PERGAMON

Computers & Geosciences 28 (2002) 501–511

COMPUTERS & GEOSCIENCES

www.elsevier.com/locate/cageo

DSISoft—a MATLAB VSP data processing package ☆, ☆☆

K.S. Beaty[a], G. Perron[b], I. Kay[b], E. Adam[b,*]

[a] *University of Alberta, Avadh Bhatia Physics Laboratory, University of Alberta, Edmonton, Alberta, Canada T6G 2J1*
[b] *Geological Survey of Canada, Continental Geosciences Division, Seismology and Electromagnetism Section, 615 Booth Street, Ottawa, ON Canada K1A 0E9*

Received 30 October 2000; received in revised form 2 April 2001; accepted 3 June 2001

Abstract

DSISoft is a public domain vertical seismic profile processing software package developed at the Geological Survey of Canada. DSISoft runs under MATLAB version 5.0 and above and hence is portable between computer operating systems supported by MATLAB (i.e. Unix, Windows, Macintosh, Linux). The package includes processing modules for reading and writing various standard seismic data formats, performing data editing, sorting, filtering, and other basic processing modules. The processing sequence can be scripted allowing batch processing and easy documentation. A structured format has been developed to ensure future additions to the package are compatible with existing modules. Interactive modules have been created using MATLAB's graphical user interface builder for displaying seismic data, picking first break times, examining frequency spectra, doing $f-k$ filtering, and plotting the trace header information. DSISoft modular design facilitates the incorporation of new processing algorithms as they are developed. This paper gives an overview of the scope of the software and serves as a guide for the addition of new modules.

Keywords: Vertical seismic profile; Seismic; Data processing; MATLAB; Freeware; Interactive

1. Introduction

The downhole seismic imaging (DSI) consortium, a partnership between the Geological Survey of Canada, Canadian universities, and Falconbridge, INCO and Noranda mining companies, was established with the aim of applying vertical seismic profiling (VSP) techniques to mineral exploration. The two objectives of the consortium are: (1) to detect mineral deposits using the seismic reflection method utilizing 3-component receivers located in exploration boreholes and (2) to transfer the technology to the mining, exploration, and geophysical services industries. Software development was included in the mandate of the consortium because it was recognized that existing packages do not contain the non-standard processing techniques that would be required. Most seismic processing software packages are proprietary and it is difficult for the end-user to add specialized modules. Having an affordable package that includes these non-standard techniques will be important in order to facilitate technology transfer at the end of the consortium lifetime. To address this shortcoming, considerable effort has been made to prepare a customized VSP processing software package, DSISoft.

DSISoft has been developed as free software with a GNU license, allowing public access to the source code. The package may be used for any purpose including educational, research and commercial applications. Users are encouraged to modify and add to the existing package to suit their specific needs. Another free

☆ Code available at http://www.cg.NRCan.gc.ca/dsisoft/ and at http://www.iamg.org/CGEditor/index.htm

☆☆ Geological Survey of Canada Contribution 2000166.

*Corresponding author. Tel.: +1-613-947-1579; fax: +1-613-943-9285.

E-mail address: eadam@nrcan.gc.ca (E. Adam).

0098-3004/02/$ - see front matter
PII: S0098-3004(01)00073-5

software package for processing seismic data, Seismic Un*x (also known as SU), is widely available (Murillo, 1996; Stockwell, 1997, 1999; Templeton and Gough, 1999). SU was developed in American National Standards Institute (ANSI) C and runs on Unix-linux platforms whereas DSISoft was developed in MATLAB. SU works well for processing large seismic datasets because compiled codes run faster but it is more difficult to add new modules to the package and especially graphic interface programs. DSISoft, on the other hand, is a specialized package for VSP processing. It is unsuitable for large datasets because of the large memory requirement for MATLAB, but is well suited for testing new techniques, for developing graphic interface programs, and for processing small sets of data. SU has the advantage of running and compiling with only public domain software while DSISoft requires a MATLAB license. Stockwell (1997) provides an extensive discussion of SU and the concept of free software.

MATLAB was chosen as a platform for DSI software development for several reasons as it provides a technical computing environment that combines numeric computation, visualization and a higher-level programming language. Code developed in MATLAB is portable between operating systems (e.g. Unix, PC, and Macintosh). MATLAB was designed to allow easy extensibility. Its built in functions and graphics capabilities provide a useful environment for testing new algorithms. With technology transfer in mind, choosing an affordable and flexible software package should be attractive to smaller geophysical service companies and academic institutions. MATLAB has proved to be a useful tool for geoscience applications (Marcotte, 1991; Tian et al., 1993; Yeung and Chakrabarty, 1993; Marcotte, 1996). Marcotte (1991) provides a review of some of the literature discussing MATLAB.

DSISoft contains sufficient modules to process VSP data, along with interactive modules for displaying seismic data, picking first break times, examining frequency spectra, performing $f-k$ filtering, and plotting the information contained in trace headers. This paper outlines the development strategy, the processing and interactive modules that are currently available, applications of the software, and plans for future improvements.

2. Development strategy

2.1. DSI variable format

Software development requires a variable structure to be defined. MATLAB works most efficiently on rectangular matrices, but it also supports cell arrays and structured variables, which have no symmetry requirements and can contain other matrices, cell arrays or structured variables within themselves. The DSI variable format, developed in MATLAB 5.0, combines all of these variable types. It is a structured variable with a *.fh* extension for the file header, a *.th* extension for the trace headers, and a *.dat* extension for the seismic traces. The file header is a cell array of arbitrary length containing information pertaining to the whole file, for example, start time, end time, number of points per trace, and sampling rate. A complete list of file header word numbers and descriptions is given in Table 3. The trace header and data parts of the variable are cell arrays, where each cell represents a record. Each record contains a matrix of seismic traces, where each column is a separate trace, and a matrix of corresponding trace headers. The trace headers contain information specific to each trace, for example, the shot location, receiver location, pick times, and kill flags. Each trace header contains 64 values. A list of trace header word numbers and descriptions can be found in Table 4. Fig. 1 shows the organization of a DSI variable and explains the syntax for accessing elements of the variable.

DSISoft has a module for reading data in SEG2 format (the standard format for engineering type seismographs) directly into DSI format within MATLAB. Modules for data exchange between ITA/Insight processing package format, ASCII text, and SEGY format (Barry et al., 1975), the oil industry standard for seismic data exchange, have also been developed.

DSI variables can be stored as *.mat* files using MATLAB's save command. This method preserves the structure of the variable and the variables can be read into the workspace from the file by using the 'load' command. It is possible to save single or multiple variables in each *mat* file. The 'save' command also provides the option to write out a variable as an ASCII file but the structured variable and cell array must first be broken down into matrices to use this option. The module 'mat2asc' does this automatically.

2.2. Processing modules

The DSI software package is set up so that the majority of the data processing can be done using a script file, which must have the file extension *.m*, that calls a series of independent processing modules. This approach makes batch processing possible, which is desirable for running large jobs and for documenting the processing steps performed on the data. Each module can also be spawned from the command line. Fig. 2 shows an example of a processing script. Notice that the first line calls 'dsi_start' and the last line calls 'dsi_end', which together, print out software information and time the processing sequence. In the example, script a variable called 'data' is passed in and out of a series of modules and along the way intermediately processed

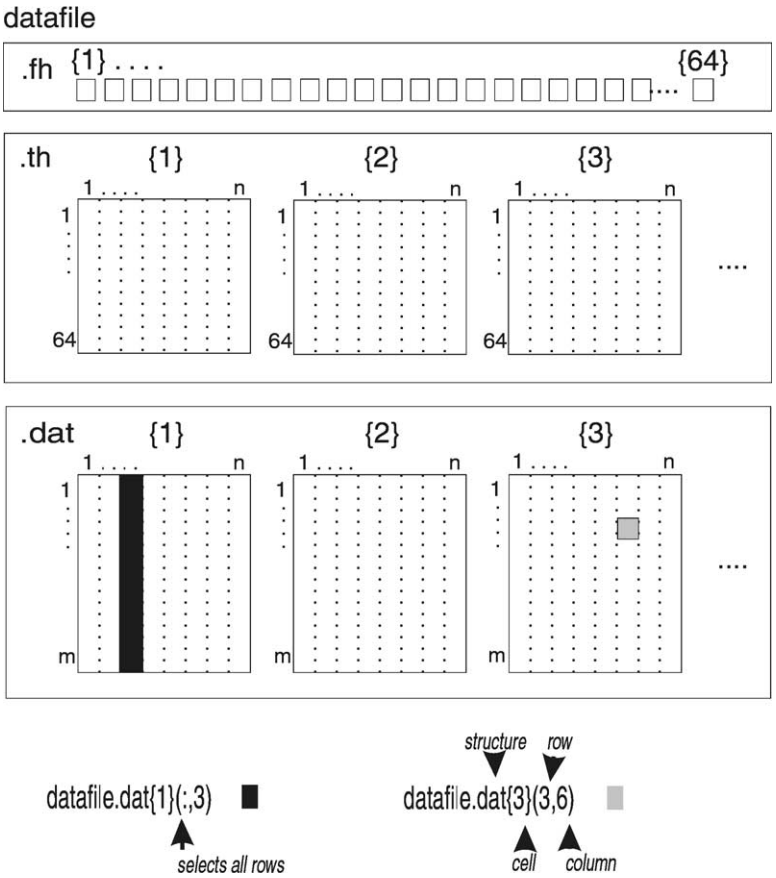

Fig. 1. Diagram showing structure of DSI variable and how to call elements of variable within MATLAB.

data files are saved under different names. This particular script is used for removing the downgoing *P*-wave from a test dataset. The results of this process is shown in Fig. 3.

Table 1 gives a summary of processing modules currently available. The concepts behind most of these modules and the application to seismic processing are described in DiSiena et al. (1984), Hardage (1985) and Yilmaz (1987). Discussions of adaptive filtering techniques for removing power line noise (used in 'harmon') are in Adam and Langlois (1995), Butler and Russell (1993) and Nyman and Gaiser (1983). DSISoft has sufficient modules to process a VSP dataset and provides a good basis on which to develop and test new processing strategies.

Some conventions have been developed regarding the form of a processing module. A coding style has been adapted as shown in Fig. 4. The function call is a single line that takes on the form:

```
function[dataout] = function_name
  (datain, parameter1, parameter2, ...)
```

The variables 'dataout' and 'datain' always refer to the output and input DSI seismic data files, respectively. The other parameters are values that should be specified by the user. Parameters are passed in the function call to allow batch processing. The next section is the help text. In MATLAB, typing 'help' followed by the name of a *.m* file displays the first block of consecutive lines that all start with a percent sign (%), the symbol used for comments. The echo section of the module displays the function call when the module runs, allowing the user to track progress as a script runs. The last section of a DSI module is the body. This is where manipulation of the data, the file headers, and the trace headers takes place. In most modules, with the exception of 'fkfilt' and 'rot3c_eig', the processing is applied to all records in the dataset. The module 'subset' may be used to limit the dataset and process fewer records if desired. Programmers should keep in mind that code runs more smoothly if large matrices are initialized before being used, and runs fastest when operating on matrices as opposed to looping over vectors, so an effort should be made to optimize the code wherever possible.

DSISoft script

```
%
%This DSIsoft script generates data used in Figure 3
%
dsi_start
load test_adap

%
% Data processing
%
[data]=butw(test_adap,20,40,140,160,3,15);
[data]=tred(data,-6000.0,0.01);
[data]=ener(data,0.006,0.039);
[data]=trim(data,0.006,0.039,0.01);
[data]=trim(data,0.006,0.039,0.01);
[data]=trim(data,0.006,0.039,0.01);
[test_flat]=trim(data,0.006,0.039,0.01);

%
% Apply a 29 trace median filter to estimate the
% downgoing P-wave
%
[test_med]=medi_filt(test_flat,29);

%
% Subtract the estimated downgoing P-wave from the
% original dataset
%
[test_sub]=subr(test_med,test_flat);

%
% save data in separate variables
%
save test_med.mat test_med
save test_flat.mat test_flat
save test_sub.mat test_sub

%
% Delete temporary variable
%
clear data

dsi_end
```

Fig. 2. Typical script for removing downgoing *P*-wave from simple VSP.

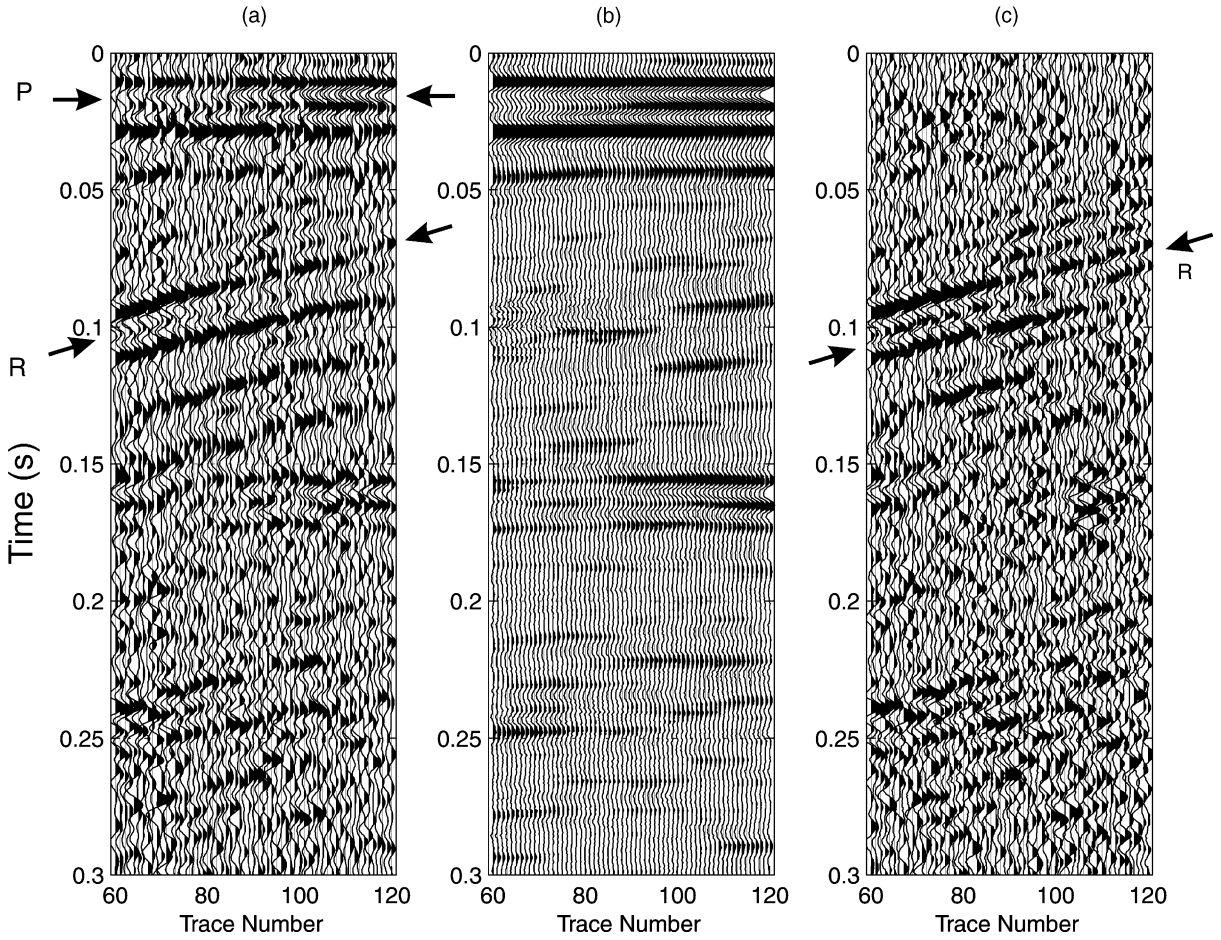

Fig. 3. Results of applying script shown in Fig. 2 to VSP dataset: (a) first breaks are flattened (test_flat); (b) median filter (test_med) applied to a; (c) result (test_sub) of subtracting b from a. Notice that the first breaks are no longer apparent.

The processing module shown in Fig. 4 performs energy balancing. Energy balancing assures that every trace has, within a specified time window, the same amount of energy. The amplitudes of each trace are normalized by the square root of the total energy for that trace occuring within the specified time window. This technique can be used to account for energy variations due to differences in shot size, variability in the coupling of shot energy, and increasing depth of receiver.

2.3. Interactive modules

Interactive modules are useful for processing VSP data, particularly for visualizing traces and frequency spectra, picking first break times and selecting $f-k$ filtering polygons. A summary of interactive modules along with their functional descriptions is given in Table 2. The interactive modules have been designed using MATLAB's graphic user interface (GUI) builder tool. As with the processing modules, a strategy has been developed for the general design of a GUI module. Interactive modules generally utilize a menu window which pops up first and has text, editable text, and pop-up menus for loading a dataset and selecting plotting options. A second window is then called up by the first and displays one or more axes on the left and buttons, sliders, editable text, and pop-up menus on the right, with similar functions grouped together (Fig. 5). Layout can be achieved by clicking and dragging objects using the GUI builder. Each object has a list of properties that can be accessed and changed using the property editor or by using the 'findobj', 'get', and 'set' functions at the command line. Usually the 'units' property of the figure and all of its objects should be set to 'normalized' so that

Table 1
DSISoft processing modules

Module name	Purpose
dsi_start	Prints software info and the data, starts timer
dsi_end	Prints out time elapsed since 'dsi_start' was called
agc	Automatic gain control
asc2mat	Converts ASCII files of seismic traces to DSI format
autocor	Autocorrelation of each trace
bandpass	Linear bandpass filter
bison2mat	Reads BISON data and converts to DSI format
b_rise	Calculates rise time of first breaks
butw	Butterworth bandpass filter (uses signal processing toolbox)
decons_ls	Spiking deconvolution, least squares (Weiner) method
ener	Energy balancing
equa	Spectral equalization
fkfilt	Apply $f-k$ filter
harmon	Adaptive filter useful for removing electrical noise
ita2mat	Converts seismic data from ITA to DSI format
kill	Flag bad traces
mat2asc	Saves a DSI variable as ASCII files
mat2segy	Saves a DSI variable in SEGY format
medi_filt	Median filter
medirm	Removes a downgoing wave using a median filter based on either first break pick times or velocity and shot static
merge_files	Merges two DSI variables into one
mix	Lateral trace mixing
mute	Mutes data either before, after, or between specified time(s)
noch	Linear notch filter
pack_good	Removes traces flagged by 'kill' from the dataset
profil	Gives profile of a DSI variable
rot3c_eig	Rotates two components to maximize energy on one using Eigenvalue method
rot3c_deg	Rotates two components to examine energy reflected from a specific direction
seg2mat	Reads SEG2 data and puts it into DSI format
segy2mat	Reads SEGY data and puts it into DSI format
seisplot	Plots seismic traces, called by most of the graphic interface modules
sft	Shifts all traces by a specified time
sortrec	Sorts records and traces within records by any trace header word
subr	Subtracts one set of traces from another
subset	Used for making a subset of data
thread	Reads specified trace headers and saves contents in an ASCII file
thwrite	Writes values stored in an ASCII file to specified trace headers
tred	Performs time shifts according to wave velocity and shot-receiver offset to flatten first breaks (linear moveout correction)
trim	Residual static correction
tune_xcorr	Cross-correlation based algorithm for tuning first break pick times
unflat	Restores flattened data according to first break pick times

the modules can be used on machines with different screen sizes.

After the windows have been satisfactorily laid out, they need to be made functional. This can be achieved using the 'callback' properties of the objects. Code in the callback function is executed each time an object is selected by the user. All of the code can be stored in the callbacks, but a superior method of organizing the callbacks is to create a separate function that is controlled by case statements and holds the code for all of the functions. There are many advantages to this method: it is easier to modify callbacks, comments can be added to annotate the code, and repetition can be reduced if two callbacks are similar or the same. In addition, variables created within the code are invisible to the workspace and help text can be added to give instructions on how to use the module. A callback function might look like the following:

```
function function_name(action)
%put some help text here
if nargin==0%if there are no input arguments
 menufig; %open up menu
 return;
end %if
switch action
case 'plot'
 %put code here that opens main window,
 %looks at all of the plotting parameter selections in
menu window
 %plots accordingly
 case 'quit'
 close all;
end. %switch
```

Then the 'callback' property of the plot button of the figure would simply be 'function_name plot;' and this would execute multiple lines of code required to accomplish the task. Not every object needs a callback. For example, the menu shown in Fig. 5 has 13 objects that can be modified by the user but only three actually have callback functions because the 'plot' callback looks at all of the other objects to find parameters before plotting. The code for the display module, 'dispseis' (Fig. 5) is an example of how callbacks can be organized into a case controlled function.

Note that when programming callback functions, they cannot see or create variables in the workspace. This is desirable because it means that the workspace will not become cluttered but makes it more difficult to keep track of parameters. Two solutions to the problem of seeing and passing variables to the functions are either to have the function look at the properties of objects to find parameter values or to use global variables, which are visible to any function that acknowledges their existence. The display module shown in Fig. 5 has an editable text box where a record number can be specified. The callback for this object plots the data stored in the specified record. In order to do this, the callback must somehow find the plotting parameters (e.g. scale factor, trace numbers, time scale, etc.). One way to do this is to have the callback look at the 'string' properties of all of the editable text boxes containing pertinent information. Alternately, the callback for the 'plot' push button on the menu could save all of these parameters as a cell array or structured variable in the 'userdata' property of a figure or an object, thereby making it accessible to any callback function. This second option may be more efficient in cases where parameters from several objects are needed (see 'dispseis' code for an example of this method). Using global variables is another option that makes the code simpler. An example of how they can be used can be found in the picking modules (e.g. pickfb, pick1comp) where global variables are used to store pick times and kill flags because these variables need to be accessed repeatedly by a number of callback functions.

3. Future development and outlook

Software development for the DSI consortium has reached the stage where there are sufficient modules to process data from a simple VSP. Most of the necessary standard processing modules have been implemented in MATLAB along with some modules that are not usually available in other VSP processing packages. Interactive modules have been developed for picking first breaks on three components or else on a single record, displaying traces, displaying frequency spectra, performing $f-k$ filtering, and plotting information stored in trace headers. Modules have been developed to read data from SEG2, SEGY, ASCII, and ITA formats. It is possible to write out SEGY format, MATLAB format and ASCII format. A module has

Table 2
DSISoft graphic interface modules

Module name	Purpose
aspec	Shows plots of amplitude and phase versus frequency
dispseis	Display module for looking at seismic traces
fkpoly	Plots the $f-k$ spectrum of seismic traces and allows interactive picking of a polygon to be used for $f-k$ filtering with options to look at filtered results of passing or rejecting contents of polygon
pick1comp	Interactive first break picking on one component with options to tune, kill traces, and flatten according to pick times
pickfb	Interactive first break picking simultaneously on three components with options to tune according to component with most energy, kill traces, flatten according to pick times, and rotate components on the fly
plothd	Plots contents of trace headers in two or three dimensions as selected from a menu that identifies the significance of each trace header word

Table 3
File header words

Word number	Significance
1	Number of traces contained in file
2	Line number (from SEGY)
3	Reel number (from SEGY)
4	Seismic attributes variable flag (1 for seismic attributes, 0 or [] for seismic data
5	Currently unused
6	Currently unused
7	Number of samples per trace
8	Sampling interval (s)
9	Trace start time (s)
10	Trace end time (s)
11	Currently unused
12	Number of records in file
13	Maximum record fold

Table 4
Trace header words

Word number	Significance
1[a]	Original trace number
2	Field file ID (FFID)
3	CDP number
4	Recording component
5	Angle of rotation
6	Trace status (flags dead traces)
7	Trace polarity
8	Original channel number
9	Shot-receiver azimuth
10	Rise time (s)
11	Currently unused
12*	Number of traces in record
13	Trace number within record
14	Bit mask for picks
15	Pick times 1
16	Pick times 2
17	Pick times 3
18	Pick times 4
19	Pick times 5
20	Elevation static shift
21	Refraction shot static
22	Refraction receiver static
23	Residual shot static
24	Residual receiver static
25	Trim static
26	Shot point number
27	Receiver point number
28	Shot receiver offset
29	Source northing
30	Currently unused
31	Source easting
32	Currently unused
33	Source elevation
34	Currently unused
35	Receiver northing
36	Currently unused
37	Receiver easting
38	Currently unused
39	Receiver elevation
40	Currently unused
41	CDP northing
42	CDP northing bin number
43	CDP easting
44	CDP easting bin number
45	CDP elevation
46	CDP elevation bin number
47	CMP northing
48	Currently unused
49	CMP easting
50	Currently unused
51	Datum elevation
52	Replacement velocity
53	Source-to-receiver offset
54	Shot depth
55	Uphole time
56	Wireline depth
57	Total residual shot static
58	Total residual receiver static
59	Ray parameter
60	Stack fold
61	Total datum static
62	Total refraction shot static
63	Total refraction receiver static
64	Total static

[a] It is important to preserve these headers.

been developed to find reflection points from uniformly dipping and striking beds for given shot and receiver geometries in three dimensions. Our future plan is to develop a 3D common depth point transform algorithm to use on DSI data. Other modules will be developed as non-standard processing techniques will be explored. The DSI software will expand to include programs that reflect other aspects of the DSI consortium as well. For example, MATLAB is a useful environment for modeling the seismic response of ore bodies. Already a version of 'bmod3d' (Eaton, 1997), a 3D seismic modeling program based on the Born approximation, exists in MATLAB format. A set of programs for presenting sonic and density logs with borehole lithology, written in MATLAB, also exists. The DSI software being released at present provides a solid platform that allows for rapid prototyping of new processing, modeling and imaging schemes.

```
function [dataout]=ener(datain,T1,T2)
```

Function call

```
%[dataout]=ener(datain,T1,T2)
%
%This function will do energy balancing on the traces in datain based
%on the amplitudes of the data in the time window between times T1
%and T2 (both in seconds).  Each trace is multiplied by a factor equal to
%the sum over the specified time window of one over the amplitude squared
%of that trace.
%
%DSI customized VSP processing software
%Version 1
%written by Kristen Beaty Nov. 1997
%last modified April 28, 1998
```

Help Text

```
%Copyright (C) 1998 Seismology and Electromagnetic Section/
%Continental Geosciences Division/Geological Survey of Canada
%
%This library is free software; you can redistribute it and/or
%modify it under the terms of the GNU Library General Public
%License as published by the Free Software Foundation; either
%version 2 of the License, or (at your option) any later version.
%
%This library is distributed in the hope that it will be useful,
%but WITHOUT ANY WARRANTY; without even the implied warranty of
%MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
%Library General Public License for more details.
%
%You should have received a copy of the GNU Library General Public
%License along with this library; if not, write to the
%Free Software Foundation, Inc., 59 Temple Place - Suite 330,
%Boston, MA  02111-1307, USA.
%
%DSI Consortium
%Continental Geosciences Division
%Geological Survey of Canada
%615 Booth St.
%Ottawa, Ontario
%K1A 0E9
%
%email: dsi@cg.nrcan.gc.ca
```

License

```
disp('[dataout]=ener(datain,T1,T2)')
```

Echo

```
dataout=datain;
tstart=datain.fh{9}; %start time in seconds
int=datain.fh{8}; %sampling interval in seconds
npts=datain.fh{7}; %number of points per trace

%find indexes of T1 and T2
T1=round((T1-tstart)./int)+1;
T2=round((T2-tstart)./int)+1;
nrec=datain.fh{12}; %number of records in datain

for COUNT=1:nrec
  x=datain.dat{COUNT}(T1:T2,:);
  x=x.^2; fact=sum(x); fact=fact.^0.5;
  i=find(fact==0); fact(i)=1; %avoid divide by zero error for dead traces
  factgr=meshgrid(fact,1:npts);
  dataout.dat{COUNT}=datain.dat{COUNT}./factgr;
end %loop over records
```

Body

Fig. 4. Code for typical processing module. This module performs energy balancing on set of VSP data.

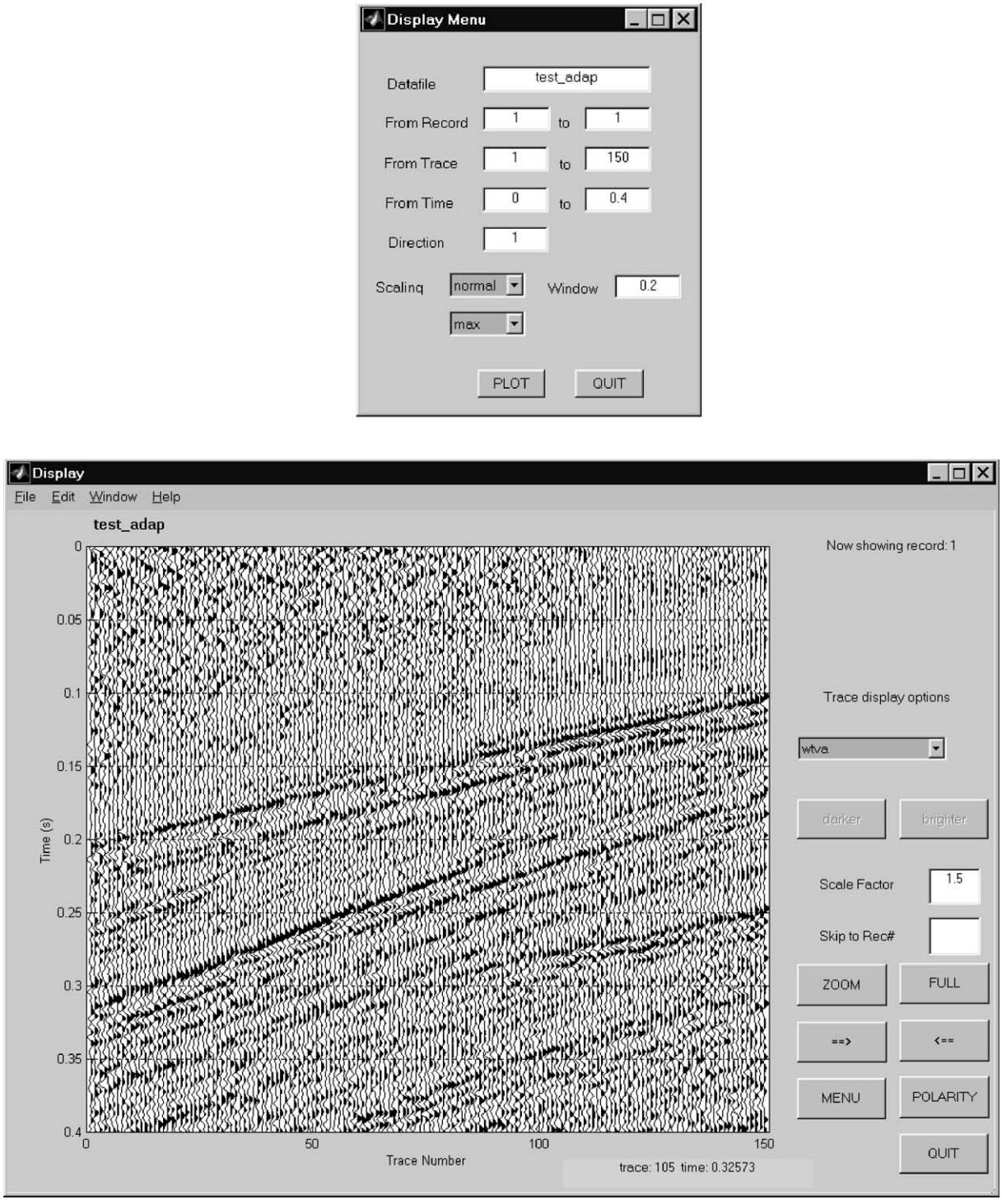

Fig. 5. Menu and display for GUI module 'dispseis'.

4. Conclusions

DSISoft was created to facilitate the development of new VSP processing techniques and to ensure that these techniques can be transferred to industry and to other researchers. The DSISoft package has been released as free software under a GNU license by the Geological Survey of Canada to promote its use and improvement by others. The package contains numerous basic processing modules as well as several customized interactive programs forming a solid framework for standard VSP processing and future development. New modules will be developed as the research of the DSI consortium progresses.

5. Software availability

DSISoft can be down loaded from http://www.cg.NRCan.gc.ca/dsisoft/. DSISoft is free software that can be redistributed and/or modified under the terms of the GNU Library General Public License as published by the Free Software Foundation. Use of DSISoft is not limited to educational purposes and research; it may be used for commercial applications.

Acknowledgements

Development of this software was funded by a collaborative research agreement between the Continental Geosciences Division of the Geological Survey of Canada, INCO, Falconbridge, and Noranda Inc. The first author acknowledges support from the co-op and FSWEP student employment programs of the Government of Canada. David Eaton, Brian Roberts, and John McGaughey have contributed code to DSISoft. Comments from I. Asudeh, D. Boerner, and B. Roberts have improved this manuscript. The authors are grateful to INCO, Falconbridge, and Noranda for allowing the public release of DSISoft.

References

Adam, E., Langlois, P., 1995. Elimination of monofrequency noise from seismic records. Lithoprobe Seismic Processing Facility Newsletter 8, 59–65.

Barry, K.M., Cavers, D.A., Kneale, C.W., 1975. Recommended standards for digital tape formats. Geophysics 40, 344–352.

Butler, K.E., Russell, R.D., 1993. Subtraction of powerline harmonics from geophysical records. Geophysics 58, 898–903.

DiSiena, J.P., Gaiser, J.E., Corrigan, D., 1984. Horizontal components and shear wave analysis of three-component VSP. In: Toksöz, M.N., Stewart, R.R. (Eds.), Vertical Seismic Profiling Part B: Advanced Concepts. Geophysical Press, London, pp. 177–188.

Eaton, D.W., 1997. BMOD3D: a program for three-dimensional seismic modeling using the Born approximation. Open File 3357, Geological Survey of Canada, Ottawa, Ontario, 25pp.

Hardage, B.A., 1985. Vertical Seismic Profiling Part A: Principles, 2nd ed. Geophysical Press, London, 509pp.

Marcotte, D., 1991. Cockriging with MATLAB. Computers & Geosciences 17 (9), 1265–1280.

Marcotte, D., 1996. Fast variogram computation with FFT. Computers & Geosciences 22 (10), 1175–1186.

Murillo, A.E., 1996. DSU: distributed parallel processing with seismic unix. Society of Exploration Geophysicists, 66th Annual International Meeting. Abstracts volume, Denver, USA, pp. 997–1000.

Nyman, D.C., Gaiser, J.E., 1983. Adaptive rejection of high-line contamination. Society of Exploration Geophysicists, 53rd Annual International Meeting. Abstracts volume, Tulsa, USA, pp. 321–323.

Stockwell Jr., J.W., 1997. Free software in education: a case study of cwp/su: seismic un*x. The Leading Edge 16 (7), 1045–1049.

Stockwell Jr., J.W., 1999. The cwp/su: seismic un*x package. Computers & Geosciences 25 (4), 415–419.

Templeton, M.E., Gough, C.A., 1999. Web seismic un*x: making seismic reflection processing more accessible. Computers & Geosciences 25 (4), 421–430.

Tian, D., Sorochian, S., Myers, D.E., 1993. Correspondence analysis with matlab. Computers & Geosciences 19 (7), 1007–1022.

Yeung, K., Chakrabarty, C., 1993. An algorithm for transient pressure analysis in arbitrarily shaped reservoirs. Computers & Geosciences 19 (3), 391–397.

Yilmaz, Ö., 1987. Seismic data processing. Society of Exploration Geophysicists, Tulsa, OK, 526pp.