ARSRP Signal Processing Software

Lisa A. Pflug
Jerald W. Caruthers
Richard R. Slater
Ocean Acoustics Branch
Center for Environmental Acoustics Division

February 1993

Approved for public release; distribution is unlimited.
**Title and Subtitle.**

ARSRP Signal Processing Software

**Authors.**

Lisa Pflug, J. W. Caruthers, and R. R. Slater

**Performing Organization.**

Naval Research Laboratory – Detachment Center for Environmental Acoustics

Stennis Space Center, MS 39529-5004

**Sponsoring/Monitoring Agency.**

Office of Naval Research

800 N. Quincy Street

Arlington, VA 22217-5000

**Supplementary Notes.**

Approved for public release; distribution is unlimited. Naval Research Laboratory, Washington, DC 20376-5320.

**Abstract (Maximum 200 words).**

Software written in PV-WAVE to process data collected aboard the RV *Cory Chouest* during the Acoustic Reverberation Special Research Program Reconnaissance (ARSRP) Experiment conducted in the summer of 1991 is documented in this memorandum report. Although developed specifically for ARSRP-collected data, the software will work on any data collected aboard the RV *Cory Chouest* and written to 9-track tape for processing with the Off Line Processor (OLP) software in the "beam tape" (BTS) format — this includes beamformed and hydrophone pass-through data. Software written by the Scripps Institution of Oceanography to read OLP tapes and output files in Society of Exploration Geophysicists 'Y' (SEGY) format is discussed. ARSRP Signal Processing Software (ASPS) discussed in the report includes reading SEGY format archive tapes into ASCII format, conversion of ASCII data files to unformatted form for compact storage and fast access by PV-WAVE, and beamforming and matched filtering with PV-WAVE programs. A program to create waterfall display plots of hydrophone data, beamformed data, or matched filtered data is included. A printout of each program is given in an appendix. Since this software is intended to reproduce real-time data processing created by the Monitoring Support Software (MSS) aboard the *Cory*, some of the real-time data displays are shown for comparison. Since there is only a limited capability of ARSRP investigators to access the OLP tapes in their standard formats, high-priority segments of the ARSRP reconnaissance data has been converted to exabyte tapes in SEGY format for routine access with UNIX equipment using ASPS or other software. A listing of available exabyte tapes is given in an appendix. For ARSRP program use, a limited number of these tapes may be obtained from the authors.

**Subject Terms.**

- Bottom Scattering
- Reverberation

**Security Classification.**

- Unclassified

**Distribution/Availability Statement.**

Approved for public release; distribution is unlimited. Naval Research Laboratory, Washington, DC 20376-5320.
CONTENTS

I. INTRODUCTION 1

II. ORIGINAL DATA FORMATS 1

III. PROGRAM SUMMARY 3

IV. CONVERSION FROM SEGY TO ASCII 4

V. CONVERSION FROM ASCII TO BINARY 6
   Data Containing Real and Imaginary Parts 6
   Data Containing Beam Number, Time Sample, Real and Imaginary Parts 7

VI. READING DATA INTO PV-WAVE 8

VII. BEAMFORMING 10

VIII. WAVEFORMS 13
   HFM Upsweep 13
   CW Tonals 14

IX. MATCHED FILTERING 16

X. WATERFALL PLOTS 17

XI. STORING OUTPUT 25
   Storing Output in Binary 25
   Storing Output in ASCII 25

XII. PROCESSING CW DATA 26

ACKNOWLEDGMENTS 32

REFERENCES 32

APPENDICES 33
   Appendix A - Reading OLP BBN formatted tapes 33
   Appendix B - OLP and SEGY tapes held by NRL/SSC 37
   Appendix C - Program Code 41
ARSRP Signal Processing Software

I. INTRODUCTION

Software written in PV-WAVE, Version 3.0, is used to process Acoustic Reverberation Special Research Program (ARSRP) Reconnaissance Experiment data collected by the RV Cory Chouest during the summer of 1991. This software is referred to as the ARSRP Signal Processing Software (ASPS) and was written to reproduce the real-time processing done during the experiment using a software system aboard the Cory referred to as the Monitoring Support Software (MSS). ARSRP Signal Processing Software (ASPS) discussed in this report includes reading SEGY format archive tapes into ASCII format, conversion of ASCII data files to unformatted form for compact storage and fast access by PV-WAVE, and beamforming and matched filtering with PV-WAVE programs. A program to create waterfall display plots of hydrophone data, beamformed data, or matched filtered data is included. Information about the experiment is documented in the Acoustic Reverberation Special Research Program Initial Report of 19 August 1991¹, and displays of the real-time processed data can be found in the Acoustic Reverberation Special Research Program Monitoring Support Software Supplement of 7 April 1992².

Programs and data discussed in this report are available as follows:

<table>
<thead>
<tr>
<th>Programs and data</th>
<th>Jerald W. Caruthers (601) 688-5438</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>telnet - <a href="mailto:jwc@milo.noarl.navy.mil">jwc@milo.noarl.navy.mil</a></td>
</tr>
<tr>
<td>PV-WAVE programs</td>
<td>Lisa A. Pflug (601) 688-5574</td>
</tr>
<tr>
<td></td>
<td>telnet - <a href="mailto:pflug@pfester.noarl.navy.mil">pflug@pfester.noarl.navy.mil</a></td>
</tr>
<tr>
<td>Modified SIO C prg.</td>
<td>Richard R. Slater (601) 688-5472</td>
</tr>
<tr>
<td></td>
<td>telnet - <a href="mailto:slater@gandalf.noarl.navy.mil">slater@gandalf.noarl.navy.mil</a></td>
</tr>
</tbody>
</table>

II. ORIGINAL DATA FORMATS

The original real-time data recording of the full waveforms for each wavetrain is done on HDDR tapes aboard the Cory. An original pair of 9-track tapes containing complex demodulated signals is written during Real-Time Processing (RTP). These two 9-tracks are referred to as "beamformed tapes" (BF or BTS) and "matched filter tapes" (MF or MFT) and each is in a format referred to in ref. 1 as "Format A," which is half (62) hydrophones (called "pass throughs"), half (64) beams, and 2
desensitized phones. The HDDR and original 9-track tapes are at the Naval Research Laboratory archives in Washington, D.C. (NRL/DC). Several other institutions, including Naval Research Laboratory at Stennis Space Center, MS (NRL/SSC), hold some duplicates of the Format A archive tapes held by NRL/DC and also 9-track tapes written from the original HDDRs in other formats, i.e., format B - all beams (126) and format C - all phones (126) (each include 2 desensitized phones). Additional details relevant to the tape setups may be found in ref. 1, RTP SETUP pp 149-151 or Section 8, "Real Time Processing System" pp 216-227. Details on what institutions hold what tapes in which formats can be found in ref. 1, pp 183 to 186.

All 9-track tapes are in a special VAX format (developed by BBN, Inc.) written for a system called the Off-Line Processor (OLP, ref. 1, pp 235-241). Only NRL/DC and NRL/SSC have the OLP hardware/software to read the 9-tracks. ARSRP investigators at Scripps Institution of Oceanography (SIO) have developed the software to use standard UNIX equipment to read the tapes and produce a Society of Exploration Geophysicists "Y" (SEGY, pronounced "seg - Y") formatted exabyte tape. (Appendix A includes a brief description of the SIO software in the form of an OMNET message posted by John Orcutt in March 1992.) We have adopted SIO software, with modifications, for our routine processing in lieu of the dedicated OLP system. Although the primary subject of this report is the development and application of additional software for beamforming and matched filtering, a brief section on reading the original OLP 9-track tapes and putting into SEGY format on exabyte and then reading the SEGY tapes as we have it implemented at NRL/SSC is included in Appendix A.

NRL/SSC holds over one hundred 9-track OLP tapes. A selected set of the tapes have been converted to SEGY format and written to exabyte tapes. Listings of 9-track and exabyte tapes held by NRL/SSC is given in Appendix B. NRL/DC is the archival depot for all 9-track tapes in the A format and all HDDR tapes.

As mentioned previously, we have implemented our beamformer and matched filter in PV-WAVE coding and, since PV-WAVE does not read SEGY format directly, we have an additional step of converting to ASCII when reading in the exabyte tapes. Finally, although some of the original 9-track data tapes contain half beamformed data and half hydrophone data, the processing discussed in this document applies to all hydrophone or all beam
data. The beamforming program default uses 126 phones but can easily be modified to beamform 62 channels (BTS format A). Note that by default, channel numbers (hydrophone or beam) begin with 0, and that all data is complex due to complex demodulation (basebanding) performed during data collection. Data processed with this software matches the Monitoring Support Software (MSS) plots, although the choice of default angles for beamforming are different, as will be discussed later.

III. PROGRAM SUMMARY

The following table lists the existing software in ASPS with brief descriptions of their functions. Each program is discussed in a separate section following the overview, and the code for all but bbn2segty.c (i.e., the SIO software) is included in Appendix C. All programs ending with ".pro" are PV-WAVE programs. Examples of interactive sessions (italics indicate user input) and figures are included in this document.

<table>
<thead>
<tr>
<th>Program</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>bbn2segty.c</td>
<td>Scripps program (modified) to read the 9-track BBN formatted tapes and convert to SEGY.</td>
</tr>
<tr>
<td>segy2ascii.c</td>
<td>'C' program that converts SEGY data on archive tape to ASCII format.</td>
</tr>
<tr>
<td>pvconvert2arr.pro</td>
<td>Converts one ASCII file containing 128 sequential channels of data of the form (real part, imaginary part) to unformatted output.</td>
</tr>
<tr>
<td>pvconvert4arr.pro</td>
<td>Second version of pvconvert2arr.pro which converts one ASCII file containing 128 sequential channels of data of the form (channel number, time sample, real part, imaginary part) to unformatted output.</td>
</tr>
<tr>
<td>readdata.pro</td>
<td>Opens and reads the unformatted data created by pvconvert2arr.pro or pvconvert4arr.pro into PV-WAVE variables.</td>
</tr>
</tbody>
</table>
beamform.pro Beamforms phone data at a default or user-defined set of angles.

hfmsource.pro Creates a 210 - 280 Hz HFM upsweep source function for use in match filtering data (waveform ID SPSS053).

cwtonal.pro Creates one of nine 2-second CW tonals for use in match filtering data.

mfilt.pro Match filters beamformed data with a user supplied source.

waterfall.pro Creates a waterfall plot of amplitudes for channel (phone, beamformed, or match filtered data) versus time.

unformatoutput.pro Stores data in a binary file which can later be read into PV-WAVE variables by readdata.pro.

asciioutput.pro Stores data in an ASCII file in one of two formats which can be later be converted to unformatted data by pvconvert2arr.pro or pvconvert4arr.pro.

The programs hfmsource.pro and cwtonal.pro are included in the software to reproduce the wavetrains WTRP001 and WTRP008, respectively (p 124, ref. 1). Like the data, the wavetrains are basebanded and sampled at 128 samples/second.

Figure 1 contains a reference flow chart of the ASPS programs included in this list.

IV. CONVERSION FROM SEGY TO ASCII

Program Name: segy2ascii.c
Syntax: % segy2ascii

This 'C' program converts data in SEGY format to ASCII format. The SEGY tapes contain data separated into records. The first record contains 66.5 seconds of data and the remaining records contain 59.5 seconds of data. Except for the CW exabyte tapes, the exabyte archive tapes at NRL/SSC contain
Figure 1. Flow chart of ASPS programs.
approximately 145 seconds (less than two records) of data for one ping (the HFM upsweep) from Run 5. The program allows the user to choose the number of records to extract.

NOTE: For each of the following examples the number distinguishes different processing sequences of different data sets and the letter between different processing sequences within the same data set.

Example 1(a). Remove first record of data from tape btsc218_0948.392b (phone data for ping 190 of Run 5). To do this, load the appropriate 8mm tape. The command on acoustics (NRL/SSC specific) (Silicon Graphics) to copy the file is:

```
% dd if=/dev/mt/tps0d6v of=my-dir-name/btsc218_0948.392b conv=swab bs=2048
```

where my-dir-name is the full path name where the file is to be copied. (Other organizations would make appropriate changes.)

Now run segy2ascii.c.

```
% segy2ascii
Enter input file name: btsc218_0948.392b
Enter output file name: ascii btsc218_0948.392b
Enter starting record number [1-N]: 1
Enter ending record number [1-N]: 1
Writing channel 1, record 1
.
.
Writing channel 128, record 128
```

V. CONVERSION FROM ASCII TO BINARY

Data Containing Real and Imaginary Parts

Program Name: pvconvert2arr.pro
Syntax: WAVE>pvconvert2arr,<infile>,<outfile>
Variables: infile - name of input file containing ASCII data
outfile - name of output file containing unformatted data

The program pvconvert2arr.pro reads in an ASCII input file (created by segy2ascii.c) which contains the phone or beam data to be processed in sequential order. The file contains two arrays: real part of data and imaginary part of data. There are no headers in the file. The user must know the number of
points present in each channel which is \((8512 + 7616k)\) points, where \(k\) is the number of records minus 1, if the first record is included. If the first channel is not included, the number of points is 7616k. The output of this program is simply the input data in unformatted form and usually requires approximately half the storage space of the ASCII file. However, the main purpose of this program is to store the data in a format which can be accessed quickly by PV-WAVE for a processing session. The first line of the output file contains the channel number zero and the number of sample points in that channel, both integers. The second line contains the time array, the third line the real part of the data, and the fourth line, the imaginary part of the data, all for channel zero. The next line contains the channel number 1 and the number of sample points in that channel (number of points per channel should be constant), etc. This form is read automatically by the program readdata.pro. The conversion from ASCII to unformatted output needs to be done only once, and readdata.pro used to access the data thereafter. The programs signals each time a channel is read from the input file and written to the output file.

Example 1(b). Convert ASCII file ascii_btsc218_0948.392b to unformatted form and store in unf_btsc218_0948.392b. Number phones from 0 to 127.

WAVE>pvconvert2arr,'ascii_btsc218_0948.392b','unf_btsc218_0948.392b'
Enter start phone/beam number in file (usually 0 or 1).
: 0
Enter number of points per phone/beam.
: 8512
Channel number converted: 0
Channel number converted: 1
.
.
Channel number converted: 127
Unformatted output in unf_btsc218_0948.392b

Data Containing Beam Number, Time Sample, and Real and Imaginary Parts

Program Name: pvconvert4arr.pro
Syntax: WAVE>pvconvert4arr,<infile>,<outfile>
Variables:     infile – name of input file containing ASCII data
               outfile – name of output file containing unformatted data

This program is a second version of pvconvert2arr.pro with the major difference being that the input file contains four arrays instead of two: channel number, time sample, real part of data, imaginary part of data. The program segy2ascii.c can be modified to produce ASCII output in this format. In this case, a sample input file containing three channels of four data points each, sampled at time increment .5 should be in the form (without headers):

<table>
<thead>
<tr>
<th>BEAM NUMBER</th>
<th>TIME SAMPLES</th>
<th>REAL PART</th>
<th>IMAGINARY PART</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0</td>
<td>3.5</td>
<td>1.0</td>
</tr>
<tr>
<td>0</td>
<td>0.5</td>
<td>2.1</td>
<td>-0.2</td>
</tr>
<tr>
<td>0</td>
<td>1.0</td>
<td>-0.2</td>
<td>1.1</td>
</tr>
<tr>
<td>0</td>
<td>1.5</td>
<td>0.4</td>
<td>0.2</td>
</tr>
<tr>
<td>1</td>
<td>0.0</td>
<td>1.3</td>
<td>1.2</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>-0.3</td>
<td>1.2</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>-0.2</td>
<td>1.3</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>-0.1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.0</td>
<td>3.2</td>
<td>2.0</td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td>1.1</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>-0.2</td>
<td>0.0</td>
</tr>
<tr>
<td>2</td>
<td>1.5</td>
<td>-0.4</td>
<td>0.1</td>
</tr>
</tbody>
</table>

The ARSRP data is sampled at 128 samples/second. Thus, given the initial array size of 2100000 in the code (see Appendix C), a maximum of 128 seconds of data for 128 beams may be converted and stored if pvconvert4arr.pro is used.

VI. READING DATA INTO PV-WAVE

Program Name: readdata.pro
Syntax:      WAVE>readdata,timedata,realdata,imagdata
Variables:   timedata – one-dimensional array of time samples (output)
               realdata – two-dimensional array of real part of data for each channel (output)
imagdata - two-dimensional array of imaginary part of data for each channel (output)

The file created by pvconvert2arr.pro or pvconvert4arr.pro are read into a current PV-WAVE session by using readdata.pro where the arrays realdata and imagdata contain the real part of the data and the imaginary part of the data for each beam, respectively, and the array timedata contains the time domain samples over which each beam is defined. The first dimension contains the time sample, real part, or imaginary part, and the second dimension contains the channel number (the channel number corresponds directly to the array position since PV-WAVE arrays use zero as the starting position). A message giving the channel number and number of sample points is printed to the screen after each channel is read. The user is given the option of keeping a subset of the time samples and/or a subset of the channels. The first channel in the subset is assigned to the zero position in the array, however, when plotting, the user is prompted for the first and last channel numbers in the subset for accurate plot labels. The program prints the channel number and number of points to the screen after each channel is read.

Example 1(c). Read file unf_btsc218_0948.392b into PV-WAVE variables. Keep all the phones with 8192 points each to use in beamforming program.

WAVE>readdata,timedata,realdata,imagdata
Enter input file name.
 : unf_btsc218_0948.392b
Enter number of channels in file.
 : 128
Number of channels read = 128.0
Enter beginning and ending channel numbers to keep
 - possible 0 to 127
 : 0 127
Number of channels to keep = 128.0
Number of points per channel - 8512
Enter number of points per channel to keep.
 : 8192
Data read for channel number - 0
Number of points per channel - 8512
Data read for channel number - 1
Number of points per channel - 8512.
Data read for channel number - 127
Number of points per channel - 8512

VII. BEAMFORMING

Program Name: beamform.pro
Syntax: WAVE>beamform, timedata, realdata, imagdata, angledata, realbeam, imagbeam
Variables: timedata - one-dimensional array of time samples (input)
realdata - two-dimensional array of real part of phone data for each channel (input)
imagdata - two-dimensional array of imaginary part of phone data for each channel (input)
angledata - one-dimensional array of beamforming angles (output)
realbeam - two-dimensional array of real part of beamformed data for each angle (output)
imagbeam - two-dimensional array of imaginary part of beamformed data for each angle (output)

Given 128 channels of hydrophone phone data, and a default set of 126 angles or a user-defined set of an arbitrary number of angles, beamform.pro creates a beam for each angle. For a user-defined set of angles, the user must supply the program with the total number of angles, the starting angle in degrees, and the angle increment in degrees. The program writes the array of angles, whether user-defined or default before processing. At this point, the user should interrupt the program if the array of angles has been entered incorrectly. The default set of angles used in the ASPS software does not match the set of angles used by the MSS. It is unclear how the angles given in references (1) and (2) were calculated.

The beamformer is a multiple frequency version of a time-delay beamformer such as that described by Burdic, which uses frequency-dependent phase shifts for calculations in the frequency domain. It includes interpolation for dead phones (dead phones in the program are specifically for Run 5 and should be modified for different tracks) and a time-domain Hamming window. Phones 0 and 1 are the forward and aft desensitized phones. These are not included in the beamforming
process, but are moved from phones 0 and 1 to the last two beams.

To match the real-time beamformed data, 64 seconds of data (8192 sample points) should be used when beamforming. For 8192 sample points, the program takes approximately 5 minutes to beamform at one angle after the initial data transformation from the time-domain to the frequency-domain. ARSRP data considers forward as 0 degrees and aft as 180 degrees, the program implements a temporary -90 degree shift so that broadside is 0 degrees in the calculations.

There is a +3 dB amplitude difference between the real-time recorded beamformed data and the output from beamform.pro which is not accounted for in the program. To match the output of the PV-WAVE beamformer to the real-time beamform, the speed of sound used is 1510 m/s. However, the speed of sound hardwired into the program is 1525 m/s, which we believe to be the correct value. See Figure 2 for comparisons to the real-time MSS beam at angle 71.85. Near broadside, the difference in output is minimal, but increases as the angle leaves broadside. Note that the output of this beamformer is not converted to dB.

To beamform 62 phones (format A) instead of the 126 default, the constant nphones in beamform.pro should be changed from 126 to 62. Also, the constant numangles should be changed from 126 to 62 for the default set of angles to span 0 to 180 degrees evenly in cosine space.

**Example 1(d).** Beamform the data read in Example 1(c) at angles 69.85 to 71.85 with angle increment 0.1.

```
WAVE>beamform, timedata, realdata, imagdata, angledata, realbeam, imagbeam
Enter number of points per phone.
: 8192
User defined set of angles or default? U=user  D=default
: U
Enter number of angles.
: 61
Enter start angle.
: 69.85
Enter angle increment (theta) in degrees.
: 0.1
```
Figure 2. (a) Real-time beamformed data for Run 5, Ping 190 at angle 71.85. (b) PV-WAVE beamformed data for Run 5, Ping 190 at angle 71.85 with sound speed 1510 m/s. (c) Same as (b) except with sound speed 1525 m/s.
Angle array = 69.85  69.95  70.05  ...  71.75  71.85
Input data Fourier transformed.
Finished beam for angle  69.85
Finished beam for angle  69.95
.
.
Finished beam for angle  71.85

VIII. WAVEFORMS

Several different waveforms were used in the ARSRP reconn experiment (cf. ref. 1, pp 90-126). Within the ASPS software we have implemented match filtering for only two of them: the HFM upsweep and the CW tonals. A discussion for each of these follows:

HFM Upsweep

Program Name: hfmsource.pro
Syntax: WAVE>hfmsource,f,freqhfm
Variables: f - frequency array (output)
           freqhfm - frequency-domain HFM source signal (output)

This program creates a 210 - 280 Hz HFM upsweep source (Waveform ID SPSS053), and basebands the 186 to 314 Hz region. To do this, the source is actually basebanded at a center frequency of 250 Hz using the complex demodulation process described in Marple4. Then a low pass filter from -64 to 64 Hz is applied. This is then inverse transformed into the time-domain and zero-padded to a user given number of samples. The fourier transform of the zero-padded source results in a 128 Hz bandwidth frequency domain source stored in the variable freqhfm, which can be used to match filter the data, also of bandwidth 128 Hz. Note that the source function is rearranged to have negative frequencies preceding positive frequencies (-64 to 64 Hz), and then assigned to frequencies 0 to 128 Hz to match the bandwidth of the data. The algorithm uses a center frequency of 250 Hz instead of 186 Hz only to avoid an extra step of rearranging the data. An equivalent algorithm using 186 Hz center frequency could be implemented, however, the original source would be required to have a minimum of 628 samples/second
before basebanding and decimation. For large time series, it would be prohibitive.

Since the beamformer implemented in this package does not exactly match the real-time beamformer due to sound speed differences, Example 2 is not a continuation of Example 1. Instead of phone data for ping 190, real-time beamformed data from archive tape for ping 190 found in file btsb218_0948.393a will be used for all of Example 2.

**Example 2(a).** Create a 8192 point hfm source function for use in the matched filter.

```plaintext
WAVE>hfmsource, f, freqhfm
time increment = 0.000976562
Nyquist = 512.0
freq increment = 0.50
Enter number of points in source signal (sampled at 128 samples/sec.):
: 8192
zero-domain hfm source points = 8192.0
new time increment = 0.0078125
new freq increment = 0.015625
new Nyquist freq = 64.0
```

The frequency variable f is not needed in other programs, but simply output for plotting purposes. The result of Example 2(a) is shown in Figure 3.

**CW Tonals**

Program Name: cwtonal.pro
Syntax: WAVE>cwtonal, f, freqcw
Variables: f - frequency array (output) freqcw - frequency-domain CW tonal (output)

This program creates one of nine tonal waveforms in the wavetrain WRTP008. Each is a two second signal centered at the given frequency (Waveform ID's SRP010, SRP011, ..., SRP018), and basebands the 186 to 314 Hz region in the manner described for the HFM source. Processing of CW tonal data differs from that of the HFM data, and is discussed separately in Section XII.
Program Name: mfilt.pro
Syntax: WAVE>mfilt, source, timedata, realbeam, imagbeam, timemf, realmf, imagmf

Variables: source - frequency domain source (input)
timedata - one-dimensional array of time samples (input)
realbeam - two-dimensional array of real part of beamformed data for each beam (input)
imagbeam - two-dimensional array of imaginary part of beamformed data for each beam (input)
timemf - time array shifted to beginning of source transmission (output)
realmf - two-dimensional array of real part of matched filtered data for each beam (output)
imagmf - two-dimensional array of imaginary part of matched filtered data (output)

Beamformed data which has been created by beamform.pro or read from archive tape is matched filtered with a user supplied frequency-domain source.

A subset of the beams and/or time may be used for matched filtering. To do this, one can remove unwanted data in readdata.pro by choice of beams and number of points to keep, or alternately, read all the data, match filter, and then plot only the selected subsets. The latter is recommended to avoid confusion caused by reassignment of channel numbers. For Run 5, hfmsource.pro can be used to create the basebanded source for matched filtering. For 64 second of data (8192 points), the program mfilt.pro takes approximately 2 or 3 minutes to run.

Time is shifted so that the origin denotes the time sound is emitted from the source using the formula: shift = t1 - t2 = x/c, where t1 = actual direct arrival time of source on forward desensitized phone, t2 = expected direct arrival time of source on forward desensitized phone, x = distance between source and midpoint of hydrophone array (881m), and c = speed of sound (1525m/s). Thus negative time represents time between the start of recording and source transmission. The user is asked whether the data is phone or beam data. If the user supplies beamformed data, then the two desensitized phones have been moved by the beamformer from channels 0 and 1 to the last two channels. If the user supplies phone, or matched filtered data, then the two
desensitized phones are found in channels 0 and 1. Since the
time origin is calculated using the two desensitized phones, it
is important that the user supply the correct answer to this
question.

The output of the matched filter is converted to dB. Note
that during the conversion to dB, a small constant is added to
avoid taking the logarithm of zero.

Example 2(b). First read unformatted data into
timedata, realbeam, and imagbeam using readdata.pro. For this
example, all the data is read in and match filtered with the hfm
source produced in Example 2(a).

WAVE>mffilt, freqhfm, timedata, realbeam, imagbeam, timemf, realmf, imagmf
Enter number of channels in data file.
: 128
Enter number of points per channel- should be equal to number
of points in source.
: 8192
Enter phone or beam data? P or B
: b
Time shift = -7.28167

The time shift indicates that original beginning and ending
times of 0 and 64 seconds has been shifted to -7.28167 and
56.7813 seconds, respectively.

X. WATERFALL PLOTS

Program Name: waterfall.pro
Syntax: WAVE>waterfall, ti, re, im (general form)
WAVE>waterfall, timedata, realdata, imagdata (for
phone data)
WAVE>waterfall, timedata, realbeam, imagbeam (for
beamformed data)
WAVE>waterfall, timemf, realmf, imagmf (for matched
filtered data)

Variables: ti - one-dimensional array of time samples (input)
re - two-dimensional array of real part of data
for each channel (input)
im - two-dimensional array of imaginary part of
data for each channel (input)
The variables ti, re, and im can hold either phone, beamformed, or matched filtered data. One only needs to pass the appropriate variables, as listed above. The program creates a waterfall plot of the magnitude of the complex data. The y-axis of each plot is labelled with the integer phone or beam number. Amplitudes are not calibrated. The user has many options for plotting, however, the data can be smoothed using a given number of points only once during the program. To use a different number of points in the smoothing, the program should be terminated and started again. After the data is smoothed (or averaged), the program allows the user to create many plots with varying sets of beams, times, gain factors, and thresholds. This information is given in the upper right-hand corner of each plot. After each plot is created, the user can indicate whether or not a hard copy is wanted, and the name of the file in which to store the postscript output.

Example 2(c). Plot the matched filtered data created in Example 2(b).

```
WAVE>waterfall,timemf,realmf,imagmf
Enter number of channels.
: 128
Enter number of points per channel.
: 8192
Enter number of magnitude points to average.
: 16
Begin plots.
Enter title.
: b0948
Minimum magnitude (in dB) = -90.0457
Maximum magnitude (in dB) = 78.5482
Threshold data? Y or N
: Y
Enter min and max threshold values (dB).
: 45 80
Enter gain factor.
: 4.0
Enter first channel and last channel to plot.
: 0 127
Enter channel increment (integer).
: 2
Label y-axis with channel number of angle? C or A
: C
```
Time shift in seconds = -7.28167
Min, Max times =  -7.28167  56.7105
Enter min and max time in seconds to plot.
: 0 35
Plot finished.
Hardcopy of plot? Y or N
: Y
Enter name of postscript output file.
: wavel.ps
Postscript file created.

Note: At this point, the file wave.ps is ready to be sent to the printer. Since the postscript files are usually large, use the command lpr -s wavel.ps to create a symbolic link rather than copying to the spool area. The program continues:
Another plot? Y or N
: Y
Enter title.
: Subset of b0948
Minimum magnitude (in dB) = -90.0457
Maximum magnitude (in dB) = 78.5482
Threshold data? Y or N
: Y
Enter min and max threshold values (dB).
: 35.0 80.0
Enter gain factor.
: 1.5
Enter first channel and last channel to plot.
: 41 45
Enter channel increment (integer).
: 1
Label y-axis with channel number of angle? C or A
: C
Time shift in seconds = -7.28167
Min, Max times =  -7.28167  56.7105
Enter min and max time in seconds to plot.
: 5.0 34.0
Plot finished.
Hardcopy of plot? Y or N
: Y
Enter name of postscript output file.
: wave2.ps
Postscript file created.
Another plot? Y or N
: N
The two plots created and stored in wavel.ps and wave2.ps are shown in Figures 4 and 5 with the corresponding MSS real-time plots from the ARSRP Monitoring Support Software Supplement shown in Figure 6 and 7. The time origins for the MSS plots are not the same as the software discussed here, but are plotted with the same total time.
Figure 4. PV-WAVE match filtered HFM data for Run 5, Ping 190.
Figure 5. PV-WAVE match filtered HFM data for Run 5, Ping 190.
Figure 6. Real-time MSS match filtered HFM data for Run 5, Ping 190.
Figure 7. Real-time MSS match filtered HFM data for Run 5, Ping 190.
XI. STORING OUTPUT

Storing Output in Binary

Program Name: unformatoutput.pro
Syntax: WAVE>unformatoutput, timeout, realout, imagout (general form)
        WAVE>unformatoutput, timedata, realbeam, imagbeam (for beamformed data)
        WAVE>unformatoutput, timemf, realmf, imagmf (for matched filtered data)

Variables: timeout - one-dimensional array of time samples (input)
           realout - two-dimensional array of real part of data for each channel (input)
           imagout - two-dimensional array of imaginary part of data for each channel (input)

Once data has been beamformed or matched filtered, it may be saved using unformatoutput.pro in a binary (unformatted) file in a form which can be read directly by readdata.pro in another PV-WAVE session. The program line

    openw,1,outfile

should be replaced with

    openw,1, output,/f77_unformatted

if the user wishes to create a binary file which can be read by a fortran program. The program readdata.pro can be modified in the same manner to read the fortran compatible file into PV-WAVE.

Storing Output in ASCII

Program Name: asciioutput.pro
Syntax: WAVE>asciioutput, timeout, realout, imagout (general form)
        WAVE>asciioutput, timedata, realbeam, imagbeam (for beamformed data)
        WAVE>asciioutput, timemf, realmf, imagmf (for matched filtered data)

Variables: timeout - one-dimensional array of time samples (input)
realout - two-dimensional array of real part of data for each channel (input)
imagout - two-dimensional array of imaginary part of data for each channel (input)

Once data has been beamformed or matched filtered, it may be saved using unformatoutput.pro in an ASCII file in one of two formats. The first choice creates a file which contains only the real and imaginary parts of the data and can be converted to binary using pvconvert2arr.pro for use in another PV-WAVE session. The second choice creates a file which contains four arrays instead of two, namely, channel number, time sample, real part, and imaginary part, and can be converted to binary using pvconvert4arr.pro. No headers are present in either format.

XII. PROCESSING CW DATA

In Run 5, a sequence of nine two second tonals sources with Hamming taper (Wavetrain ID WTRP008) were transmitted one second apart. The software to create each of the nine replicas is cwtonal.pro.

The CW tonal hydrophone data for Pings 188, 189, 190, and 191 currently exists on four exabyte tapes, one per ping. Each CW epoch begins at 600 seconds. Data before 600 seconds is HFM data which can be found on other tapes. To extract the proper epoch, the user should read record number 10 and 11, or if one can ignore the first two seconds of the ping, only record 11. Record 11 contains data from 602 to 661.5 seconds.

Unlike the HFM data, which is beamformed and then matched filtered, the CW data is first match filtered to extract one CW, and then beamformed. This process is repeated nine times, once for each CW tonal, creating nine sets of acoustic data. The following example is shown to illustrate this process.

Example 3(a). Extract the acoustic data field produced by the 250 Hz CW tonal for Ping 190.

First, use segy2ascii.c to extract record 11 from the tape labelled CWC218_0948.393A and put into an ASCII file of the same name. Use pvconvert2arr.pro to convert from ASCII to PV-WAVE binary.
WAVE>pvconvert2arr,'cwc218_0948.393a','cwc218_0948unf.393a'
Enter start phone/beam number in file (usually 0 or 1).
: 0
Enter number of points per phone/beam.
: 7616
Channel number converted: 0

Channel number converted: 127
Unformatted output in cwc218_0948unf.393a.

Example 3(b). Use readdata to read the data into PV-WAVE variables, timedata, realdata, imagdata.

WAVE>readdata,timedata,realdata,imagdata
Enter input file name.
: cwc218_0948unf.393a
Enter number of channels in file.
: 128
Number of channels read = 128.0
Enter beginning and ending channel numbers to keep
-possible 0 to 127
: 0 127
Number of channels to keep = 128.0
Number of points per channel - 7616
Enter number of points per channel to keep.
: 7616
Data read for channel number - 0
Number of points per channel - 7616

Data read for channel number - 127
Number of points per channel - 7616

Example 3(c). Create a 7616 point 290 Hz CW tonal source to be used as the matched filter.

WAVE>cwtonal,f,freqcw
time increment = 0.000976562
Nyquist = 512.000
freq increment = 0.50000
Enter frequency (210,220,230,240,250,260,270,280,290)
: 290
Enter number of points in source signal (sampled at 128 samples/sec).
: 7616
zero padded time-domain CW source points = 7616
new time increment = 0.00781250
new freq increment = 0.0168067
new Nyquist frequency = 64.0000

Example 3(d). Match filter the data with the 290 Hz CW source.

WAVE>mfilt, freqcw, timedata, realdata, imagdata, timemf, realmf, imagmf
Enter number of channels in data file.
: 128
Enter number of points per channel – should be equal to number of points in source.
: 7616
Enter phone or beam data? P or B
: P
Time shift = -7.24261

Example 3(e). Beamform the matched filtered data from angles 25 to 104 degrees with an angle increment of one degree.

WAVE>beamform, timemf, realmf, imagmf, anglearray, realbeam, imagbeam
Enter number of points per phone.
: 7616
User defined set of angles or default? U=user D=default
: u
Enter number of angles.
: 80
Enter start angle.
: 25
Enter angle increment (theta) in degrees.
: 1
Angle array = 25.0000 26.0000 27.0000 ... 103.0000 104.0000

Input data Fourier transformed.
Finished beam for angle 25.0000
Finished beam for angle 26.0000
.
.
Finished beam for angle 104.0000

28
Example 3(f). Plot the data from 16 to 35 seconds. First, throw away the desensitized phones. Remember that the matched filtered data shifts the time array, thus, timemf is used for plotting.

WAVE>realbeam = realbeam(*,0:79)
WAVE>imagbeam = imagbeam(*,0:79)
WAVE>waterfall, timemf, realbeam, imagbeam
Enter number of channels.
: 80
Enter number of points per channel.
: 7616
Enter number of magnitude points to average.
: 1
Begin plots.
Enter title.
: P190F290.CW
Minimum magnitude (in dB) = -21.6739
Maximum magnitude (in dB) = 66.1557
Threshold data? Y or N
: Y
Enter min and max threshold values (dB).
: 45 70
Enter gain factor.
: 5.0
Enter first channel and last channel to plot.
: 0 79
Enter channel increment (integer).
: 1
Label y-axis with channel number of angle? C or A
: A
Enter number of labels (integer).
: 9
Enter 9 labels (one per line).
: 25
: 35
: 45
: 55
: 65
: 75
: 85
: 95
: 105
Time shift in seconds = -7.24261
Min, Max times = -7.24261  52.2496
Enter min and max time in seconds to plot.
: 16   35
Plot finished.
Hardcopy of plot?  Y or N
: Y
Enter name of postscript output file.
: output.ps
Postscript file created.
Another Plot?  Y or N
: N

A hardcopy of this plot is shown in Figure 8.
Min, Max Magnitude (dB): -21.6739, 66.1557
Threshold (dB): 45.0000 - 70.0000
Magnitude points avg: 1.00000
Gain factor: 5.00000
Beam increment: 1
Time Shift: -7.24261

Figure 8. CW tonal data from Ping 190, match filtered with the 290 Hz source using PV-WAVE.
ACKNOWLEDGEMENTS

The authors wish to express their appreciation to Prof. John Orcutt and his associates at Scripps Institute of Oceanography, especially Mr. Paul Henkart, for making the original of the BBN to SEGY conversion program available to us; to Prof. Donald Tufts and his graduate students at the University of Rhode Island for parts of the routines for beamforming and matched filtering in Matlab which we used as examples for parts of our PV-WAVE programs; and to Messrs. E.J. Yoerger and James Showalter for assistance in programming and data handling. Finally, we express our appreciation to Office of Naval Research Program Managers Drs. Marshall Orr and Moheen Badiey for interest in and funding of this work. This work was funded under PE 0601153N.

REFERENCES


APPENDIX A

Reading OLP BBN formatted tapes

In early March the Scripps Institute of Oceanography (SIO) supplied some software to the ARSRP community which allowed its users to read the 9-track data tapes produced by the Off Line Processor (OLP) on the RV Cory Chouest. The following is a copy of the message posted to OMNET/BOTMREV.ACOUSTICS by John Orcutt:

Posted: Fri, Mar 13, 1992 1:50 PM EST
From: J.ORCUTT
To: botmrev.acoustics
Subj: UNIX software

A set of software for reading the SRP 9-track tapes on a UNIX machine has been put on a SUN at IGPP which can be accessed by anonymous FTP protocols. The address is "sioseis.ucsd.edu" and the required code as well as a "read-me" file is in the FTP user area. The code could be made available on other media, but this is probably the simplest approach. If there are problems please call either myself or Paul Henkart (the author) at (619)534-3487. Paul’s Internet address is "henkart@sioseis.ucsd.edu" and mine is "orcutt@bull.ucsd.edu." The programs read the nine track tapes and write the time series in modified SEGY format. Much of the header information, like ship’s position and time, is written to the SEGY headers. The software will allow you to read the tapes and save the data in a standard format for plotting or processing. We’ve found that MATLAB running on a workstation is pretty efficient for much of the signal processing. As changes are made to the software at your institutions, it would be a good idea to share improvements throughout the community.

As often happens with such software, it was not as portable as its authors had hoped, and it was necessary to modify it further before it could be used locally. This turned out to be a more difficult exercise than expected, because several subtle compiler differences had to be located and corrected before the time-series data could be read and demultiplexed.

The SIO software has now been successfully used to read OLP tapes produced during the ARSRP Reconnaissance Cruise, and it should prove equally useful with CST and LFA tapes. Not only have results been verified as numerically correct, but the speed
of reading any data tape is several orders of magnitude faster than the local OLP hardware/software. As an example, it took more than eight hours for the OLP to read track 5 data and convert them to an ASCII format file. The current software can read and convert the same data in fewer than five minutes.

The 9-track tapes produced on the Cory contain several types of data, but the format in which the data are written is not standard and was selected by the BBN Corporation. A 9-track tape typically contains the following types of information:

a) Configuration data for the real time processor (beam directions, transmission times, beamformer bias).

b) Non-acoustic source (NAS) data (latitude, longitude, and other navigation data).

c) Multiplexed time series data in 16 bit BBN floating point format.

d) System time in DEC VMS format.

The SIO software reads the 9-track tapes and produces a disk file of navigation and time-series data which adhere to two widely used standards: (1) The BBN floating point numbers are converted to IEEE floating point numbers so that computers such as Sun workstations and desk top microcomputers can use them directly and (2) the navigation and digitization parameters are combined with the data samples in a format known as the Society of Exploration Geophysicist's Y-format (SEGY). The results are a sequence of records - called traces - each one of which has a trace header preceding the time-series data. The trace header contains navigation and digitization information and is immediately followed by the complex time-series data. One "trace" corresponds to either the output data from one hydrophone (passthrough data) or to the output of one beam (beamformed data). Thus, the output file contains important additional information such as the sampling rate, and there is much less chance of losing information needed to process and interpret the data. The SEGY format is easily converted to others. The program segy2ascii.c described in the main text converts the SEGY format to ASCII format.

It is very important that the software is compiled and
linked on the Sun workstation where it will be used. Failure to do so may result in software which operates incorrectly or not at all. A later section discusses how to do this.

The heart of the software is a program called bbn2segy. This program reads the 9-track BBN formatted tapes, converts and demultiplexes the data, and produces one or more SEGY files. There are only a few questions the user needs to answer, and they are all straightforward. The user will be asked for the number of minutes of data which are to appear per beam or hydrophone after each ping and the number of pings to be read. Be aware that the SIO software uses a slightly peculiar definition of a minute. A minute contains exactly 66.5 seconds of data or 8512 complex samples. The program does not allow the user to read less than one "minute" of data per ping, and will only accept an integer number of minutes. If the user elects to read more than one minute of data, there will be a seven second overlap of data during each succeeding minute. That is, each 66.5 second data trace written to the file will contain the last seven seconds of the preceding data trace at its beginning. Because of the way in which the tape is structured, it is necessary to deal with 448 complex points at a time (3.5 seconds). This leads to the 66.5 second minute and to the overlap given above.

The software actually means "epoch" when it asks for the number of pings to read, and it is important not to confuse a software ping with the more usual use of the word. If, for example, you want to reach some CW data which follows a series of FM chirps, you must read all of the preceding data in the epoch.

The software produces one SEGY disk file per epoch (ping) of data, and it will ask the user to make up a name for each disk file to be written before it reads anything from tape. That is, if the user selects eight epochs, the program will ask for eight file names right at the start of the run. Each file will require approximately 8,750,000 bytes for each minute of data read. There is no additional information required, other than the name of the tape drive on which the 9-track tape will be found.

Usually each tape contains two epochs of data, and the attempt to read more epochs than are present will cause the software to issue instructions to the operator to mount another
tape. Should that happen, restart the software so that only the number of epochs on one tape are required.

There is additional software (see the installation instructions which follow) in the SIO package, but it is mostly used for examining a tape prior to reading it in detail. There are programs to print the navigation and configuration data to the screen (dump.nasdata, dump.configdata) as well as one program to very briefly list the number of NAS data packets on the tape. Most people will probably not be interested in the output of these additional programs, although the program dump.configdata can be used to count the number of epochs on a tape by counting how many times it will run before the end of tape indicator is reached.

To obtain the software, use the ftp file transfer program from slater@gandalf.noarl.navy.mil to copy the file "scripps.tar". Place the file in the directory area where you want the software to self-install, and type

```
tar -xvf scripps.tar
```

to extract all of the software. This should result in two new subdirectories called src and lib, as well as a README file. There should also be a file called "install" present, and it should be marked as executable (it is a script file). Type

```
install
```

and the script file will attempt to generate the library, followed by the executable program "bbn2segy." If this results in an "access denied" message, type

```
csh install.
```

In order for the script file to work, the Sun f77 compiler must be available as well as a C compiler. The Sun workstations have so far come with the cc compiler installed, and so the cc compiler is assumed. If any error messages are observed, check the path to be sure that both cc and f77 can be accessed. Install simply invokes the make utility for each of the SIO programs, and is therefore not complicated.
APPENDIX B

OLP and SEGY tapes held by NRL/SSC

The following table lists the 9-track OLP tapes held by NRL/SSC. (Not all the information in the columns are readily available, but it can be determined with a little effort.) Note that in addition to the A format (1/2 phones and 1/2 beams) previously discussed, the B format is all beams and the C format is all phones. The sequence number is our tape ID.

9-track OLP tapes

<table>
<thead>
<tr>
<th>RUN</th>
<th>STRT TIME</th>
<th>STOP TIME</th>
<th>PNG ID#</th>
<th>TYP</th>
<th>FM</th>
<th>SEQ#</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>216:0615Z</td>
<td>005,006</td>
<td>BTS</td>
<td>B</td>
<td></td>
<td>353B</td>
</tr>
<tr>
<td>1</td>
<td>216:0615Z</td>
<td>216:0645Z</td>
<td>005,006</td>
<td>B</td>
<td></td>
<td>356B</td>
</tr>
<tr>
<td>1</td>
<td>216:0645Z</td>
<td>007,008</td>
<td>BTS</td>
<td>B</td>
<td>345B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:0700Z</td>
<td>007</td>
<td>BTS</td>
<td>C</td>
<td>357B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:0700Z</td>
<td>216:0730Z</td>
<td>008,009</td>
<td>BTS</td>
<td>358B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:0715Z</td>
<td>009,010</td>
<td>BTS</td>
<td>B</td>
<td>355B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:0730Z</td>
<td>216:0745Z</td>
<td>010</td>
<td>BTS</td>
<td>359B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1349Z</td>
<td>216:1444Z</td>
<td>BTS</td>
<td>A</td>
<td>016B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1440Z</td>
<td>216:1640Z</td>
<td>MFT</td>
<td>A</td>
<td>005A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1440Z</td>
<td>216:1646Z</td>
<td>MFT</td>
<td>A</td>
<td>005B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1444Z</td>
<td>216:1515Z</td>
<td>BTS</td>
<td>A</td>
<td>017B</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1544Z</td>
<td>216:1614Z</td>
<td>BTS</td>
<td>A</td>
<td>019A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1544Z</td>
<td>216:1614Z</td>
<td>BTS</td>
<td>A</td>
<td>019A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1545Z</td>
<td>216:1615Z</td>
<td>041,042</td>
<td>BTS</td>
<td>363A</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>216:1545Z</td>
<td>216:1615Z</td>
<td>040,041</td>
<td>B</td>
<td>364A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>216:2105Z</td>
<td>216:2135Z</td>
<td>BTS</td>
<td>A</td>
<td>028A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>216:2105Z</td>
<td>216:2135Z</td>
<td>BTS</td>
<td>A</td>
<td>028A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>216:2108Z</td>
<td>216:2327Z</td>
<td>MFT</td>
<td>A</td>
<td>008A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>216:2108Z</td>
<td>216:2327Z</td>
<td>MFT</td>
<td>A</td>
<td>008A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>216:2206Z</td>
<td>216:2236Z</td>
<td>062,063</td>
<td>BTS</td>
<td>365A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>217:0151Z</td>
<td>217:0206Z</td>
<td>075</td>
<td>BTS</td>
<td>372A</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>217:0251Z</td>
<td>217:0306Z</td>
<td>079</td>
<td>BTS</td>
<td>368A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1354Z</td>
<td>217:1408Z</td>
<td>BTS</td>
<td>A</td>
<td>059A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1354Z</td>
<td>217:1424Z</td>
<td>120,121</td>
<td>BTS</td>
<td>375A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1354Z</td>
<td>217:1424Z</td>
<td>124,125?</td>
<td>BTS</td>
<td>375A</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1408Z</td>
<td>217:1438Z</td>
<td>BTS</td>
<td>A</td>
<td>060A</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1420Z</td>
<td>217:1620Z</td>
<td>MFT A 016A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1420Z</td>
<td>217:1620Z</td>
<td>MFT A 016A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1424Z</td>
<td>217:1524Z</td>
<td>122,123 BTS C 374A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1438Z</td>
<td>217:1508Z</td>
<td>BTS A 061A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:1508Z</td>
<td>217:1538Z</td>
<td>BTS A 062A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:2220Z</td>
<td>218:0025Z</td>
<td>MFT A 020A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>217:2220Z</td>
<td>218:0025Z</td>
<td>MFT A 020A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>217:2244Z</td>
<td>217:2314Z</td>
<td>BTS A 075A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>217:2244Z</td>
<td>217:2314Z</td>
<td>BTS A 075A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>217:2314Z</td>
<td>217:2344Z</td>
<td>BTS A 076A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>217:2344Z</td>
<td>218:0014Z</td>
<td>BTS A 077B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>218:0025Z</td>
<td>218:0235Z</td>
<td>MFT A 021A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0900Z</td>
<td>218:0930Z</td>
<td>BTS A 094A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0918Z</td>
<td>218:0948Z</td>
<td>188,189 BTS C 391A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0918Z</td>
<td>218:0933Z</td>
<td>MFT A 026A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0918Z</td>
<td>218:0948Z</td>
<td>188,189 BTS C 391</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0930Z</td>
<td>218:1002Z</td>
<td>189,190 BTS A 095A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0948Z</td>
<td>218:1018Z</td>
<td>190,191 BTS C 392B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:0948Z</td>
<td>218:1018Z</td>
<td>190,191 BTS B 393A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1002Z</td>
<td>218:1033Z</td>
<td>191,192 BTS A 096A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1018Z</td>
<td>218:1033Z</td>
<td>192 BTS B 394A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1018Z</td>
<td>218:1038Z</td>
<td>192 BTS B 395B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1038Z</td>
<td>218:1301Z</td>
<td>MFT A 027A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1038Z</td>
<td>218:1301Z</td>
<td>MFT A 027A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1103Z</td>
<td>218:1148Z</td>
<td>BTS A 098A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1103Z</td>
<td>218:1148Z</td>
<td>BTS A 098A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1103Z</td>
<td>218:1113Z</td>
<td>195 BTS C 396A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1103Z</td>
<td>195 BTS C 396</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1203Z</td>
<td>218:1233Z</td>
<td>198,199 BTS C 397A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1203Z</td>
<td>198,199 BTS B 399</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1233Z</td>
<td>218:1248Z</td>
<td>200 BTS B 398A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1248Z</td>
<td>218:1318Z</td>
<td>BTS A 101A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1301Z</td>
<td>218:1506Z</td>
<td>MFT A 028A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1310Z</td>
<td>218:1506Z</td>
<td>MFT A 028A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1329Z</td>
<td>218:1347Z</td>
<td>BTS A 102A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1347Z</td>
<td>218:1417Z</td>
<td>BTS A 103A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1347Z</td>
<td>218:1417Z</td>
<td>BTS A 103A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1403Z</td>
<td>218:1418Z</td>
<td>206 BTS C 401A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1403Z</td>
<td>218:1418Z</td>
<td>206 BTS C 401</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>218:1417Z</td>
<td>218:1447Z</td>
<td>BTS A 104A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5P</td>
<td>218:1719Z</td>
<td>218:1739Z</td>
<td>BTS A 110A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5P</td>
<td>218:1739Z</td>
<td>218:1800Z</td>
<td>BTS A 111A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5P</td>
<td>218:1740Z</td>
<td>218:1800Z</td>
<td>218 BTS C 402A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5P</td>
<td>218:1740Z</td>
<td>218 BTS C 402</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page</td>
<td>Time</td>
<td>Time</td>
<td>Event</td>
<td>Location</td>
<td>Info</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
<td>----------</td>
<td>------------------</td>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>218:2331Z</td>
<td>219:0135Z</td>
<td>MFT A 033A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>218:2354Z</td>
<td>218:0024Z</td>
<td>BTS A 124A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0024Z</td>
<td>219:0054Z</td>
<td>BTS A 125A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0054Z</td>
<td>219:0124Z</td>
<td>BTS A 126A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0124Z</td>
<td>219:0154Z</td>
<td>BTS A 127A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0135Z</td>
<td>219:0245Z</td>
<td>MFT A 034A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0140Z</td>
<td>219:0210Z</td>
<td>246, 247 BTS C 412A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0140Z</td>
<td>219:0210Z</td>
<td>246, 247 BTS C 412</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0154Z</td>
<td>218:0204Z</td>
<td>BTS A 128A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0245Z</td>
<td>219:0320Z</td>
<td>BTS A 130A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0320Z</td>
<td>219:0410Z</td>
<td>BTS A 131A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0345Z</td>
<td>219:0627Z</td>
<td>MFT A 036A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0410Z</td>
<td>219:0425Z</td>
<td>253 BTS C 415A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>219:0410Z</td>
<td>219:0425Z</td>
<td>253 BTS C 415</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>219:0510Z</td>
<td>219:0540Z</td>
<td>BTS A 134A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>219:0510Z</td>
<td>219:0525Z</td>
<td>257 BTS B 417A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>219:0510Z</td>
<td>219:0525Z</td>
<td>257 BTS C 417</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>219:0540Z</td>
<td>219:0610Z</td>
<td>BTS A 135A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>219:0610Z</td>
<td>219:0640Z</td>
<td>BTS A 136A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>222:0127Z</td>
<td>222:0159Z</td>
<td>BTS A 256A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>222:0159Z</td>
<td>222:0231Z</td>
<td>BTS A 257A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>222:0231Z</td>
<td>222:0304Z</td>
<td>BTS A 258A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2119Z</td>
<td>222:2158Z</td>
<td>BTS A 295A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2140Z</td>
<td>222:2220Z</td>
<td>580, 581 BTS B 449A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2140Z</td>
<td>222:2240Z</td>
<td>580, 581 BTS B 449</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2158Z</td>
<td>222:2239Z</td>
<td>BTS A 296A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2220Z</td>
<td>222:2240Z</td>
<td>582 BTS B 450A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2220Z</td>
<td>222:2240Z</td>
<td>582 BTS B 450</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>222:2239Z</td>
<td>222:2300Z</td>
<td>BTS A 297A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>223:0250Z</td>
<td>223:0329Z</td>
<td>BTS A 304A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>223:0310Z</td>
<td>223:0350Z</td>
<td>596, 597 BTS B 451B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>223:0310Z</td>
<td>223:0350Z</td>
<td>596, 597 BTS B 451</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>223:0350Z</td>
<td>223:0430Z</td>
<td>598, 599 BTS B 452B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>223:0409Z</td>
<td>223:0448Z</td>
<td>BTS A 306A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>223:0800Z</td>
<td>223:0840Z</td>
<td>610 BTS B 453B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>223:0819Z</td>
<td>223:0900Z</td>
<td>BTS A 312A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>223:0840Z</td>
<td>223:0900Z</td>
<td>611 BTS B 454B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>223:0900Z</td>
<td>223:0940Z</td>
<td>BTS A 313A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>223:0940Z</td>
<td>223:1001Z</td>
<td>BTS A 314A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>223:1350Z</td>
<td>223:1430Z</td>
<td>BTS A 320A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>223:1430Z</td>
<td>223:1509Z</td>
<td>BTS A 321A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>223:1509Z</td>
<td>223:1530Z</td>
<td>BTS A 322A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>223:1923Z</td>
<td>223:2019Z</td>
<td>BTS A 329A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>223:2019Z</td>
<td>223:2100Z</td>
<td>BTS A 330A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Exabyte SEGY tapes

The following exabyte SEGY tapes have been created from their corresponding OLP tapes. The tape names are constructed as followings: example BTSB218_0948.363A -- "BTS"-beam tapes, "A"-format A (also B or C), "218_0948"-Julian date and time Zulu, ".363A"-OLP tape sequence number. All "BTS" exabyte tapes listed below are HFM upsweeps with the initial 145 sec recorded. "CWC" tapes are all CW tonals in the C format.

BTSB218_0948.393A  BTSC218_1740.402A
BTSB218_1033.393A  BTSC216_0615.356B
BTSB218_1018.394A  BTSC216_0630.356B
BTSC218_0948.392B  BTSC216_0645.357B
BTSC218_1003.392A  BTSC216_0700.358B
BTSC218_0918.391A  BTSC216_0715.358B
BTSC218_0933.391A  BTSC216_0730.358B
BTSC218_1103.396A  BTSC216_0730.359B
BTSC218_1203.399   BTSC216_1545.364A
BTSC218_1218.399   BTSC216_1600.364A
BTSC218_1233.398A  BTSC216_2206.365A
BTSC219_0510.417A  BTSC216_2221.365A
BTSC219_0510.417   BTSC217_0151.368A
BTSB216_0615.353B  BTSC217_0251.368A
BTSB216_0630.353B  BTSC217_1354.373A
BTSB216_0645.354B  BTSC217_1409.373A
BTSB216_0700.354B  BTSC217_1424.374A
BTSB216_0715.355B  BTSC217_1439.374A
BTSB216_0730.355B  BTSC218_1403.401A
BTSB216_1545.363A  BTSC218_0155.412A
BTSB216_1600.363A  BTSC219_0140.412A
BTSB216_2140.449A  BTSC219_0410.415A
BTSB222_2155.449A  BTSC217_1354.375A
BTSB222_2220.450A  BTSC217_1409.375A
BTSB223_0310.451B  CWC218_0948.392B
BTSB223_0325.451B  CWC218_1003.392B
BTSB223_0350.452B  CWC218_0918.391A
BTSB223_0405.452B  CWC218_0933.391A
BTSB223_0800.453B  CWC218_1103.396A
APPENDIX C

Program Code

The program listings are given for the following programs:

segy2ascii.c
pvconvert2arr.pro
pvconvert4arr.pro
readdata.pro
beamform.pro
hfmsource.pro
cwtonal.pro
mfilt.pro
waterfall.pro
unformatoutput.pro
asciioutput.pro
/* segy2ascii

/* Small program to read bbn2segy files and write ascii files. */
/* This program writes all channels sequentially to the output file */
/* by scanning the input segy file repeatedly for records belonging */
/* to the desired channel. */

/* A "record" is a segy record, corresponding to 66.5 - 7 = 58.5 seconds */
/* of data. This program ignores the first 7 seconds of every record */
/* beyond the first, because that is duplicated in the previous record. */

/* The program can be invoked entirely from the command line with the */
/* syntax: segy2ascii input_file output_file begin_trace end_trace */

/* Slater - NRL-SSC */

#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#define SEEK_SET 0
#define SEEK_CUR 1
#define SEEK_END 2

/* SEGY trace header structure. */

struct thd {
    char dum0[8];
    long recno, chno;
    char dum1[12];
    short chflag;
    char dum2[42];
    float sourcelat, sourcelong, reclat, reclong;
    char dum3[26];
    unsigned short nsamp, sint;
    char dum4[38];
    short year, day, hour, minute, sec, msecs;
    char dum5[24];
    float flsint;
    char dum6[44];
};

main( argc, argv )
int argc;
char *argv[]; {
    FILE *fpin, *fpout;
    char inpath[80], oupath[80];
    static struct thd hdr;
    float buffer[17024];
    int count=0, beam=1, start=1, stop=0;

    if ( argc < 2 )
    |
        /* Query for and open input file. */
        printf( "Enter input file name: ");
        gets( inpath );
    |
    else
    |
        strcpy( inpath, argv[1] );
    |
    if( (fpin = fopen( inpath, "rb" )) == NULL )
    |
        printf( "Can't open input file.\n" );
        exit( 1 );
    |
    if ( argc < 3 )
    |
        /* Query for output file name. */

printf( "Enter output file name: ");
gets( oupath );
}
else
{    
    strcpy( oupath, argv[2] );
}

/* Position file past reel and binary headers. */
if ( !( fseek( fpin, 3600L, SEEK_SET) == NULL ) )
{    
    printf( "Can’t position input file.\n" );
    exit( 1 );
}
if ( (fpout=fopen(oupath,"wt")) == NULL )
{    
    printf( "Can’t open output file %s\n", oupath);    
    exit(1);
}
if( argc < 4)
{    
    printf("Enter starting record number [1-N]: ");
    gets( inpath );
}
else strcpy( inpath, argv[3] );
if( sscanf(inpath, "%d", &start) !=1 ) exit(1);
if( argc < 5)
{    
    printf("Enter ending record number [%d-N]: ", start);    
    gets( inpath );
}
else strcpy( inpath, argv[4] );
if( sscanf(inpath, "%d", &stop) !=1 ) exit(1);
if( start > stop )
{    
    printf("Record numbers specified incorrectly.\n");
    exit(1);
}

/* Adjust start and stop for later use. */
start--;    
stop++;    
while ( !feof( fpin) )
{    
    if ( fread( &hdr, (unsigned) sizeof(struct thd), 1, fpin ) !=0 )
    {    
        if ( hdr.recno > start && hdr.recno < stop )
        {    
            if ( hdr.chno == beam)
            {    
                if ( !(fread( buffer, (unsigned) sizeof(float), hdr.nsamp,    
                  fpin) == hdr.nsamp ){
                    printf("Can’t read trace.\n");
                    exit(1);    
                }
            }    
        }    
    }    
    /* If this is the first record, get all of the samples. */
    /* Otherwise, skip the duplicated 7 seconds of data. */
    if ( hdr.recno == 1 )
    {    
        printf( "Writing channel %ld, record %ld\n", hdr.chno, hdr.recno);    
        for (count = 0; count < hdr.nsamp; count+=2)
            fprintf( fpout, "%g %g\n", buffer[count], buffer[count+1]);
    }    
    else    
    {    
        printf( "Writing channel %ld, record %ld\n", hdr.chno, hdr.recno);    
        for (count = 1792; count < hdr.nsamp; count+=2)
            fprintf( fpout, "%g %g\n", buffer[count], buffer[count+1]);
    }
}
else fseek (fpin, 68096L, SEEK_CUR); /* skip data */
}
else if (hdr.recno >= stop) fseek (fpin, 0L, SEEK_END);
else fseek (fpin, 68096L, SEEK_CUR); } /* EOF or skip */
else { /* EOF found. Reset if not done. */
    beam++;
    if (beam < 129) fseek (fpin, 3600L, SEEK_SET);
}
fclose (fpout);
exito;
pro pvconvert2arr,infile,outfile

; August 1992, L. A. Pflug
; Program to convert ascii output (created by segy2ascii.pro)
; to unformatted data which can be read more quickly by pv-wave.
; Input data should consist of two arrays: real part, imaginary part.

; open input and output files
openr,1,infile
openw,2,outfile

; user must input first channel number and points per channel
nchannel=0
arrsize=0
print,'Enter start and end channel numbers in file (possible 0 to 127).'
read,nchannelstart,nchannelend
print,'Enter number of points per phone/beam.'
read,arrsize

; initialize data arrays
; (output time array is redundant for this program but is
; compatible with readdatapro
outtime=fltarr(arrsize)
outreal=fltarr(arrsize)
outimag=fltarr(arrsize)

; define dt as the sampling rate
dt=1/128.

for iloop=nchannelstart,nchannelend do begin
; read each channel and create time array
for ipoint=0,arrsize-1 do begin
readf,1,tempy,tempz
outtime(ipoint)=t
outreal(ipoint)=tempy
outimag(ipoint)=tempz
endfor
; output channel in unformatted form
print,'Channel number converted: ',iloop
writeu,2,fix(iloop),fix(arrsize)
writeu,2,outtime
writeu,2,outreal
writeu,2,outimag
endfor

; close input and output files
close,1
close,2

print,'Unformatted output in ',outfile
end
pro pvconvert4arr,infile,outfile

; June 1992, L. A. Pflug
; Program to convert ascii output to PV-WAVE unformatted data
; which can be read more quickly by pv-wave.
; Input data should consist of four arrays:
; phone/beam number, time, real part, imaginary part
; output data numbers channels from 0 to nchannel-1.

;open input file
openr, 1, infile
openw, 2, outfile

;following two lines should be uncommented if a header is present
;header=''
;readf, 1, header

;initialize nchannel to integer 0
nchannel=0

;user input number of channels and points per channel.
print,'Enter start phone/beam number in file (usually 0 or 1).'
read, nchannel
print,'Enter number of points per phone/beam.'
read, arrsize

;initialize input arrays for unknown number of points (max of 128 seconds
data) and output arrays
timearr=fltarr(2100000)
realarr=fltarr(2100000)
imagarr=fltarr(2100000)
outtime=fltarr(arrsize)
outreal=fltarr(arrsize)
outimag=fltarr(arrsize)

;initialize nptstotal (total number of points overall) to long integer
;initialize nsize (number of points per channel) and nchannelnum to 0
nptstotal = 0L
nsize=0
nchannelnum=0

;loop through each channel
while not eof(l) do begin
; read one channel
readf,l,nchannelnum,tempx,tempy,tempz
timearr(nptstotal)=tempx
realarr(nptstotal)=tempy
imagarr(nptstotal)=tempz

; output one channel in unformatted form
if (nchannelnum gt nchannel) or (eof(l)) then begin
if (not eof(l)) then begin
print,'Channel number converted: ',nchannelnum-1
print,'Number points read = ',nsize
writeu,2,fix(nchannelnum-1),fix(nsize)
endif
assign channel to output array
outtime=timearr(nptstotal-nsize:nptstotal-1)
outreal=realarr(nptstotal-nsize:nptstotal-1)
outimag=imagarr(nptstotal-nsize:nptstotal-1)
writeu,2,outtime
writeu,2,outreal
writeu,2,outimag
endif
if eof(l) then begin
print,'Channel number converted: ',nchannelnum
print,'Number points read = ',nsize+1
writeu,2,nchannelnum,nsize+1
assign channel to output array
outtime=timearr(nptstotal-nsize:nptstotal)
outreal=realarr(nptstotal-nsize:nptstotal)
outimag=imagarr(nptstotal-nsize:nptstotal)
writeu,2,outtime
writeu,2,outreal
writeu,2,outimag
endif
nchannel=nchannel+1
; re-initialize output and variables and nsize
outtime=fltarr(arrsize)
outreal=fltarr(arrsize)
outimag=fltarr(arrsize)
nsize=0
endif

; increment nsize and nptstotal
nsize=nsize+1
nptstotal=nptstotal+1
endwhile
print,'Total number points for all channels= ',nptstotal
; close input and output files
close,1
close,2
print,'Unformatted output in ',outfile
end
pro readdata, timedata, realdata, imagdata

; June 1992, L. A. Pflug
; Reads binary phone or beam data from input file.

; open input file
filename = ' ,
print,'Enter input file name.'
read, filename
openr, 1, filename

; get number of channels and points per channel
print,'Enter number of channels in file.'
read, nchannels
print,'Number of channels read = ', nchannels
print,'Enter beginning and ending channel numbers to keep'
print,' - possible 0 to 127'
read, nchannelstart, nchannelend
nchannelkeep = nchannelend - nchannelstart + 1
print,'Number of channels to keep = ', nchannelkeep

; read header line with channel number and number of points
firstline = intarr (2)
readu, 1, firstline
print, firstline
nchannel = firstline (0)
pts = firstline (1)

; points per channel to keep
print,'Number of points per channel = ', pts
print,'Enter number of points per channel to keep.'
read, ptkeep

; initialize input and temporary data arrays
temps = fltarr (pts)
tempreal = fltarr (pts)
tempimag = fltarr (pts)
timedata = fltarr (ptkeep, nchannels)
realdata = fltarr (ptkeep, nchannels)
imagdata = fltarr (ptkeep, nchannels)

; read each phone/beam
for ichannel = 0, nchannels - 1 do begin
if (ichannel gt 0) then begin
; read header line for this channel
readu, 1, firstline
nchannel = firstline (0)
pts = firstline (1)
endif
; read three arrays
readu, 1, temps
readu, 1, tempreal
readu, 1, tempimag
; throw away unwanted points
timedata = temps (0:ptkeep-1)
realdata (0:ptkeep-1, ichannel) = tempreal (0:ptkeep-1)
imagdata (0:ptkeep-1, ichannel) = tempimag (0:ptkeep-1)
; output current status
print,' Data read for channel number - ', nchannel
print,' Number of points per channel - ', pts
endfor

; close input file
close, 1

; throw away unwanted beams
realdata = realdata (*, nchannelstart : nchannelend)
imagdata = imagdata (*, nchannelstart : nchannelend)

end
pro beamform, timedata, realdata, imagdata, anglearray, realbeam, imagbeam

; Uses phone data to time-delay add beamform using phase shifts in frequency.
; Beamformed data is output in realbeam, imagbeam.
; First two phones are desensitized and output as last two beams.

; define number of phones
nphones=128

; define constants
pi = 3.1415926536
twopi=2*pi
; dist = distance between hydrophones in meters
dist = 2.5
; c = sound velocity in meters/second
c=1525.0
;r = sampling rate in points/second
r=128.0
; downshift = downshift in frequency used for basebanding phone data
downshift = 186.0

; Get number of points per phone.
print,'Enter number of points per phone.'
read, ptnum

; Create angle array in degrees.
angleanswer=string(1)
print,'User defined set of angles or default? U-user D=default'
read, angleanswer

; user-defined angles
if (angleanswer eq 'U' or angleanswer eq 'u') then begin
print,'Enter number of angles.'
read, numangles
anglearray=fltarr(numangles)
print,'Enter start angle.'
read, ang
anglearray(0)=ang
if (numangles gt 1) then begin
print,'Enter angle increment (theta) in degrees.'
read, theta
for i=1,numangles-1 do begin
anglearray(i)=anglearray(0)+theta*i
endfor
endif
endif

; default angles (126 angles evenly spaced in cosine)
if (angleanswer eq 'D' or angleanswer eq 'd') then begin
numangles=126
anglearray=fltarr(numangles)
cosincrement=2./125.
cosnumber=1.
for i=0,numangles-1 do begin
anglearray(i)=180*(acos(cosnumber))/pi
cosnumber=cosnumber-cosincrement
endfor
endif
print,'Angle array = ', anglearray

; Create real and complex arrays.
phonedata=complexarr(ptnum, nphones)
frqphonedata=complexarr(ptnum, nphones)
realbeam=fltarr(ptnum, numangles+2)
imagbeam=fltarr(ptnum, numangles+2)

; Adjust for dead phones in Run 5. (9,29,34,58,70,74,100,104,121,126)
ndead=10
deadarray=[9,29,34,58,70,74,100,104,121,126]
for iphone=0,ndead-1 do begin
if (nphones gt deadarray(iphone)) then begin
realdata(*,deadarray(iphone))=(realdata(*,deadarray(iphone)-1) +
+realdata(*,deadarray(iphone)+1))/2.0
imagdata(*,deadarray(iphone))=(imagdata(*,deadarray(iphone)-1) +
+imagdata(*,deadarray(iphone)+1))/2.0
endif
; Convert realdata and imagdata to complex arrays for easier computation
; and calculate Fourier transform of time-domain phone data array.

dt=1/128.
df=1/(ptnum*dt)
for i=2,nphones-1 do begin
    phonedata(*,i)=complex(realdata(*,i),imagdata(*,i))
endfor
freqphonedata(*,i)=fft(phonedata(*,i),-1)

; Create Hamming window positioned evenly over data phones
w=fltarr(nphones)
for i=2,nphones-1 do begin
    w(i)=0.54-(0.46*cos(twopi*(i-2)/(nphones-3)))
endfor

; Create frequency array.
f=fltarr(ptnum)
for j=0,ptnum-1 do begin
    f(j)=(j+(downshift*ptnum/r))*(r/ptnum)
endfor

; Beamform at each angle.
for angle=0,numangles-1 do begin
    ; Shift angle by -90 degrees to simulate broadside as 0 degrees
    anglearray(angle)=-90-anglearray(angle)
    taufactor=(dist/c)*sin(pi*anglearray(angle)/180.)
    ; Initialize sum
    sum=complexarr(ptnum)
    sum=replicate(0.0,ptnum)
    for i=2,nphones-1 do begin
        tau=taufactor*(nphones-1-i)
        argument=complexarr(ptnum)
        for j=0,ptnum-1 do begin
            argument(j)=complex(0.0,-twopi*f(j)*tau)
        endfor
        expfactor=complexarr(ptnum)
        expfactor=exp(argument)
        ; Initialize shiftdata
        shiftdata=complexarr(ptnum)
        shiftdata=w(i)*freqphonedata(*,i)*expfactor
        ; Inverse Fourier transform and sum
        sum = sum + fft(shiftdata,1)
    endfor
    ; Store in appropriate position in beam arrays (possible 0-125)
    realbeam(*,angle)=(float(sum))/(nphones-2)
    imagbeam(*,angle)=(imaginary(sum))/(nphones-2)
    print,'Finished beam for angle ',90.0-anglearray~angle)
endfor

; Move desensitized phones 0 and 1 into last two beams
realbeam(*,numangles)=realdata(*,0)
imagbeam(*,numangles)=imagdata(*,0)
realbeam(*,numangles+1)=realdata(*,1)
imagbeam(*,numangles+1)=imagdata(*,1)

close,1
end
pro hmsource,f,freqhfm

;Create HFM upsweep 210 to 280 Hz source (SOURCE ID SPSS053)
:and downshift by 186 Hz.

;define time duration of source and original sampling rate
sourcetime=2.
origpts=sourcetime*1024.

;initialize original time, freq, and time domain source arrays
torig=fltarr(origpts)
forig=fltarr(origpts)
timehfm=fltarr(origpts)

;define beginning and ending frequencies for source
fl=210.0
f2=280.0

;create original source
constant=(2*3.1415921*3ourcetime*fl*f2)/(f2-fl)
for i=0,origpts-1 do begin
torig(i) = i*dt
forig(i) = i*df
if (i gt origpts/2) then forig(i)=forig(i)-2*fnyq
arg=f2-((f2-fl)*torig(i)/sourcetime )
phi=constant*(alog(f2)-alog(arg))
timehfm(i) =sin(phi)
endfor

;baseband signal by phase shifting in time (Marple, p. 52)
:moves 250 Hz down to 0 Hz
:source will be complex instead of real
timehfm=timehfm
tempfreqhfm=complexarr(origpts)
centerfreq=250.0
const=2*3.1415921*centerfreq*dt
for i=0,origpts-1 do begin
arg=const*i
timehfm(i) = complex(tempfreqhfm(i)*cos(arg),tempfreqhfm(i)*sin(arg))
endfor

;use -64 to 64 Hz filter on freq-domain signal (128 Hz total
:bandwidth with 256 pts)
pts=256

;create new time and frequency arrays, and freq-domain source
f=fltarr(pts)
t=fltarr(pts)
freqhfm=complexarr(pts)
fft time domain source
tempfreqhfm=fft(timehfm,-1)
rearrange order of freq array and freq-domain shifted source (put
:negative freq on left)
fbegl=0
fbeb2=pts/2-1
fendl=origpts-pts/2
fend2=origpts-1
f(f0:pts/2-1)=forig(fbegl:fbeb2)
for f(x:pts/2:pts-1)=forig(fendl:fend2)

;throw away frequencies outside -64 to 64 Hz
freqhfm=freqhfm(0:pts-1)

;inverse fft for new time-domain hfm with appropriate dt
timehfm=complexarr(pts)
timehfm=fft(freqhfm,1)
zero pad to desired time duration
print,'Enter number of points in source signal (sampled at 128 samples/sec).'
read,newpts
oldpts=pts
fulltime=newpts/(oldpts/2)
print,'zero padded time-domain hfm source points = ',newpts
newtimehfm=complexarr(newpts)
newtimehfm(0:oldpts-1)=timehfm(0:oldpts-1)

; transform to get freq domain equivalent
temp=complexarr(newpts)
temp=fft(newtimehfm,-1)

; reverse fft order on source (negative freq on left)
freqhfm=complexarr(newpts)
freqhfm(0:(newpts/2)-1)=temp(newpts/2:newpts-1)
freqhfm(newpts/2:newpts-1)=temp(0:(newpts/2)-1)

; find new dt,df and nyquist freq
dt=fulltime/newpts
df=1./(newpts*dt)
fnyq=1./(2.*dt)
print,'new time increment = ',dt
print,'new freq increment = ',df
print,'new Nyquist freq = ',fnyq

; create new time and frequency arrays and shift frequencies to 0 to 128 Hz.
t=fltarr(newpts)
f=fltarr(newpts)
for i=0,newpts-1 do begin
t(i)=i*dt
f(i)=(i*df)
extfor
end
procedure create CW tone
  ; create one CW tone at a given frequency
  ; and downshift by 186 Hz
  ; define time duration of source and original sampling rate
  ; source time = 2
  ; orig pts = source time * 1024
  ; initialize original time, freq, and time domain source arrays
  ; freq = orig time * orig pts
  ; time CW = orig time * orig pts
  ; determine time and frequency increments and Nyquist frequency of
  ; original source
  ; dt = source time / orig pts
  ; df = l. / (orig pts * dt)
  ; f nyq = l. / (2. * dt)
  ; print, 'time increment = ', dt
  ; print, 'Nyquist = ', f nyq
  ; print, 'freq increment = ', df
  ; input frequency for source
  ; print, 'Enter frequency (210, 220, 230, 240, 250, 260, 270, 280, 290)'
  ; read, freq
  ; create original source with Hanning weighting
  ; pi = 3.1415927
  ; for i = 0, orig pts - 1 do begin
  ;   torig(i) = i * dt
  ;   forig(i) = i * df
  ;   if (i gt orig pts / 2) then forig(i) = forig(i) - 2 * f nyq
  ;   temp = sin(pi * torig(i) / source time) / 2
  ;   time CW(i) = temp * cos(2 * pi * freq * torig(i))
  ; endfor
  ; baseband signal by phase shifting in time (Harple, p. 52)
  ; moves 250 Hz down to 0 Hz
  ; source will be complex instead of real
  ; temp CW = time CW
  ; time CW = complex array (orig pts)
  ; center freq = 250.0
  ; const = 2 * 3.1415927 * center freq * dt
  ; for i = 0, orig pts - 1 do begin
  ;   arg = const
  ;   time CW(i) = complex(temp CW(i) * cos(arg), temp CW(i) * sin(arg))
  ; endfor
  ; use -64 to 64 Hz filter on freq-domain signal (128 Hz total
  ; bandwidth with 256 pts)
  ; pts = 256
  ; create new time and frequency arrays, and freq-domain source
  ; f = fltarr(pts)
  ; t = fltarr(pts)
  ; freq CW = complex array (pts)
  ; fft time domain source
  ; temp freq CW = fft (time CW, -1)
  ; rearrange order of freq array and freq-domain shifted source (put
  ; negative freq on left)
  ; freq beg = 0
  ; f beg = (pts / 2) - 1
  ; f end = orig pts - (pts / 2)
  ; f (0:pts / 2 - 1) = orig (f beg: f beg - 1) + (pts / 2:pts - 1) = orig (f end: f end - 1)
  ; freq CW (0:pts / 2 - 1) = temp freq CW (f beg: f beg - 1) + temp freq CW (f end: f end - 1)
  ; throw away frequencies outside -64 to 64 Hz
  ; freq CW = freq CW (0:pts - 1)
  ; inverse fft for new time-domain CW with appropriate dt
  ; time CW = complex array (pts)
  ; time CW = fft (freq CW, 1)
  ; zero pad to desired time duration
print,'Enter number of points in source signal (sampled at 128 samples/sec).'
read,newpts
oldpts=pts
fulltime=newpts/(oldpts/2)
print,'zero padded time-domain cw source points = ',newpts
newtimecw=complexarr(newpts)
newtimecw(0:oldpts-1)=timecw(0:oldpts-1)

;transform to get freq domain equivalent
temp=complexarr(newpts)
temp=fft(newtimecw,-1)

;reverse fft order on source (negative freq on left)
freqcw=complexarr(newpts)
freqcw(0:(newpts/2)-1) = temp(newpts/2:newpts-1)
freqcw(newpts/2:newpts-1) = temp(0:(newpts/2)-1)

;find new dt,df and nyquist freq
dt=fulltime/newpts
df=1./(newpts*dt)
fnyq=1./(2.*dt)
print,'new time increment = ',dt
print,'new freq increment = ',df
print,'new Nyquist freq = ',fnyq

;create new time and frequency arrays and shift frequencies to 0 to 128 Hz.
t=fltarr(newpts)
f=fltarr(newpts)
for i=0,newpts-1 do begin
  t(i)=i*dt
  f(i)=i*df
endfor
end
program match filters complex data with a user supplied complex frequency-domain source.

constant speed of sound

get number of channels and points per channel
print, 'Enter number of channels in data file.'
read, nchannels
print, 'Enter number of points per channel - should be equal to number of 9 points in source.'
read, ptnum

match filter phone or beam data?
phonebeam=string(1)
print, 'Enter phone or beam data? P or B'
read, phonebeam

initialize output variables
timemf=fltarr(ptnum)
realmf=fltarr(ptnum, nchannels)
imagmf=fltarr(ptnum, nchannels)

fourier transform each channel and match filter source with data in frequency domain (conjugate on source)
newdata=complexarr(ptnum)
fftdata=complexarr(ptnum)
for i=0,nchannels-1 do begin
fftdata(0:ptnum-l)=complex(realbeam(*,i),imagbeam(*,i))
fftdata=fft(fftdata,-1)
newdata=conj(source)*fftdata
newdata=fft(newdata, 1)
realmf(*,i)=float(newdata)
imagmf(*,i)=imaginary(newdata)
endfor

time shift origin to beginning of source transmission
find peaks of matched filtered desensitized phones
if (phonebeam eq 'B' or phonebeam eq 'b') then iplace=nchannels
maxl=max(abs(complexrealmf(*,iplace-2), imagingmf(*,iplace-2))), max_subscript
timel=timedata(max_subscript)
use distance between source and first phone (881 m) to find true time
time2=881.0/c
difference between true time and peak of first mf phone is time shift
tshift=timel-time2
timemf=timedata-tshift
print, 'Time shift = ', -tshift
end
pro waterfall,ti, re, im

; June 1992, L. A. Pflug
; waterfall plot of magnitude versus time for many channels of data
; input file should be read by readdata.pro

!p.region=[0, 0, 1.3, .8]

print, 'Enter number of channels'
read, nchannel
minchannel=0
maxchannel=nchannel-1
print, 'Enter number of points per channel'
read, ptnum

; create variables
nhardcopy=string(1)
magndata=fltarr(ptnum, nchannel)
magntemp=fltarr(ptnum, nchannel)
print, 'Enter number of magnitude points to average.'
read, navg

for ichannel=0, nchannel-1 do begin
; calculate magndata (magnitude of data)
magndata(*, ichannel)=sqrt((re(*, ichannel))^2+(im(*, ichannel))^2)
; average magnitudes of user given number of points
if (navg gt 1) then begin
  for ibeg=0, ptnum-1 do begin
    iend=ibeg+navg-1
    if (iend le ptnum-1) then magndata(ibeg, ichannel)= $total(magndata(ibeg: iend, ichannel))/navg
    if (iend gt ptnum-1) then magndata(ibeg, ichannel)= $total(magndata(ibeg: ptnum-1, ichannel))/navg
  endfor
  endif
; convert magnitude to dB after adding small constant
; minimum dB value is -120 dB for constant=0.000001
magndata(*, ichannel)=magndata(*, ichannel)+.000001
magndata(*, ichannel)=20*alog10(magndata(*, ichannel))
endfor

; make plots
print, 'Begin plots.'

; get title of plot
jumpl: mytitle = ''
print, 'Enter title.'
read, mytitle

; threshold magnitude data
; downshift data to zero for plotting purposes
magntemp=magndata
minmag=min(magndata)
maxmag=max(magndata)
print, 'Minimum magnitude (in dB) = ', minmag
print, 'Maximum magnitude (in dB) = ', maxmag
print, 'Threshold data? Y or N'
threshans=string(1)
read, threshans
if (threshans eq 'Y' or threshans eq 'y') then begin
  print, 'Enter min and max threshold values (dB).'
  read, threshmin, threshmax
  if (threshmin gt min(magndata)) then magntemp(where(magndata lt $threshmin)) = threshmin
  if (threshmax lt max(magndata)) then magntemp(where(magndata gt $threshmax)) = threshmax
  magntemp=magntemp-threshmin
  endif
  if (threshans eq 'N' or threshans eq 'n') then magntemp=magntemp-min(magntemp)

; normalize to unit height and multiply by gain factor for plotting purposes
magntemp=magntemp/max(abs(magntemp))
print, 'Enter gain factor.'
read, gain
magntemp=magntemp*gain
print, 'gain*min, max magnitude = ', min(magntemp), max(magntemp)
print,'Enter first channel and last channel to plot.'
read,nchannelbeg,nchannelend
print,'Enter channel increment (integer).'
read,ndeltachannel

y-axis labels for channels or angles
label=string(1)
print,'Label y-axis with channel number or angle? C or A'
read,ylabel
if (ylabel eq 'C' or ylabel eq 'c') then begin
  ytit='Channel'
endif
if (ylabel eq 'A' or ylabel eq 'a') then begin
  ytit='Angle'
ylabsize=0
  print,'Enter number of labels (integer).'
  read,ylabsize
  ystring=strarr(ylabsize)
  print,strcomp('Enter '+string(ylabsize)+' labels (one per line).')
  read,ystring
endif

;get times to plot
tshift=ti(0)
print,'Time shift in seconds = ',tshift
print,'Min, Max times = ',ti(0),ti(ptnum-1)
print,'Enter min and max time in seconds to plot.'
read,mintime,maxtime

delta=nchannelend
;loop through each channel, and konstant for plotting, and plot
for iloop=nchannelbeg,nchannelend do begin
  ichannel=iloop-nchannelbeg
  ;first plot
  if (iloop eq nchannelbeg) then begin
    konstant=nchannelbeg-1/2.
    if (ylabel eq 'C' or ylabel eq 'c') then plot,ti,$
      magntemp(*, ichannel)+konstant,xrange=[mintime,maxtime],$
      yrange=[nchannelbeg-.5,nchannelend+1],$
      xtitle='Time (sec)',ytitle=ytit,xstyle=1,ystyle=1,$
      title=mytitle
    if (ylabel eq 'A' or ylabel eq 'a') then plot,ti,$
      magntemp(*, ichannel)+konstant,xrange=[mintime,maxtime],$
      yrange=[nchannelbeg-.5,nchannelend+1],$
      xtitle='Time (sec)',ytitle=ytit,xstyle=1,ystyle=1,$
      title=mytitle
  endif
  ;remaining plots
  if (iloop gt nchannelbeg) then begin
    oplot,ti,magntemp(*, ichannel)+konstant
    ndelta=ndelta+ndeltachannel
  endif
  konstant = konstant + 1.
endfor

;output pertinent information to plot
xyouts,/normal,.63,.98,strcomp('Min, Max Magnitude (dB): '+string(minmag)+','$
  +string(maxmag))
if(threshans eq 'Y' or threshans eq 'y') then xyouts,/normal,.63,.95,$
  strcomp('Threshold (dB): '+string(threshmin)+'-'+string(threshmax))
if(threshans eq 'N' or threshans eq 'n') then xyouts,/normal,.63,.95,$
  strcomp('Magnitude points avg: '+string(navg))
xyouts,/normal,.63,.89,strcomp('Gain factor: '+string(gain))
xyouts,/normal,.63,.86,strcomp('Beam increment: '+string(fix(ndeltachannel)))
print,'Plot finished'

;set up for postscript output and repeat plot
nhardcopy='N'
print,'Hardcopy of plot? Y or N'
read,nhardcopy
if (nhardcopy eq 'Y') or (nhardcopy eq 'y') then begin
set_plot,'ps'
postfile=''
print,'Enter name of postscript output file.'
read,postfile
device,/landscape,/inches,xsize=9.0,ysize=6.5,filename=postfile
ndelta=nchannelbeg
for iloop=nchannelbeg,nchannelend do begin
  ichannel=iloop-minchannel
  if iloop eq nchannelbeg then begin
    konstant=nchannelbeg-1/2.
    if (ylabel eq 'c' or ylabel eq 'C') then plot,ti,$
      magntemp(*,ichannel)+konstant,xrange=[mintime,maxtime],$
      yrange=[nchannelbeg-.5,nchannelend+1],$
      xtitle='Time (sec)',ytitle=ytit,xstyle=1,ystyle=1,$
      title=mytitle
    if (ylabel eq 'a' or ylabel eq 'A') then plot,ti,$
      magntemp(*,ichannel)+konstant,xrange=[mintime,maxtime],$
      yrange=[nchannelbeg-.5,nchannelend+1],$
      xtitle='Time (sec)',ytitle=ytit,xstyle=1,ystyle=1,$
      title=mytitle,ytickname=ystrin,ytick3=ylabsize-1
  endif
  if (iloop gt nchannelbeg) then begin
    if (iloop eq ndelta+ndeltachannel) then begin
      oplot,ti,magntemp(*,ichannel)+konstant
    endif
    konstant=konstant+1.
  endfor
endif
output pertinent information to plot
xyouts,/normal,.63,.98,strcompress('Min, Max Magnitude (dB): '+string(minmag)+'+','$
  +string(maxmag))
if(thresholds eq 'Y' or thresholds eq 'y') then xyouts,/normal,.63,.95,$
  strcompress('Threshold (dB): '+'string(threshmin)+'-','+string(threshmax))
if(thresholds eq 'N' or thresholds eq 'n') then xyouts,/normal,.63,.95,$
  strcompress('Threshold (dB): '+'None')
xyouts,/normal,.63,.86,strcompress('Magnitude points avg: '+'string(navg))
xyouts,/normal,.63,.89,strcompress('Gain factor: '+'string(gain))
xyouts,/normal,.63,.86,strcompress('Beam increment: '+'string(fix(ndeltachannel))
xyouts,/normal,.63,.83,strcompress('Time Shift: '+'string(tshift))
reinitialize device variables
empty
device,/close
print,'Postscript file created.'
set_plot,'x'
endif
;loop through for more plots
print,'Another plot? Y or N'
another=string(1)
read,another
if (another eq 'Y' or another eq 'y') then goto,jump1
end
pro unformatoutput, timeout, realout, imagout

;August, 1992 - L. A. Pflug
;Write real and imaginary parts of data to PV-WAVE unformatted file.
;Channels written sequentially.

;determine size of matrices
realsize=size(realout)
ptnum=fix(realsize(1))
umchannels=fix(realsize(2))

print,'Real part:
print,strcompress('number points = '+string(ptnum))
print,strcompress('number of channels = '+string(numchannels))

imagsize=size(imagout)
ptnum=fix(imagsize(1))
umchannels=fix(imagsize(2))

print,'Imaginary part:
print,strcompress('number points = '+string(ptnum))
print,strcompress('number of channels = '+string(numchannels))

;open output file
outfile=''
print,'Enter name of output file.'
read,outfile
openw,1,outfile

;write data to file
for ichannel=0,numchannels-1 do begin
  writeu,1,ichannel,ptnum
  writeu,1,timeout
  writeu,1,realout(*,ichannel)
  writeu,1,imagout(*,ichannel)
  print,strcompress('Channel number '+string(ichannel)+' written to file.')
endfor
print,'Output in file = ',outfile

;close file
close,1
end
pro asciioutput,timeout,realout,imagout

:August, 1992 - L. A. Pflug
:Write real and imaginary parts of data to ascii file.
:Channels written sequentially.

:determine size of matrices
timesize=size(timeout)
ptnum=fix(timesize(1))
print,'Time array:'
realsize=size(realout)
ptnum=fix(realsize(1))
numchannels=fix(realsize(2))
print,'Real part:'
numchannels=fix(imagsize(1))
numchannels=fix(imagsize(2))
print,'Imaginary part:'

; determine format of output
print,'Ascii output form:'
print,1=(real part, imaginary part)"'
print,'- compatible with pvconvert2arr.pro'
print,2=(beam number, time sample, real part, imaginary part)"'
print,'- compatible with pvconvert4arr.pro'
print,'Enter 1 or 2.'
read,nform

:open output file
outfile="'
print,'Enter name of output file.'
read,outfile
openw,1,outfile

;write data to file
for ichannel=0,numchannels-1 do begin
  for ipoint=0,ptnum-1 do begin
    if (nform eq 1) then printf,l,realout(ipoint,ichannel),S
    imagout(ipoint,ichannel)
    if (nform eq 2) then printf,l,ichannel,timeout(ipoint),S
    realout(ipoint,ichannel),imagout(ipoint,ichannel)
  endfor
  print,strcompress('Channel number '+'string(ichannel)+' written to file.')
endfor
print,strcompress('Output in file '+outfile)

;close file
close,1
end