction
25.0,100.0              ; The line starts at x=25 cm and ends at x=100 cm.
101                     ; Number of points to calculate.
```

In this case, the line endpoints are specified as absolute coordinates rather than distances from an origin.

The simulation allows the calculation of pressures along an arc, but only in the x-y plane or the x-z plane. The plane which the arc lies in is specified as "y" for the x-y plane or "z" for the x-z plane. The arc is determined by its origin, radius, and extent. The extent of the arc can be designated in one of two ways. The first way is chosen by specifying "angle" and the angles of the endpoints in degrees from the x-axis. The second way is indicated by specifying "position" and the y or z coordinates of the endpoints in cm from the x-axis.

A section of a parameter file using the first method looks like:

```
arc                     ; Calculate along an arc
y                       ; Arc is in the x-y plane.
300,0,0                 ; Origin of arc.
10                      ; Radius of arc
angle                   ; Endpoints are angles in degrees from x-axis.
-10,10                  ; Arc runs from -10 to 10 degrees.
101                     ; Number of points to calculate.
```

An example of a section of a parameter file using the second method:

```
arc                     ; Calculate along an arc
y                       ; Arc is in the x-y plane.
300,0,0                 ; Origin of arc.
10                      ; Radius of arc
position                ; Endpoints are distance from x-axis (y-direction).
-5,5                    ; Arc runs from y=-5 to y=5.
101                     ; Number of points to calculate.
```

For some lens systems, the trajectory which an element must follow in order to simulate the movement of the source is more complicated than a line or an arc. This is especially true for sources which are positioned at large off-axis angles. In these cases, it is better to calculate beam patterns by following the trajectory of focal points which the point source makes when moving along an arc. See the utility command FOCI_CIRC for more information on calculating focal trajectories. An example of the parameter file which is used to calculate the beam pattern once the trajectory has been defined follows:

```
trajectory              ; Use saved trajectory for beam pattern.
foci.save               ; File containing trajectory information.
5000,0,0                ; Lens system origin
20,30                   ; Range of source angles to consider, in degrees.
101                     ; Number of points to calculate.
```

In some situations, it may be advantageous to calculate a beam pattern by holding both the source and element positions fixed and rotating the element. This can be done by specifying "rotation" as the calculation type, then listing the origin of the element and the range of angles (in degrees) to rotate from/to. An example parameter file:

```
rotation                ; Calculate beam pattern by rotating an element.
300,0,0                 ; Element origin (the fixed center of the element.)
-10,10                  ; Rotate the element from -10 to 10 degrees.
101                     ; Number of rotation angles to calculate at.
```

Pressure Fields

The simulation will also calculate pressures in a region of space. This region can be defined as a rectangular volume, a planar rectangle or circle, or as a curved version of a rectangle (i.e. a section of a cylinder.)

A volume of space is specified by two corners in three dimensions and its sampling density in each dimension. By setting one or more dimension's size(s) equal to zero, a rectangular or linear region of space can be selected. An example parameter file section looks like:

```
volume                  ; Calculate pressure field within a rectangular volume of space.
275,-5,-2.5             ; First corner is at (275,-5,-2.5) cm.
325,5,2.5               ; Opposite corner is at (325,5,2.5) cm.
1,0.5,0.25              ; Volume will be sampled at 1 cm intervals in the x-dimension,
                        ; 0.5 cm intervals in the y-dimension, and 0.25 cm intervals in
                        ; the z-dimension.
```

Each of the other three regions are planar (or planar with a curve), and are specified by their origin, size, sampling density, and orientation. There are two ways to define a region's orientation in space. The simplest is by specifying the plane in which it lies: xy, xz, or yz. In these cases the sizes are specified directly in terms of the plane in which the region lies. If the xy plane is specified, then the first size refers to the x-direction and the second to the y-direction. Similarly, if the yz plane is specified, then the first refers to the y and the second to the z.

For more general orientations, the region's normal can be specified to define the direction in which it faces. A normal of [0,-1,0] would put the region in the x-z plane (it's normal is parallel to the y-axis), and a normal of [-1,-1,0] would put the region parallel to the z-axis with its projection into the x-y plane a line with 45 degree angle.

Since the region is assumed to start in the y-z plane, care must be taken to specify the size correctly if another plane is desired. For example, for a rectangular region in the x-y plane with size in the x-direction of 10 cm and size in the y-direction of 5 cm, the orientation vector will be [0,0,1], but the size should be specified as 5,10 rather than 10,5, since the region will rotate around the y-axis, moving the second number from z to x but leaving the first number as y.

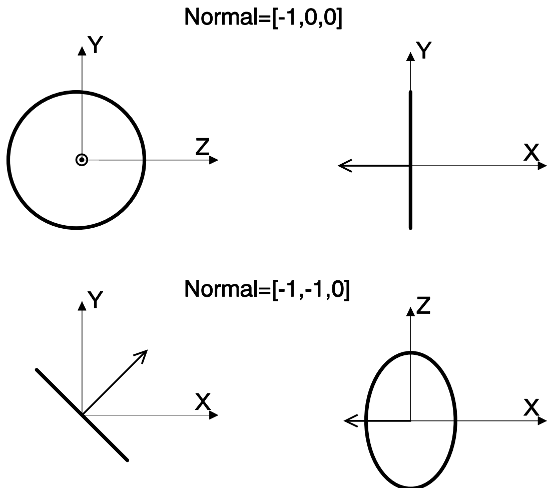

Figure 7 - Orientation Examples

A rectangle is specified by its center (or origin), its size in two dimensions, the sampling density on its surface (in each dimension), and its orientation in space. For the orientation given as an example above, the parameter file could be specified as either:

```
rectangle          ; Calculate within a rectangular section of space.
300,0,0            ; Origin (center) of rectangle
10,5               ; Size of rectangle (x_size=10, y_size=5)
0.5,0.25           ; Sampling density in x and y directions, respectively.
xy                 ; Rectangle is in x-y plane.

or:
rectangle          ; Calculate within a rectangular section of space.
300,0,0            ; Origin (center) of rectangle
5,10               ; Size of rectangle (starting in y-z plane, so y_size=10, z_size=5)
0.5,0.25           ; Sampling density in y and z directions, respectively.
0,0,1              ; Rotate rectangle into x-y plane (normal is along z-axis.)
```

A curved rectangle is a rectangle which is curved in one dimension, that is, a section of a cylinder wall. The specified size of the rectangle in the curved dimension refers to the arc length, not the chord length. The radius of curvature of the rectangle may be non-zero in either the y or z direction, but not in both. A zero radius of curvature means a flat surface. If both radii of curvature are set to zero, then a curved rectangle is identical to a rectangle with the same parameters. An example parameter file section is:

```
curved                  ; Calculate on a curved rectangular section of space.
300,0,0                 ; Origin (center) of curved rectangle
10,5                    ; Size of curved rectangle (y_size=10, z_size=5)
5,0                     ; Curvature of rectangle (y_rad=5, z_rad=0 (flat))
0.5,0.25                ; Sampling density in y and z directions, respectively.
-1,0,0                  ; Region’s normal is in -x direction, so it lies in the yz plane.
```

The pressure field can also be calculated within a planar circle. In this case, the circle is specified identically to the rectangle, with the substitution of its radius for the rectangle’s dimensions. An example:

```
circle                  ; Calculate on a planar circle.
300,0,0                 ; Origin (center) of circle
5                       ; Radius of circle, in cm.
0.5,0.25                ; Sampling density in y and z directions, respectively.
0,1,0                   ; Region’s normal is in y direction, so it lies in the xz plane.
```

Focal Point Determination

The focal point of a lens system is defined as the point of maximum absolute pressure. The focal point can be found along a line or in a plane. The algorithm to find the focal point along a line is much faster than that for a plane, so should be used whenever possible. For searching along a line, the line is specified by its origin and angle in the y and z directions. The search range is given as distances from the origin along the line. If the line is parallel to an axis, then the appropriate angles can be set to zero, or a simplified form can be used. In the latter case, the axis which the line is parallel to is specified along with the fixed coordinates of the line in the other two dimensions. For searching in a plane, the corners of a rectangle within which to search, as well as the plane’s identity, are given.

In all cases, a coarse search is done first over the search range in order to avoid local maxima, so the number of coarse search points must be specified. The maximum value of those found in the coarse search is chosen, then the maxima is refined until the accuracy condition is met. The accuracy is the distance, in cm, between the maxima and the nearest two search points. The algorithm used is a binary search, so the actual accuracy will be slightly higher than that given, depending on where the first subdivision which exceeds the specification lies.

Example parameter file for searching along a line:

```
line_max            ; Find maximum along a line.
300,0,0             ; Origin of line.
25.0,100.0          ; Search range along line (distances from origin).
30,0                ; Angle of line from x-axis (in y and z directions)
31,0.001            ; Number of coarse search points, maxima accuracy
```

Example parameter file for searching along a line which is parallel to an axis:

```
axial_max           ; Find maximum along a line parallel to an axis.
x                   ; Axis which line is parallel to.
2.0,3.5             ; Fixed coordinates (in this case, y and z) of line.
25.0,100.0          ; Search range along line.
31,0.001            ; Number of coarse search points, maxima accuracy
```

Example parameter file for searching in a plane:

```
dual_axial_max      ; Find maximum in a plane.
xy                  ; Use xy plane
5.5,0.0             ; Initial guess for y, fixed value of z.
320,400             ; Search range in x-direction.
2.5,8.5             ; Search range in y-direction.
31,0.001            ; Number of coarse search points, maxima accuracy
```

The planar algorithm finds the maximum parallel to the first axis at a fixed distance from the second axis (given by the initial guess.) Then it uses that point as the fixed distance from the first axis, and searches parallel to the second axis. This is repeated until the given accuracy is achieved. When searching along a ridge, however, this algorithm is very slow to converge. Therefore, when the maximum is known to lie along a certain line, it is much faster to use the line_max or axial_max routines.

Element Types

There are seven element types available in ALSSP. The first is a point element. This is simply a point in space. The next four; line, rectangular, curved rectangular, and circular, are elements whose faces form the appropriate shape. They are calculated by calculating a complex sum of the pressure field over their faces. The final two element types; sinc and jinc, are faster and more accurate ways to calculate the specific cases of a line element in a two dimensional system and a circular element in a three dimensional system. The sinc function sin(x)/x, is the Fourier transform of a uniform line element. Similarly, the jinc function $J_1(x)/x$, is the Fourier transform of a uniform circular element.

The parameter file to describe any calculation ends with the element description. For a point element, this is simply the string "point". For more complicated elements, more description is required.

For a point element:
```
point                   ; Use a point element.
```

For a line element:
```
rect                    ; Use a rectangular element.
1.0,0.0                 ; Element is 1.0 cm high but has zero width.
```

For a rectangular element:
```
rect                    ; Use a rectangular element.
1.0,0.5                 ; Element is 1.0 cm high (y) and 0.5 cm wide (z)
```

For a curved rectangular element:
```
curved                  ; Use a curved rectangular element.
1.0,0.5                 ; Element is 1.0 cm high (y) and 0.5 cm wide (z)
10.0,0.0                ; Radius of curvature of 10 cm in the y-direction.
```

For a circular element:
```
round                   ; Use a round element.
1.5                     ; Element’s radius is 1.5 cm.
```

For a sinc element:
```
sinc                    ; Use the sinc function to simulate a line element.
1.5                     ; Element’s length is 1.5 cm.
```

For a jinc element:

```
jinc                    ; Use the jinc function to simulate a round element.
1.5                     ; Element's diameter is 1.5 cm.
```

Element Pose and Sampling Density

For any element except a point element, the element's orientation and sampling density must also be specified. The element can be oriented in one of two ways: so that it is always facing a specified direction, or so that is always faces a specific point, regardless of the element's position. For each element except a point element, the sinc element, and the jinc element, the number of samples taken on the element's face must also be listed.

Two examples:

```
point                   ; Element will always face the same point.
5000,0,0                ; That point is (5000,0,0).
10                      ; Number of points on element.

direction               ; Element will always face in the same direction.
-1,0,0                  ; That direction is the -x direction.
25                      ; Sample 25 points on the element.
```

UTILITY PROGRAMS

In addition to CALC_LENS, ALSSP contains a variety of utility programs designed to help in the design and analysis of lens systems.

ANNOTATE,*text* [,*/alignment*,*/file*]

Adds text to a plot using the mouse to choose the text location. It can also save the resulting command in a file (for use as a batch file to reproduce or print the plot, for example.) The text string to add to the plot is held in *text*. Alignment is the same as for the XYOUTS command. Namely, the default alignment of 0.0 left-justifies the text at the mouse position. Setting the alignment to 0.5 centers the text on the mouse position, and an alignment of 1.0 right-justifies the text. If */file* is set to a string, then the command is appended to the end of the file of that name.

CALC_POINTS,*filename* [,*/xrange*,*/yrange*]

Calculate the complex pressure at a point in the xy plane specified by the mouse cursor. Prints the x and y coordinates of the point selected by pressing the left mouse button and the magnitude and phase of the pressure at that point. *Filename* is the name of the lens system parameter file. The optional keywords */xrange* and */yrange* describe the region of the ray-trace which will be displayed.

CLOSE_ALL

Closes all open files and frees their LUNs.

CONCAT_RESULTS,*/files*,*/angles*,*/prefix*,*/suffix*,*/outfile*

Concatenates results of multiple source angle pressure calculations into a single beam pattern. Can be used to create an ideal beam pattern to compare to approximate beam patterns. Either */files* or */angles* must be set. */files* should be set to an array of file names which are the CALC_LENS output files to be processed. If */files* is set then */angles, /prefix,* and */suffix* are ignored. If */angles* is set instead of */files,* then it should contain an array of source angles corresponding to CALC_LENS output files. The file names to be processed will be formed by /prefix+sangle+/suffix, where sangle is a string version of the angle (i.e. 10, 1, 3.5, 0.2, etc.) If */prefix* is not set it will default to ’t’ and if */suffix* is not set it will default to ’_20b.out’. The resulting beam pattern will be saved in */outfile,* which defaults to ’concat.out’ if not set. MAKE_BEAMS can be used to generate the multiple source angle pressure calculations.

CURSORP

Prints the location of the mouse cursor on the screen when the left button is pressed. Pressing the right button exits.

DRAW_LINE,*startp,endp* [,keyword parameters]

Draws a line from *startp* to *endp*, which are (x,y) pairs. Available keyword parameters are */noerase, /xrange, /yrange, /psym, /title, /linestyle,* and */color.* All work the same as for PLOT.

FIND_ARRAY_MAX,*array,xlabels,ylabels*

Prints the value and location of the maximum element of an array.

FOCI_CIRC(*points* [,*/origin,/plot,/noerase,/mirror*])

Calculates a piecewise-circular fit to data. Returns the origin and radius of a circle fit to each data point and its two neighbors. The parameter *points* is usually the output of GET_LOG. If */origin* is set, then it is used as the origin of the lens system. Otherwise, the smallest x-coordinate is used. The origin is subtracted out in order to increase the accuracy of the fit. If */plot* is set, then the data points and curve-fit are plotted. If */noerase* is set, then the current plot is not erased. If */mirror* is set, then the data is mirrored over the x-axis first. FOCI_CIRC returns an array which is *points* with four extra columns. The first three extra columns are the x,y,z coordinates of the center of each arc. The final column is the radius of each arc. For example, if *points* was:

```
48.1998 0.00000 0.00000 0.00000 0.0171502
48.1888 0.69622 0.00000 1.00000 0.0171280
48.0898 1.73744 0.00000 2.50000 0.0171480
47.7226 3.45114 0.00000 5.00000 0.0172910
46.3198 6.72023 0.00000 10.0000 0.0177150
37.3790 14.1998 0.00000 25.0000 0.0154940
```

then *foci* would be:

```
48.1998 0.00000 0.00000 0.00000 0.0171502 0.00000 0.00000  0.00000 0.00000
48.1888 0.69622 0.00000 1.00000 0.0171280 37.1873 0.174849 0.00000 11.0138
48.0898 1.73744 0.00000 2.50000 0.0171480 36.1563 0.076738 0.00000 12.0484
47.7226 3.45114 0.00000 5.00000 0.0172910 34.5418 -0.26918 0.00000 13.6957
46.3198 6.72023 0.00000 10.0000 0.0177150 31.9396 -1.38579 0.00000 16.5075
37.3790 14.1998 0.00000 25.0000 0.0154940 0.00000 0.00000  0.00000 0.00000
```

FOCI_CIRC1(*points* [,*/mirror*])

Finds the second principal points for data points. Returns the origin and radius of a circle which is equivalent to the source arc. Thus moving an element along the arc is approximately equivalent to moving a point source by the same angle. The parameter *points* is usually the output of GET_LOG. If */mirror* is set, then the data is mirrored over the x-axis first. The output array *foci* is of the same form as that returned by FOCI_CIRC. For the same *points* array used in the FOCI_CIRC example, *foci* would be:

```
48.1998 0.00000 0.00000 0.00000 0.017150 8.30271 0.00000 0.00000 39.8971
48.1888 0.69622 0.00000 1.00000 0.017128 8.30271 0.00000 0.00000 39.8922
48.0898 1.73744 0.00000 2.50000 0.017148 8.29583 0.00000 0.00000 39.8319
47.7226 3.45114 0.00000 5.00000 0.017291 8.27588 0.00000 0.00000 39.5974
46.3198 6.72023 0.00000 10.0000 0.017715 8.20752 0.00000 0.00000 38.7003
37.3790 14.1998 0.00000 25.0000 0.015494 6.92735 0.00000 0.00000 33.5997
```

GENERATE_SYSTEM,*filename* [,keyword parameters]

The program GENERATE_SYSTEM uses the output of BEAM THREE to generate a lens system. It makes parameter files which describe the system for ray-tracing and also generates parameter files which find the focal point(s) of the system. The parameter *filename* specifies a file which holds the OPTICS table from BEAM3 or BEAM4 or the .ZMX file from ZEMAX. The keyword parameters for GENERATE_SYSTEM are:

/angles	A vector of source angles to set up parameter files for. Defaults to [0.0].
/batchfile	The name of the file to which will be saved the commands required to run the generated parameter files. Defaults to 'batch'.
/box	If set, specifies the area within which to search for the focal points of the system generated.
/distance	The distance from the lens origin (see */origin*), in cm, at which the point source will be placed. Defaults to 500.0 cm.
/freq	The frequency of sound used in the lens system. This is normally set in the header file specified by */sys_header* but can be overridden here.
/head	The prefix for generated parameter files. Defaults to 't'.
/maxfiles	An array of strings containing the names of the template files to be used when creating the parameter files to find the maximum response of the lens system. This allows finding the maximum for several different types of elements. Defaults to ['template/point'].
/mouse	If set, allows the user to select the focal point search area using the mouse while viewing the ray trace plot.
/nomax	If set, will not generate files to find the focal points.
/origin	The origin of the lens system. Usually the center of the first lens surface. Defaults to [0.0,0.0,0.0].
/sys_header	The header file for the lens system. Specifies the water salinity, depth, and attenuation, the frequency of operation, number of rays to trace, and

	the view ranges, as well as the command parameters for CALC_LENS. Defaults to './template/system_header'.
/tail	The suffix for generated parameter files. Defaults to '_'+Temperature.
/Temperature	The ambient temperature at which the system will be operating, in degrees C. Defaults to 20 degrees C.
/verbose	If set, reports on progress as it goes.
/wateratten	The sound attenuation rate of water. Defaults to 0.04 db/cm.

If the output of BEAM THREE was stored in the file 'simple.opt':

```
3 surfaces                    SING128.OPT                Units cm
 Index    Zvx          F DX   DY      CX      Curv        Mir/Ls
--------  :--------    :-:------ :------ :----- :--.------ :------  :
 1.00     : 300.       :S: 4.   : 20.0  : 0.   :-0.026314  ? Lens  :
 0.5789   : 301.5      :S: 4.   : 20.0  : 0.   : 0.031084  ? Lens  :
 1.00     : 345.9191   ?S: 4.   : 30.   : 0.   :           : Film  :
 0.5789 : Syntactic Foam : 0.6983 : 2613.5 : -1.25 : 1.0 : 0.0
```

then GENERATE_SYSTEM would be run by: GENERATE_SYSTEM,'simple.opt'

Alternatively, this same data could be entered as:

```
header=['Index','Zvx','DX','DY','CX','Curv']
data=   [string([1.00,300.0,4.0,20.0,0.0,-0.026314])] $
        ,[string([0.5789,301.5,4.0,20.0,0.0,0.031084])] $
        ,[string([1.00,345.9191,4.0,30.0,0.0,0.0])]]
mdata= ['0.5789 : Syntactic Foam : 0.6983 : 2613.5 : -1.25 : 1.0 : 0.0']
```

and run by: GENERATE_SYSTEM,header,data,mdata

In either case, GENERATE_SYSTEM would generate a file called 't0_20':

```
/logfile            ; Use log file 'batch.log'
/nopressure         ; Do not do pressure calculation, only ray trace.
/trace2d            ; Do ray trace in xy plane only.
/summation          ; Do ray trace for summation method.
/savefile           ; Save results in filename.save
Sal=0               ; Water Salinity, parts per thousand
Depth=1.0           ; Water Depth, meters
freq=900k           ; Frequency, Hz
xrange=[0,65]       ; X range to display
```

```
yrange=[-25,25]        ; Y range to display
num_rays=301           ; Number of rays to trace
waterdensity=1.0       ; Water Density, g/cm^3
wateratten=0.0         ; Water Attenuation Rate, dB/cm
Source=point           ; Source Type
-500.0,0.0,0.0         ; Point Source Location, cm
origin=[0.0,0.0,0.0]   ; Lens System Origin, cm
T=20.0                 ; System Temperature, degrees C
waterspeed=0.0         ; Calculate Speed of Sound in Water
2                      ; Number of Interfaces
aspherical             ; Lens Interface Type
300.0,0,0              ; Interface Vertex, cm
-0.0263140             ; Lens Curvature, 1/cm
1                      ; Lens Shape Factor, 1/cm
20.0000                ; Lens Aperture, cm
2613.5,-1.25           ; Sound Speed Coefficient for Syntactic Foam, m/sec
0.6983                 ; Syntactic Foam Density, g/cm^3
2.5                    ; Syntactic Foam Attenuation Rate, dB/cm
aspherical             ; Lens Interface Type
301.5,0,0              ; Interface Vertex, cm
0.0310840              ; Lens Curvature, 1/cm
1                      ; Lens Shape Factor, 1/cm
20.0000                ; Lens Aperture, cm
0.0                    ; Use Sound Speed for Water
1.00                   ; Water Density, g/cm^3
0.0                    ; Water Attenuation Rate, dB/cm
```

It would also generate a file to find the focal point of the system called 't0_20am':

```
restorefile=t0_20.save     ; Restore ray trace results.
/logfile                   ; Use log file batch.log.
/summation                 ; Use summation method of calculation.
line_max                   ; Find focal point along line in a plane.
0.00,0.00,0.00             ; Origin of line.
340.75,345.0               ; Range along line to search.
0.00,0                     ; Angle of line in xy plane, in xz plane, in degrees.
31,0.05                    ; Number of coarse search points, accuracy of search.
point                      ; Use a point element
```

GET_LOG(*logfile* [,*/verbose*,*/tail*])

Searches a log file for focus information. Returns an array containing the focal positions (in x,y,z triples), the angles corresponding to those focal positions (in degrees), and the magnitude of response at each focal position (in absolute units, not dB). The keyword parameter */tail* specifies the parameter file suffix to search for. It defaults to 'am'. If */verbose* is set, then GET_LOG reports its progress as it runs.

KCLEANPLOT

Resets all plot system variables to their defaults. Identical to CLEANPLOT, but doesn't reset !p.color to 255 (since that messes up Postscript output.)

newstring=KILL_SUFFIX(*oldstring*)

Returns a string without its suffix. See SUFFIX.

outstring=KILL_ZEROS(*instring* [,*/all*])

Deletes all trailing zeros off of a string. If the resulting string ends in a decimal point ('.'), then one zero is added back on, unless */all* is set, in which case the trailing decimal point is deleted. Examples:

```
outstring=KILL_ZEROS('25.00000')          ; outstring is '25.0'
outstring=KILL_ZEROS('25.00000',/ALL)     ; outstring is '25'
outstring=KILL_ZEROS('25.7500')           ; outstring is '25.75'
outstring=KILL_ZEROS('25.000,34.5400')    ; outstring is '25.0,34.54'
outstring=KILL_ZEROS('25.')               ; outstring is '25.'
```

KLEGEND,*labels* [,optional parameters] [,keyword parameters]

Adds a legend to a plot. Any of the parameters (*labels, plotsyms, linethick, xstart, ystart, linelength,* and *yspacing*) may be specified by a keyword parameter of the same name instead of as a parameter. Keyword parameters are:

/labels	An array of strings which are the labels for each legend item.
/plotsyms	An array of the plot symbols for each item. If */plotsyms* is a scalar, then all the plot symbols will be that number. Defaults to an array filled with the system default of !psym (normally a solid line with no marker symbols.)

/symsize	An array of symbol sizes, one for each entry in the legend. If */symsize* is a scalar, then all symbol sizes will be that number. Defaults to an array of 1's.
/linethick	An array of the line thickness indices associated with each item. If */linethick* is a scalar, then all the line thicknesses will be that number. Defaults to an array filled with the system default of !p.thick (normally 1).
/xstart	The x-coordinate of the left-hand side of the legend. See below for default.
/ystart	The y-coordinate of the top of the legend. See below for default.
/linelength	The amount of space allotted for the legend symbols (and possibly lines). Defaults to 1/20th of the plot width.
/yspacing	The distance between consecutive legend items. Defaults to 1/20th of the plot height.
/title	If set, the legend title.
/box	If set, a box is drawn around the legend.

Either *labels* or */labels* must be specified, but KLEGEND will choose defaults for everything else if not given. */xstart* will default to a value which results in the legend being centered in the center of the plot. */ystart* will default to a value which results in the legend ending near the bottom of the plot. The other default values are listed with the parameter descriptions.

KPLOT,*filename* [,keyword parameters]
KPLOT,*positions,values* [,keyword parameters]

The program KPLOT is designed to display most of the data which CALC_LENS generates. It can plot beam patterns, surface plots, contour plots, and display images. The type of plot generated depends on the character of the data. Beam patterns can be displayed in rectangular or polar coordinates (to emulate a paper chart-recorder). In addition, it can find and display the 3 dB points of beam patterns and help the user to find sidelobe positions and measure sidelobe levels. Beam patterns can be displayed in magnitude or decibels, and if desired, KPLOT will simulate two_way beam patterns by squaring the data before plotting it. Many display options are available, including complete control over tick placement, grid styles, overplots, normalization or offsets, and colors.

There are two ways to invoke KPLOT. The first is with the file name of an output file generated by CALC_LENS. If *filename* has no tail (any string following a period) then the tail '.out' is appended. The second way to invoke KPLOT is by specifying the actual data which it is to plot. Two parameters must be given. The first, *positions*, is an Nx3 array holding the xyz coordinates of the data points. The second, *values*, is either an N element vector or an MxP element array (where N=M*P) which holds the actual complex data points generated by CALC_LENS.

In either case, the available keyword parameters are:

/contour	If the data is a matrix, then contour plot it instead of surface plotting it. If set to 2, display an image. If set to 3, display a smoothed image.
/db	Transform the data to dB (20.0*alog10(abs(data))).
/find3db	Find the 3 dB points on a beam pattern. Sets */db* and */normalize*. If */find3db* is set to 2, also finds the sidelobe heights and positions (formerly */sidelobes*).
/force_line	Forces KPLOT to interpret the data as points along a straight line (rather than angles, for example.)
/gridstyle	Line style of grid for plot.
/mirror	Mirror the data over the x-axis (for symmetric data).
/normalize	Normalize the data to magnitude of 1 or 0 dB.
/offset	Amount to offset the data. If set to a scalar, then offset is along the y-axis. If set to a two-element vector, then the first element is the offset along the x-axis and the second is along the y-axis.

/origin	Origin of beam pattern arc.
/shift	Amount to shift the data along the x-axis.
/print	Print the plot rather than displaying it on the screen.
/polar	Plot in polar form instead of rectangular.
/theta_scale	Scaling factor for x-axis values.
/two_way	Transform to two_way beam pattern (i.e. square the data).
/xticki	X tick interval (distance between ticks along x-axis).
/yticki	Y tick interval (distance between ticks along y-axis).

Also available are the standard keywords for plot, contour, and surface: */ax*, */az*, */charsize*, */color*, */c_color*, */nlevels*, */noerase*, */psym*, */symsize*, */thick*, */ticklen*, */verbose*, */title*, */subtitle*, */xtitle*, */ytitle*, */ztitle*, */xrange*, */yrange*, */zrange*, */xstyle*, */ystyle*, */zstyle*, */xticks*, */yticks*, */zticks*, */xtickv*, */ytickv*, */ztickv*, */xminor*, */yminor*, and */zminor*.

The default action for KPLOT depends on the type of data it is operating on. If the data is a vector, then the program assumes that it represents a beam pattern. It then looks at the vector of sampling positions. If this is simply a vector, then those values are assumed to be angles. If the sampling positions are given as points in 3-space, then POS2RETANG is called to convert them into angles (based on an arc centered at */origin*) or points along a line. If the data is an array, then KPLOT will display it as a surface plot, unless */contour* is set, in which case it will be displayed as a contour plot. If */contour* is set to 2, then the data will be displayed as an image. If */contour* is set to 3, then the image will be interpolated to provide a smoother display.

By linking several KPLOT commands together, a contour plot can be overlayed on top of an image:

```
KPLOT,filename,/db,/contour,nlevels=1
KPLOT,filename,contour=3,/noerase
KPLOT,filename,/db,/contour,/noerase
```

The magnitude of the data, which is normally complex, will be displayed unless */two_way* or */db* are set. If */two_way* is set, then the data will be squared to simulate a two-way beam pattern. If */db* is set, then the data will be converted to decibels, 20*alog10(abs(data)). If both */two_way* and */db* are set, then the data will be squared, then converted to decibels.

If */normalize* is set, then the data will be normalized. If */db* is not set, then the maximum magnitude will be set to 1. If */db* is set, then the maximum value will be set to 0 dB.

If */offset* is set, then the data will be shifted by the value of */offset*. This is useful for comparing the attenuation displayed by various beam patterns. */normalize* takes precedence over */offset*, so if */normalize* is set, then any value of */offset* will be ignored.

If */theta_scale* is set, then the theta (or x) axis will be scaled by */theta_scale*.

If */shift* is set, then the entire plot will be shifted along the x-axis by */shift*.

If */mirror* is set, then the data will be mirrored so that data from 0 to max will be extended to the range -max to max. For symmetric situations, this can halve the amount of computations required to be performed when calculating the beam pattern or pressure field.

/xticki and */yticki* set the x and y tick intervals. This is useful for forcing the y tick intervals to 3 dB, for example. Normally, the x ticks will start with the minimum value of x (i.e. the left side of the plot) and the y ticks will start with the maximum value of y (i.e. the top of the plot). This can be changed, however, by setting */xticki* or */yticki* to a two element vector, where the first element is the tick interval and the second is the starting point.

If */gridstyle* is set, then a grid will be displayed on the beam pattern plot. The line style of the grid will be set to */gridstyle*. Grid lines will be drawn at the positions of the major ticks along both axes. If the tick positions aren't defined in any way, then KPLOT will attempt to choose "good" values for them, rather than let PV-WAVE do it. This is because there is no way to tell what values PV-WAVE has chosen, so the grid lines don’t match up with the ticks.

If the data is to be displayed on a contour plot, then the number of contour levels can be set by */nlevels*.

If */print* is set, then the resulting plot will be sent to the printer instead of to the screen. Note that the plot is sent before the location of the 3 dB points are indicated (see */find3db*) or sidelobes are located (see */sidelobes*), so the printed plot will not show those extra lines. If a plot with those lines is desired, then set */print* to 2 instead of 1 (i.e. print=2 instead of print=1 or /print). Also, for a screen plot without the extra lines, set */print* to -1.

If */xtitle* is not set, then it defaults to an indication of the type returned by POS2RETANG. If the beam pattern was based on positions along a line parallel to the x-axis, for example, */xtitle* will be set to ’Axial distance (x-direction), cm’. If */theta_scale* was set, then the string ’ (Scaled by: ’ and the amount of scaling will be added to */xtitle*.

If */ytitle* is not set, then it defaults to an indication of the data type. It begins with nmtitle, where nmtitle is ’Normalized’ if */normalize* was set, ’(Relative)’ if */offset* was set, and ’’ if neither was set. To nmtitle is added ’Acoustic Pressure’, then dbtitle, where dbtitle is ’ (dB) (20log(abs))’ if */db* was set (or ’ (dB) (20log(abs^2))’ if */two_way* was set). If */db* was not set, then dbtitle is set to ’(abs)’ (or ’(abs^2)’ if */two_way* was set).

If */polar* is set, then the plot will be made in polar coordinates. A few options work differently when */polar* is set. The */gridstyle* and */sidelobe* options are not available. A grid will always be drawn with */xticks* grid lines spaced */xticki* degrees apart in the theta direction. The grid in the magnitude (y) direction will be the same as if */polar* was not set (although the grid lines will be curves instead of straight lines.) The default value for */xticki* will be 1 degree and for */yticki* will be 3 dB.

If */find3db* is set, then the program will attempt to locate the 3 dB points, display them on the plot, and print the 3 dB beamwidth (the distance between the 3 dB points). This option also sets */db* and */normalize*. In addition, */dec* can be set to fix the number of decimal points that will be displayed for the 3 dB beam width (the distance between the 3 dB points.)

If */sidelobes* is set, then the user will be able to find sidelobes or other points of interest using the mouse. Hitting the left button will choose a point for one side of a search range. Hitting it again on another point will complete the search range, whereupon the program will print the maximum value within that range and the position of that maximum. Hitting the center button will choose whichever data point is closest to the cursor position and will print the value and position of that point. Hitting the right button will exit the program.

LASER [,*/square,/encapsulated*]

Changes the output device to postscript. The default output size is 7x5 inches in portrait mode, with the plot positioned near the top of the page. If */square* is set, then the output size is 7x6.729 inches in order to ensure a square aspect ratio. If */encapsulated* is set, then an encapsulated postscript file is created (for inclusion into other documents.)

LASER_SEND [,*/printer*]

After plotting a figure with the output device set to postscript, laser_send will send it to the printer. If */printer* is not specified, then laser_send will attempt to get the default printer name from the environment variable PRINTER. If PRINTER isn't set either, then 'lp428' will be used as the default. After sending the plot, LASER_SEND will reset the output device to 'x'.

MAKE_BEAMS,*angs* [,*/batchfile,/position,/head,/tail,/elementfile,/wateratten*]

Generates parameter files which will calculate the response of an element at a single point. The results of these calculations can be concatenated together using

CONCAT_RESULTS to form beam patterns. *angs* is an array of source angles which are to be calculated. The optional keyword parameters are:

/batchfile	The name of the file which will hold the batch instructions for running the generated parameter files. Defaults to 'batchb'.
/position	The xyz coordinates of the position of the element. Defaults to [5034.65,16.1582,0.0].
/head	The prefix for the output parameter files. Defaults to 't'.
/tail	The suffix for the output parameter files. Defaults to 'a'.
/elementfile	The template file for the element type. Defaults to '/home/kfink/work/template/point'.
/wateratten	The sound attenuation rate of water. Defaults to 0.04 db/cm.

MAKE_BEAMS1,*angs,width,focifile,origin* [,keyword parameters]

Generates parameter files which will calculate the beam pattern for an element by moving the element along the focal trajectory. *angs* is an array of source angles which are to be calculated. *width* is the range of angles in degrees to calculate for each source angle. *focifile* is the filename of the trajectory file generated by FOCI_CIRC. *origin* is the origin of the lens system. Optional keyword parameters are:

/batchfile	The name of the file which will hold the batch instructions for running the generated parameter files. Defaults to 'batch'.
/head	The prefix for the output parameter files. Defaults to 't'.
/tail	The suffix for the output parameter files. Defaults to 'a'.
/elementfile	The template file for the element type. Defaults to 'template/point'.
/numpos	The number of points to calculate for each source angle. Defaults to 101.

MAKE_BEAMS2,*origin*,*radius*,*angles*,*rayfile* [,keyword parameters]

Creates parameter files to calculate beam patterns along an arc. The origin of the arc is specified by *origin*, and may be a 3-element vector, a scalar, an nx1 ($n \neq 3$) vector, or an nx3 vector. In the first case, the origin is used as-is. In the second case, the origin is expanded to [origin,0.0,0.0]. In the third case, each element is expanded as in the second case. In the fourth case (and the third, after expansion), multiple parameter files are generated, one for each 3-element vector.

Radius may be a scalar, in which case it is used as the radius of curvature of the arc for all origin values. It may also be a vector, in which case each element generates a separate parameter file. If both the origin and radius represent multiple files, then one of two things can happen. If each represents the same number of files, then it is assumed that they are paired. Otherwise, a parameter file is generated for each combination of radius and origin.

Each arc generated must have the same endpoints, which are specified as *angles* in degrees. The saved ray-trace data is specified in *rayfile*. *Rayfile*, without the (optional) trailing '.save', is also used as the base for the parameter file names. Onto the base is added */tail* then consecutive numbers starting with */start*.

Optional keyword parameters are:

/batchfile	The name of the file which will hold the batch instructions for running the generated parameter files. Defaults to 'batch'.
/head	The prefix for the output parameter files. Defaults to 't'.
/tail	The suffix for the output parameter files. Defaults to 'a'.
/start	The starting number for the output parameter files. Defaults to 1.
/elementfile	The template file for the element. This file is concatenated onto the end of each generated parameter file. Defaults to '/home/kfink/work/template/point'.
/numpos	The number of points to calculate for each beam pattern. Defaults to 101.

MAKE_LINE_MAX,*filename* [,*/savefile,/plot,/afile,/tail,/batchfile*]

Generates a parameter file which will find the maximum absolute pressure of a lens system along a line. *Filename* is the file name of the lens system parameter file. If */savefile* is not set to a string, then it will be set to *filename* with '.save' appended to the end, which matches the default save file name generated by CALC_LENS. */Savefile* is used as the name of the PV-WAVE data file which will be restored to provide information about the ray trace results. The parameter file will be saved in *filename+/tail*, where */tail* defaults to 'am'. The template file for the element is */afile,* which defaults to '/home/kfink/work/template/point'. If */plot* is set, then the line which will be searched for the maximum will be displayed. The command to run the parameter file from PV-WAVE will be saved in the file */batchfile,* which defaults to 'batch'.

MAKE_MAX_REGION,*file1* [,*file2,/overwrite,/batchfile,/box*]

This program generates a CALC_LENS parameter file which will find the maximum pressure within a region. The region can be selected by setting the keyword parameter */box* to the coordinates of the region's lower-left and upper-right corners, or by using the mouse to select an area of the plot (see MOUSEBOX). If called with one parameter, that parameter is used as the output file name and the current plot is used for input. If called with two parameters, then the first is a CALC_LENS system parameter filename which is used to display the ray trace plot and the second is the output file name. If */overwrite* is set, then the output file will be automatically overwritten, if it exists. Otherwise, the program will query the user if the output file already exists. The instructions for MOUSEBOX describe how to choose the region. The command to run the parameter file from PV-WAVE will be saved in the file */batchfile,* which defaults to 'batch'.

MAKE_VOLUME,*file1* [,*file2,/xpts,/ypts,/overwrite,/dec,/batchfile*]

This program works identically to MAKE_MAX_REGION, but generates a parameter file which will calculate an array of pressures within the region. The keyword parameters */xpts* and */ypts* are the number of points in each direction to specify in the parameter file. The default values are both 25, for a total of 625 points (this takes approximately 1 minute, 40 seconds to compute). The keyword parameter */overwrite* forces MAKE_VOLUME to overwrite *file2* if it exists. Otherwise, MAKE_VOLUME will ask the user whether or not to erase the old file. */dec* specifies the number of decimal points to round the coordinates of the selected region's boundaries to. MAKE_VOLUME saves the command needed to execute the parameter file it creates in the batch file specified by */batchfile,* which defaults to 'batch'.

MAN,*program_name*

This program gives a brief summary of a PV-WAVE program, including the header line, which shows the parameters and keyword parameters. It displays any comment lines (those starting with a semi-colon) before the header line, plus any comments which immediately follow the header line. Example: MAN,'calc_lens'

point=MOUSEBOX([*file*])

This program allows the user to select a region of a plot using the mouse. It returns the lower-left and upper-right coordinates of the region. If *file* is not specified, then the current screen plot is used. If *file* is given as a CALC_LENS system parameter file, then CALC_LENS will be called to display the ray trace for it. This also enables the zoom features of mousebox.

Pressing the left mouse button fixes the first corner of the box. The box will be drawn from that point to the current mouse position until the left button is released, which fixes the opposite corner of the box. This can be repeated until the box covers the desired region. Pressing the right button chooses the current box, returning the coordinates. Zooming: If *file* is given mousebox can redraw the display to any scale. With the cursor in the plot window, pressing the center button will zoom the display in to the selected region. With the cursor in the PV-WAVE window, there are several options. Pressing the ’r’ key will redraw the plot at the present scale. Pressing the ’0’ key will reset the plot to the original scale. Pressing the ’o’ key will zoom out by 50%. Pressing the ’i’ key will zoom in by 50%. Pressing a number key from 1 to 9 will zoom out by 10 to 90%.

angles=POS2RETANG(*positions* ,[*/origin,/xy*])

Returns the angles or positions associated with the points in 3-space held in *positions*. The final element of *angles* holds the type of conversion performed to form *angles* from *positions*. If *positions* is a vector, then it is returned with a final element of 0 (no conversion) added. If */xy* is then POS2RETANG assumes that the first two columns of *positions* contain the x and y coordinates of data points on an arc, and the type is set to 6. If */force_line* is set, then POS2RETANG returns the data in spatial coordinates (as opposed to angles). Otherwise, if two columns are constant, then the third is returned with type 1, 2, or 3, depending on whether that column was the first, second, or third. If only one column is constant, then the other two are assumed to be data points on an arc, and the type is set to 4, 5, or 6, depending on which column was constant (x, y, or z, respectively).

If POS2RETANG decides that the data corresponds to points on an arc (or */xy* was set), then the data is converted to angles in degrees. The arc is centered at */origin*, which defaults to [0,0,0].

outvector=RANGE(*min,max,num*)

Returns a vector of *num* elements evenly spaced from *min* to *max.* If *num* is 1, then the average of *min* and *max* is returned.

answer=SIGN(*value*)

Returns the sign (+/- 1) of *value*.

outstring=SPLIT_STRING(*instring* [,*/separator*])

Breaks *instring* up into an array of strings. The input string is broken at each occurrence of */separator*, which defaults to ','. The */separator* string is not included in the output array.

answer=STACK(*in0* [,*in1,in2,in3,in4,in5,in6,in7,in8,in9*])

Returns concatenation of input vectors "stacked" next to each other. Each input vector forms a column in the output array. Up to 10 vectors can be stacked per call.

array=STRING2ARRAY(*instring* [,*/len*])

Break a string up into an array of floating point numbers. Discards comments at the end of the line and any square brackets (' [' and ']'), then converts the remaining comma or space-delimited string into floats. If */len* is set, then the resulting array always has */len* elements. If the input string was too long, then the output array is truncated. If it was too short, then the output array is padded with zeros.

out=STRPOSS(*string,object* [,*/pos*])

Similar to STRPOS (see PV-WAVE manual), but finds ALL occurrences of the object string rather than just the first. Returns an array containing the indices of each occurrence of object in string starting at position */pos.* If */pos* is not specified, it defaults to 0 (the first character of the string.)

outstring=SUFFIX(*instring*)

Returns the suffix of *instring* (usually a file name.) The suffix is defined as that part which lies after the final period ('.') in a string.

tabstring=TAB(*number*)

Returns a string containing *number* tab characters.

tabstring=TAB(*string*)

Returns a string containing 3-strlen(*string*)/8 tabs. Used to create parameter files in which the beginnings of the comments line up at 3 tabs over.

outstring=UNQUOTE(*instring*)

Removes single and double quotes (’ and ") from *instring*.

phase=UNWRAP(*data* [,*/tolerance*])

Performs a phase unwrapping operation on a vector. *Data* should be a complex-valued vector. Returns the unwrapped phase. */Tolerance*, which defaults to 0.8, determines how large a phase jump must be to be considered wrap-around.

phase=UNWRAP2D(*data* [,*/tolerance*)]

Same as unwrap, but works on an complex-valued array instead of a vector.

answer=VEQ(*vec1*,*vec2*)

Returns 1 if two vectors are equal. If one vector is a scalar, then returns 1 if every element of the other vector is equal to that number.

len=VLEN(*vector*)

Returns Euclidean length of a vector (the dot product of the vector with itself.)

WIN,*num* [,keyword parameters]

Simple way to open windows. The parameter *num* can be either 0 or 1. Window 0 is in the lower-right-hand corner of the screen and window 1 is in the upper-right-hand corner. Keyword parameters:

/square	Makes window square instead of rectangular.

/xsize	Number of pixels in x-direction. Default is 565.
/ysize	Number of pixels in y-direction. Default is 454.

X

Resets the output device to ’x’ after closing the Postscript device.

TROUBLESHOOTING GUIDE

Sometimes dual_axial_max will "lose" a peak. This may happen when the search range includes several local maxima and the initial guess happens to hit a local maxima instead of a global maxima. In order to exclude the local maxima from the search range, the pressure field in that area can be calculated and displayed:

MAKE_VOLUME,*system_file*,*output_file*
CALC_LENS,*output_file*
KPLOT,*outputfile*,*/contour*,*/db*

Then the search range in the maximum-finding parameter file can be refined appropriately. It takes about 2 minutes to generate the pressure field for the default sampling of MAKE_VOLUME.

Another way to avoid the lost peak problem is to change the initial guess in the y-direction. The situation which usually causes lost peaks is when the pressure field looks like:

```
          x
                X
          x
```

where the small x’s are local maxima and the large X is the global maxima. If the y guess is in line with one of the little x’s instead of the large X, the routine will get stuck on the little x’s and jump back and forth between the x’s until it gives up and "loses" the peak. Changing the y guess to a position in between the two x’s will solve the problem.

A third way to avoid this situation is to search in the y direction first instead of second (by specifying yx instead of xy as the search type in the parameter file.) GENERATE_SYSTEM uses this approach. In addition, it sets the initial guess in the x direction to a point 2/3 of the way from the left side of the search range in order to avoid the local maxima shown above.

INDEX