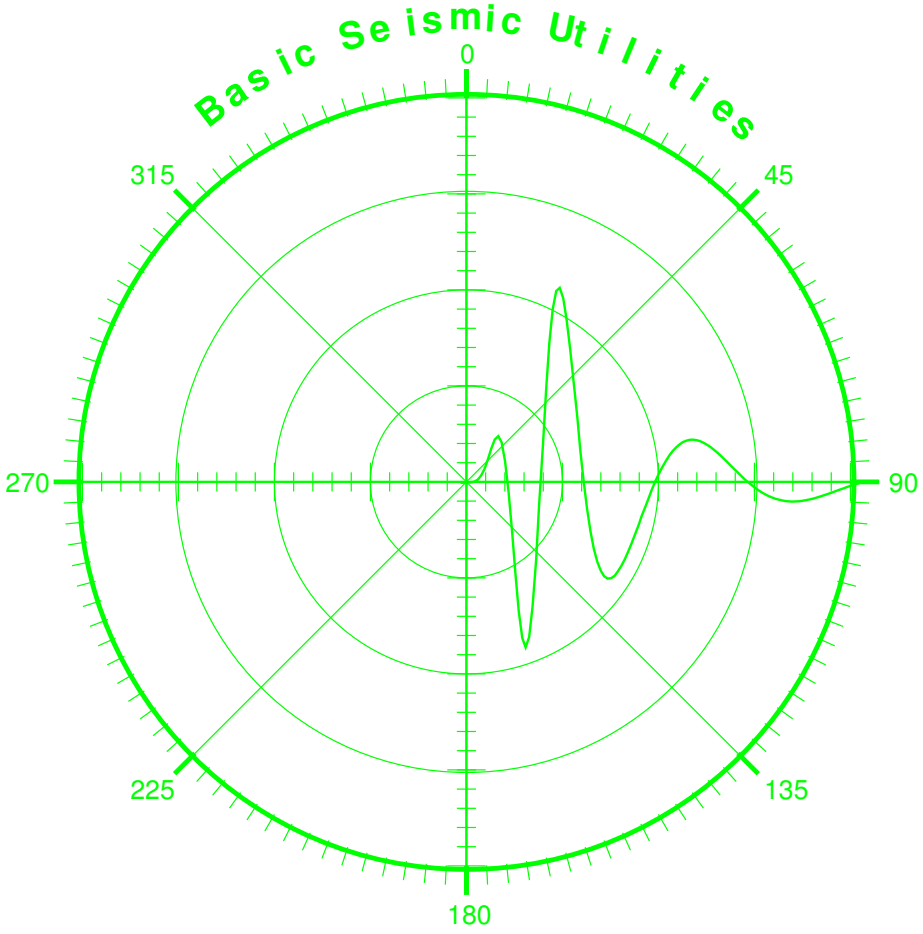
Basic Seismic Utilities User's Guide

Dr. P. Michaels, PE
<paulmichaels@boisestate.edu>

October 19, 2020
Version 3.0.2

Software for Engineering Geophysics



Boise State Geosciences, Engineering Geophysics
Boise State University
Boise, Idaho 83725

Acknowledgements

This software is an updated version of the original release that was done during a sabbatical leave 7 years ago. On this sabbatical, the goal has been portability. That is, while some new programs have been added, much of the software has been carried forward, and thus is now available for use on a variety of operating systems. Thus, I remain indebted to the work of many others in the development of this package. I would like to thank Enders A. Robinson and the Holden-Day Inc., Liquidation Trust (1259 S.W. 14th Street, Boca Raton, FL 33486, Phone: 561.750-9229 Fax: 561.394.6809) for license to include and distribute under the GNU license subroutines found in Dr. Robinson's 1967 book [19] , Multichannel time series analysis with digital computer programs. This book is currently out of print, but contains a wealth of algorithms, several of which I have found useful and included in the BSU Fortran77/gfortran subroutine library (sublib4.a). This has saved me considerable time.

In other cases, subroutines taken from the book Numerical Recipes [17] had to be replaced (the publisher did not give permission to distribute). While this is an excellent book, and very instructional for those interested in the theory of the algorithms, future authors of software should know that the algorithms given in that book are NOT GNU. Replacement software was found in the *GNU Scientific Library (GSL*), and in the *CMLIB*.

For plotting, I remain indebted to the developers of *PLPLOT*. *PLPLOT* credits have grown to be too many to list. However, there are a number of instances where I ran into dependency problems with some operating systems, particularly the Microsoft family. So I have added GNUPLOT alternatives.

Where there was a need to solve for eigenvalues, or invert a matrix, I have relied on *LAPACK*. This excellent package is well worth installing, and I acknowledge the contributions of the many authors of *LAPACK* and *BLAS*.

Lastly, the author acknowledges financial support over the years from various clients. Financial support included that provided by grant DAAH04-96-1-0318 from the U.S. Army Research Office. Views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Army Research Office or the U.S. Government. This material is also based upon work supported by the National Science Foundation under Grant No. 0321233. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation. Other support has been provided by the Idaho State Board of Education, Boise State University Sabbatical Committee, Idaho Transportation Department, and Idaho Power. Again, views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the views of those who have provided the author support.

Copyright

Contents

List of Figures

1 Description of BSU

Basic Seismic Utilities (BSU) is a collection of seismic signal processing programs. Also included in BSU are some modeling programs (computation of the near-field, and solution to Lamb's problem). These programs are written in Fortran 77 and the C-language. At the time of this writing, the software community is moving from Fortran77 to "gfortran". The g77 codes compile well under gfortran, and the only major issue is linking to the correct versions of libraries like "lapack" and "gsl". Libraries must be compiled with the same compiler that you will be using for compiling BSU. Since PLPLOT no longer supports g77, some codes have been converted to Fortran 90. When looking at source code, if the file name ends in *.F90, it is Fortran 90, and if it ends in *.f, the code format will be the older style. The configuration script should handle most situations. Tips for different Linux distributions can also be found in the README_* files included in the TAR archive (the * represents your operating system distribution or environment, like Debian, Ubuntu, CentOS, Mingw32, etc).

The binary data file format of BSU (BSEGY) is derived from SEG-Y (omit the reel header, 240 byte trace header, floating point data, 4 bytes per sample value). This format is compatible with the Seismic Unix (SU) (Cohen [3]) package (Colorado School of Mines, CWP), but differs in some optional header definitions. While SU is designed primarily for CDP reflection data, BSU is designed for engineering geophysical surveys (down-hole, refraction, near field, and surface waves). The BSU package is well suited for 3-component data collection (headers include source and geophone polarization information). Crooked line and data acquisition in irregular patterns are easily handled by geometry setting procedures which integrate electronic distance measuring survey files (NEZ) with the file formats from common engineering seismographs (SEG-2 and BISON). The BSU package also includes Octave|Matlab procedures for reading the BSU binary file format, BSEGY.

These codes were written over a number of years, some as early as 1967 (26 keypunch, BCD, later 29 keypunch, EBCD, up to the present state of affairs). Do not be surprised that the code is a bit patch worked and mixed language. It was not designed, but evolved. As I look backward, a lot has changed since my first connection between two computers in the military and the present day Internet with the WWW.

Finally, BSU evolved to fill needs I have had as a practicing engineering geophysicist, and to serve my needs in research and education. Like SU, it is meant to be compiled, modified, and extended. If you read further, you will find that the Fortran and C-language masters (*bmst.f* and *cmst.c*) provide enough of an example to have one programming in BSU within a day. There are usually 3 steps:

1. Read a trace (*bsegin.f* or *c_bsegin.c*)
2. Do some computations.
3. Write a trace (*bsegout.f* or *c_bsegout.c*)

1.1 What's New or Changed

This release is a complete rebuild and debug of earlier versions. There have been significant changes to make the project more easy to use for Microsoft users (cross-compiling on Linux to create *.exe binaries), and a number of new coding decisions have been made for both Linux and Microsoft users. In addition, the project has been successfully compiled on Apple's Macbook Pro platform. Upgrade to Debian 10 from Debian 9 requires changes again to PLPLOT library configurations.

- **PLPLOT** The pkg-config files have changed names for Fortran. Debian 9 used **plplotd-f95.pc**. Now, Debian 10 packages use the file name **plplot-fortran.pc**.
- **Program seg2dump.c** Does a raw dump of SEG-2 formatted files. Dumps to a text file exactly how numbers stored before applying any corrections (as in the case with egg2seg or topcon2).
- **Program genwaw.c** A geometry setting program which is helpful for walk-a-way shooting and non pattern situations. Use with SEG-2 format data.
- **Octave | Matlab** Scilab programs have been replaced with Octave programs. In most cases, Matlab will also run these with just a few exceptions. The exceptions are primarily those cases which depend on disper.oct (FwdR1.m, invR1.m moho.m, mastercurve.m and rayleigh.m).

- **disper.oct** This is an ELF LSB shared object which should be compiled after installing BSU on an operating system. The **build_disper.oct** script is found with all the *.m files in either the source code (Octave directory) or when installed, in the `/usr/local/share/octave/site-m` directory.

- **Fortran 90 vs g77** Since PLPLOT no longer supports Fortran 77, several programs were converted to Fortran 90. Thes programs include: **bamx.F90, bamp.F90, bhod.F90, caplot.F90, bvax.F90, bvas.F90, genwav.F90, and waves.F90**. All the Fortran 77 codes remain with the *.f suffix.

- **Gnuplot** Due to PLPLOT dependency problems on some platforms and the incompatibilities that have developed during revisions of PLPLOT, the configuration script now permits using Gnuplot as an alternative to PLPLOT. The default is to compile with Gnuplot. To compile with PLPLOT, the user must decide on the new or old version of PLPLOT. Old in this context would be version 5.9.9-5. An instance of the New PLPLOT would be version 5.10.0. If you are running the Old version, the configuration command would be **configure –with-plplotlib –with-plplot-old**. If you are running with the New version, the command would be **configure –with-plplotlib**. NOTE: with these options, two dashes precede the "with" and one dash "plplot".

- **Cygwin replaced with Mingw32** Microsoft windows users often had problems recognizing the need to install Cygwin to run BSU codes. So Cygwin has been dropped, most of the codes have been cross-compiled for Microsoft windows using Mingw32. The result is a collection of static compiled files ending in the suffix *.exe. These can be run directly on a Microsoft platform from a Microsoft terminal (Power Shell is preferred). The static built binaries are a bit larger than they would be with shared libraries, but that greatly reduces the number of dependency issues. Still, it is highly recommended that Microsoft users install a windows version of Gnuplot to take full advantage of the graphics capabilities. It is also recommended that Microsoft users install a windows version of Mingw32 to take advantage of the many linux commands which become available in Power Shell (but this is only a recommendation, the *.exe files should run on their own).

- **Program bis2seg.c** The conversion from Bison formatted data to BSEGY format code has been fixed to correctly convert **Bison Floats**. The integer format worked OK, but Bison floats are quite different from most standards (Bison uses a 16 bit 2's complement mantissa, 4 bit exponent normalized float, in effect a 2.5 word format in the context of a Bison data bus). Conversion of Bison floats requires examining the 4 bit exponent (is it greater than 7? If so, then is a negative number). This 20 bit code revision was inspired by Jens Hartmann code found in Center for Wave Propagation, CWP Third Party codes found in Seismic Unix. However, the Hartmann code may need revision to work with floats (as did mine). The problem is that Bison float data are fairly rare in the wild. I only discovered this problem when I encountered some Bison float data.

- **Program ba2s.c** Revised code, converts ASCII format file to a BSEGY binary file, row or column order for the time axis.

- **Program seg2csv.c** New code to convert from BSEGY to Comma Separated Variable (CSV spreadsheet format).

- **Program tplt.c** New code to plot a single signal in a multi-trace BSEGY file using Gnuplot (no PLPLOT version of this). The result is an interactive plot in the window system, and an output file, **graph.gp**, which is in the Gnuplot language. One can edit the graph.gp file for a different output terminal or file type.

- **Program qplt.c** New code to quickly plot all signals in a BSEGY file using Gnuplot (no PLPLOT version of this). The result is an interactive plot in the window system, and an output file, **qgraph.gp**, which is in the Gnuplot language. One can edit the qgraph.gp file for a different output terminal or file type.

- **Program seg2txt.f** Converts BSEGY data to ASCII text file, time in row order.

- **Miscellaneous** See ChangeLog file for many other minor changes. The ChangeLog file is packaged with the source code archive.

2 Obtaining and Installing BSU

The BSU package can be down loaded from my web page at https://www.boisestate.edu/earth/staff-members/paul-michaels/ or at https://173.255.241.228. Click on the download link and find the format for your distribution or operating system.

2.1 Package Managers

- **APT Package Manager** – For **Debian, Mint, Ubuntu, Xubuntu**, there is a *.deb package.
- **Slackpkg Package Manager** – For **Slackware**, the best approach is to build from the source (see next section).
- **Pacman Package Manager** – For **Arch** the best approach is to build from the source (see next section).
- **RPM, Yast2, Zypper Package Manager** – For **CentOs, Redhat Enterprise, OpenSuse** there is an *.rpm package.
- **Apple Package Management** – For users with Apple products like the **MacBook Pro**, the best approach is to build from the source. The Apple environment has a number of package managers (Fink, Brew, MacPorts), and detection of the Darwin system is included in the configuration script. Because there are multiple possible package management instances in a single system, the user is cautioned to avoid crashing installed directories. The next section gives an example of how installation on a **MacBook Pro** can be done.
- **Microsoft Package Manager** – There is no install wizard. For **Windows XP or Windows 8.1**, the user may download a ZIP archive of static compiled *.exe files. These have been cross-compiled on a Linux host using the MingW32 tool chain. Simply download **BSU_EXEC.zip** and unzip the archive in an executable directory which is in your execution path. The man pages are available for download in **man-bsu3-html.zip** and can be viewd using a web browser. The Octave scripts are available in **Octave.zip**, but some codes will not work if they require **disper.oct**. To build that, you will need a windows equivalent of **mkoctfile** on your system, in addition to a windows version of Octave. I recommend that the windows user run the codes in a PowerShell environment. Further, to increase the utility of the codes, I recommend that the user install a windows MingW32 package. CAUTION: Windows is not a true case sensitive system, and both upper and lower case names will be viewed as being the same.
- **Chrome Books and APT Package Management** – For Chrome book users, the source can be compiled if developer mode is installed. Be aware of your CPU (Intel or Arm), since you will want to make sure any dependencies are met with the same CPU. I was able to compile on a Samsung Chrome Book set to developer mode with **Crouton** and **XFC4 window manager**, and an **armv7l CPU**. See the next section for more details.

2.2 Compile Tips

If you wish to compile the codes, a traditional Linux location will be in the /usr/local directory tree (I untar the archive in /usr/local/src). This produces the top directory in /usr/local/src/bsu-3.0.2, and that is where you will issue the configure command. To learn about alternative options, issue the command “configure –help”. For example, to compile with the GNUPLOT alternative, just “configure” will work. To use the PLPLOT options, the configure command would be “configure –with-plplotlib ”. In addition, you might also need the old version of PLPLOT libraries. In that case, “configure –with-plplotlib –with-plplot-old” would be the command to invoke.

2.2.1 Most Common Problem

The most common problem will be an unmet dependency. These are often caught by the configuration process. The solution on Linux is to use your package manager and add a missing package (you need gfortran to compile fortran for example). The other issue may be a missing library. For example, on Debian 8 (Jessie), the bplt.c code requires the libshp-dev package be installed. Different Linux distributions package things in different ways, so read the error message if you get one, and start looking for the missing dependency.

2.3 Building From Source

The following gives my experience on a number of systems when building BSU from the source tar archive. These tips and comments will have a limited lifetime since the build environment and dependencies will always be in a state of change over time.

Dependencies like Plplot and Gnuplot are difficult to support in a single configure script, since these can change at any moment. In particular, Plplot libraries have at times even changed the names of some function calls in a non-backward compatible way. The current configure script includes an option to use Gnuplot as an alternative to Plplot if one becomes desperate. But even Gnuplot has introduced some changes, so one should note which versions of Gnuplot and Plplot exist on the user's system.

2.3.1 Dependencies

The following should be installed on the system:

- libblas and (libblas-dev see last item below)
- liblapack and liblapack-dev
- libgsl, gsl-bin (GNU scientific library)
- libgfortran, gfortran (compiler for fortran)
- gcc (compiler for C-language)
- gnuplot (plotting package)
- plplot (best plots, but is optional, default configure uses gnuplot)
- libshp (needed by plplot)
- Development packages for libraries (blas, lapack, gsl, plplot) These will end with -dev or -devel in the package name for APT or RPM respectively.

2.3.2 General Directions

The recommended location to untar the archive file, `bsu-3.0.2.tar.gz`, is `/usr/local/src`. There are two major ways to compile BSU software. While some codes will always use GNUPLOT for some graphics, PLPLOT will often result in the best results for those codes that have this as an option. While you could change into directory bsu-3.0.2 and then issue build commands, an alternative is to leave that directory alone and create some alternative build directories that refer back to that directory (see section 2.3.3 below for details. However, if you wish to work in the bsu-3.0.2 directory, these will be the build commands:

configure *options if any*

make

make install

sudo ldconfig
(this last command may be needed and directs the system to use the shared libraries that are built)

NOTE: On linux, have the **locate** command installed and update the database before configure (**update db**) for best results.

2.3.3 Debian 10 Buster

This is the build environment most likely to work (it is my OS). Refer to the general directions, 2.3.2, above when opening the TAR archive.

2.3.3.1 Dependencies (at time of this writing)

```
Install PLPLOT, BLAS, LAPACK development packages and locate package
liboctave-dev  (needed for mkoctfile)
libblas-dev:amd64
libblas3:amd64
libgslcblas0:amd64
liblastfm5-1:amd64
libopenblas-base:amd64
liblapack-dev:amd64
liblapack-doc
liblapack-doc-man
liblapack3:amd64
liblapacke:amd64
liblapacke-dev:amd64
libplplot-dev
libplplot-lua:amd64
libplplot-ocaml
libplplot16:amd64
libplplotcxx14:amd64
libplplotfortran0:amd64
libplplotqt2:amd64
libplplotwxwidgets1:amd64
octave-plplot:amd64
plplot-doc
plplot-driver-qt
plplot-driver-wxwidgets:amd64
plplot-driver-xwin:amd64
plplot-tcl
plplot-tcl-bin
plplot-tcl-dev
python3-plplot
python3-plplot-qt
```

2.3.3.2 Directories to Create Untar **bsu-3.0.2-tar.gz** in `/usr/local/src`. This will create a directory, **/usr/local/src/bsu-3.0.2**. Then make one of the following directories to do the build in:

```
/usr/local/src/bsu+plplot
/usr/local/src/bsu+gnuplot
```

Choose which way to build and install.

- **bsu+gnuplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

  ```
  ../bsu-3.0.2/configure  >&configure.out
  make >&make.out
  make install  >&install.out
  sudo ldconfig
  ```

- **bsu+plplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

  ```
  ../bsu-3.0.2/configure --with-plplotlib  >&configure.out
  make  >&make.out
  make install  >&install.out
  sudo ldconfig
  ```

2.3.4 Slackware

Follow the general directions, 2.3.2, above when opening the TAR archive. These instructions are based on version 14.2 of Slackware. If you don't have BLAS, LAPACK, and PLPLOT installed yet, these dependencies should be built first. PLPLOT uses **cmake**, and you may need to make sure that your version is compatible with the PLPLOT version. Build the dependencies as follows in **/usr/local/src**.

- **plplot-5.11.1** First build plplot shared libraries. To do this, untar and cd into plplot-5.11.1

```
mkdir build_dir
cd build_dir
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../ >& cmake.out
make >&make.out
make install >&install.out
```

- **BLAS** Build BLAS static library, untar BLAS-3.8.0 and cd into BLAS-3.8.0

```
make Makefile
cp blas_LINUX.a  /usr/local/lib/libblas.a
```

- **LAPACK** Build LAPACK and more BLAS libraries, untar lapack-3.8.0, cd into lapack-3.8.0

```
ln -s make.inc.example make.inc
make all

[Note: it may bomb on testing, Makefile has specific targets]
When done with any extra make targets, should have
liblapack.a
libtmglib.a
librefblas.a
libcblas.a
Can install with copy command:

cp lib*.a /usr/local/lib
```

- **BSU-3.0.2** Untar bsu-3.0.2 in /usr/local/src, then build in either a gnuplot or plplot directory. ONLY install one or the other, gnuplot or plplot. Best graphics with plplot.
  - **bsu+gnuplot**

```
mkdir /usr/local/src/bsu+gnuplot
cd bsu+gnuplot
../bsu-3.0.2/configure  >&configure.out
make
make install
sudo ldconfig
```

  - **bsu+plplot**

```
mkdir /usr/local/src/bsu+plplot
cd bsu+plplot
../bsu-3.0.2/configure --with-plplotlib  >&configure.out
make >&make.out
make install >&install.out
sudo ldconfig
```

2.3.5 Arch Linux

Follow the general directions, 2.3.2, above when opening the TAR archive.

- **plplot-5.15.0** Build plplot shared libraries, untar and cd into plplot-5.15.0. Note, **cmake** is like **configure** in other build situations. WARNING: When building plplot with cmake, more than one instance may exist after installing the library. The BSU configure script uses "locate" and looks for the first instance, so you may wish to remove the plplot in the build directory, leaving only the installed to be found. It may work without this, but maybe not.

```
mkdir build_dir
cd build_dir
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../ >&cmake.out
make >&make.out
make >&install.out
```

- **xfig** Build xfig if you wish to use it, untar xfig-3.2.7 and then cd into xfig-3.2.7

```
configure
make
make install
```

- **bsu-3.0.2** Build bsu-3.0.2, untar. Then build in either a gnuplot or plplot directory. Untar bsu-3.0.2.tar.gz in /usr/local/src directory. Decide on building with either PLPLOT or GNUPLOT. ONLY install one or the other, gnuplot or plplot. Best graphics with plplot.

  - **bsu+gnuplot** From /usr/local/src:

```
mkdir bsu+gnuplot
cd bsu+gnuplot
../bsu-3.0.2/configure  >&configure.out
make >&make.out
make install  >&install.out
sudo ldconfig
```

  - **bsu+plplot** From /usr/local/src:

```
mkdir bsu+plplot
cd bsu+plplot
../bsu-3.0.2/configure --with-plplotlib  >&configure.out
make  >&make.out
make install  >&install.out
sudo ldconfig
```

2.3.6 CentOS 7

Follow the general directions, 2.3.2, above when opening the TAR archive. The tested build was done on **Kernel 3.10.0-1127.19.1.el7.x86_64**, 64 bit system.

2.3.6.1 Dependencies (at time of this writing)

Install PLPLOT, BLAS, LAPACK and development packages:

```
lapack-3.4.2-8.el7.x86_64
lapack-devel-3.4.2-8.el7.x86_64
lapack-static-3.4.2-8.el7.x86_64
blas-3.4.2-8.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
blas-static-3.4.2-8.el7.x86_64
plplot-5.10.0-10.el7.x86_64
plplot-ada-5.10.0-10.el7.x86_64
plplot-ada-devel-5.10.0-10.el7.x86_64
plplot-devel-5.10.0-10.el7.x86_64
plplot-doc-5.10.0-10.el7.noarch
plplot-fortran-devel-5.10.0-10.el7.x86_64
plplot-java-5.10.0-10.el7.x86_64
plplot-java-devel-5.10.0-10.el7.x86_64
plplot-libs-5.10.0-10.el7.x86_64
plplot-lua-5.10.0-10.el7.x86_64
plplot-perl-5.10.0-10.el7.x86_64
plplot-pyqt-5.10.0-10.el7.x86_64
plplot-qt-5.10.0-10.el7.x86_64
plplot-qt-devel-5.10.0-10.el7.x86_64
plplot-tk-5.10.0-10.el7.x86_64
plplot-tk-devel-5.10.0-10.el7.x86_64
plplot-wxGTK-5.10.0-10.el7.x86_64
plplot-wxGTK-devel-5.10.0-10.el7.x86_64
xfig-3.2.5-44.c.el7.x86_64
xfig-common-3.2.5-44.c.el7.x86_64
```

2.3.6.2 Directories to Create

Untar **bsu-3.0.2-tar.gz** in `/usr/local/src`. This will create a directory, **/usr/local/src/bsu-3.0.2**. Then make one of the following directories to do the build in:

```
/usr/local/src/bsu+plplot
/usr/local/src/bsu+gnuplot
```

Choose which way to build and install.

- **bsu+gnuplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

  ```
  ../bsu-3.0.2/configure  >&configure.out
  make >&make.out
  make install  >&install.out
  sudo ldconfig
  ```

- **bsu+plplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

```
../bsu-3.0.2/configure --with-plplotlib  >&configure.out
make  >&make.out
make install  >&install.out
sudo ldconfig
```

2.3.7 CentOS 8

At the date of this writing, several problems came up with CentOS_8. The plplot RPMs compiled with BSU, but the PDF driver did not work. One can use the PostScript (PS) driver and then use the **ps2pdf** program. Also, GhostView (GV) program could not display any Postscript files, regardless of where they come from. However, the *.ps files generated could be displayed with program **evince** in CentOS 8. The Postscript files appear OK when displayed on other operating systems. There is an alternative approach, see 2.3.7.1. However, if you want to use the current PLPLOT RPM's, then the list of RPMS needed are:

```
ocaml-plplot.x86_64
ocaml-plplot-devel.x86_64
plplot.x86_64
plplot-data.noarch
plplot-devel.x86_64
plplot-doc.x86_64
plplot-fortran-devel.x86_64
plplot-java.x86_64
plplot-java-devel.x86_64
plplot-libs.x86_64
plplot-lua.x86_64
plplot-pyqt.x86_64
plplot-qt.x86_64
plplot-tk.x86_64
plplot-tk-devel.x86_64
plplot-wxGTK.x86_64
plplot-wxGTK-devel.x86_64
lapack.x86_64
lapack-devel.x86_64
blas.x86_64
blas-devel.x86_64
openblas.x86_64
openblas-srpm-macros.noarch
openblas-threads.x86_64
gsl.x86_64
gsl-devel.x86_64
gnuplot.x86_64
gnuplot-common.x86_64
xfig.x86_64
transfig.x86_64
octave.x86_64
octave-devel.x86_64
```

2.3.7.1 Alternative Approach One can download the latest PLPLOT source archive and compile it. But to build it, you need a newer version of CMAKE. Files to download at the time of this writing from the home web pages of CMAKE and PLPLOT projects are:

```
cmake-3.18.3.tar.gz
plplot-5.15.0.tar.gz
```

- **CMAKE** Untar the source in /usr/local/src. Then,

```
cd cmake-3.18.3
mkdir build
cd build
../bootstrap && make  && sudo make install
```

- **PLPLOT** Untar the source in /usr/local/src. Then,

```
cd plplot-5.15.0
mkdir build
cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../ >& cmake.out
make
make install
cd ..
rm -r -f build   (to avoid multiple instances of plplot-fortran.pc, etc)
sudo updatedb    (so configure calls to locate can find *.pc files etc)
```

  NOTE: The **jpeg** driver was not available, so a work-a-round is to manually reply with SVG when programs like BPLT pause. Clearly, there is room for improvement with CENTOS 8. No problems like this with CENTOS 7 or other operating systems like Debian 10.

- **BSU-3.0.2** Build as in the case for CENTOS7 (see 2.3.6.2).

2.3.8 Chrome Book

Follow the general directions, 2.3.2, above when opening the TAR archive. The build was done on **Kernel 3.8.11 armv7l**, 64 bit system.
Switch to developer mode.
https://www.chromium.org/chromium-os/developer-information-for-chrome-os-devices
Then install Crouton and the Ubuntu Trusty release.
https://github.com/dnschneid/crouton
Crouton – Switch to a linux crosh terminal (**Ctrl+Alt+T**) and **sudo startxfce4**.

2.3.8.1 Dependencies Install the following packages if you installed the Trusty release of Ubuntu.

```
gsl-bin
libgsl0-dev
libgsl0ldbl
cl-plplot
libplplot-c++11
libplplot-dev
libplplot-fortran11
libplplot-java
libplplot-lua
libplplot-ocaml
```

```
libplplot12
plplot-doc
plplot-tcl
plplot-tcl-dev
plplot12-driver-gd
plplot12-driver-qt
plplot12-driver-wxwidgets
plplot12-driver-xwin
python-plplot
python-plplot-qt
liblapack-dev
liblapack3
libblas-dev
libblas3
gnuplot
gnuplot-nox
gnuplot-x11
xfig
xfig-libs
octave
octave-common
liboctave-dev
liboctave2:armhf
```

Untar **bsu-3.0.2.tar.gz** in the **/usr/local/src** directory. Then make one of the following directories to do the build in:

```
/usr/local/src/bsu+plplot
/usr/local/src/bsu+gnuplot
```

Choose which way to build and install.

- **bsu+gnuplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

  ```
  ../bsu-3.0.2/configure  >&configure.out
  make >&make.out
  make install  >&install.out
  sudo ldconfig
  ```

- **bsu+plplot** Change into this directory, then issue commands. The *.out files record what happened if there is a problem.

  ```
  ../bsu-3.0.2/configure --with-plplotlib  >&configure.out
  make  >&make.out
  make install  >&install.out
  sudo ldconfig
  ```

2.3.9 MacBook Pro

Follow the general directions, 2.3.2, above when opening the TAR archive.

- The tested build was done on **OSX Sierra, Darwin, kernel 16.1.0**, 64 bit system. From the top directory one runs configure with appropriate options. Choose one of the following (note: options start with a double dash "- -" before the "with" and a single dash after the "with":

  configure
  (will use gnuplot5 instead of plplot)

MACBOOK TIPS
The fact that there are more than one package manager that might be used, and that dependencies might be installed in different locations makes a configure script challenging. The following gives the choices that will work with the current configure script when run on a MacBook Pro.

Fink Package Management – Packages are installed under /sw directory. This is very good and avoids conflict with other install locations from other package managers like Homebrew or Macports. The default install location of BSU is `/usr/local`.

X11 Environment – I installed XQuartz to get X11 running on Darwin. *WARNING: if you update your system, you likely will need to reinstall XQuartz since Apple's update procedures may break XQuartz.*

GSL: GNU Scientific Library – This test installed GSL using Fink. Configuration builds make files with `AM_CFLAG= -I/sw/include` and `GSL_LIB=-L/sw/lib`

Macport installs under /opt directory tree.

Homebrew installs under `/usr/local`. This can be a problem if BSU is also installed under `/usr/local` (the default). WARNING: If Octave is installed using Homebrew, it likely will break gfortran for the BSU build.

Fortran and Darwin – Configure tests if Darwin, then compiles most of fortran codes with -static flag, sublibF4 since libsubF4.so does not pass common block /header/ to main programs.

2.3.10 Redhat Enterprise

Follow the general directions, 2.3.2, above when opening the TAR archive. This instance is for Red Hat Enterprise Linux Server release 7.7 (Maipo) Kernel: Linux 3.10.0-1062.4.1.el7.x86_64 .

- **Dependencies** Install MLOCATE, BLAS, LAPACK development packages using the package manager, like yum:

```
mlocate-0.26-8.el7.x86_64
blas-3.4.2-8.el7.x86_64
blas-devel-3.4.2-8.el7.x86_64
lapack-3.4.2-8.el7.x86_64
lapack-devel-3.4.2-8.el7.x86_64
```

- **PLPLOT** Build and install plplot [Note cmake-2.8.12.2-2.el7.x86_64 is available with this OS, so you can't use the most recent plplot 5.15.0 which requires a more advanced version of cmake]

```
untar plplot-5.11.0
cd plplot-5.11.0
mkdir build_dir
cd build_dir
cmake -DCMAKE_INSTALL_PREFIX=/usr/local ../ >&cmake.out
    [check cmake.out for any missing dependencies]
```

```
make >&make.out
    [check make.out for any problems]
make install >&install.out
    [check install.out for any problems]
```

Depending on what else is installed, you may not get full functionality on the plplot devices (like jpg), but you will likely get postscript. Be sure to install following for xwin device:

```
wxBase-2.8.12-20.el7.x86_64
wxGTK-2.8.12-20.el7.x86_64
wxGTK-devel-2.8.12-20.el7.x86_64
wxGTK-gl-2.8.12-20.el7.x86_64
wxGTK-media-2.8.12-20.el7.x86_64
```

- **bsu-3.0.2** Untar bsu-3.0.2.tar.gz in /usr/local/src. Then build in either a gnuplot or plplot directory. ONLY install one or the other, gnuplot or plplot. Best graphics with plplot.
  - **bsu+gnuplot** From /usr/local/src:

    ```
    mkdir bsu+gnuplot
    ../bsu-3.0.2/configure  >&configure.out
    make >&make.out
    make install  >&install.out
    sudo ldconfig
    ```

  - **bsu+plplot** From /usr/local/src:

    ```
    mkdir bsu+plplot
    ../bsu-3.0.2/configure --with-plplotlib \
    PKG_CONFIG_PATH=/usr/local/lib/pkgconfig >&configure.out
    make  >&make.out
    make install  >&install.out
    sudo ldconfig
    ```

2.3.11 Microsoft Windows

BSU is best run from a PowerShell. There are several options.

- A static build is available on the website, BSU_EXEC.zip. This is a precompiled version that just includes binaries and assumes that gnuplot has been installed on the Windows platform. Unzip it in a directory that is in your execution path.
- Cross-compile on a Linux system using MingW32. For this option, download Mingwbsu-3.0.2.tgz and untar in /usr/local/src. Then change into the top directory.

  make

  make install

  This will create a directory, EXEC, with all the *.exe binaries. For this to work, you will need to install the MingW32 tool chain on your Linux platform.
- You can try taking the linux source tar archive and build it in cygwin. I have dropped cygwin support in bsu-3.0.2 because most Microsoft users find cygwin a challenge, and don't understand the relationship between cygwin and Windows. Most Microsoft users will rather just install the *.exe files and be done with it. Man pages are in *man-bsu3-html.zip*, Octave scripts in *Octave.zip*.

2.4 Octave

There are some octave programs included with BSU-3.0.2/Octave that will need to build an ELF LSB shared object that extends the octave package. There is an executable script, `build_disper_oct` that can be run to create disper.oct. This shared object is required for the following programs:

FwdR1.m – Computes Rayleigh wave forward problem, compares to data.

moho.m – Computes Rayleigh wave forward problem for a built in velocity profile.

rayleigh.m – Simple computation of Rayleigh wave forward problem example.

invR1.m – Inverse Rayleigh wave inversion with data.

mastercurve.m – Computes Rayleigh wave master curves for layer over half-space.

To build the shared object, be sure to have octave development package installed. For Debian users, this is `liboctave-dev-3.8.2-4.deb` or later.

2.5 What to download

If you just wish to run BSU, download a binary package that matches your packaging system or compile from the tar archive. If you plan on modifying or extending BSU, then you will want to download either a suitable source package, or the source TAR archive.

Operating System	TAR (Source) File	Binary Package	Source Package
Debian 10 (Buster)	bsu-3.0.2.tar.gz bsu-3.0.2.tar.gz.asc	bsu_3.0.2-1_amd64.deb bsu_3.0.2-1_amd64.deb.asc bsu_3.0.2-1+plplot_amd64.deb bsu_3.0.2-1+plplot_amd64.deb.asc bsu_3.0.2-1+static_amd64.deb bsu_3.0.2-1+static_amd64.deb.asc	bsu_3.0.2-1.dsc bsu_3.0.2.orig.tar.gz
Debian 7 and 8	bsu-3.0.2.tar.gz	none	none
Ubuntu	bsu-3.0.2.tar.gz	none	none
CentOS 7	bsu-3.0.2.tar.gz bsu-3.0.2+plplot.tar.gz	bsu-3.0.2-1.x86_64.rpm bsu-3.0.2+plplot-1.x86_64.rpm	bsu-3.0.2-1.src.rpm bsu-3.0.2+plplot-1.src.rpm
CentOS 8	bsu-3.0.2.tar.gz bsu-3.0.2+plplot.tar.gz	bsu-3.0.2-1.x86_64.rpm bsu-3.0.2+plplot-1.x86_64.rpm	bsu-3.0.2-1.src.rpm bsu-3.0.2+plplot-1.src.rpm
Redhat Enterprise	bsu-3.0.2.tar.gz	none	none
Open Suse	bsu-3.0.2.tar.gz	none	none
Slackware	bsu-3.0.2.tar.gz	none	none
Arch Linux	bsu-3.0.2.tar.gz	none	none
MacBook Pro	bsu-3.0.2.tar.gz	none	none
Chrome Book	bsu-3.0.2.tar.gz	none	none
MicroSoft Win 8.1	MingwBSU-3.0.2.tgz	BSU_EXEC-3.0.2.zip man-bsu3-html.zip Octave.zip	none html man pages

NOTES: Above packages are 64 bit. Source code should compile on both 32 and 64 bit systems. The APT package managed systems (Debian, Ubuntu, Mint) may work with packages for Debian 10 if they are at the Debian 10 level or more recent. The earlier versions of Debian should be compiled from source using appropriate configure options (see above, sections 2.3.2 and 2.3.3).

The RPM package managed systems (CENTOS, Open Suse, Redhat Enterprise) have particular considerations. Since CENTOS 7 has a conflict when installing under the /usr directory tree (but CENTOS 6 has no such conflict with a filesystem package). So installing in the /usr/local directory tree was chosen in the *.spec files since that location works for both versions of CENTOS.

The Microsoft case was compiled on Linux using the Mingw32 tool chain, producing most of the programs as *.exe files that will run well in a PowerShell terminal. Just unzip the file BSU_EXEC.zip in a location of choice that is included in the execution path.

In all cases, compiling from source is always an option, particularly if one wishes to add or modify codes in BSU. The *.asc files are GPG detached signatures signed with my Key 4812C85C (see my web page for a copy of the key).

2.6 Installing Binary Packages

The two package managers supported are APT (*.deb) and RPM (*.rpm). The supplied packages are limited to a few instances of operating systems and versions. These are:

- Debian 10 Buster (APT)
- CentOS 7 and 8 (RPM)

A challenge in building packages exists due to the versions of dependent packages. Once built, a package can become broken or out of date as dependent packages evolve. The compressed TAR source archives contain package building files (*.dsc for APT, *.spec for RPM). These provide a starting point for those wishing to build custom packages for their own specific operating system instance. If you have a 32 bit system, these package building files may also be useful if you wish to go the package route. Otherwise, the most likely route to success is to build from the source (see section 2.3).

2.6.1 Debian, Mint, Ubuntu, or Chrome Book Install (APT)

The command to install BSU on a 64 bit machine with just Gnuplot graphics is:

```
sudo dpkg -i bsu_3.0.2-1_amd64.deb
```

If you wish to install on a 64 bit machine with Plplot, the command is:

```
sudo dpkg -i bsu_3.0.2-1+plplot_amd64.deb
```

If you wish to install static built executables (ONLY Gnuplot version), the command is:

```
sudo dpkg -i bsu_3.0.2-1+static_amd64.deb
```

These packages should work for both AMD and Intel cpu's. As an alternative to "dpkg -i", I recommend using the "gdebi" command, as it checks for dependencies. For example:

```
sudo gdebi bsu_3.0.2-1+plplot_amd64.deb
```

2.6.2 RedHat Package Install (RPM)

For systems using the RedHat package manager (rpm, yum, yumex, urpmi, rpmdrake), the command for the Gnuplot only version is:

```
sudo rpm -ihv bsu-3.0.2-1.x86_64.rpm
```

For Plplot graphics in addition to Gnuplot, the CentOS-7 or CentOS-8 command is:

```
sudo rpm -ihv bsu-3.0.2+plplot-1.x86_64.rpm
```

2.6.3 Slackware or Arch Linux Package Install

There are no packages provided for these two systems at this time. It is left to the reader to build a slackpkg or pacman package if desired. However, building from the source is recommended over building more packages.

2.6.4 MacBook Pro Darwin Package Install

There are no packages provided for the MacBook Pro or any Apple OS. It is left to the reader to build a package if desired. However, building from the source is recommended over building more packages.

2.6.5 Microsoft Package Install

From within a power shell, create a directory for the *.exe executables or use an existing directory that is already in your execution path. For example:

```
unzip BSU_EXEC-3.0.2.zip
```

This will install the static built executables (*.exe files). You will also need to install the Microsoft Windows package for Gnuplot if the graphics are to work. Installing Mingw32 will add additional tools that may be of use, but is optional. You can modify the execution path with a *.ps1 file: `Documents\PowerShell\path.ps1`. For example, path.ps1 could contain this:

```
$env:path +=';H:\MingW\bin;H:\MingW\msys\1.0\bin;H:\gnuplot\bin;H:\EXEC'
```

2.7 Installing Source Code

Installing the source code and compiling is essential if one wishes to extend or modify BSU. You might choose to compile the software to adapt it to new shared libraries with a system upgrade.

```
cd /usr/local/src
tar xvzf bsu-3.0.2.tar.gz
cd bsu-3.0.2
./configure <build options>
make
make install
```

There are a number of options that may be given to the configure command. To see these, change to the source top directory and issue the command (two dashes before help):

```
configure --help
```

2.7.1 TAR Source: Linux or Darwin

This is the best choice if you plan on editing or extending BSU. The old school approach to installing a TAR archive (tarball) is to unpack the archive under the `/usr/local/src` directory tree. This would be desirable if other users will want to execute the programs (to be installed in `/usr/local/bin`). This assumes that you have write privileges in that directory (ie. you can be come root or use the sudo command for this purpose).

If you are working on a host machine that is administered by a university or business, and do not have write privileges in `/usr/local`, or if you will be the only one using BSU, then you can do everything in your own home directory tree.

Begin by downloading the following file into your home directory (see section 2 for the web address):

`bsu-3.0.2.tar.gz`

Change to the top directory where you will unpack the archive. Execute the *tar* command on the downloaded file in your home directory. Then change into the bsu-3.0.2 directory. An example of the commands would be:

```
cd /usr/local/src
tar xvzf ~/bsu-3.0.2.tar.gz
cd bsu-3.0.2
```

Next, you can run the configure script specifying your explicit options. For suggestions on which configure options to use on which operating system, see section 2.3.

2.7.1.1 Additional Hints on Configure Options Additional options that you may wish to customize are `--prefix=` and `--exec-prefix=` options of configure. The prefix option controls where the executables, libraries, and man pages will be installed.

- **Building for Multiple users**

  An example of the command sequence follows. It assumes you have sudo or root privileges, and are installing software system wide. Assume you have installed the tar archive in `/usr/local/src`. Enter the top directory, bsu-3.0.2, and then type the following commands in a terminal.

```
configure <desired options appropriate for your OS>
make
sudo make install
```

- **Building for a Single user**

  The next example illustrates how BSU could be built in the home directory tree for use by a single user. The example assumes you have created a `local` directory under you home, and unpacked the tarball under this local directory. After changing into the `bsu-3.0.2` directory, you would issue the following command sequence.

```
export PREFX=${HOME}/bin
./configure --prefix=$PREFX --enable-shared=no \
            --libdir=$PREFX/lib/bsu \
            --includedir=$PREFX/include/bsu  \
            --datadir=$PREFX/share/bsu \
            --docdir=$PREFX/share/doc/bsu


make
make install
```

  Watch for warnings during the configure step. Common problems are often resolved by installing a missing development library.

2.7.2 Install Source: Linux Packages

If you just want to compile the existing set of software and have an active packaging system like Debian (*.deb) or "RedHat" (*.rpm), you could install the source package and compile it. In that case, download the needed package as described in section 2.5.

2.7.2.1 Debian Source Package Build Place the following files in a working directory

```
bsu_3.0.2-1.diff.gz  (<--if any exists)
bsu_3.0.2.orig.tar.gz
bsu_3.0.2-1.dsc
```

You should have installed both Fortran and C compilers. It is also good to have helper packages installed (debhelper, dpkg-dev, dh-make, devscripts, lintian). Unpack the source with the following command:

```
dpkg-source -x bsu_3.0.2-1.dsc
```

Your screen will output the following (assuming a *.diff.gz file exists)

```
dpkg-source: extracting bsu in bsu-3.0.2
dpkg-source: unpacking bsu_3.0.2.orig.tar.gz
dpkg-source: applying ./bsu_3.0.2-1.diff.gz
```

Then change into the debian directory and use your favorite editor (like vi) to edit the changelog file:

```
cd bsu-3.0.2
cd debian
vi changelog
```

The changelog file will look something like this:

```
bsu (3.0.2-1) stable; urgency=low

  * Initial release (Closes: #0000)

 -- P. Michaels <pm@cgiss.boisestate.edu>  Wed, 09 Apr 2009 16:14:28 -0600
```

Modify the first line with some ID to indicate that the package has been built by you. Let's say your initials are "JMS", then the modified file would look like this:

```
bsu (2.0.1-1JMS) stable; urgency=low

  * Initial release (Closes: #0000)

 -- P. Michaels <pm@cgiss.boisestate.edu>  Wed, 09 Apr 2009 16:14:28 -0600
```

This would be the minimum change. Changelog files have a rigorous format, so quit while you are ahead, and save the modified file. Then move up a directory and execute the build command:

```
cd ..
dpkg-buildpackage -rfakeroot -uc -us
```

If all goes well, you will have a *.deb binary file built (it will be located in the same directory as the *.dsc file). The new *.deb file can be installed as above in section 2.6.1. If all does not go well, it probably will mean that you need to install a package or compiler, or something that BSU needs. Just read the error message.

2.7.2.2 Redhat Source Package Build

Download the following file

`bsu-3.0.2-1.src.rpm`

You can either build it in the `/usr/src/redhat` directory tree, or somewhere else where you have write privileges (like your home directory tree). An alternative location can be set with a `.rpmmacros` file in your home directory. This is an example of my `.rpmmacros` file.

```
%packager  P. Michaels <pm@cgiss.boisestate.edu>
%vendor BSU
%_topdir /home/pm/redhat
%_prefix /usr
%_exec_prefix    %{_prefix}
%_mandir         %{_prefix}/share/man
%_datadir        %{_prefix}/share
%_sysconfdir     %{_prefix}/etc
%_bindir         %{_exec_prefix}/bin
%_gpg_name      Paul Michaels
```

The `%_topdir` variable specifies where the package will be built. The steps are to unpack the source RPM, then cd into the SPECS directory and issue a build command. The unpack command is

```
rpm -ihv bsu-3.0.2-1.src.rpm
```

The directory tree will look something like this:

```
redhat/
       |-- BUILD
       |-- RPMS
```

```
|   |-- i386
|   |-- noarch
|   `-- x86_64
|
|-- SOURCES
|   `-- bsu-3.0.2.tar.gz
|-- SPECS
|   `-- bsu-3.0.2-1.spec
`-- SRPMS
```

Then change into the SPECS directory and edit the BSU spec file:

```
cd redhat/SPECS
vi bsu-3.0.2-1.spec
```

You can edit the bsu-3.0.2-1.spec file which starts like this:

```
%define name bsu
%define version 3.0.2
%define release 1
```

And change the release to match your initials, say “JMS":

```
%define name bsu
%define version 3.0.2
%define release 1JMS
```

Save the edited spec file and then build the binary rpm:

```
rpmbuild -ba bsu-3.0.2-1.spec
```

If all goes well, it will build and locate a binary rpm in the `redhat\RPMS` directory tree. Install the binary rpm as described in section 2.6.2.

2.8 Security

Integrity of the packages can be verified with my public GPG key and finger print posted on the security web page:

http://cgiss.boisestate.edu/~pm/security.php

You will also find md5sum files for each package or TAR archive.

2.8.1 GPG Signature, RPM Packages

To verify the integrity of an RPM package, do the following:

1. Download my public GPG key from my security web page, save it in a file named “pmkey.asc“
2. Import my public key into RPM
3. Execute the rpm command with the -K option on the downloaded *.rpm file.

EXAMPLE:

```
sudo rpm --import pmkey.asc
rpm  -K  bsu-3.0.2-1.x86_64.rpm
```

The output from the ”rpm -K ” command should verify both md5 sum and gpg in one line:

```
bsu-3.0.2-1.x86_64.rpm:  sha1 md5 OK
```

2.8.1.1 GPG Signature, DEB Packages

To verify the integrity of a *.deb package, do the following:

1. Download my public GPG key from my security web page, save it in a file named "pmkey.asc"
2. Import my public key into your gpg keyring
3. Execute the gpg command with the --verify option on the *.dsc or *.changes file

EXAMPLE: First we import the key. We can list our keys to check and see if it is there (for those who may be paranoid). Below are the commands and the output from the gpg program.

Command:

```
gpg --import pmkey.asc
```

Output:

```
gpg: key 4812C85C: public key "Paul Michaels (January 2002)
<pm@cgiss.boisestate.edu>" imported
gpg: Total number processed: 1
gpg:               imported: 1
```

Command:

```
gpg --list-keys
```

Output:

```
/home/pm/.gnupg/pubring.gpg
---------------------------
pub   1024D/4812C85C 2002-02-01
uid                  Paul Michaels (January 2002) <pm@cgiss.boisestate.edu>
sub   1024g/1430F4BF 2002-02-01
```

Next, we verify the signature embedded in the *.dsc or *.changes file.

Command:

```
gpg --verify bsu_3.0.2-1.dsc
```

Output:

```
gpg: Signature made Tue 13 Jun 2017 03:05:50 PM MDT using DSA key ID 4812C85C
gpg: Good signature from "Paul Michaels (January 2002)
<pm@cgiss.boisestate.edu>"
```

The warning results if you have not certified the signature with a level of trust. You can edit the key and change the trust, as well as sign it (assuming you have your own public and private key). But this is not necessary to check *.changes or *.dsc file (assuming you are confident of my key). If you obtained my key from some source other than my web page, you may wish to compare it with the listing below:

```
 -----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.6 (GNU/Linux)

mQGiBDxZ9GIRBADJ/p9510p5ubYWlhGZgMNJUEcP2rkMoA/jtxFqqaWERjvMKwW4
kbWytJ2VLlIhiFI0vUMtViHpppFz6TfDR+1qvIzBqxobieIuPxotCa1e+KDWpaCI
Rb2+4ny2TlbZ3JBsK9rzMZkIVsUa7aCFbHmtLpBRwf2T97AEm3+lSQFBEwCgiWqC
wc2tKBGeZ6rdmGWkbmUzdN8EAMnoXaW5lo+WhbitR4OqQ5YwEOGwXMcD+/QhMjCv
353YPbnPzkhFokQ6dVkk3rTQBV5jg0p0YsjNVaWwQo7oNXLOLLhC2d+/mLPGjPH+
afgKGFmyXkxUuLHmht0JZsuiLfr8oOEBQyHwQC+y1Ccd94nEefTvQE8I8Sqy0kP/
01ezA/9B/+Xy+L4mJvGGJ/cOOV4yzzR8BJ+koYhGVvNEq2I5jy67KhBpSPWxRPb5
dWu31WCnkzk6i9NUAx3QecvXLTR6AZMvw1TL8kGmCsG7vWWNB1Mg2P62AMARxbOl
fPY5Y9tzDFXaNan6axTGMKOto/5RDfp5X9nO8bfiUiFK0iVH6bQ2UGF1bCBNaWNo
YWVscyAoSmFudWFyeSAyMDAyKSA8cG1AY2dpc3MuYm9pc2VzdGF0ZS5lZHU+iFcE
ExECABcFAjxZ9GIFCwcKAwQDFQMCAxYCAQIXgAAKCRDdrrznSBLIXCYzAJ9XDNs+
/Ue7F/hFQdsM8Xb3K1EUsACeIbhzpowmmOAWkcW/H77fUg4O6G+5AQOEPFn0dRAE
AIBVO6W+vPZimewQeBIAaou+81RMGmBcMQ3fUjLdXUQubjOM4LYjS4WP+AtzIvuj
2GXMBkh0eOAiw0Icn9UD5Qv1ogBrRBmSGmP4XLtin6PgGdG9Ak6PQtb2ZKj5kGq3
8fG/OOtFSYHJuD8MPenL3mMQwSMtoFgMqpU3b/1ONVdDAAMFA/0WjVrD0vMw2O1R
```

```
4owGbsu9VdS5V3BDwssgVy1V7GZEB1iCJqKPf87wNYaZWQWuCx6SmVQe+XrP67MC
Zbm8Pk8bFFaNa3aOXHfqB+kzXofiKfCNVdqy7jAyZrhN753pZlJYMvq/EnNa5qMm
PrNak+/8XZMA1I76l9//ybQMwuK6gYhGBBgRAgAGBQI8WfR1AAoJEN2uvOdIEshc
sToAni1CjyZvwbsYb0uVSkZuP4dEUAOkAJ9FIBzX2e/16FVW222yNKOl0shLRw==
=zS6T
-----END PGP PUBLIC KEY BLOCK-----
```

2.8.1.2 Detatched GPG Signatures Files that can not be internally signed have a detached GPG signature. For example, the TAR archive `bsu-3.0.2.tar.gz` has been signed with my default key using the command:

```
gpg --sign --armor -b bsu-3.0.2.tar.gz
```

This produces a file `bsu-3.0.2.tar.gz.asc` which is the detached signature. This signature may be verified with my public key. If the *.asc and *.tar.gz file are in the same directory, one would issue the following command:

```
gpg --verify bsu-3.0.2.tar.gz.asc
```

A more complete command would also include the file being checked as an additional argument:

```
gpg --verify bsu-3.0.2.tar.gz.asc bsu-3.0.2.tar.gz
```

3 Other Software

3.1 PLPLOT

At the time of this writing, the home web page for PLPLOT is http://plplot.sourceforge.net/. Most linux distributions have PLPLOT binary packages available in their respective repositories. The PLPLOT library is used by BSU to generate the following types of graphics:

Program	Graphics Produced	Comment
bplt.c	X-window (xwin/wxt) Postscript xfig jpeg PDF	plot seismic waveform data
bvsp.f	Postscript	down-hole vertical seismic profile velocity inversion
bhod.F90	Postscript	hodogram analysis of down-hole tool orientation
bvas.F90	Postscript	velocity dispersion of **Down-hole** data
bamp.F90	Postscript	amplitude decay of **Down-hole** data
bvax.F90	Postscript	velocity dispersion of **Surface Wave** data
bamx.F90	Postscript	amplitude decay of **Surface Wave** data
caplot.F90	Postscript	composite plot of bvas and bamp results
bfit.c	Postscript	down-hole vertical seismic profile vertical time analysis
bgaz.c	Postscript	broad-band analysis of amplitude decay with **vertical depth**
bgar.c	Postscript	broad-band analysis of amplitude decay with **horizontal offset**

3.2 BLAS and LAPACK

BLAS and LAPACK are available from http://www.netlib.org/lapack/. These libraries are used by BSU when linear algebra functionality is needed.

Program	Comments
bhod.F90	hodogram analysis of down-hole tool orientation
bvsp.f	down-hole vertical seismic profile velocity inversion

3.3 GSL and CBLAS

GSL and CBLAS are available from http://www.gnu.org/software/gsl/. CBLAS is the C-language version of BLAS (see above). GSL is used in the following case:

Program	Comments
lamb.c	solution to Lamb's problem (waves generated from a vertical impact on 1/2 space)

3.4 CMLIB

The **CMLIB** is quite large and not essential to install, since the needed code is included in the Fortran77 **sublib4.a** (*rand.f* and *runif.f*). However, you may wish to explore what is available from **CMLIB**, so the site to visit is http://gams.nist.gov/serve.cgi/PackageModules/CMLIB.

3.5 Octave

Included with BSU is a directory with some procedures written in the Octave language. For example, the procedure, *bsegin.m*, can read the binary BSEGY data format. If you have used Matlab, you will find Octave easy to use (the syntax is very similar). Also included in BSU are other scripts to perform seismic refraction interpretation and inversion of down-hole transmission seismograms for soil stiffness and damping. Octave is particularly useful for plotting, and some interactive GUI driven plotting procedures may prove useful to the user. You may install Octave with your package manager or download Octave from the web site, http://www.gnu.org/software/octave.

3.6 Seismic Unix

While BSU is completely independent of Seismic Unix (SU), it is highly compatible with it, but only if SU has been compiled without *XDR*. The *XDR* option in SU originated around 1997 and is essentially a big endian byte order with IEEE encoding of floats. See section 6.6.4 for more on this topic. I use both, often plotting BSU results with SU software (see the bash scripts *xPlot-su* and *psPlot-su* in the BSU source archive scripts directory). The structure of the binary data formats are the same (240 byte trace header followed by 4 byte floating point data). The number of samples and sample interval headers are identical. The header formats do differ in some important ways (BSEGY has definitions for source and receiver polarizations). In short, SU has a program, *segyclean,* which permits SU to read BSU files, should a difficulty be encountered. To download SU, visit GITHUB https://github.com/JohnWStockwellJr/SeisUnix.

3.7 Xfig

Xfig is a CAD program that permits you to draft figures (like Autocad or Microstation). The Octave procedures included in BSU often provide the option to save plots in Xfig format. In particular, the plotting sections have scaling options for both Xfig and Postscript. It is possible to control the axes lengths in desired distance units on a printed page. It is also worth noting that Xfig is a valid device format for the PLPLOT library. That is, all you need to do is change the device definition in a BSU code that uses Postscript, and it will output a file in Xfig format. Then you can draft on the figure. If you don't already have Xfig (it comes with most Linux distributions), then download it from, https://sourceforge.net/projects/mcj.

3.8 Trouble Shooting

There are basically two ways to install software in Linux. One can either install packages, or unpack TAR archives in the `/usr/local` directory tree followed by a compile of the source code. Mixing these two methods can sometimes lead to problems. Packaging data bases are often unaware of programs compiled in the `/usr/local` tree.

3.8.1 Example: PLPLOT tar, BSU rpm

At the time of this writing, the PLPLOT libraries are not well represented in the RPM packaging effort (APT packaging is quite good by contrast). Installing PLPLOT on Redhat Enterprise Linux is a good example. Developing a SPEC file for PLPLOT with all the various sub-packages is a big job, and so unpacking in the `/usr/local` tree is quite attractive. I did this with the **plplot-5.10.0.tar.gz** TAR archive. PLPLOT uses *cmake* rather than *configure*. To build plplot libraries, untar plplot-5.10.0.tar.gz in the /usr/local/src directory. The following is the sequence of commands:

```
cd /usr/local/src/plplot-5.10.0
mkdir build_dir
export CC="gcc -O2"
export CXX="g++ -O2"
export FC="gfortran -O2"
cd build_dir
cmake -DCMAKE_INSTALL_PREFIX=/usr/lib64 ../ >& cmake.out

make >& make.out
sudo make install >& make_install.out
```

The X11 driver requires that the **libX11-devel** RPM package be installed. Failing to install needed *-devel.rpm packages is likely to result in a driver not being built.

In this example, the installed libraries are located in /usr/lib64. If you put the libraries somewhere else like `/usr/local/lib` then they may not be found when needed. For example, if `/usr/local/lib` is not already in `/etc/ld.so.conf`, you should add it, and then execute

```
sudo ldconfig -v
```

to make the system aware of the new libraries. If you intend to install BSU as an RPM, it is likely that the next problem will be that RPM will not be aware of these libraries, and can block the install of the package. That is, even though these libraries have been installed under `/usr/local/lib` and the system is aware of it, the RPM data base is not aware of it, and so blocks the install. The solution is to use the --nodeps option:

```
rpm -ihv bsu-3.0.2+plplot-1.x86_64.rpm --nodeps
```

This overrides the RPM data base, and permits the install. At object time, the programs requiring the PLPLOT libraries will be able to find them, and all will be well. Of course, these types of problems are avoided when building BSU from a source tarball, or from installing Plplot libraries in standard locations.

4 Programming in BSU

BSU is composed of Fortran77, Fortran90, and C-language codes. I continue to write in both Fortran and C-language, taking advantage of pre-existing libraries and software. The BSU paradigm has been kept very simple. To add a new program is quite easy. Here are some guidelines.

4.1 Programming Guidelines

1. **Decide on which language to program in**. If you only know one language (and it is either Fortran or C) then this will be an easy decision. If you are bilingual, consider the subroutine libraries.

2. **Examine the BSU and some 3rd Party subroutine libraries**

   - `bsu-3.0.2/src/sublibC4` – These are the C-language functions
   - `bsu-3.0.2/src/sublibF4` – These are the Fortran subroutines
   - `bsu-3.0.2/src/sublibC` – These are C-functions which can be called by Fortran
   - `bsu-3.0.2/src/libIBM` – The `xdrfloa.c` function is in libibm.a and is used to produce IBM floating point needed for the SEGY exchange format (see program bcnv.c).
   - `bsu-3.0.2/src/subLAPACK` – This produces a static library that consists of a subset of the Lapack functions needed for a static build of BSU.

   Consult these libraries to see which pre-existing subroutines might serve your needs. Your choice of programming language may depend on what is available in each language. Of course, you can always add to the subroutine libraries, so this is not the only issue.

3. **Examine Other 3rd party libraries that you may need**. PLPLOT can be called from both C and Fortran. For linear algebra, the LAPACK libraries are available in both Fortran and C. Further, the GSL library has some duplication of LAPACK in it (strictly C only). For other scientific functions, the GSL libraries are in C, and the CMLIB material is Fortran.

4. **Copy a master example to a new file name.** The programs are:
   `bsu-3.0.2/src/Fort/bsegy/bmst.f`
   `bsu-3.0.2/src/C/bsegy/cmst.c`
   These programs are simple examples which you can copy to a new file, and then edit. What they do is read traces from a BSEGY data set. As each trace is read, it is rectified with an absolute value function. The rectified trace is output. What you will want to do is replace the absolute value part with your own calculations. The source code for main programs should be kept located in the appropriate **bsegy** directory.

5. **Edit the new program.** You will need to change the process name character string (4 characters), the input parameter function/subroutine, and the output listing function/subroutine to meet your needs (see section 4.2 below). You will replace the computation section with your own code.

6. **Edit *Makefile.am***. You will want to add your program to the file.
   a). Add your process to the list of executables at the top (`bin_PROGRAMS=`).
   b). Add a `filename_SOURCES=filename.c` or `filename_SOURCES=filename.f`
   c). If you are adding to the one of the subroutine libraries, then all you need to do to the *Makefile.am* in the subroutine directory is to add the name of the new source file to the list at the top of *Makefile.am*.
   (ie. `libsubF4_la_SOURCES=` or `libsubC4_la_SOURCES=`).

7. **Re-configure the build tree** Change back to the `bsu-3.0.2` top directory and re-run configure.

8. **Compile your new program** Type:

   ```
   make
   ```

9. **Install your new program**. Type:

   ```
   make install
   ```

   (this will install everything, so you might only want to explicitly install the executable manually).

10. **Write a man page for your new program**. Start with an existing man page, and modify it to meet your specific needs. Use `man1` for main programs, `man3` for subroutines. Install in the appropriate directory:

```
bsu-3.0.2
        |-- man
        |   |-- man1
        |   |-- man3
        |   |-- man5
        |   `-- man7
```

4.2 Conventions and Process Flow Description

4.2.1 File Naming Conventions

BSU main programs should be named with a 4 character process name, starting with the letter "b". This permits about 17,576 names. An example would be *bfoo.f* or *bfoo.c* in Fortran or C. The reason for this is that an output file name is constructed from the input file name (all input file names need to be at least 4 characters, and the first 4 characters are captured to form part of the output file name). In retrospect, the Seismic Unix convention is far superior (using stdin and stdout). But, as you might have guessed, I went through a MSDOS phase (file names restricted to xxxxxxxx.yyy format), and this is an example of the inertia that all ideas have (both good and bad). It isn't absolutely necessary to start a program name with "b", and the BSU package includes even more deviant examples. This was how I started, and future versions of BSU may abandon this convention.

The file naming convention is to form the output file name as "*bfoobbar.seg*", where *bfoo* is the current process running, and *bbar* is the first 4 characters of the input file name. In a limited way, the file name becomes a processing history (of rather short memory span). Thus if process *bfoo* were to read a file *babsw001.seg*, the output file would be named *bfoobabs.seg*. Another example would be if *bfoo* were to read a file *xyzfileseven.seg*, the output file name would be *bfooxyzf.seg*. The primary advantage of this scheme is that it generates predictable file names that can be counted on in writing bash scripts that run a string of processes. The disadvantages are too numerous to list. You will probably note that I have deviated from this convention on several occasions. See one of those programs for an example if you need to deviate from the convention.

4.2.2 Input Parameter Conventions

All input parameters may be entered on the command line, but their location is restricted to a predefined order. To quickly ascertain the order, use the "-h" option. For example, if you type

```
bmed -h
```

Then the following output will be printed to the screen:

```
 |--------------------------------|
 | Copyright (C) 2017  P. Michaels |
 |       All rights reserved       |
 |see GNU General Public License   |
 |--------------------------------|

 |----------------------------------------|
 | Basic Seismic Utilities     FORTRAN    |
 | ONLINE HELP:                           |
 |----------------------------------------|
 | bmed: Median mix of seismic traces     |
 | across the trace direction.            |
 |----------------------------------------|

 bmed infile mix

 infile  =input file name
 mix     =mix width <21
```

This gives one a short description of the process, and the command line arguments (in this case, *infile* and *mix*). You may then run the process providing any number of command line arguments. The user will be prompted for any arguments not included on the command line. The input of parameters is done by a call to subroutine *GETPRM(. . .)* in Fortran, and by a function *getparm(. . .)* in C.

The other way to learn about the use of any program or subroutine/function of sublibF4 or sublibC4 is to use the man page. For example, you could type:

```
man bmed
```

for a more complete description of the program and its input parameters.

4.2.3 Process Flow, Fortran Codes

The following is a description of the process flow for *bmst.f*, and any codes based on this master.

1. **The application prefix** (4 characters) is written into character variable *aplc*. You will want to edit this according to your new process name.

2. **Subroutine GETPRM:**
`call GETPRM(nargsx,infil,parm1, . . .)`. This subroutine is located inline with every main program. You only need to modify the arguments starting with "parm". This is your interactive/command line input parameter subroutine.

3. **Subroutine CHKTRC:**
`call CHKTRC(iunit1,icall1,npts,s1,fsamin,infil,bar,invbar,ibar,ntrace,ndim)` checks the input file and outputs number of samples, sample interval, number of traces, and some additional parameters needed for the execution progress bar. Type:

   ```
   man chktrc
   ```

   for more on this subroutine.

4. **Subroutine LSTPRM:**
`call LSTPRM(io11,ntrace,aplc,outlst,outfil, parm1, . . .)` This subroutine provides an echo check of the input parameters by writing to a listing file. Examining the listing file is sometimes useful when you wish to see what you have previously done. The listing file name is the same as the output data file name, but with a different suffix. Thus, if the ouput data file were named *bfoobbar.seg*, the listing file name would be, *bfoobbar.lst*.

5. Then there is the trace loop. The **do** loop runs over the number of traces found in the *chktrc()* call. The sequence in *bmst.f* is:

```
c...data loop=======================================....
      do 100 jrec=1,ntrace
c
c...input a trace
      call BSEGIN(iunit1,ndim,
     +icall1,npts,s1,fsamin,infil,jrec,iexit)
c
      ntr = ntr + 1
  . . . .    [  DO SOME COMPUTATIONS ON THE SIGNAL S1 ]
c...output trace
      iunit2=2
      call BSEGOUT(iunit2,
     +icall2,npts,s2,fsamin,outfil,jrec)
c...display progress....
      call pltbar(bar,invbar,ibar,jrec,ntrace)
c...END LOOP=======================================....
  100 continue
```

where BSEGIN reads a trace, BSEGOUT writes a trace, and PLTBAR writes a progress bar to the screen in real time. You would insert your own code at the "do some computations" location in the trace loop. The secret to supressing a carriage return and line feed for the progress bar is the non standard "$" format for Linux. In other operating systems, it may be different, try "\" if you plan on porting these codes to another OS and "$" doesn't work. This is a simple example, and more complicated flows with multiple loops are possible. The other programs in the BSU package provide examples.

4.2.4 Process Flow, C-Language Codes

The following is a description of the process flow for *cmst.*c, and any codes based on this master. This is the C version of *bmst.f*, and was named differently to allow both executables to be located in the same directory.

1. **The application prefix** (4 characters) is written into character variable *pid*. You will want to edit this according to your new process name.

2. **Function "getparm"**
   `getparm(argc,argv,&iparm1. . .);` This function is located inline with every main program. You only need to modify the arguments starting with "iparm". This is your interactive/command line input parameter function.

3. **Function "outlst"**
   `outlst(iparm1,fparm1,infile,pid,ofile,lstfile,h3);` This function (located inline with main program file) provides an echo check of the input parameters by writing to a listing file. Examining the listing file is sometimes useful when you wish to see what you have previously done. The listing file name is the same as the output data file name, but with a different suffix. Thus, if the ouput data file were named *bfoobbar.seg*, the listing file name would be, *bfoobbar.lst*.

4. **Function "in_chk"**
   `in_chk(&ntraces,&npts,&fsamin,hd,h1);` Checks the input file and outputs number of samples, sample interval, number of traces. Type:

   ```
   man in_chk
   ```

   for more on this subroutine.

5. **Function "bargrid"**
   `bargrid(ntraces,&barr,&ibar,&invbar);` Computes the parameters for the real time progress bar drawn by *exbar()* to the screen.

6. Then there is the trace loop. The **for** loop runs over the number of traces found in the *in_chk()* call. The sequence in *bmst.c* is:

```
      for (jrec=0;jrec<ntraces;jrec++)         //...trace loop
     {
        if(c_bsegin(jrec,s1,npts,&hd,h1)!=0)   //...read a trace
       {
        fprintf(stderr,"ABORT--c_bsegin");
        goto quit_it;
       }
. . . . .     [ DO SOME COMPUTATIONS ON SIGNAL S1 ]
        if(c_bsegout(jrec,s2,npts,&hd,h2)!=0)  //...output a trace
       {
        fprintf(stderr,"ABORT--c_bsegout");
        goto quit_it;
       }
      exbar(ntraces,bar,ibar,invbar,jrec+1);   //...display progress bar
     } //  end trace loop-----------------------------
```

where `c_bsegin` reads a trace and `c_bsegout` writes a trace, and `exbar` writes the progress bar to the screen in real time. Again, you would insert your own code at the "do some computations" location in the trace loop. This is a simple example, and more complicated flows with multiple loops are possible. The other programs in the BSU package provide examples.

4.2.5 Locations of Functions and Subroutines

For the Fortran codes, the subroutines *GETPRM()* and *LSTPRM()* are *kept inline with the main program file*. The reason is that these subroutines will change with every new program (since each new program will have different input parameters). The same is true for C-language functions, *getparm()* and *outlst()*. For other subroutines and functions, they should be located in the subroutine library, especially if they are likely to be reused by other programs. Exceptions would be functions unique to a particular program, or which might have a library equivalent, but for some reason, like extra I/O needs, not be appropriate in a general library. For Fortran codes, the library is located in **bsu-3.0.2/src/sublibC4**, and for C-language programs, the library is located in **bsu-3.0.2/src/sublibF4.**

5 BSU Documentation

BSU has several forms of documentation. These are:

- The help option on the command line.
- The program *bhelp*
- Man Pages (including *whatis* or *mandb* and *apropos*)
- Running BSU (**RunningBSU-3.0.2.tgz**)
- This User's Guide (**BSU-guide3-2.tgz**)

5.1 Command Line Help

Each executable main program tests the command line for arguments. If no arguments are found on the command line, then the program will prompt for input as described above. One can see a complete list of command line arguments by invoking the command line help option. The syntax for a command line help is:

```
bmed -h
```

5.2 The *bhelp* Program

While the command line help is useful when you know what program to run, there will be times when you want to briefly scan a list of available programs and functions. To browse this list, run *bhelp* and pipe it through the *less* program:

```
bhelp | less
```

For those not familiar with *less*, you exit the *less* program by pressing the letter "q" on the keyboard while the X-term window has focus. The programming language is indicated by the suffix of the file name. Program *bhelp* lists the following:

- **main programs**
- **C-functions** (in `bsu-3.0.2/src/sublibC4`)
- **Fortran subroutines** (in `bsu-3.0.2/src/sublibF4`)
- **Octave procedures** (in `bsu-3.0.2/Octave`)

See **Appendix A** for the *bhelp* listing.

5.3 BSU Man Pages

The manual pages are viewed by using the *man* command. The main programs, subroutines, and functions packaged with BSU all have man pages. The location of the man page files will depend on your choice of the option, `--prefix=` set during the compilation process. The standard location in Debian and Redhat packages would be `--prefix=/usr`. However, it may be different. For example, you may have chosen `--prefix=/usr/local`. If that were the case, man page files will be located under the directory `/usr/local/man`. Whatever your prefix is, the man directory under that prefix should be in your man search path. If it isn't, define and export the MANPATH variable. For example, if you use bash and want to set MANPATH for all users, you might wish to edit the file, /etc/profile, and add the line:

```
export MANPATH=$MANPATH:/usr/local/man
```

This will add the above directory to any existing MANPATH. Options for setting the man search path can vary among the Linux distributions. For example, on Ubuntu, you may wish to edit the file `/etc/manpath.config`. Examine your `/etc` directory for a file with "man" and some variation of "conf" or "config". To view a man page, type man followed by the program name (bmed in example below):

EXAMPLE

```
man bmed
```

You will be able to see the man page in an X-term window. To generate a Postscript version of a man page, type:

```
man -T bmed >bmed.ps
```

Then you can print or view the file *bmed.ps* with a ghostscript program (gv, evince, kpdf, etc . . .).

The other feature of man pages is the *whatis* and *apropos* data base. If you know the full name of the man page you wish to view, then typing

```
whatis bmed
```

will give you a one line description of the man page contents.

```
bmed (1) - BSU program median mixes seismic data across the trace direction
```

If you are not sure of the entire spelling, type

```
apropos bme
```

to get a list of programs that have that character string in the *whatis* database. Not all programs listed will be relevant to your inquiry.

```
bmed (1) - BSU program median mixes seismic data across the trace direction
obmenu (1) - a menu editor for openbox
```

5.4 BSU User's Guide and Running BSU

That is this document, **bsu-user-guide3-2.pdf** and **Running_BSU-3.0.2.pdf**. They can be found in directory `/usr/share/doc/bsu` if you have done a standard package install (*.rpm or *.deb). Alternatively, it might be in `/usr/local/share/doc/bsu` if you have compiled from the source in the /usr/local/src directory. Also in the doc directory are the GNU license documentation and a README file. Further information follows in section 6.

6 Using BSU

This section contains examples demonstrating how to use BSU. While each main program has a limited task to perform, you may create more complicated processing flows using executable scripts. Examples of scripts may be found in `/usr/share/bsu/scripts/`. When running scripts, you may wish to redirect the progress bar to the bit bucket (`/dev/null`). Fore example, you could do this by:

```
bfoo argument1 argument2 >/dev/null
```

The above would be for a program *bfoo* with two command line arguments. You will need to specify all the command line arguments if you do this. For examples of script flows, see the *Merge* scripts in Appendix B and Appendix C.

This section includes:

- Data formats
- Converting data formats
- Setting Geometry
- Plotting programs
- Bash scripting with BSU and Seismic Unix (SU)
- Using Octave with bsegy data.
- Processing a down-hole (Vertical Seismic Profile) survey acquired with a source produces both SH- and P-waves.
- Refraction delay time processing.
- Synthetic Seismograms, Lamb's Problem
- Synthetic Seismograms: Near-Field, Far-Field Body Waves, and Rayleigh Surface Waves

6.1 BSU Data Format, BSEGY

The definition of BSEGY format data is contained in the include files packaged with the source code. See directory `bsu-3.0.2/src/Fort/include/` for the fortran programs, `bsu-3.0.2/src/C/include/` for the C-language programs. The format is derived from the SEGY data exchange format published by the Society of Exploration Geophysicists (SEG) [2]. As is the case in Seismic Unix (SU), the tape reel header is dropped. Each signal begins with a 240 byte trace header, followed by floating point data. The trace header specifies locations of the source and receiver, number of samples in the signal, sample interval, etc. The BSEGY format differs from the SU format in a key area. The SU format is well designed for reflection data, typically vertical component receivers. The BSEGY format is designed with header elements which record the source and receiver polarizations, permitting a full description of the inclination and azimuth of the geophone element as well as an equivalent force description of the source effort.

6.1.1 Data Format Conversion

Engineering seismic data come in a variety of data formats, as recorded on seismographs. Examples include BISON and SEG-2. Further, there are exchange formats, and other formats for published earthquake data. The BSU package contains some software to convert data from one format to another. Programs `bis2seg` and `egg2seg` have only one command line argument, the name of the input file which should be at least 4 characters long.

- **BISON:** Program *bis2seg* converts BISON format data to BSEGY format.
- **SEG-2:** Program *egg2seg* converts EGG Geometrics SEG-2 data to BSEGY format.

- **SEGY:** Program *bcnv* converts either way between BSU's BSEGY and SEG's SEGY formats. Since BSU data names originated with the 8.3 format, SEGY data generated from BSU will have the suffix *sgy* rather than *segy*. You can always rename the file after it is generated. The online help (bcnv -h) is:

```
   bcnv  infile endian compliance idirec idfc iunits hedfil

     infile   =  input file name
     endian  0= Little Endian host (Linux PC)
             1= Big    Endian host (IBM Mainframe)
  compliance 1= SEGY Compliant (EBCDIC, IBM Float, BigEndian)
             0= (ASCII Reel Header, Float and endian of host)
     idirec  0= BSEGY ==>SEG-Y (new *.sgy)
             1= SEG-Y ==>BSEGY (new *.seg)
     idfc    1= floating point 4 byte output
             2= long integer   4 byte output
             3= short integer  2 byte output
             (uses reel header if SEG-Y input data)
     iunits  1= meters
             2= feet
             (uses reel header if SEG-Y input data)

NOTE:
 hedfil only input if idirec=0  BSEGY --> SEG-Y
 (hedfil contains 3200 bytes, 40 records, 80char each
 If hedfil='none', then blank lines after C used
```

6.1.2 SEGY Exchange Format

Despite its age, the SEGY data exchange format remains popular. Originally designed for 1/2 inch tape, it has evolved into a disk format as well. It consists of the following:

- **Text Reel Header** 3200 byte EBCDIC encoded 80 character records, beginning with the letter "C". Origins of EBCDIC (Expanded Binary Coded Decimal) are with the main frame computers, largely built by IBM at the time SEGY was created. This is not ASCII encoding, typical of today's personal computers and work stations. As an aside, EBCDIC (29 key punch) evolved from an earlier, BCD (Binary Coded Decimal, 26 key punch), format. The use of IBM cards is extremely rare today.
- **Binary Reel Header** 400 byte binary integer encoded information about the entire reel of data. This is in Big Endian byte order (most significant byte first).

Where the Seismic Unix (SU) program *segywrite* uses a pipe to the *dd* program for the conversions between ASCII and EBCDIC, the BSU program *bcnv.c* uses library functions found in `bsu-3.0.2/src/libIBM` to do the conversions. The requisite functions were downloaded from IBM in the original EBCDIC, converted to ASCII and then edited to make the small static library `libibm.a`.

The reel headers are then followed by (trace header, data) signals. These are encoded as follows for each signal:

- **Trace Header** 240 byte mixture of 2 byte and 4 byte integers as defined in 6.1 above. Big Endian byte order.
- **Data** Each trace has the same number of digital samples, stored in either 4 byte IBM floating point, 4 byte integer, or 2 byte integer format. Only one format is used in a data set. The sample interval is always the same for all the traces. NOTE: The IBM floating point format is different than the native format, IEEE floats, found on today's workstations and personal computers Both IBM and IEEE are 4 byte floats. This difference goes beyond byte order.

Program *bcnv* is used to convert between SEGY (exchange format above) and BSEGY (used by the BSU software). If you really have data on tape, recommend that you use Seismic Unix (SU) programs, *segyread* and *segywrite*. The program, *bcnv*, has been updated. The current version of *bcnv* will create SEGY data that can be read by SU program *segyread*. The reverse direction is also true. That is, SEGY data created by SU program *segywrite* can be read by BSU program *bcnv*. This has been tested successfully on 4 byte floats.

6.1.3 IBM and IEEE Floats

The program *bcnv* uses an IBM C-language function contained in a published library, *libascii.tar.Z*, downloaded from <http://www-03.ibm.com/systems/z/os/zos/features/unix/libascii.html>. The file, xdrfloa.c, was extracted and used in BSU library *libIBM*. The downloaded library had to be translated from EBCDIC encoding using a Perl script.

```
#!/usr/bin/perl -w
 use Convert::EBCDIC;
#  $Id: Ebcdic2Ascii,v 1.1 2010-06-16 22:41:38 pm Exp $


 $translator = new Convert::EBCDIC;
#$ebcdic_string = $translator->toebcdic($a);
$filename=$ARGV[0];
open (fp1,$filename) ||
 die "error opening file \n";
$e=<fp1>;
close(fp1);

$ascii_string = $translator->toascii($e);


open (STDOUT,'| tr  \'\205\' \'\n\' | tr \'\335\' \'[\' |tr \'\250\' \']\'  ');

print $ascii_string;
```

The above Perl script requires a Convert-EBCDIC package found at:
<http://search.cpan.org/ cxl/Convert-EBCDIC-0.06/lib/Convert/EBCDIC.pm>.

Since there are a number of versions of EBCDIC, the translate pipe converts the left, "[", and right, "]" characters to produce a correct ASCII translation.

The function, **ConvertIEEEToFloat(src,dst)**, produces the IBM floating point needed for SEGY exchange format, and is used by my C program *bcnv.c*. The function, **ConvertFloatToIEEE(src,dst)**, is used to produce IEEE from floating points read from a SEGY file. Because IBM main frames are big endian, and Linux PC's are little endian, some byte swapping became necessary.

6.1.3.1 IBM FLOAT

The 4 byte float (8 bits per byte) is defined as:

```
     byte1       byte2       byte3        byte4
   xxxx xxxx   xxxx xxxx   xxxx xxxx    xxxx xxxx
  |s|<=ex=>|   |<==============fr==============>|
      (7)                    (24)
sign|exponent  |            fraction            |

    value =  (-1)^s  *  16^(ex-64) *  .fr

             5E-79< value < 7E+75
```

6.1.3.2 IEEE FLOAT The 4 byte float (8 bits per byte) is defined as:

```
     byte1       byte2       byte3        byte4
   xxxx xxxx   xxxx xxxx   xxxx xxxx    xxxx xxxx
  |s|<=ex=====>|<==============fr==============>|
      (8)                    (23)
sign|exponent  |            fraction            |

    value =  (-1)^s  *  2^(ex-127) *  1.fr

             1E-38< value < 3E+38
```

In converting between the two float formats, an underflow is set to zero, and an overflow is set to the maximum value defined by the format.

6.2 Checking Binary Files with hexdump

Sooner or later, the question about what format a data file exists in will come up. One can use the hexdump program with the "-C" option to view a binary file. For example, the following command

```
hexdump -C data.seg | head -n 20
```

produced the following listing. The 240 byte trace header is listed from **byte 00000000** to **byte 000000f0**. The first sample is at **byte 000000f0**, and the 4 byte float value is `F7 EC 97 40`. This is a IEEE float representation of the first data sample with a value of `4.7476763725280761`. The float is single precision representation.

```
00000000  00 00 00 00 00 00 00 00  e9 03 00 00 18 00 00 00  |................|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 03 00  |................|
00000020  00 00 01 00 11 00 00 00  e4 44 01 00 d7 4b 01 00  |.........D...K..|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000040  00 00 00 00 9c ff 9c ff  40 42 0f 00 ef 41 0f 00  |........@B...A..|
00000050  40 42 0f 00 34 42 0f 00  00 00 00 00 00 00 00 00  |@B..4B..........|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 f6 ff 00 00  |................|
00000070  00 00 a0 0f fa 00 00 00  00 00 18 00 00 00 00 00  |................|
00000080  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  00 00 00 00 00 00 e8 03  00 00 00 00 d3 07 81 00  |................|
000000a0  0d 00 20 00 37 00 00 00  00 00 00 00 00 00 00 00  |.. .7...........|
000000b0  00 00 00 00 00 00 00 00  00 00 00 00 b4 00 00 00  |................|
000000c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 30 30 30 31  |............0001|
000000e0  30 30 30 31 00 00 00 00  41 31 30 30 00 00 00 00  |0001....A100....|
000000f0  f7 ec 97 40 a7 eb b1 40  05 8d f9 3e 4e d5 a9 c0  |...@...@...>N...|
00000100  8e 09 95 c0 ca e1 13 3e  ea 7b 09 40 bc 42 0f 3f  |.......>.{.@.B.?|
00000110  91 65 01 c0 6f 9d 95 c0  31 97 a0 c0 35 f3 0c c0  |.e..o...1...5...|
00000120  ad a3 0a bf 2e 3b b4 bf  d8 80 98 bf 05 8d f9 be  |.....;..........|
00000130  91 65 01 c0 b2 2e 4a c0  1a e4 d6 bf 91 65 01 3e  |.e....J......e.>|
```

6.3 Preparing data for BSU processing

Before processing with BSU software, one must first convert the data to *bsegy* format. Later sections in this document give examples on how to do that conversion. Often, there will be two steps. Conversion of the data (using a program from the above list), along with a transfer of survey or "geometry" data to the headers. The need for this second "setting geometry" step is the common practice of not transferring survey information into the digital data. Either a format may not include options for this information, or it may be that so much was going on in the field that the observer was stressed and was lucky to just get it all down on "observer's logs". These are paper sheets with the information about shot and receiver location, sample interval, etc.

In section 6.7.5, an example is presented for down-hole BISON data. You can download this data set from the same web site that you downloaded BSU. The Geologan data example illustrates both the conversion of the digital signals and the transfer of geometry to the BSEGY data set.

6.4 Conversion Programs: BSEGY <–> [SEGY | ASCII | CVS | Bison | SEG2]

Section 6.1.1 above lists programs to convert between BSU's BSEGY format and the instrument formats (Bison and SEG-2). It also gives the program used to convert between the exchange format (SEGY) and the BSU format (BSEGY). Recognizing that some users will only wish to work with ASCII or Comma Separated Variable (CSV, spread sheet) values instead of remaining in the BSU family of programs, the following programs were composed.

- **seg2txt** This program converts from BSEGY to ASCII text.
- **seg2csv** This program converts from BSEGY to CSV (spread sheet compatible)
- **ba2s** – This program converts from ASCII text to BSEGY binary format.

6.4.1 seg2txt

This Fortran program would follow after first converting from either an instrument format (Bison | SEG-2) or from the SEGY exchange format to the BSU format, BSEGY. The online help shown below describes how one can limit the data either by time (tmin & tmax) or by trace number (fstrc & lstrc). The output file will be columns (traces) and time sample (down the page). One can also output an optional first column with sample time. Microsoft users often find the *.exe versions of these programs useful.

```
seg2txt infil tmin tmax fstrc lstrc timelist

 infil =input file name
 tmin  =minimum time
 tmax  =maximum time
 fstrc =first trace
 lstrc =last trace
 timelist = 0 do NOT add column of sample times
          = 1 Do add first column of sample times

 EXAMPLE:
  seg2txt 1000.seg .05 .10 2  5 0
 would generate text file with 5-2+1=4 columns
 No additional time column added
```

6.4.2 seg2csv

This C-language program would follow after first converting from either an instrument format (Bison | SEG-2) or from the SEGY exchange format to the BSU format, BSEGY. The online help shown below reveals that this program does not limit the data the way program `seg2txt` above does. For example, if the input file is 48 traces, 2500 samples per trace, the output *.csv file will have 2500 rows and 49 columns (the extra column being sample time, and is the first column). The advantage of this approach is that the user need not know how many channels or samples are in the file, and the disadvantage is that a very big file could swamp the limits of a spread sheet program.

```
   seg2csv  infile

     infile  = input file name (4char minimum)

From the Man page:
      EXAMPLE:
     seg2csv 0001.seg

     Input file 0001.seg   will  be  converted  and  output  as  file  named
     0001.csv
```

6.4.3 ba2s

This C-language program converts an ASCII text file to a binary BSEGY file. It converts in the opposite direction of the above programs `seg2txt` and `seg2csv`. The online help shows how one can read either a row or column order text file. Only minimal headers are generated. The user can employ the program `bhed` to edit the headers.

```
   ba2s  infile iorder ncol nrow dt

     infile   =  input file name
     iorder   =  1 each row is a trace, columns are time axis (int)
              =  0 each col is a trace, rows are time axis (int)
     ncol     =  number of columns in matrix (int)
     nrow     =  number of rows in matrix (int)
     dt       =  sample interval in micro seconds (int)

From the Man page:
   EXAMPLE:
       ba2s 8800.txt 0  62  5000 1000

       File 8800.txt is processed by ba2s.  Rows are time axis, 62 columns
       (ie. channels).  The number of samples per trace is 5000.  Sample
       interval is 1000 microseconds.
```

The procedure to edit headers includes running program `bhed` twice. First to download the current headers into a file. This file can be edited, and then the edited file uploaded back into the data. For reference, the bhed help is:

```
  bhed infile  header_file  iupdn

 infile       =input file name
 header_file  =file with selected header info
 iupdn        =1 download headers to header_file
              =0 upload headers to BSEGY data set
```

An example flow is as follows:

```
bhed 8800.seg 8800.hed 1
(edit file 8800.hed)
bhed 8800.seg 8800.hed 0
mv bhed8800.seg 8800.seg
```

Note that the file 8800.hed is created on the first bhed run. Use your favorite text editor to modify this. Then use the modified 8800.hed in the second run. This produces a file bhed8800.seg which can be renamed using the move (mv) command.

6.5 Setting Geometry

This is the most painful and most critical step in processing seismic data. The BSEGY format is derivative of SEGY format, and consists of a 240 byte trace header (which includes the geometry; source and receiver locations, elevations, sample interval, geophone polarity and orientation, etc.). The trace header is followed by the digital data for that source-receiver pair.

It is often the case that the Bison, EGG Geometrics, or whatever recording instrument does not contain enough information to complete the trace header. This is why paper observer's logs are filled out. Here is where the surveyor information is merged to complete the setting of trace headers. Header information like sample interval or number of samples are usually OK and included in the trace header when tools like **bis2seg, egg2seg, topcon, topcon2, bhed, bnez, gensetg, setgeom, genref** are run.

6.5.1 Setting Geometry SEG-2 Data: Example 1 [bnez-> gensetg-> egg2seg-> setgeom]

This example is for data recorded in SEG-2 format. If you wish to follow along, this sample is for a data set, recorded on an EGG Geometrics seismograph. Those data and scripts below can be downloaded from the SEGY-Data archive:

Go to https://cgiss.boisestate.edu/ pm/BSU
Check the **BSU Campus** check box and then submit button. The data are downloaded as archive **ID-100.zip**. Unzip the archive and go to Example 1. We begin with the program **bnez**.

```
#!/bin/bash
# Script gofirst
bnez 000001.nez 13 2 1 0 0 0 01 0 0 0 1
mv bnez.lst bnezshots.lst
bnez 000002.nez 30 1 14 0 0 0 01 0 1 0 1
mv bnez.lst bnezphones.lst
mv 000001.nez LAB001.nez
cat 000002.nez >>LAB001.nez
rm 000002.nez
```

The first command, **bnez**, automates the generation of source positions for an NEZ survey file (Northing=Y, Easting=X, Elevation=Z). Here is the online help when you type `bnez -h`

```
bnez  outfile  n-points tag so yo xo zo ido dy dx dz did

     outfile  = output file name (ex. aaaa0001.nez

     n-points = number of survey points to generate
     tag      = 1 tag=VP
              = 2 tag=SP
     so       = first value of sequence number
     yo       = northing of first point
     xo       = easting of first point
     zo       = elevation of first point
     ido      = initial ID number
     dy       = spacing between points in north direction
     dx       = spacing between points in east direction
     dz       = spacing in elevation between points
     did      = interval in ID between points
```

The contents of the file, **000001.nez**, contains 5 columns, [ID,Y,X,Z,Tag] and is as follows:

```
 1    0.000000    0.000000    0.000000 SP001
 2    0.000000    0.000000    0.000000 SP002
 3    0.000000    0.000000    0.000000 SP003
 4    0.000000    0.000000    0.000000 SP004
 5    0.000000    0.000000    0.000000 SP005
 6    0.000000    0.000000    0.000000 SP006
 7    0.000000    0.000000    0.000000 SP007
 8    0.000000    0.000000    0.000000 SP008
 9    0.000000    0.000000    0.000000 SP009
10    0.000000    0.000000    0.000000 SP010
11    0.000000    0.000000    0.000000 SP011
12    0.000000    0.000000    0.000000 SP012
13    0.000000    0.000000    0.000000 SP013
```

One generates a second file for the geophone locations, **000002.nez**, and then the two NEZ files are concatenated into a merged file **LAB001.nez**. Note the source did not move for all 13 shot record efforts.

The second **bnez** command produces file **000002.nez** shown below:

```
14    0.000000    0.000000    0.000000 VP001
15    0.000000    1.000000    0.000000 VP002
16    0.000000    2.000000    0.000000 VP003
17    0.000000    3.000000    0.000000 VP004
18    0.000000    4.000000    0.000000 VP005
19    0.000000    5.000000    0.000000 VP006
20    0.000000    6.000000    0.000000 VP007
21    0.000000    7.000000    0.000000 VP008
22    0.000000    8.000000    0.000000 VP009
23    0.000000    9.000000    0.000000 VP010
24    0.000000   10.000000    0.000000 VP011
25    0.000000   11.000000    0.000000 VP012
26    0.000000   12.000000    0.000000 VP013
27    0.000000   13.000000    0.000000 VP014
28    0.000000   14.000000    0.000000 VP015
29    0.000000   15.000000    0.000000 VP016
30    0.000000   16.000000    0.000000 VP017
31    0.000000   17.000000    0.000000 VP018
32    0.000000   18.000000    0.000000 VP019
33    0.000000   19.000000    0.000000 VP020
34    0.000000   20.000000    0.000000 VP021
35    0.000000   21.000000    0.000000 VP022
36    0.000000   22.000000    0.000000 VP023
37    0.000000   23.000000    0.000000 VP024
38    0.000000   24.000000    0.000000 VP025
39    0.000000   25.000000    0.000000 VP026
40    0.000000   26.000000    0.000000 VP027
41    0.000000   27.000000    0.000000 VP028
42    0.000000   28.000000    0.000000 VP029
43    0.000000   29.000000    0.000000 VP030
```

Once the merged NEZ file has been created, we can then run program **gensetg**. This is an interactive program, and the following is a log of the execution captured from the terminal:

```
 $ gensetg
 |----------------------------------------|
 | Copyright (C) 2009 P. Michaels         |
 |       All rights reserved              |
 | See GNU General Public License         |
 |----------------------------------------|
 gset:  TIME: 15:44:00 DATE: 29/Dec/2016
SHOTS: --------------------------
 Enter first shot file NAME number
1001
 Enter last shot file NAME number
1013
 Enter first SP label NUMBER
01
 Enter increment for SP label NUMBER
1
PHONES: --------------------------
 Enter number of BLOCKS to define channels
1
```

```
BLOCK Number------- 1
Channels (1) through (?)
 Enter last channel for this block
30
 Enter first label VP NUMBER for this block
01
 Enter label VP increment for this block
1
```

The result of gensetg is the creation of 2 files, shots.txt, which relates the data sets to the shot locations, and phones.txt, which relates the channel to VP location in the NEZ file.

File **shots.txt** is as follows:

```
1001.seg  SP001
1002.seg  SP002
1003.seg  SP003
1004.seg  SP004
1005.seg  SP005
1006.seg  SP006
1007.seg  SP007
1008.seg  SP008
1009.seg  SP009
1010.seg  SP010
1011.seg  SP011
1012.seg  SP012
1013.seg  SP013
```

File **phones.txt** is as follows:

```
01  VP001
02  VP002
03  VP003
04  VP004
05  VP005
06  VP006
07  VP007
08  VP008
09  VP009
10  VP010
11  VP011
12  VP012
13  VP013
14  VP014
15  VP015
16  VP016
17  VP017
18  VP018
19  VP019
20  VP020
21  VP021
22  VP022
23  VP023
24  VP024
25  VP025
26  VP026
27  VP027
28  VP028
29  VP029
30  VP030
```

We conclude by transferring the geometry to the seismic traces. The following script is used in this case:

```
#!/bin/bash

if ! [ -e LAB001.nez ]
then
echo "LAB001.nez missing, run gofirst"
else
  if ! [ -e shots.txt ]
  then
  echo "shots.txt missing, run gosecond (gensetg)"
  else
    if ! [ -e phones.txt ]
    then
    echo "phones.txt missing, run gosecond (gensetg)"
    else
# convert SEG2 files to BSEGY (*.seg), headers not final
FILES=‘ls -1 *.DAT |sort ‘
for f in $FILES; do
egg2seg $f;
done
rm *.lst

# apply correct geometry to *.seg files created above
setgeom shots.txt phones.txt LAB001.nez

#rename files (like setg1001.seg to 1001.seg)
for f in $FILES; do
NAME=‘basename $f .DAT‘
mv setg${NAME}.seg ${NAME}.seg
done

fi
  fi
    fi
```

The after checking for needed files, the bash script loops through data files, **1001.DAT through 1013.DAT** converting these SEG-2 files recorded with the EGG Geometrics. The result is files **1001.seg through 1013.seg** are created, but do not yet have the correct geometry. Program, setgeom, runs and applies the correct geometry to the BSEGY files, **setg1001.seg through setg1013.seg**. The last step in the script renames the files back to **1001.seg through 1013.seg**.

We can view the headers created using the **bdump** program. For example, the command, **bdump 1001.seg 0** produces the following text file. Note that the above process corrects the locations of sources and receivers. The program **egg2seg**, has inserted the trace length, filters, shot date and time. The assumption of shot and receiver orientation (180 degrees from vertical) is a default setting and may need to be corrected. For that, program **bhed** can be used.

```
                    |------------------------------|
                    |   PARTIAL SEGY HEADER DUMP   |
                    |                              |
                    |          1001.seg            |
                    |                              |
                    |------------------------------|

--------------------------------------------------------------------------------
  Length =  2000 samples             |  Shot Elevation =        0.0
  Sample Interval =    0.00025 sec.  |  Shot Depth =        0.0
  Delay Time =    0 msec.            |  Up Hole Time =     0 msec
  Low Cut Filter  =    0 Hz.         |  Shot X-COORD =       0.00
  High Cut Filter = 1000 Hz.         |  Shot Y-COORD =       0.00
  Line ID: 001                      |  Shot Date (year.moday) = 2004.0406
  Shot Orientation:                  |  Shot Time (hr:min)   = 14:16
  Azimuth=  0 Deg.  Vertical=180 Deg.|  Charge Size (grams)=     0
--------------------------------------------------------------------------------
TRACE|SHOT| STATION | OFFSET|        RECEIVER          |VERT|1STBRK|K-GAIN|AZI|VER|
  #  |REC.|SHOT  REC|       |  ELEV.  X-COORD   Y-COORD|FOLD|(SEC.)| (dB) |   |   |
-----|----|---------|-------|--------------------------|--|------|------|---|---|
   1 |   1| 001  001|   0.00|    0.00      0.00      0.00| 2|0.0000|   24 |  0|180|
   2 |   1| 001  002|   1.00|    0.00      1.00      0.00| 2|0.0000|   24 |  0|180|
   3 |   1| 001  003|   2.00|    0.00      2.00      0.00| 2|0.0000|   24 |  0|180|
   4 |   1| 001  004|   3.00|    0.00      3.00      0.00| 2|0.0000|   24 |  0|180|
   5 |   1| 001  005|   4.00|    0.00      4.00      0.00| 2|0.0000|   24 |  0|180|
   6 |   1| 001  006|   5.00|    0.00      5.00      0.00| 2|0.0000|   24 |  0|180|
   7 |   1| 001  007|   6.00|    0.00      6.00      0.00| 2|0.0000|   24 |  0|180|
   8 |   1| 001  008|   7.00|    0.00      7.00      0.00| 2|0.0000|   24 |  0|180|
   9 |   1| 001  009|   8.00|    0.00      8.00      0.00| 2|0.0000|   24 |  0|180|
  10 |   1| 001  010|   9.00|    0.00      9.00      0.00| 2|0.0000|   24 |  0|180|
  11 |   1| 001  011|  10.00|    0.00     10.00      0.00| 2|0.0000|   24 |  0|180|
  12 |   1| 001  012|  11.00|    0.00     11.00      0.00| 2|0.0000|   24 |  0|180|
  13 |   1| 001  013|  12.00|    0.00     12.00      0.00| 2|0.0000|   24 |  0|180|
  14 |   1| 001  014|  13.00|    0.00     13.00      0.00| 2|0.0000|   24 |  0|180|
  15 |   1| 001  015|  14.00|    0.00     14.00      0.00| 2|0.0000|   24 |  0|180|
  16 |   1| 001  016|  15.00|    0.00     15.00      0.00| 2|0.0000|   24 |  0|180|
  17 |   1| 001  017|  16.00|    0.00     16.00      0.00| 2|0.0000|   24 |  0|180|
  18 |   1| 001  018|  17.00|    0.00     17.00      0.00| 2|0.0000|   24 |  0|180|
  19 |   1| 001  019|  18.00|    0.00     18.00      0.00| 2|0.0000|   24 |  0|180|
  20 |   1| 001  020|  19.00|    0.00     19.00      0.00| 2|0.0000|   24 |  0|180|
  21 |   1| 001  021|  20.00|    0.00     20.00      0.00| 2|0.0000|   24 |  0|180|
  22 |   1| 001  022|  21.00|    0.00     21.00      0.00| 2|0.0000|   24 |  0|180|
  23 |   1| 001  023|  22.00|    0.00     22.00      0.00| 2|0.0000|   24 |  0|180|
  24 |   1| 001  024|  23.00|    0.00     23.00      0.00| 2|0.0000|   24 |  0|180|
  25 |   1| 001  025|  24.00|    0.00     24.00      0.00| 2|0.0000|   24 |  0|180|
  26 |   1| 001  026|  25.00|    0.00     25.00      0.00| 2|0.0000|   24 |  0|180|
  27 |   1| 001  027|  26.00|    0.00     26.00      0.00| 2|0.0000|   24 |  0|180|
  28 |   1| 001  028|  27.00|    0.00     27.00      0.00| 2|0.0000|   24 |  0|180|
  29 |   1| 001  029|  28.00|    0.00     28.00      0.00| 2|0.0000|   24 |  0|180|
  30 |   1| 001  030|  29.00|    0.00     29.00      0.00| 2|0.0000|   24 |  0|180|
```

6.5.2 Setting Geometry SEG-2 Data: Example 2 [bnez-> topcon2]

This is an alternative approach to illustrate the use of the **topcon2** program. The first step is the same as in Example 1. Program **bnez** is run to generate the NEZ data set of merged shot and geophone coordinates. The other steps, **gensetg, ee2seg, and setgeom** are replaced with calls to the **topcon2** program. The downside is that there is no interactive generator (gensetg). So this is good for small problems where one just edits a bash script to run.

```
#!/bin/bash
topcon2 LAB001.nez 1001.DAT  0001 0.0  1 30 001 030   1  0.   0   180   0   0
topcon2 LAB001.nez 1002.DAT  0001 0.0  2 30 001 030   2  0.   0   180   0   0
topcon2 LAB001.nez 1003.DAT  0001 0.0  3 30 001 030   3  0.   0   180   0   0
topcon2 LAB001.nez 1004.DAT  0001 0.0  4 30 001 030   4  0.   0   180   0   0
topcon2 LAB001.nez 1005.DAT  0001 0.0  5 30 001 030   5  0.   0   180   0   0
topcon2 LAB001.nez 1006.DAT  0001 0.0  6 30 001 030   6  0.   0   180   0   0
topcon2 LAB001.nez 1007.DAT  0001 0.0  7 30 001 030   7  0.   0   180   0   0
topcon2 LAB001.nez 1008.DAT  0001 0.0  8 30 001 030   8  0.   0   180   0   0
topcon2 LAB001.nez 1009.DAT  0001 0.0  9 30 001 030   9  0.   0   180   0   0
topcon2 LAB001.nez 1010.DAT  0001 0.0 10 30 001 030  10  0.   0   180   0   0
topcon2 LAB001.nez 1011.DAT  0001 0.0 11 30 001 030  11  0.   0   180   0   0
topcon2 LAB001.nez 1012.DAT  0001 0.0 12 30 001 030  12  0.   0   180   0   0
topcon2 LAB001.nez 1013.DAT  0001 0.0 13 30 001 030  13  0.   0   180   0   0
```

The topcon2 program converts an *.DAT (SEG-2) file to an *.seg (BSEGY) file, (what egg2seg does above). However, it does this with the aid of an NEZ file which contains the [Y, X, Z] coordinates of shots and receivers. In addition, the command line of topcon2 selects information from the NEZ file and permits insertion of shot and geophone orientations. The online terminal help is as follows:

```
  topcon2  topf seg2f lid shdp is nch vpl vpn ir esh isa isv ira ita

     topf   =  topcon file name
     seg2f  =  seg-2 file name
     lid    =  line ID
     shdp   =  shot depth
     is     =  shot location number
     nch    =  number of channels (nch<66)
     vp1    =  geophone station channel 1
     vpn    =  geophone station channel n
     ir     =  shot record number
     esh    =  elevation adjustment to be added
     isa    =  source polarization azimuth (deg.)
     isv    =  source polarization vertical (deg.)
     ira    =  reference phone polarization R-axis (deg.)
     ita    =  reference phone polarization T-axis (deg.)
```

6.5.2.1 NEZ Format The Northing, Easting, Elevation (NEZ) format is that which the FC-4 unit of a Topcon EDM (electronic distance measurement) survey tool outputs. In terms of a fortran specification, it is 5A12 format. Strings are converted to numbers by programs that read NEZ files. The best NEZ data are actual survey data. But absent that, one can use tools to build an NEZ file (like **bnez**).

```
columns:
123456789111111111122222222223333333333444444444455555
         012345678901234567890123456789012345678901234567890123

           1    0.000000    0.000000    0.000000 SP001
           2    0.000000    0.000000    0.000000 SP002
           3    0.000000    0.000000    0.000000 SP003
           4    0.000000    0.000000    0.000000 SP004

123456789111111111122222222223333333333444444444455555
         012345678901234567890123456789012345678901234567890123
```

Program topcon2 is written in the C-language. For those with Bison files, see the next section.

6.5.3 Setting Geometry Bison Data: [genref-> geom->geom2(go1)]

The program **genref** is an interactive program to aid in setting geometry. It creates files **geom, geom2, go1, *.xyz, *.nez**. For example, type genref from a text window. The following is a captured log:

```
  CDP Roll-a-long Pattern Generator
  Bison Format Data

 -----------SOURCES-----------------------------
   Enter 6 char. name for nez file (ex. STP001)
REF001
   Enter 4 char. LINEID
0001
   Enter Z-Datum: Elevation
500.
   Enter number of shots
1
   Enter Shot Record Names 8char: First
LOST0001
   Enter Shot Record Names 8char: Last
LOST0001
  Enter First Shot Station Number
01
   Enter First Source: x, y, z
0, 0, 100.
   Enter Last Source: x, y, z
0, 0, 100.
  Enter number of receivers in a shot gather
12
  Enter TOTAL NUMBER of stations on line
12
   Enter First Geophone Station: x, y, z
0, 140., 100.
   Enter Last Geophone Station: x, y, z
0, 250., 100.
  Enter first shot NEAR GEOPHONE station
01
  Enter first shot FAR GEOPHONE station
12
```

In this example, there is only one shot gather, Bison file named LOST0001. We are creating an *.nez file, REF001.nez. The Z-Datum is the elevation that all following elevations are measured from. Here, we set that to 500. Then later, the geophones and shots are specified as being 100. meters above that (final elevations are thus 600. meters). The generated file, REF001.nez is:

```
0001            0.0000      0.0000    600.0000  SP001
0001          140.0000      0.0000    600.0000  VP001
0002          150.0000      0.0000    600.0000  VP002
0003          160.0000      0.0000    600.0000  VP003
0004          170.0000      0.0000    600.0000  VP004
0005          180.0000      0.0000    600.0000  VP005
0006          190.0000      0.0000    600.0000  VP006
0007          200.0000      0.0000    600.0000  VP007
0008          210.0000      0.0000    600.0000  VP008
0009          220.0000      0.0000    600.0000  VP009
0010          230.0000      0.0000    600.0000  VP010
0011          240.0000      0.0000    600.0000  VP011
0012          250.0000      0.0000    600.0000  VP012
```

The generated files **geom, geom2, go1** must be set to executable using the command
`chmod +x g*`
(assuming there are no other files starting with the letter "g"). File **geom** has a record for each shot. Here, there is only one line:

```
topcon REF001.nez LOST0001 0001 0.0  1 12 001 012   1  0.   0   0   0   0
```

The call to topcon captures sample interval, number of samples, date of shooting, etc from LOST0001, the Bison file. It combines that information with the REF001.nez file geometry to create a file LOST0001.xyz which is then needed later by script **go1** which is called from **geom2** shown next:

```
go1 001
```

Here is the script **go1**:

```
 bis2seg LOST0$1
 bhed LOST0$1.seg LOST0$1.xyz 0
mv bhedLOST.seg L$1.seg
rm LOST0$1.seg
```

$ In summary after running **genref**, make the scripts executable and run them in sequence (geom, then geom2). The result will be file L001.seg in this example. The program **bdump** produces a partial listing of the headers:

```
                    |-------------------------------|
                    |   PARTIAL SEGY HEADER DUMP    |
                    |                               |
                    |           L001.seg            |
                    |                               |
                    |-------------------------------|

 ------------------------------------------------------------------------------------
   Length =  2000 samples             |  Shot Elevation =      600.0
   Sample Interval =    0.00020 sec.  |  Shot Depth =        0.0
   Delay Time =    0 msec.            |  Up Hole Time =     0 msec
   Low Cut Filter  =   16 Hz.         |  Shot X-COORD =       0.00
   High Cut Filter =  500 Hz.         |  Shot Y-COORD =       0.00
   Line ID: 0001                      |  Shot Date (year.moday) = 1992.0303
   Shot Orientation:                  |  Shot Time (hr:min)   = 17:07
   Azimuth=  0 Deg.  Vertical=  0 Deg.|  Charge Size (grams)=     0
 ------------------------------------------------------------------------------------
 TRACE|SHOT| STATION | OFFSET|        RECEIVER          |VERT|1STBRK|K-GAIN|AZI|VER|
   #  |REC.|SHOT  REC|       |  ELEV.  X-COORD   Y-COORD|FOLD|(SEC.)| (dB) |   |   |
 -----|----|---------|-------|--------------------------|--|------|------|---|---|
    1 |   1|001  001 | 140.00|  600.00      0.00    140.00| 1|0.0000|   60 |  0|  0|
    2 |   1|001  002 | 150.00|  600.00      0.00    150.00| 1|0.0000|   60 |  0|  0|
    3 |   1|001  003 | 160.00|  600.00      0.00    160.00| 1|0.0000|   60 |  0|  0|
    4 |   1|001  004 | 170.00|  600.00      0.00    170.00| 1|0.0000|   60 |  0|  0|
    5 |   1|001  005 | 180.00|  600.00      0.00    180.00| 1|0.0000|   60 |  0|  0|
    6 |   1|001  006 | 190.00|  600.00      0.00    190.00| 1|0.0000|   60 |  0|  0|
    7 |   1|001  007 | 200.00|  600.00      0.00    200.00| 1|0.0000|   60 |  0|  0|
    8 |   1|001  008 | 210.00|  600.00      0.00    210.00| 1|0.0000|   60 |  0|  0|
    9 |   1|001  009 | 220.00|  600.00      0.00    220.00| 1|0.0000|   60 |  0|  0|
   10 |   1|001  010 | 230.00|  600.00      0.00    230.00| 1|0.0000|   60 |  0|  0|
   11 |   1|001  011 | 240.00|  600.00      0.00    240.00| 1|0.0000|   60 |  0|  0|
   12 |   1|001  012 | 250.00|  600.00      0.00    250.00| 1|0.0000|   60 |  0|  0|
```

If there are any problems, or if one wishes to customize the geophone or shot orientations, that can be done by editing the *.xyz file and rerunning **geom2**. For setting down-hole data, see 6.7.5.

6.6 Plotting Seismic Data

Displaying seismic data in *bsegy* format is fundamental to evaluating data quality, processing of the data, and interpretation of the data. Much can be gained or lost from our view depending on how we decide to view or plot the data. BSU has its own program, *bplt*, which can be used in conjunction with other processing steps to view engineering seismic data (or any other digital signals, including those from earthquake recordings). In addition to the BSU program *bplt*, one can employ other freely available software to view *bsegy* format data. For the examples here, we will assume that one has a BSEGY data set and done the conversions to BSEGY (see section 6.1.1 for an example of how to do this).

6.6.1 Using bplt

The help facility for *bplt* is displayed by typing the following in an x-terminal: `bplt -h`

This will produce a listing of the command line parameters. The list is a bit long, as is shown below:

```
 |-------------------------------------------|
 | Basic Seismic Utilities (BSU) C-LANGUAGE  |
 | ONLINE HELP:                              |
 |-------------------------------------------|
 |Plot Seismic Traces using X or PostScript  |
 |for the output device                      |
 |-------------------------------------------|

  bplt  infile idev iorient itype 1tr Ltr tmin tmax istyl amp percnt xaxis
yaxis

    infile   =  input file name
    idev     =  output device
            0=    xwin/wxt  (Linux/MS Windows)
            1=    Post Script
            2=    xfig
            3=    jpeg
            4=    PDF
    iorient  =  orientation
            0=    landscape
            1=    portrait
    itype    =  select non-time axis type
            0=    trace number
            1=    offset
            2=    geophone z-coord
            3=    geophone x-coord
            4=    geophone y-coord
            5=    shot z-coord
            6=    shot x-coord
            7=    shot y-coord
    1tr      =  first trace to plot
    Ltr      =  last trace to plot
    tmin     =  minimum time to plot
    tmax     =  maximum time to plot
    istyl    =  style of plot
            0=    wiggle plot
            1=   black/white variable area
            2=   black/grey variable area
    amp      =  amplitude for 1 trace deflection
    percnt   =  percent overplot 100= 1 trace
    xaxis    =  length of x-axis (non-time) in inches
    yaxis    =  length of y-axis (time) in inches
    (if xaxis and yaxis absent, 6.0 by 4.0 inches
```

6.6.1.1 Example bplt There is a wide range of amplitude variation in most data. Program *bplt* plots the numbers as given. Here is an example using the "postscript" choice of "idev". Figure 1 shows the plot produced.

```
bplt twav.seg 1 1 2 1 78 0 .2 1 400 200 5 2
```

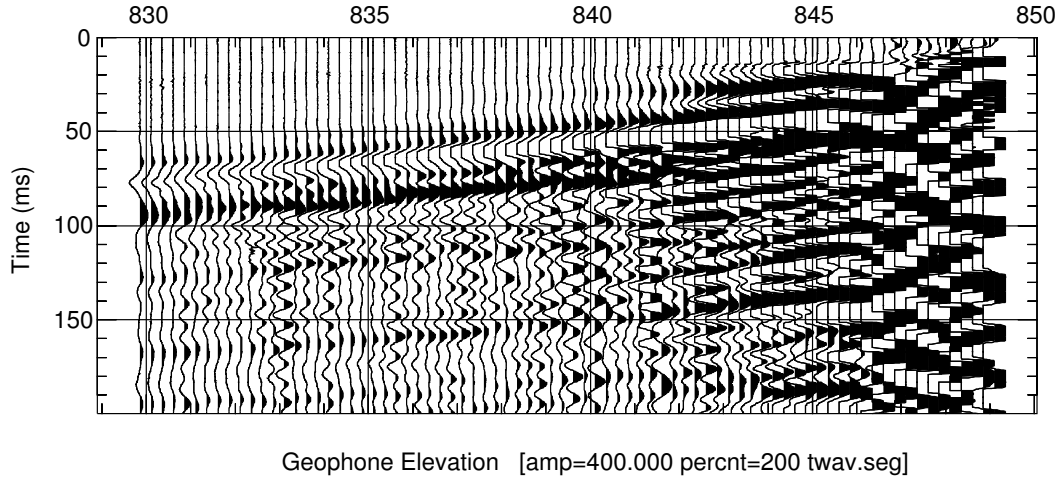


Figure 1: Example use of *bplt* to generate Post Script plot of down-hole data. Traces are plotted by geophone elevation. Data from a Boise River gravel borehole, B5.

We can zoom in, and plot a single trace. Doing so illustrates how much the data have been plotter clipped. For example, Figure 2 a single trace, number 41 (geophone at about 840 m elevation).

```
bplt twav.seg 1 1 0 41 41 0 .20 1 1500 300 5 2
```

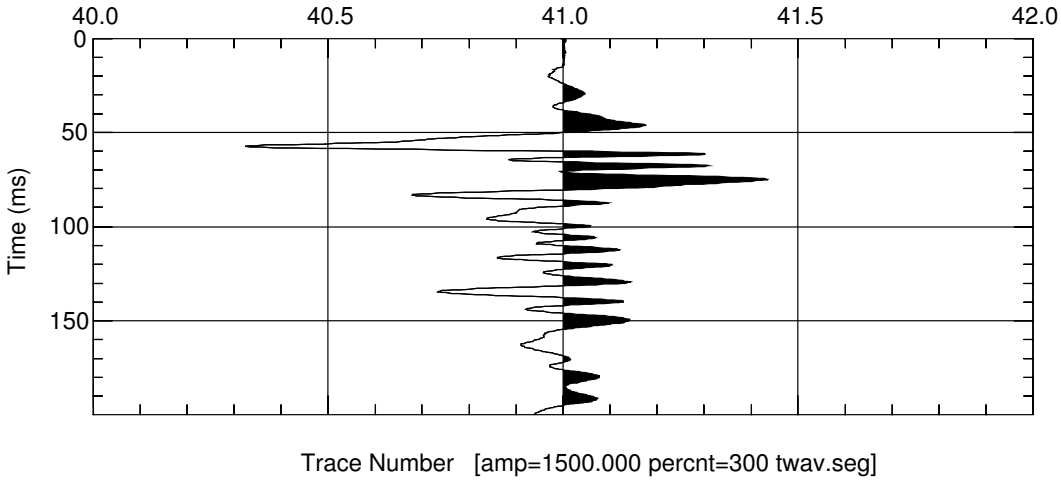


Figure 2: Example use of *bplt* to focus in on a single signal, trace 41.

Several arguments on the command line have changed. The first and last trace change from 1–78 to 41–41. The "amp" option changed from 400 to 1500. The larger the number, the smaller the deflection produced on the plot. Before, a value of 400 would plot at 1 trace spacing, now a larger sample value (1500) plots one trace deflection. *But, there is a limitation on this, the plotter clip*. The "percnt" option sets a limit the clips the plot of a sample value in terms of trace spacings. For example, in Figure 1, this was set to 200%, or 2 trace spacings. This is done to prevent traces plotting over each other. When we plot a single trace, as in Figure 2, we use two arguments to bring the entire amplitude range for that trace into view. We expand the permitted deflection to 300%, and set the plot deflection for a single trace spacing to require a larger amplitude (1500)

6.6.2 Using bplt in a bash script

BSU comes with some example bash scripts. The scripts *xplot* and *psplot* are examples of how we can combine trace equalization with plotting, and default a number of *bplt* parameters. As can be seen from the listing below, the script prompts the user for some values, then scales the data with either *bscl* or *bequ* before plotting.

```
#!/bin/sh
#  $Id: psplot,v 1.2 2009-05-05 19:43:43 pm Exp $
#
# Script to plot seismic data in Postscript format using BSU program bplt $
#Author: P. Michaels  <pm@cgiss.boisestate.edu>
#set -x
XDIM=5
YDIM=2

if test "$1" = "-h"
then
echo "USAGE: xplot  filename  tmax  scaling"
echo 'Scaling Choices:'
echo '1= Peak Absolute Value of profile'
echo '2= L2 Norm of profile'
echo '3= Trace by trace L2 Norm'
else
if test "$1" = ''
then
echo 'Enter input file name'
read FILEN
else
FILEN=$1
fi

if test "$2" = ''
then
echo 'Enter tmax'
read TMAX
else
TMAX=$2
fi

if test "$3" = ''
then
echo 'Enter Scaling Choice'
echo '1= Peak Absolute Value of profile'
echo '2= L2 Norm of profile'
echo '3= Trace by trace L2 Norm'
read SCL
else
SCL=$3
fi

NAME=`basename $FILEN .seg`
NAME4=`echo $NAME | gawk -F "" '{print $1$2$3$4}' `
case $SCL in
1)
bscl $FILEN 1 500 3
AMP=`gawk '/Peak Absolute Value/ {print $4}' bscl$NAME4.lst`
rm -f bscl$NAME4.*
PFILEN=$FILEN
bplt $PFILEN 1 1 0 1 500 0 $TMAX 1 $AMP 200 $XDIM $YDIM
 ;;
2)
bscl $FILEN 1 500 1
AMP=`gawk '/L2 Norm of Data Set=/ {print $6}' bscl$NAME4.lst`
rm -f bscl$NAME4.*
PFILEN=$FILEN
bplt $PFILEN 1 1 0 1 500 0 $TMAX 1 $AMP 200 $XDIM $YDIM
;;
3)
bequ $FILEN 0 $TMAX
PFILEN=bequ$NAME4.seg
AMP=4
bplt $PFILEN 1 1 0 1 500 0 $TMAX 1 $AMP 200 $XDIM $YDIM
rm -f bequ$NAME4.*
;;
esac
rm -f bplt*.lst
fi
```

There are two ways to run this script. One can simply type *psplot* and then be prompted for the data set name, the maximum time to plot, and the type of scaling to use. Alternatively, one can type all the arguments on the command line:

```
psplot twav.seg .2 3
```

which produces a plot as shown in Figure 3.

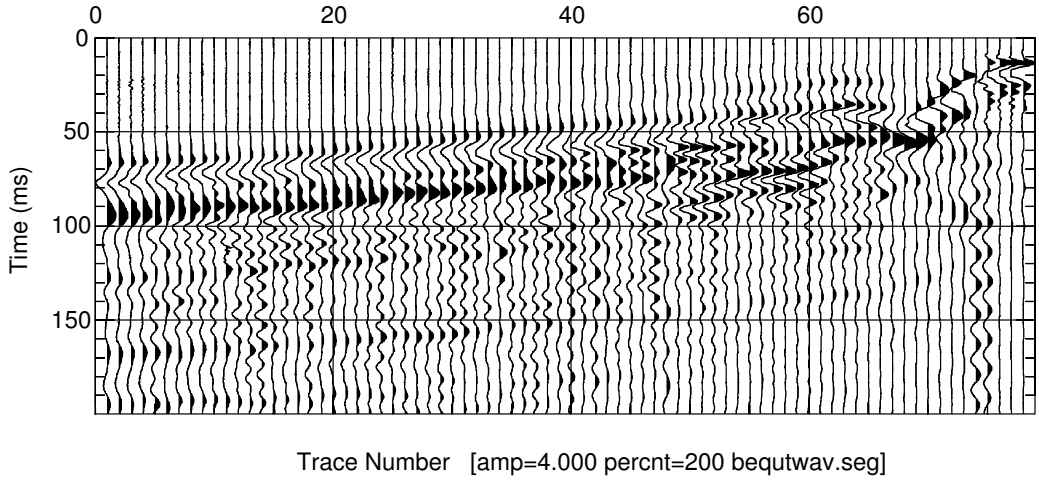


Figure 3: Example use of the *psplot* script. Data are rescaled by the L2 norm (option 3) of each trace before plotting by *bplt*

Another useful script is *xplot*. It plots the data using the X11 x-window device, and is very similar to the *psplot* script.

6.6.3 Plotting with traplt

If you really want to examine the data, sample by sample value, then use the program *traplt*. This program generates an old school lineprinter type listing. In addition to a plot of the signal, you also get a frequency spectrum. Let's say we wish to examine the first major trough in trace 41, using *bplt*, we might execute this command:

```
bplt twav.seg 1 1 0 41 41 .05 .06 2 1500 300 5 2
```

That would produce a plot as shown in Figure 4.

However, if we wish to see the actual sample values (in this case, in units of μvolts), we execute the following command:

```
traplt twav.seg 0.  .2 41 0 1
```

and page down to samples 251 through 300. This is the large negative also visible in Figure 2

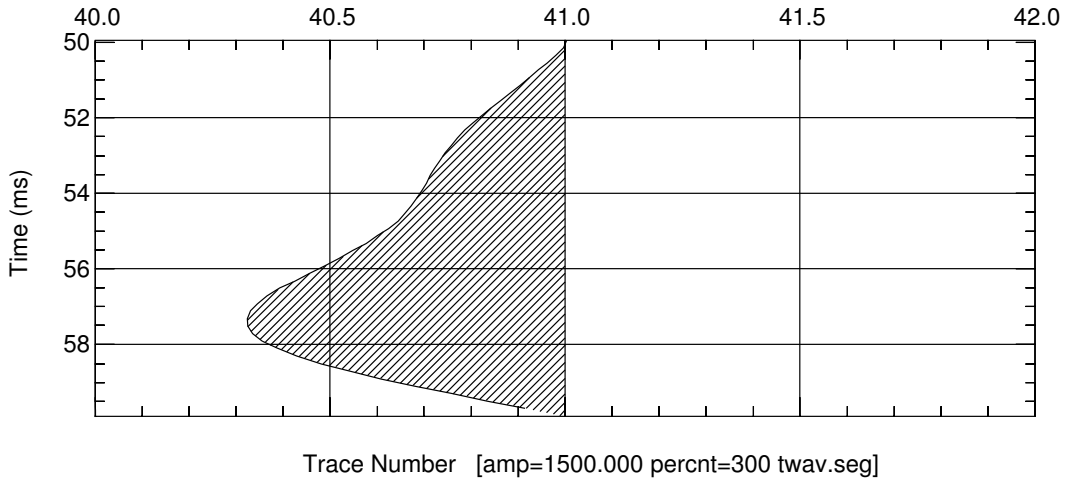


Figure 4: Using *bplt* to focus on trace 41 from 50 to 60 msec in time.

```
  max= 0.6539936E+03 min=-0.1014899E+04
   j           x(j) ..................................................
 251   0.3774579E+02 |                              .*                 |
 252   0.1636029E+02 |                              *                  |
 253  -0.5172668E+01 |                              *                  |
 254  -0.2949243E+02 |                             *.                  |
 255  -0.5731537E+02 |                            **.                  |
 256  -0.8488977E+02 |                           ***.                  |
 257  -0.1139543E+03 |                          ****.                  |
 258  -0.1435989E+03 |                          ****.                  |
 259  -0.1739700E+03 |                         *****.                  |
 260  -0.2052856E+03 |                        ******.                  |
 261  -0.2369359E+03 |                       *******.                  |
 262  -0.2657293E+03 |                      ********.                  |
 263  -0.2946696E+03 |                     *********.                  |
 264  -0.3198569E+03 |                    **********.                  |
 265  -0.3436373E+03 |                    **********.                  |
 266  -0.3639295E+03 |                   ***********.                  |
 267  -0.3823533E+03 |                  ************.                  |
 268  -0.4009266E+03 |                  ************.                  |
 269  -0.4172845E+03 |                 *************.                  |
 270  -0.4305879E+03 |                 *************.                  |
 271  -0.4431530E+03 |                 *************.                  |
 272  -0.4572617E+03 |                **************.                  |
 273  -0.4737784E+03 |                **************.                  |
 274  -0.4913515E+03 |               ***************.                  |
 275  -0.5107348E+03 |               ***************.                  |
 276  -0.5340413E+03 |              ****************.                  |
 277  -0.5633405E+03 |             *****************.                  |
 278  -0.5983062E+03 |            ******************.                  |
 279  -0.6352686E+03 |           *******************.                  |
 280  -0.6775207E+03 |          ********************.                  |
 281  -0.7210417E+03 |        **********************.                  |
 282  -0.7679259E+03 |       ***********************.                  |
 283  -0.8153746E+03 |     *************************.                  |
 284  -0.8634789E+03 |    **************************.                  |
 285  -0.9100562E+03 |   ***************************.                  |
 286  -0.9509911E+03 | ****************************.                  |
 287  -0.9814030E+03 | ****************************.                  |
 288  -0.1004137E+04 |*****************************.                  |
 289  -0.1014899E+04 |*****************************.                  |
 290  -0.1012347E+04 |*****************************.                  |
 291  -0.9976479E+03 |*****************************.                  |
 292  -0.9661345E+03 | ****************************.                  |
 293  -0.9193480E+03 |  ***************************.                  |
 294  -0.8567044E+03 |    *************************.                  |
 295  -0.7788740E+03 |       **********************.                  |
 296  -0.6881360E+03 |         ********************.                  |
 297  -0.5870426E+03 |            *****************.                  |
 298  -0.4769774E+03 |                ************.                  |
 299  -0.3637875E+03 |                   ***********.                  |
 300  -0.2464730E+03 |                       *******.                  |
                     ..................................................
  TMIN=  0.000 TMAX=  0.200 TRACE= 41 DELT=0.0002 FILE: twav.seg
```

The above example illustrates how one may sort out digital clipping from plotter clip. Trace 41 was recorded at a depth near 840 m elevation. In Figure 1, this first major trough appears clipped. This is only plotter clip, due to the choice of plotting parameters. Figures 2 and 4 show that one could choose other parameters for *bplt* to investigate the question of clipping. However, *traplt* takes us right to the sample values themselves. Data which have been digitally clipped are quite different. Such data clipping occurs when the signal is too large for either the geophone or the amplifier. Then a sequence of samples will have identical values out to the precision of the A/D converter. A quick way to quality check for digital clipping is to integrate the data. Clipped data will drift either positive or negative when integrated . You can integrate data with the program *bint*.

6.6.4 Plotting with SU

If you have installed Seismic Unix (SU), you might wish to use the example scripts *xPlot-su* and *psPlot-su*. The former can be used to generate a handy interactive X11 x-window plot. The latter creates a Postscript file, analogous to the BSU script *psplot*. The *psPlot-su* script is as follows:

```
 #!/bin/sh
# Script to call Seismic Unix Plotting
# fill=2 for grey fill of troughs
# Trace equalization with sugain if pbal=1
# d1num controls time lines, f2 controls vertical axis label
# Author: P. Michaels
set -x
if test "$1" = ''
then
echo 'Enter input file name'
read FILEN
else
FILEN=$1
fi
if test "$2" = ''
then
echo 'Enter tmax'
read TMAX
else
TMAX=$2
fi
if test "$3" = ''
then
echo 'Enter Title'
read TITLET
else
TITLET=$3
fi
segyclean <$FILEN |     \
suwind tmax=$TMAX       |       \
sugain pbal=1           |
supswigp xcur=1. style=normal   \
clip=1                          \
fill=2                          \
linewidth=0                     \
        xbox=1.0  wbox=4.0      \
        perc=98.                \
        ybox=6.5  hbox=3.65     \
        nbpi=300 grid1=solid grid2=solid        \
        titlesize=10 labelsize=10  \
        title="$TITLET $FILEN"  \
        label1=" Time (s)"      \
        d1num=.05               \
        f2=-19.60 d2=.25 label1=" Time (s)" label2="Elevation (m)"      \
         >$FILEN.ps
gv $FILEN.ps
```

The above script assumes you have ghost view "gv" installed at the end. If you don't, just comment out or delete that line and use your favorite tool to view the Postscript file generated. The script uses the SU program *segyclean* to remove BSEGY header info that conflicts with SU formatted data. The SU program, *suwind* windows the date for the time limits to be plotted. Then SU program, *sugain*, is analogous to BSU program *bequ* and trace equalizes the data. Finally, the SU program, *supswigp* does the actual plot generation. The line starting with "f2" is customized for the data set inquestion and sets the depth axis labels. Use BSU program, *bdump* to determine the deepest geophone depth (here that is -19.6 meters). The geophone increment is 0.25 meters. Figure 5 is the result of the following command:

```
psPlot-su twav.seg .2 pbal=1
```

The command line arguments are the data file name, maximum time to plot, and an arbitrary text string to label the plot. Here, that string is "pbal=1".

If you don't have SU, I recommend it highly, as it is well supported, but must be compiled from an unpacked TAR archive. To obtain SU, point your web browser to http://www.cwp.mines.edu/cwpcodes/index.html. Running SU is beyond the scope of this document. The CWP project has excellent documentation which also can be downloaded from their web site. The purpose here is to demonstrate the ease of processing between the two environments. This ease results from the similarity of the data formats unless ...

MAJOR CAUTION: *Beginning around 1997, seismic unix can be compiled two different ways. They have a new external data floating point format, XDR. Seismic Unix provides a program that converts old format SU data to new XDR format. That command is* ***suoldtonew <oldsu >newsu****. It does not appear that a inverse program has been distributed. Because BSU does not use XDR format, you will not be able to enjoy the ease of mixing BSU with SU processes, should you compile SU with the XDR format chosen in the Makefile.in file. However, I have a work-around.*

Work-A-Round: With this most recent release of BSU, you can now use the byte swap program, *bswp*, to toggle between the original SU data format and the XDR version. You could also generate a SEGY data set from a SU XDR data set (using SU program *segywrite*). Then use the BSU program, *bcnv*, to read it in to the BSU environment. But byte swapping is faster since both the traditional SU, BSU, and SUXDR are all IEEE encodings of floats. Only SEGY uses IBM big endian floats. What I do is ignore SU recommendations, and compile SU without XDR support. On personal computers running linux, this seems to be the easiest thing. There may be situations where different platforms are NFS sharing a data set that makes XDR desirable. That is not, however, a typical situation for most users.

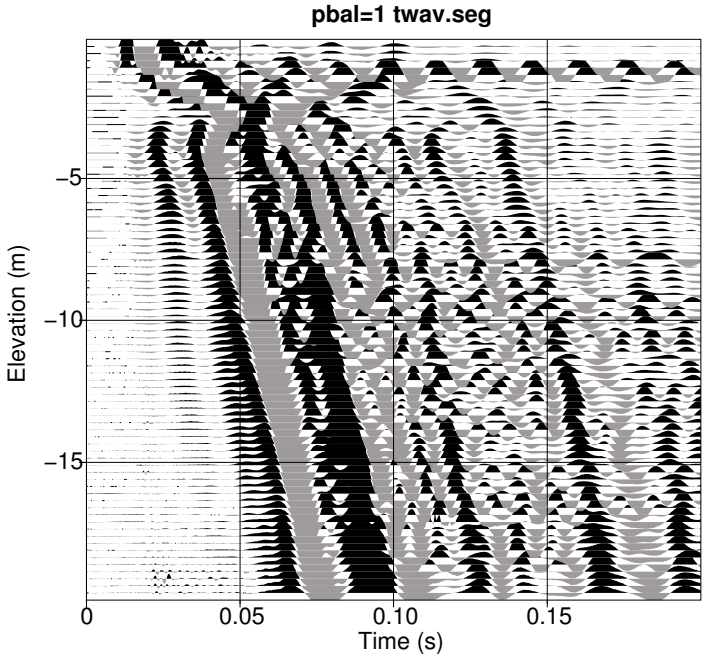


Figure 5: Using Seismic Unix (SU) to plot BSEGY data, script *psPlot-su* used.

6.6.5 Plotting with Gnuplot

As an alternative to Plplot, BSU programs can be compiled to pipe the graphics through the Gnuplot program (installed with your package manager). For compiled BSU using the `--enable-all-static` option, Gnuplot is the default for all BSU plotting programs.

IMPORTANT: After Gnuplot 4 a bug appeared so that only the positive numbers are plotted with Gnuplot 5 on a system. The fix is to compile BUS with the `--with-gnuplot5` option. Specifying this option will work for both Gnuplot 4 and 5 installations, but the *.gp files generated will be twice as large.

In any case, if BSU has been compiled with the Plplot configuration option, `--with-plplotlib`, there are still a few programs that only use the Gnuplot facility. These programs are:

- **qplt** – Quick plot for a profile. The command line options are:
  `filename tmin tmax`
  See figure 6 for an example.

- **tplt** – Single trace plot. The command line options are:
  `filename trace tmin tmax`
  See figure 7 for an example.

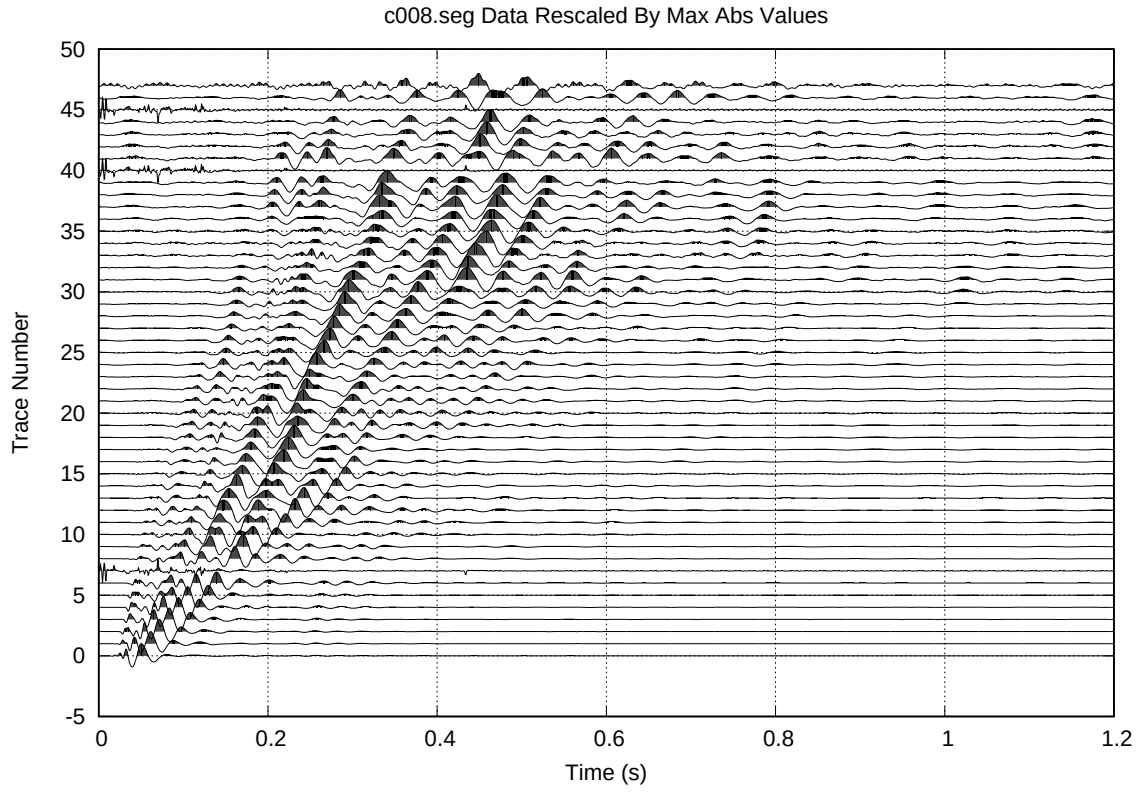


Figure 6: Plot of surface wave data using *qplt*. The qgraph.gp output file was edited to make a Postscript figure, and that is shown here.

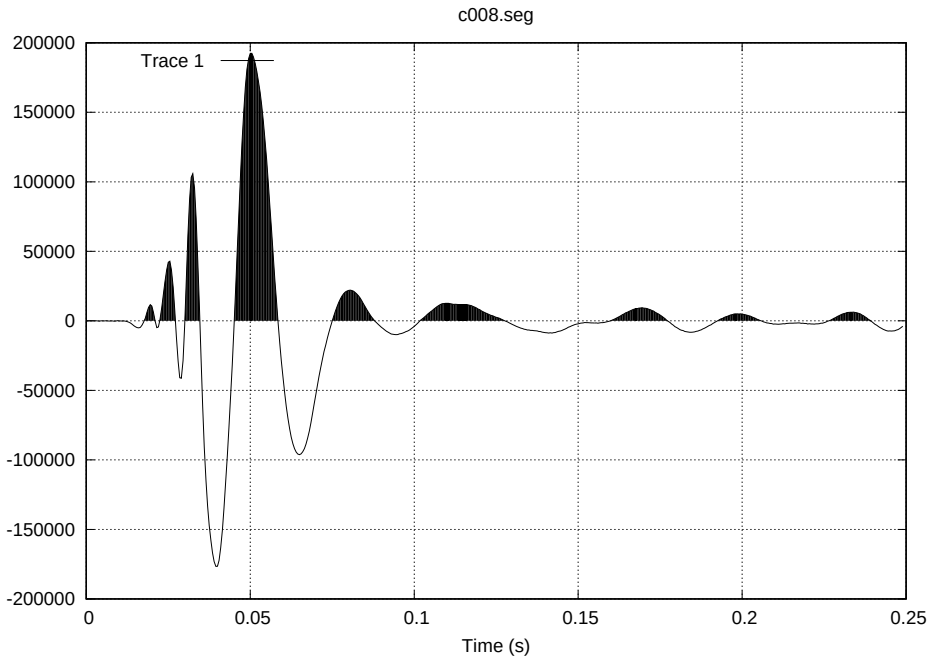


Figure 7: First trace of figure 6 surface wave data using *tplt*.

6.6.6 Plotting with Octave

BSU also comes with some example Octave programs which read the binary BSEGY format data and generate a plot. The program *traplt.m* is analogous to the BSU program *traplt* presented in section 6.6.3 above. While that was a "line printer" type of plot, the Octave version is GUI enabled. Other examples given below demonstrate other ways to plot with Octave.

6.6.6.1 Running traplt.m

1. Start an Octave session
2. In the Octave text window, enter the command:

```
traplt;
```

3. In the octave text window, enter the name of the file to be plotted when prompted.
4. The program calls two functions, segyinfo.m and bsegin.m. The first determines the number of traces in the file, the sample interval, and other details. The second is later used to read a trace from the binary BSEGY file.
5. An input GUI will pop up for entry of maximum time to plot and trace to plot. The defaults are the maximum recorded time and the last trace. Change these to what you want.
6. A plot of the signal should appear, along with a message GUI to pick a time zero for the phase plot. This will rotate the unwrapped phase.
7. Click OK on the message GUI, then move the mouse into the waveform plot and click a time zero reference.

8. Another GUI will pop up with the maximum frequency to plot in Hz. Modify the entry as desired and click OK.

Plots can be saved using the menu bars. Figure 8 shows the data from trace 10 of a surface wave data set.

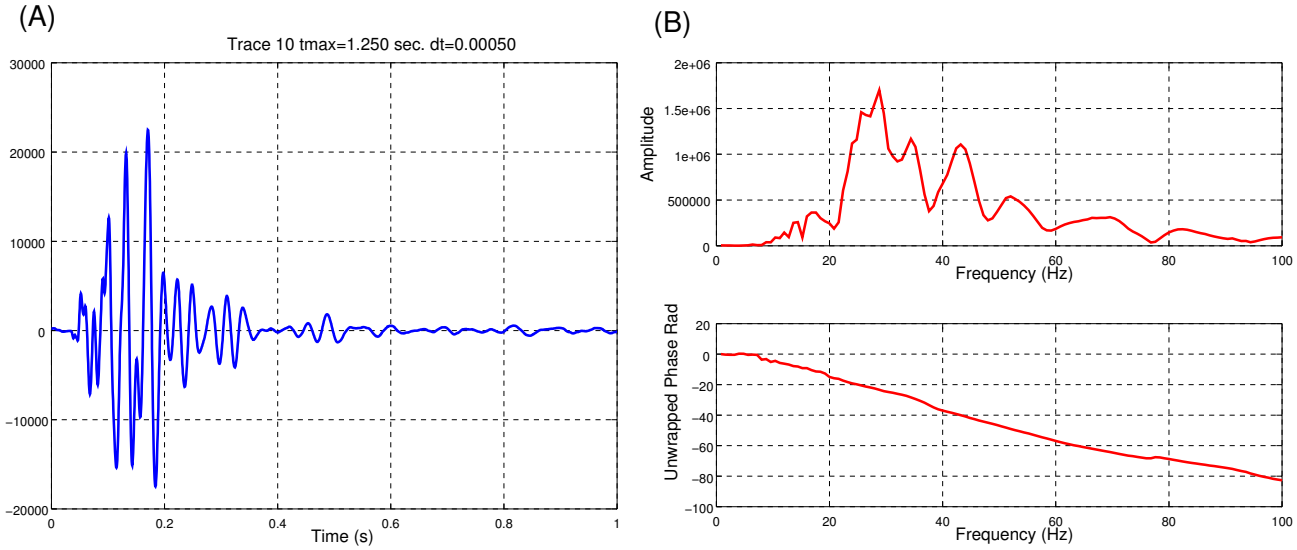


Figure 8: Plots produced by *traplt.m*. (A). Time domain (B). Frequency Domain

6.6.6.2 Running profplot.m This program produces a quick simple wiggle line plot of all the traces in a file. The plot GUI can be manipulated or saved as in Figure 9.

```
profplot;
```

6.7 Down-hole Seismic Processing

The data chosen for this example were acquired at the GeoLogan97 field day held on 15 July 1997, in Logan, Utah. GeoLogan was the first annual convention of the Geo-Institute of ASCE (American Society of Civil Engineers). The site was at a location in the valley floor, below Utah State University. Surface soils were silt. This example illustrates the steps needed to process data from the Bison Engineering Seismograph field records to a final vertical profiles for P- and SH-waves. An Octave procedure is used to invert for soil stiffness and damping properties.

You may download the Geologan97 data set at the BSU Database web page.
Go to https://doi.org/10.18122/geo_data/3/boisestate. The files are in **ID-101.zip**.
Or go to https://173.255.241.228/BSU/index.php. Click on the check boxes for 1997, Utah, and then on submit. It will list 3 download archives. ID-101.zip is the raw bison files with some scripts that will assign geometry. Run **geom** followed by **geom2**. To learn how these two scripts were generated, see section 6.7.5 below.

Alternatively, if you wish to start with SEGY data files, the other two *.zip archives are also available for download.

6.7.1 Seismic Source (SH- and P-wave)

BSU can handle a variety of seismic sources for down-hole engineering surveys. A vertical impact hammer source is used when the interest is primarily in recording P-waves on the vertical down-hole component. The hammer may be instrumented with a load cell (in which case an extra, 7th channel of data are recorded). Or the hammer can be without any instrumentation (other than contact closure for triggering). A horizontal hammer serves when the primary interest is in recording SH-waves on the horizontal down-hole components. In this example, my 135

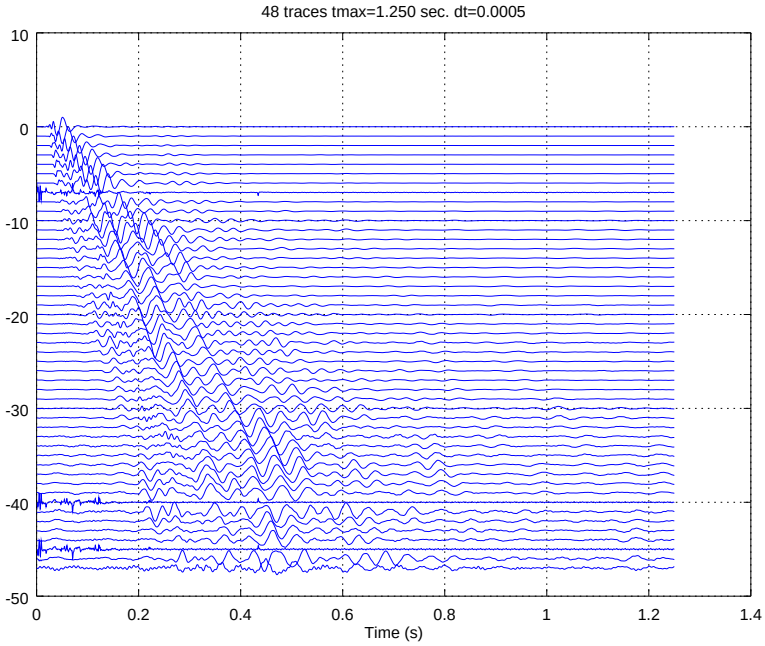


Figure 9: Plots produced by *profplot.m*.

degree inclined hammer source is used. This source delivers both horizontal and vertical motion to the ground. The source is nailed to the soil. The source is shown in Figure 10.

A sign convention for hammer sources has been adopted in BSU. One mentally associates an arrow with the blow. The arrow points in the direction of the blow, and can also be thought of as a unit vector. Thus, in Figure 10, if we assume that East is on the right, the hammer blow would be represented by the spherical coordinate designation (270,135)=(azimuth, angle from the vertical). Vertical angles are measured from zenith=0 degrees, to nadir=180 degrees. Horizontal, azimuthal angles, are measured from North=0 degrees, increasing clockwise (when viewed from above), making West=270 degrees. The BSU header stores both the horizontal and vertical angle for the source.

Data collection with this source involves two source polarizations for each sampled subsurface geophone station. In a typical survey, the geophone is lowered to the bottom of the hole, clamped with a mechanical bow spring clamp (I use a GeoStuff BHG-2 down-hole geophone [4]), and then dragged up the hole to occupy stations at a 0.25 meter interval. As the geophone remains fixed at a station, two separate seismic recordings are stored. One is for a source polarization of (270,135), and the other for a source polarization of (90,135). Subtraction of the records enhances SH-wave motion at the expense of Rayleigh and P-waves (as seen on horizontal components). Summing the two records has the opposite effect. For a more detailed discussion on wave field enhancement, see Michaels [12].

6.7.2 Down-hole and Reference Geophones

The BSU sign convention for geophones is similar to that for sources. An arrow or unit vector is associated with each geophone element. Ground motion in the direction of the arrow will generate a negative voltage at the geophone, and be recorded as a negative number on the seismic trace. For a typical moving coil “velocity” phone, this voltage is proportional to particle velocity of the soil, as sensed by the geophone at the point where it is clamped to the soil. Figure 11 shows a plan view of the typical survey for which BSU was written.

There are two 3-component geophones. One is fixed at the surface (or buried very shallow). This reference phone provides a way to monitor variations in the source waveform and recorder triggering. In Figure 11, the horizontal, R-component of the reference phone is designated (0,90), and the T-component (270,90). The vertical component arrow (not shown) is (0,0), pointing to zenith. The azimuth for the vertical component is meaningless, and so is set to zero for simplicity. The down-hole phone is free to spin on the way down, and so its orientation must be determined for each down-hole station. This is the job of the Principal Component Analysis (PCA), and

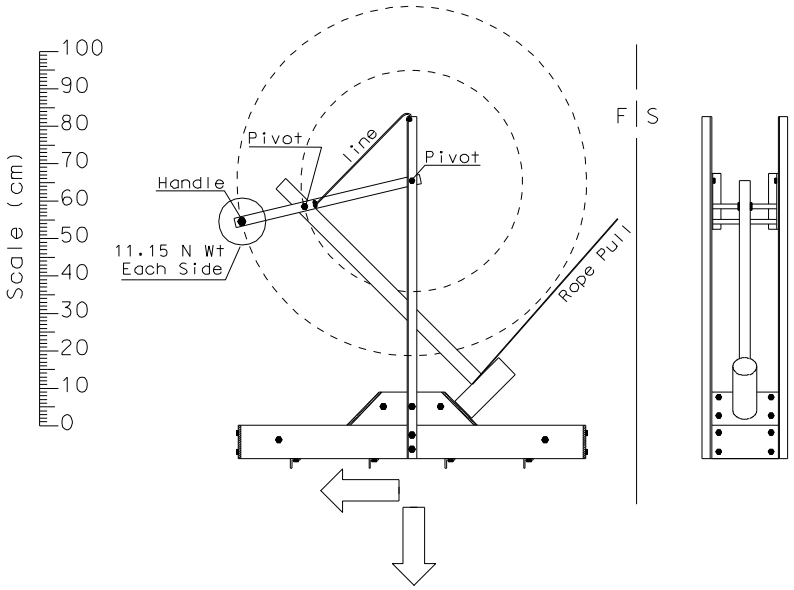


Figure 10: Source generates both horizontal and vertical motion

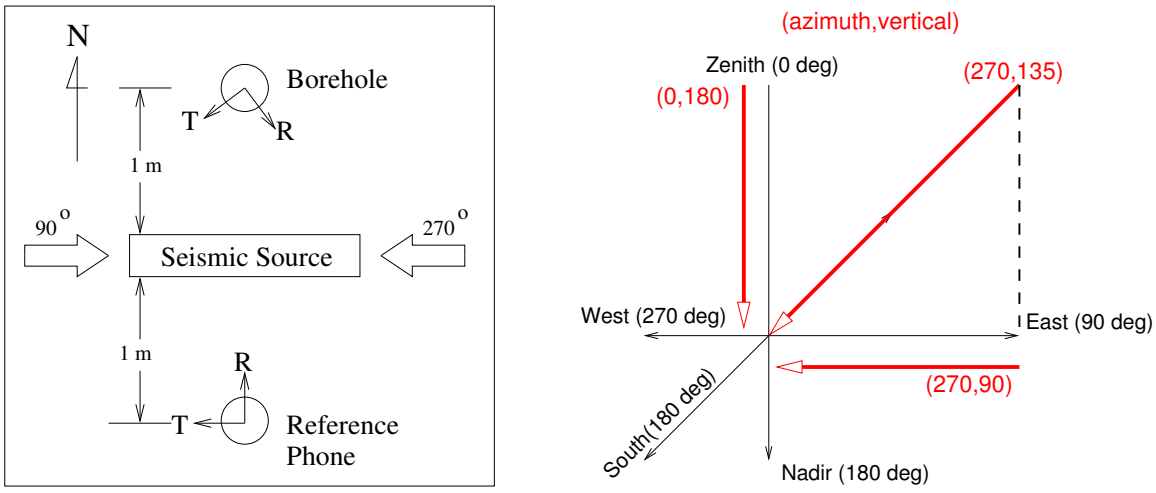


Figure 11: Plan view of a typical survey. Coordinate system for geophone components and impact forces.

will demonstrate the use of the program, *bhod,* named for the hodogram in the horizontal plane. The use of the labels R and T on horizontal components is purely arbitrary in down-hole surveys. You will note that in Figure 11, the chirality is not consistent (left handed vs right handed). Your geophones may be quite different, but the sign convention is what is important, and this can be determined by a tap test, or a shallow level where the tool orientation can be observed directly by sight. *A very important observation at the end of each survey is to note and record the bow spring azimuth as the tool exits the hole*. On the GeoStuff tool, the spring is aligned with the R-component. Knowing this direction is essential to establishing a guide vector for the PCA analysis which has an inherent 180 degree ambiguity. The vertical down-hole component points to zenith in the GeoStuff tool. Not all tools are constructed this way, so you need to check your tool to determine its actual orientation. The down-hole vertical component is designated (0,0).

Each seismic trace will have header values for the geophone and source polarization (unit vectors in spherical coordinates). These are stored as integers in the BSEGY header. Aside from looking at the include files, the header definitions can also be reviewed with the man pages. For Fortran definitions, type:

```
man bsegy
```

You will find `integer*2` header values for shot and geophone polarizations (*shtazi*, *shtver*, *geoaz*, and *geover*). For C-language definitions, type:

```
man c_bsegy
```

Thus, in C you will find the header structure elements *hd.geoaz*i and *hd.geover* contain the azimuth and vertical angles of the geophone. The corresponding shot polarization is stored in *hd.shtazi* and *hd.shtver*.

6.7.3 Sample Data Set from GeoLogan97

If you have downloaded the sample data set, it will be located in directory **PREFIX/ID-101/1997/15jul/bison**, where the PREFIX is your choice of top directory location. The READ.ME file included with the sample Bison data can be used as a reel header, and documents the experiment and local geology determined by a ConeTech survey. To run *genvsp*, we need to know the experimental geometry as recorded in the observer's log.

OBSERVER'S LOG

- **Global Coordinates of Borehole** (x,y,z)=(100.0,100.0, 1.025) in meters. BSU defines "Global" coordinates as being whatever coordinate system you are working in, usually set by the project you are tying into. Here, I have made up a value of (x,y) to illustrate how the codes work. The z-coordinate is the top of the casing stub from which down-hole measurements are taken. Normally, the casing elevation would be with respect to sea level. However, in this case, it is relative to the ground surface, since the field day did not include a survey tie in to a bench mark. **Note that the z-axis is +UP in global coordinates.**
- **Local Coordinates of the Source** (x,y,z)=(0.0, 1.24, 1.025). The local coordinate system is different, and is taken relative to the bore hole (which is at the origin). **Note that the z-axis is + DOWN in local coordinates**. The source was oriented with the long axis being East-West, the center of the source was 1.24 meters North of the bore hole.
- **Source Polarization of First (and ALL ODD) Files** (azimuth,vertical)=(90,135).
- **Source Polarization of Second (and ALL EVEN) Files** (azimuth,vertical)=(270,135).
- **Local Coordinates of the Reference Phone** (x,y,z)=(0.0, 2.17, 1.175). The ground was horizontal around the bore hole (which means that the reference phone was buried 0.15m in this case).
- **Depth to Top of Water Table**. Normally this would be measured in the bore hole. However, in this case, it appears that the PVC casing was sealed at the bottom, and the water level in the bore hole was not an indication of the local water table.
- **Geophone Station Spacing**. This was 0.25 meters.
- **Deepest Geophone Station**. This was for the first two shot records (files LOGN0001 and LOGN0002), and the phone was 19.25 meters below the casing elevation.

- **Shallow most Geophone Station**. This was for the last 2 records (files LOGN0145 and LOGN0146). The phone was at a depth of 1.25 meters below casing elevation.
- **Azimuth of bow spring on exiting the bore hole**. This was noted at 220^o, but in the dialog on PCA analysis, a guide vector of 240^o was found to require less manual corrections to the *bhod.lst* file.

6.7.4 Where to Find Scripts and Octave Codes

The location of Bash and Octave scripts depends on how BSU was installed.

- **If compiled from source**:
  Bash Scripts: /usr/local/share/scripts
  Octave Programs: /usr/local/share/octave/site-m
- **If installed as a Debian, *.deb, package:**
  Bash Scripts: /usr/share/
  Octave Programs: /usr/share/octave/site-m

6.7.5 Converting Bison Files to BSEGY Format and Setting Geometry

The conversion and geometry setting begins with program *genvsp*. Change to the directory with the Bison files. From the command line, run the interactive geometry setting program, *genvsp* by typing:

```
genvsp
```

The following is the dialog between the program and user (user response boxed).

```
|---------------------------------|
| Copyright (C) 2017  P. Michaels |
|       All rights reserved       |
|see GNU General Public License   |
|---------------------------------|

 Down-hole VSP  Pattern Generator
 For Setting Geometry
 Handles both Bison and SEG-2 File Formats

 Set Channel Order Switch
  1=ascending  1,2,3=downhole 4,5,6=reference
 -1=descending 6,5,4=downhole 3,2,1=reference
  2=ascending  1,2,3=down 4,5,6=ref,7=load_cell
 -2=descending 7=load_cell,6,5,4=down 3,2,1=ref
```

```
1
```

```
  -----------BOREHOLE----------------------------
   Enter 6 char. name for nez file (ex. STP001)
```

```
logn01
```

```
   Enter 4 char. LINEID
```

```
0001
```

```
   Enter Z-Datum: Casing Elevation
```

```
1.025
```

```
   BOREHOLE LOCATION:
   Borehole is origin of the local coordinate system
   Source and Reference phone locations are x,y
   relative to borehole.

   Following entries will shift every x,y input to
   a final global coordinate system:
   Enter Global x-coord. of borehole
```

```
100.0
```

```
   Enter Global y-coord. of borehole
```

```
100.0
```

```
   Enter number of sources
```

```
2
```

```
 FOR THIS SOURCE:
   Enter Shot Record Names 8char: First
```

```
LOGN0001
```

```
   Enter Shot Record Names 8char: Last
```

```
LOGN0145
```

```
     LOGN0001LOGN0145
   Enter Source: x, y, z_sub_CE (positive down)
```

```
0., 1.27, 1.025
```

```
   Enter Source Polarization: azi, ver
```

```
90, 135
```

```
 FOR THIS SOURCE:
   Enter Shot Record Names 8char: First
LOGN0002
   Enter Shot Record Names 8char: Last
LOGN0146

     LOGN0002LOGN0146
   Enter Source: x, y, z_sub_CE (positive down)
0., 1.27, 1.025
Enter Source Polarization: azi, ver
270, 135

 ----------REFERENCE RECEIVER-----------------
   Enter Reference: x, y, z_sub_CE (positive down)
0., 2.17, 1.175
Enter Reference Polarizations: R-azi, T-azi
0, 270

 ----------BOREHOLE PHONES--------------------
   Enter Bulk Shift (Added To Geophone Depth ONLY)
0.

For Shot: SP01 AZI= 90 VER=135

Enter Station Spacing: dz
.25
Enter First Station Depth: zmax
19.25
Enter Last Station Depth: zmin
1.25

For Shot: SP02 AZI=270 VER=135

Enter Station Spacing: dz
.25
Enter First Station Depth: zmax
19.25
Enter Last Station Depth: zmin
1.25

  Number of receivers =    73
-------------------------------------------
  CHECK DATA TYPE

  Files like XXXX0001 detected, ID=BISON
  Is above ID Correct, or overide needed?
  1=YES correct    0=NO incorrect
1

  Number of receivers =    73
-------------------------------------------
  CHECK DATA TYPE

  Files like XXXX0001 detected, ID=BISON
  Is above ID Correct, or overide needed?
  1=YES correct    0=NO incorrect
```

```
1
```

The above dialog between the program and user creates the following files:

- *logn01.nez* survey file with 5 columns: [index y-coord x-coord z-coord char_label]
- *geom* first script to run, calls program, *topcon.*
- *geom2* second script to run, calls script *go1*
- *go1* script called by *geom2,* calls programs *bis2seg* and *bhed.*

You will need to change permissions to execute (files *geom, geom2, go1*). Conversion from Bison to BSEGY and the setting of geometry is done by:

1. Run *geom* creates files LOGN0001.xyz to LOGN0146.xyz which contain geometry
2. Run *geom2* creates files L001.seg to L0146.seg which are in BSEGY format.

6.7.5.1 Post *genvsp* processing steps

1. Removing the *.xyz files if they appear OK. These are in *bhed* format. The *bhed* program was run in script *go1* called by *geom2*.
2. Removing the *.lst files if they appear OK. These files are created by *bis2seg* and contain a listing of the Bison file header information. Program, *bis2seg*, was run in the *go1* script called by *geom2*.
3. Gzip the Bison files to save disk space.
4. Move the L*.seg files to a new directory. For example:

   ```
   mv L*.seg ../seg
   ```

   assuming that the directory **ID-101/1997/15jul/seg** exists.

The newly created *.seg files in BSEGY format have the majority of the header information set. What is missing is the horizontal component orientation of the down-hole tool. For that, we must run programs *genbhod* and *bhod*.

6.7.6 Determining Down-hole Tool Orientation by PCA

The geometry will be complete once the down-hole tool horizontal component orientation has been established as described in Michaels [12]. We begin by running the *genbhod* program which builds scripts that control the hodogram analysis by program *bhod*. The following is the dialog between *genbhod* and the user (boxed). The program is run from inside the directory with the newly created `L*.seg` files (created from Bison format):

```
|---------------------------------|
| Copyright (C) 2017  P. Michaels |
|       All rights reserved       |
|see GNU General Public License   |
|---------------------------------|

WARNING: !!
See Source Code, genbhod.f, or BSU documentation
(man pages and BSU user Guide)
before you use this program. It is hardwired for
a specific type of acquisition.

enter 1char_ALPHA PREFIX
```

```
L
```

```
  enter FIRST FILE NUMBER (<=3digits)
  for which source polarization is 270 deg.
```

```
2
```

```
  enter LAST FILE NUMBER (<=3digits)
  for which source polarization is 270 deg.
```

```
146
```

```
  enter UP/DOWN SWITCH
    -1= 90 Azimuth File Number 1 LESS than 270 Az
    +1= 90 Azimuth File Number 1 MORE than 270 Az
```

```
-1
```

```
enter azimuth of bowspring(R-comp)
```

```
240
```

```
  OUTPUT====> Downhole: gobhodo
  OUTPUT====> Reference: gobhodoR
  OUTPUT====> Downhole: gorunbhod
  OUTPUT====> Reference: gorunbhodR
  ----------------------------------------
 REMEMBER to change permissions on the
 above files to execute.
  ----------------------------------------
  IF examining the Down-hole Phone

  1. Run gobhodo in directory with 6 chan
  records (3 down, 3 reference phones)

  2. Run gorunbhod in directory with files
  that are named hxxxyyy.seg

  ----------------------------------------
  IF examining the Reference Phone

  1. Run gobhodoR in the directory with
  the 6 channel records.

  2. Run gorunbhodR in the directory with
  files that are named rxxxyyy.seg

  ----------------------------------------
```

Because the tool is fixed for both source polarizations, there will be half as many orientation determinations as there are seismic files. The tips at the bottom of the dialog are reminders about which scripts are for what purpose. In a normal case, one generally is only interested in the down-hole tool orientation, and will only run scripts *gobhodo* and *gorunbhod*. If you are interested in viewing the actual rotation of the radiation from the source with time, then you will run the other two scripts, *gobhodoR* and *gorunbhodR* (the upper case "R" being a reminder that these are for the fixed reference phone). The following steps are recommended:

1. Make a new directory for the hodogram analysis under the current "seg" directory. For example,

   ```
   mkdir hodo
   ```

   will do this nicely.

2. Run the script, *gobhodo* in the current "seg" directory. This creates a lot of files.
3. Move the files beginning with "h" to the directory "hodo" created in step (1)

```
mv h*.seg hodo
```

4. Remove the *bscl* generated files.

```
rm bscl*
```

5. Copy gorunbhod to the "hodo" directory.

```
cp gorunbhod  hodo
```

6. Change into the hodo directory and run *gorunbhod*. The important file to save is *bhod.lst*. It contains a list of the file numbers with tool orientations. Copy *bhod.lst* back to the "seg" directory.
7. Run the script, *mergeplots*, provided in the script directory of the distribution (see 6.7.4). This merges all the Postscript (*.ps) files into a single PDF file for viewing in a program like *acroread (Adobe Acrobat).*
8. Remove the *.ps files, and view the file, *merge.pdf*.
9. Look for 180^oreversals in polarity, particularly if the tool had to be released during any part of the survey (and hence was free to spin). Sometimes, a slight change in the guide vector azimuth will help on a second attempt.

Figure 12 shows the plot for the files L141.seg and L142.seg. The SH-wave enhancement involved subtracting the even from the odd files (see *gobhodo* script). The result was file h141.seg. It is this file, h141.seg which was then subjected to PCA analysis by program *bhod*.

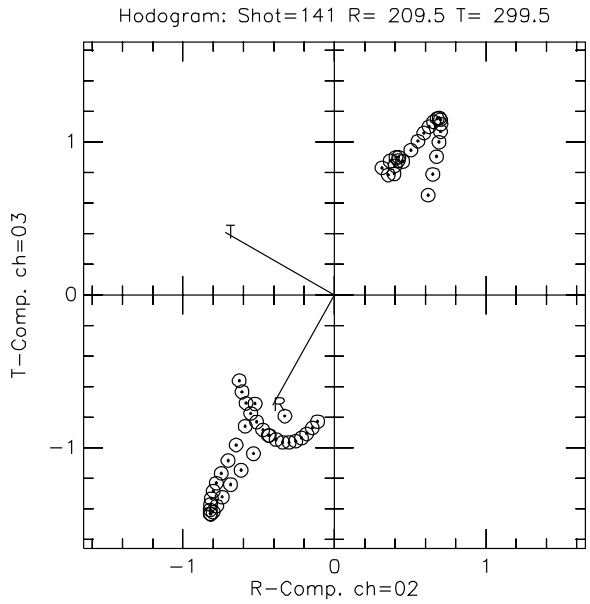


Figure 12: PCA result (file h141plt.ps) for near surface geophone station.

Since this is a shallow station, it corresponds to the observed orientation of the bow spring when the tool exited the hole. Without the guide vector (set at 240^o in the dialog above), the resulting R-component direction

could easily have been rotated by 180^o due to the inherent ambiguity in the eigenvector solution. The line in the file *bhod.lst* corresponding to this station is: `00141 209.5 299.5` from which you can see the meaning of the 3 columns. The first column corresponds to the file number, the second column the R-component azimuth, and the third column is the T-component azimuth. The values will be truncated to integers when written to the headers by program *btor*.

There is one case in which *bhod* returned a tool orientation rotated by 180^o. This is for the deepest level. The tool twisted enough during the survey to make the single guide vector insufficient. With only one problem, it is easier to edit the bhod file directly.

Figure 13 shows the problem with the deepest level. In the figure are two hodograms, one for h001.seg, and the other for h003.seg (the next shallower station). You will see from examining file *merge.pdf* that h003.seg and shallower levels resulted in a consistent set of tool orientations.

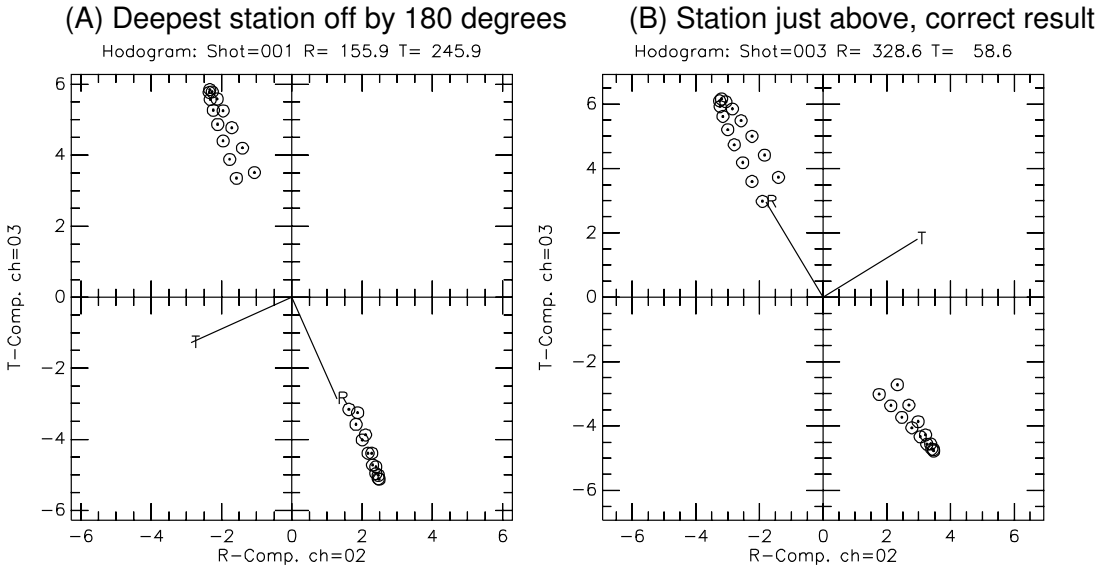


Figure 13: Deepest level (A) is 180 degrees off from desired as shown in (B)

The way to correct the deepest level here is to change the line in the *bhod.lst* file. The current first few lines are:

```
00001 155.9 245.9       <----this one is in error by 180o
00003 328.6 58.6
00005 323.1 53.1
00007 314.1 44.1
```

The first line should be manually edited so that these lines become:

```
00001 335.9 65.9        <----fixed, rotated by 180o
00003 328.6 58.6
00005 323.1 53.1
00007 314.1 44.1
```

The result is a consistent set of orientations, and one that continues to the surface and agrees with the observed orientation of the tool as it exited the hole.

6.7.7 Inserting the PCA Results to the Trace Headers (*btor*)

The next step is to copy the file *bhod.lst* (after any manual editing) back into the "seg" directory where the files L001.seg to L0146.seg are located. File *bhod.lst* will direct the processing of program *btor*. The command line help for *btor* includes the following:

```
btor -h
```

```
|--------------------------------|
| Copyright (C) 2017  P. Michaels |
|       All rights reserved       |
|see GNU General Public License   |
|--------------------------------|

|----------------------------------------|
| Basic Seismic Utilities     FORTRAN     |
| ONLINE HELP:                            |
|----------------------------------------|
| btor: Applies azimuth and vertical angles|
| to geophone trace headers (from a file)  |
|----------------------------------------|

btor lstfil, prfx, isw1 maxtr

lstfil  =input list file name (ex. bhod.lst)
prfx    =*.seg file prefix (one character)
isw1    =up/down switch
      -1=apply to *.lst file and one less
      +1=apply to *.lst file and one more
       0=IF VERTICAL IMPACT source
maxtr   =maximum number of traces in shot record
       6= 3 components down-hole, 3 ref-phone
       7= 3 down, 3 ref-phone, 1 load cell
```

In our case, there are 6 channels per shot record (maxtr=6) and the file *bhod.lst* consists of odd file numbers (1,3,5, odd). Thus, isw1=+1 so that the solution for file L001.seg will also be applied to L002.seg (2,4,6, even). The prefix for the 4 character file names is "L", and of course, the lstfil=*bhod.lst*. We can run btor with the following command:

```
btor bhod.lst  L  +1  6
```

If a load cell channel had been recorded from the hammer, the last argument would have been "maxtr=7". In the case of these data, no load cell channel was recorded.

The result of the above *btor* run will be files with names *btorL001.seg* through *btorL0146.seg*. The next step is to copy these files over the original files named *L001.seg* throu *L0146.seg*. In the script directory of the distribution is a script file, *rename-btor*, which will do the job. The only parameter it requires is the file name prefix, "L", in this particular case. The script may be run by typing

```
rename-btor L
```

or

```
rename-btor
```

and then be prompted for the prefix. This script is an excellent example of using the find and stream editor commands, and is as follows:

```
#!/bin/sh
#Script to rename files after btor process
#overwrite pxxx.seg files, p=prefix
# Author: P. Michaels     Date:April 2002   $ee GNU License

if test "$1" = ''
   then
     echo 'Enter 1 character prefix'
     echo 'Example:  w'
     echo ' for files btorw001.seg, btorw002.seg, etc...'
     read PRFX
   else
     PRFX=$1
fi

 find -name "$PRFX*.seg" |  \
 sed s/'\.\/'/' '/g |       \
 gawk '{print "mv","btor"$1,$1}'   \
  >go-rename
chmod +x go-rename
./go-rename
echo "btor files renamed"
```

Once the script has the prefix character, a find command is issued to get a list of the files L*.seg, and this is piped through sed editor to remove the leading "." and "/" characters. The result is piped through gawk (the GNU version of awk) to construct move commands which are written to a file, go-rename, that file is made executable, and then executed.

6.7.8 Checking the Headers for Source and Geophone Polarizations(*bdump*)

The *bdump* program creates a partial header dump for a BSEGY file.

The normal ascending trace order I use is defined as:

1. down-hole V (vertical)
2. down-hole R (horizontal)
3. down-hole T (horizontal)
4. reference V (vertical)
5. reference R (horizontal)
6. reference T (horizontal)

See section 6.7.5 above, at the beginning of the *genvsp* run. If you don't use this order or its reverse, then you may have to modify the codes. We can check the files L141.seg and L142.seg (recall hodogram of Figure 12) to see if the headers have been formed correctly. Type:

```
bdump  L141.seg  0
less bdump.lst
```

the result is the partial header dump for file L141.seg as shown below:

```
                    |------------------------------|
                    |   PARTIAL SEGY HEADER DUMP   |
                    |                              |
                    |          L141.seg            |
                    |                              |
                    |------------------------------|

--------------------------------------------------------------------------------
  Length =  2000 samples             |  Shot Elevation =        0.0
  Sample Interval =    0.00025 sec.  |  Shot Depth =        0.0
  Delay Time =    0 msec.            |  Up Hole Time =     0 msec
  Low Cut Filter  =    4 Hz.         |  Shot X-COORD =     100.00
  High Cut Filter = 1000 Hz.         |  Shot Y-COORD =     101.27
  Line ID: 0001                      |  Shot Date (year.day) = 1997.0715
  Shot Orientation:                  |  Shot Time (hr:min)   = 11:48
  Azimuth= 90 Deg.  Vertical=135 Deg.|  Charge Size (grams)=     0
--------------------------------------------------------------------------------
TRACE|SHOT| STATION | OFFSET|        RECEIVER          |VERT|1STBRK|K-GAIN|AZI|VER|
  #  |REC.|SHOT  REC|       |  ELEV.  X-COORD   Y-COORD|FOLD|(SEC.)| (dB) |   |   |
-----|----|---------|-------|--------------------------|--|------|------|---|---|
   1 | 141| 001  421|   1.46|   -0.73    100.00    100.00| 3|0.0000|   20 |  0|  0|
   2 | 141| 001  422|   1.46|   -0.73    100.00    100.00| 3|0.0000|   20 |209| 90|
   3 | 141| 001  423|   1.46|   -0.73    100.00    100.00| 3|0.0000|   20 |299| 90|
   4 | 141| 001  424|   0.91|   -0.15    100.00    102.17| 3|0.0000|    0 |  0|  0|
   5 | 141| 001  425|   0.91|   -0.15    100.00    102.17| 3|0.0000|    0 |  0| 90|
   6 | 141| 001  426|   0.91|   -0.15    100.00    102.17| 3|0.0000|    0 |270| 90|
```

Note that the two down-hole horizontal traces, 2 and 3, have azimuth R= 209^o and T= 299^o as determined in the Figure 12 hodogram. The dump for L142.seg is almost identical to that above, the only differences being the source polarization (which is azimuth=270^o rather than 90^o as show here), and of course the time of day.

6.7.9 Using *seisazi.m* to display azimuth headers

Examining header dumps can be tedious, so an alternative graphical scan helps evaluate the tool orientation solution. In the directory with the `Lxxx.seg` files, we can pull off selected channels of the data (using *bmrg*) and then plot the geophone azimuth header using Octave. To extract the channel 2 component for the odd shots, we give the commands:

```
bmrg L 1 146 2 2 2
mv bmrg.seg bmrg0002.seg
```

To extract the channel 3 component for odd shots

```
bmrg L 1 146 2 3 3
mv bmrg.seg bmrg0003.seg
```

We don't need more than either the odd or even channels because the tool is fixed for the two source polarizations at each depth. Insertion of the azimuth determinations will be identical for any odd-even pair at the same depth. We start an Octave session and execute the following command from within the Octave text window

```
seisazi;
```

One responds to the GUI dialog prompts for file name. Note that `seisazi.m` requires both `segyinfo.m` and `bsegin.m` to be installed in the directory with the data. One either sets up a path to the *.m files, or copies them from `/usr/local/share/octave/site-m`. Combining two runs of *seisazi.m* results in Figure 14

6.7.10 Rotating the Horizontal Data into Alignment with Source (*genbrot* and *brot*)

Once the headers have been updated with the down-hole tool orientation, the next step is to rotate the horizontal data into a single orientation. This removes any spin that has occurred while dragging the tool up the hole. The

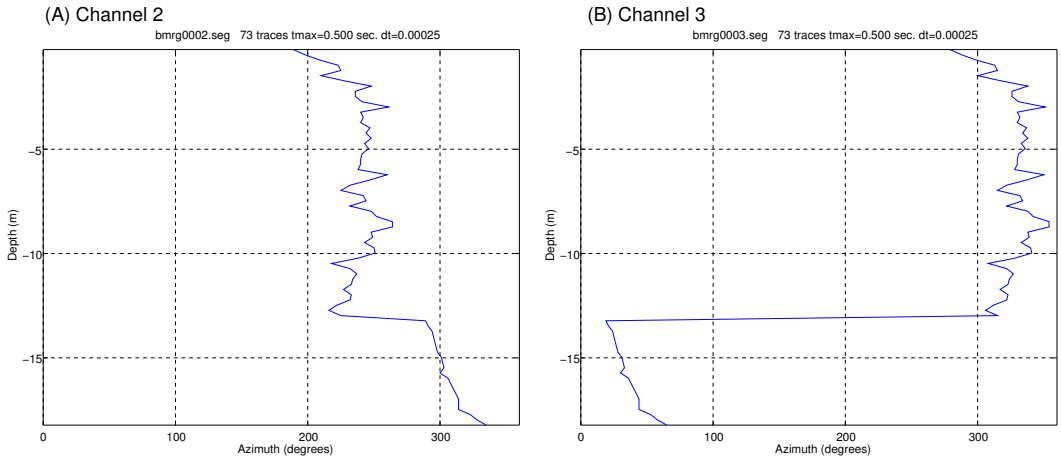


Figure 14: Plot of channel 2 and channel 3 geophone azimuth headers. The apparent discontinuity at about 12.5 m depth is exaggerated by channel 3 passing through North, 0 deg. = 360 deg.

genbrot program is run interactively to generate a script which can then be executed. The script automates running program *brot* on all the shot gathers. Program *genbrot* is set up to process data according to the channel definitions of section 6.7.8 above. Thus, only the data on traces 2 and 3 are rotated.

The program *brot* is quite flexible in how to rotate the data. Program *genbrot*, however, makes a single choice and would have to be modified if that did not suit a user's needs. The default choice imposed by *genbrot* is to align the data so that the T-component down-hole becomes aligned with the T-component of the reference phone. This particular rotation was based on the decision to align the T-component of the reference phone with the long axis of the source (see Figure 11).

The result of the above decisions is to align trace 3 with the long axis of the source, to the extent that the radiation from the source is also aligned with the long axis of the source. While one might think that the source shown in Figure 10 would radiate SH-motion polarized parallel to the long axis of the source (aligned with the hammer blows), it appears that variations in soil stiffness under the source can cause a couple, twisting the polarization axis of the radiation slightly out of alignment with the source axis (see Michaels [12]). *Thus, a more proper statement might be that the horizontal data are aligned with the major axis of the source radiation polarization ellipse, rather than with the long axis of the source itself.*

As a final caution, if a user has not acquired the data with the geometry as shown in Figure 11, the blind use of the program *genbrot* may produce unwanted and unexpected results. There are many ways to set up the source and reference phone about the bore hole (both in location and component orientation). My choice was designed to permit SH-waves to be recorded from depth to the surface, with as little interference as possible from other wave fields.

The dialog for *genbrot* would be as follows:

```
 |----------------------------------------|
 | Copyright (C) 2017 P. Michaels         |
 |       All rights reserved              |
 | See GNU General Public License         |
 |----------------------------------------|
 gbrt:  TIME: 16:12:13 DATE: 27/Apr/2017

Enter alpha prefix (char) of *.seg data to be rotated

EXAMPLE: if enter 1, then files 1001.seg to 1010.seg
                     would be processed if sequence
                     numbers 1 and 10 entered next
```

L

```
...L
Enter first file number to process
1
Enter last file number to process
146
Output in file===>gobrot
```

The next step would be to make the bash script, *gobrot*, executable:

```
chmod +x gobrot
```

After running *gobrot*, you will have files named `brotL001.seg` through `brotL0146.seg`. These data are rotated to the standard orientation described above. Not only are the data rotated, but the headers have been changed for traces 2 and 3 to reflect their new azimuths.

6.7.10.1 Post *brot* processing steps With so many files, both rotated and unrotated in the same directory, it can get a bit messy. The following steps are recommended.

1. Make a new directory for the rotated data. For example,
   ```
   mkdir brot
   ```
2. Move the rotated data to the new directory
   ```
   mv brot*.seg brot
   ```
3. Clean up the current directory:
   a). remove the list files
   ```
   rm -f brot*.lst
   ```
   b). gzip the unrotated data to save space
   ```
   gzip *.seg
   ```
4. Change to the "brot" directory, since the next steps are executed on the rotated data.

6.7.10.2 Verify Rotation with *hodoplot.m* There are basically two ways to confirm proper determination of the tool orientation by PCA, and the application of that determination by rotating the horizontal components to a standard orientation. One can select a geophone station depth and plot a hodogram of the particle motion before rotation and after. The other evidence comes later when we plot all the horizontal data and look for any abrupt changes in the direct arrival waveform which would suggest a possible rotation error.

The former is done with the Octave program, *hodoplot.m* (see 6.7.4). Start an Octave session and execute from within the Octave text window the following command (segyinfo.m, bsegin.m must also be in the path)

```
hodoplot;
```

The Octave session must be started in the same directory as the data to be plotted, and of course, a copy or link to *hodoplot.m* would also be in that directory. A number of obvious dialog GUI's come up, most of which can be defaulted. The two that are important are the selection of which trace to plot on which axis, and the time window selection. The standard BSU orientation is to have the down-hole components on channels 1,2, and 3 (V-comp, R-comp, T-comp). In that case, one select channel 2 for the x-axis, and channel 3 for the y-axis. The default time window GUI shows 0 to tmax seconds. One benefits from looking only at the direct arrival, and can step through the motion by progressively increasing the maximum time (the program loops until you click on cancel). For this example, the max times to plot were chosen as follows:

- 0 to .065 sec
- .065 to .070 sec
- .070 to .075 sec
- .075 to .078 sec

Figure 15 shows the result of plotting the data, first as recorded (A),and then a second time with the rotated data set (B).

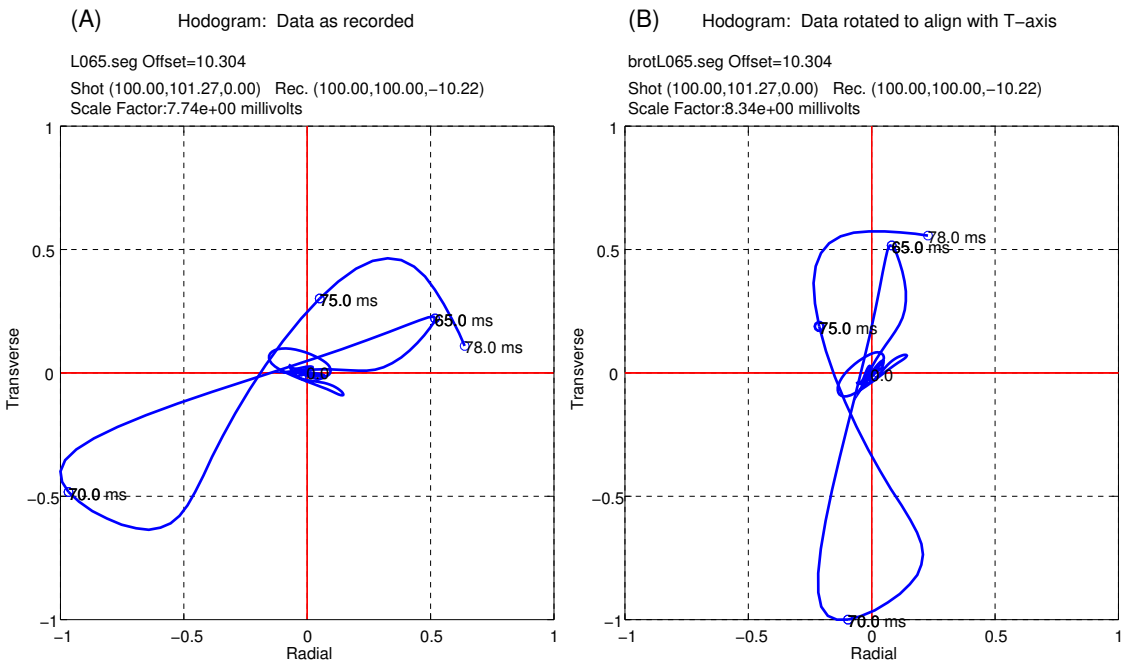


Figure 15: Plots produced by *hodoplot.m* confirms that data were rotated as desired

6.7.10.3 Using *hodo2plot.m* to plot hodograms In the above example, 6.7.10.2, the channels to be plotted were in the same file. An alternative Octave program, *hodo2plot.m*, is shipped with BSU to permit plotting hodograms when the channels to be plotted are in different files. In section 6.7.9 above, the program *bmrg* was used to collect all the channel 2 signals from all the different geophone depths. These were moved into a file, `bmrg0002.seg`. The same was done for channel 3 signals, and the result moved to a file named `bmrg0003.seg`. One could use program *bdump* to get a partial header listing and identify the trace whose geophone depth corresponded to a depth of interest. Executing *hodo2plot.m* from within the Octave text window results in a number of queries to the user. One specifies the two files and the trace number to plot as a hodogram (this would be the same for both files and is specified only once). The other queries are the same as in section 6.7.10.2, except there is one more query asking for how to label the two axes of the plot. This program, like *hodoplot.m* is useful for all multi-component data, not just down-hole data. They permit one to examine particle motion from any multi-component data, be they down-hole or surface.

6.7.11 Sorting and Merging to Common Receiver Component Gathers

What we would like to look at is not a multiplexed display of different components, but rather a gather of all one type of component. For P-waves, our first choice would be to gather all the down-hole vertical component data into a single file (trace #1 from all the files). For the SH-waves, we will want to gather all the down-hole T-component data (trace #3 from all the files). Included in the BSU distribution is a bash script, *Merge*, which produces **some** of these common component gathers. An alternative script is *Merge-all* (Appendix B) which generates all combinations of sum or difference on all components. In addition to gathering a component, the *Merge* script also computes wave field enhancements and wave shaping to remove source variations monitored by the reference phone. Begin by copying the *Merge* script into your "brot" directory which contains the *brot*.seg* files. NOTE: For vertical component sources (only one polarization) see script *Merge2* (in the scripts directory and Appendix C). *Merge2* is designed for 7 component recording (load cell on hammer is 7th signal). This script can also be modified for vertical impact sources without a load cell.

6.7.12 Edit *Merge* Script for the Specific Down-hole Survey

Since several survey details are likely to vary from one bore hole to the next, you must edit a few lines of the file, *Merge*. These lines are near the top of the file and are reproduced below. The boxed values will need to be changed.

```
#set -x
#...define parameters  YOU MUST DEFINE THESE FOR EACH NEW SURVEY
#                      SINCE IT IS LIKELY THAT THE NUMBER OF RECORDS
#                      WILL VARY WITH EACH SURVEY !!!!
odmin=01
odmax=163
evmin=02
evmax=164
PRFX=w
#         eklmax=(odmax-1)/2
eklmax=‘bc <<END
($odmax-1)/2
END‘
echo $eklmax
#         oklmax=eklmax+1
oklmax=‘bc <<END
 ($eklmax+1)
END‘
echo $oklmax
#shaping filter parameters (bshp)
tmin=0.
tmax=0.1
npf=360
stab=.0001
#polarity file definitions
az90=2
az270=1
```

The record numbers in the GeoLogan97 survey run from 1 through 146. The odd numbered records are 1, 3, 5, . . .145. Thus, *odmin=01* and *odmax=145*. You will have to edit *Merge* to change *odmax* from 163 to the current value of 145. Similarly, the even numbered records run from 2, 3, 4, . . . 146. You will have to change *evmax* to 146. In the GeoLogan97 survey, the odd numbered records had a source azimuth of 90^o and the even numbered records a source azimuth of 270^o. Thus, the variables *az90* and *az270* will have to be changed to match the current survey. Set *az90=1* and *az270=2*. These two values affect the order of subtraction between the two polarities of the source, and hence the final sign convention of the enhanced data. Finally, the prefix for the GeoLogan97 data is “L”, not “w”. Make the change so that *PRFX=L*. Depending on your views, you may also wish to modify the shaping filter parameters (see program *bshp*). The shaping filter corrects for triggering variations and source blow amplitude variations which might be mistaken for down-hole geologic effects, were it not for our monitoring of the source with a stationary reference phone.

6.7.13 Description of the *Merge* Procedure.

When you run *Merge*, the following sequence is followed:

1. Determine the maximum absolute value on the reference phone vertical component (for the last even record). This is done by first running *bscl* on the last even record (brotL146.seg in the example) and then scanning the resulting listing for the amplitude. The *bscl* command is

   ```
   bscl brot$PRFX $evmax.seg 4 1 3 1>/dev/null
   ```

and the capture of the amplitude from the listing is done by the command

```
AMP=` gawk '/Peak Absolute Value/ {print $4}' bsclbrot.lst`
```

2. A list of reduced file names is created with the file command. Thus, FILE=L001 L002 L003 . . L146 when the following command is executed:

```
FILE=` find brot*seg | sed s/\.seg/"/g |sed s/brot/"/g`
```

3. The list in FILE is then directs a *do* loop which scales each brotL*.seg file first to unity maximum absolute amplitude on the reference vertical component, followed by a rescaling that makes all the files in the list end up having the same maximum absolute value on their respective vertical reference phone signals. This single new value is of course the AMP value captured in step (1) above.

4. The newly formed scaled versions all have been scaled to the same peak signal on their vertical reference, and this is dominated by the Near Field and Rayleigh waves. The goal is to null out the Rayleigh wave, and focus on the body waves down-hole. At this point, a sequence of *bmrg* runs are made to collect common receiver gathers. The files created are:
   a) *rfv1.seg* and *rfv2.seg* [reference, Vertical for shot azimuths 270 and 90 degrees respectively]
   b) *rft1.seg* and *rft2.seg* [reference, T-comp. for shot azimuths 270 and 90 degrees respectively]
   c) *swt1.seg* and *swt2.seg* [down-hole, T-comp. for shot azimuths 270 and 90 degrees respectively]
   d) *swv1.seg* and *swv2.seg* [down-hole, Vertical for shot azimuths 270 and 90 degrees respectively]

5. Enhancements without shaping filters are computed by subtracting the az270 files from the az90 files (SH-wave enhancement), or summing the az270 and az90 files (P-wave enhancement). The resulting output files are:
   a) *twav.seg* [SH-wave enhanced, viewed on the T-component down-hole]
   b) *pwav.seg* [P-wave enhanced, viewed on the Vertical component down-hole]

6. The shaping filter versions of enhanced P- and SH-waves is done by first running *bshp* on the reference phone traces, matching each reference to a target waveform, and then applying the filters on a second pass to the down-hole data. Thus, filter design is on the reference phone data, application is to both the reference phone (for QC) and to the down-hole (removes source fluctuations from down-hole data). The resulting file comparable to step (5) are:
   a) *twave.seg* [SH-wave enhanced, viewed on the T-component down-hole]
   b) *pwave.seg* [P-wave enhanced, viewed on the Vertical component down-hole]
   Note: The only difference in the names is the extra letter "e".

In summary, the files *twav.seg* and *pwav.seg* are enhanced SH- and P-waves, without any shaping filters. The files *twave.seg* and *pwave.seg* have the additional Wiener Least Squares shaping based on observations of the reference phone.

6.7.14 Plotting the Results from *Merge*

Initial plots are produced by a call to *bplt* in *Merge*. This program can output a variety of formats, including the `*.jpg` image format in the current script. The `*.jpg` files can be viewed using a program like *display* which comes with the ImageMagick package on most Linux distributions. The *Merge* script uses the jpg option for a quick QC. The program *bplt* is new with this release of BSU, and has many more options, to be discussed later. The scripts, *xplot* and *psplot* give some additional examples.

Alternatively, one can also use Seismic Unix (SU) to plot the results. The BSU distribution includes some scripts, *xPlot-su* and *psPlot-su*, which call SU programs for plotting in an X-window or as a Postscript file. Figure 16 illustrates how plots made with *psPlot* can be combined in *xfig* with the results of a direct push survey by ConeTec. The ConeTec survey was done in preparation for the field day event.

The *twave.seg* file has been trace equalized by the program *bequ*. If each trace were not individually scaled, the large variation in amplitude would result in the deep data disappearing below the visible threshold. The SH-wave

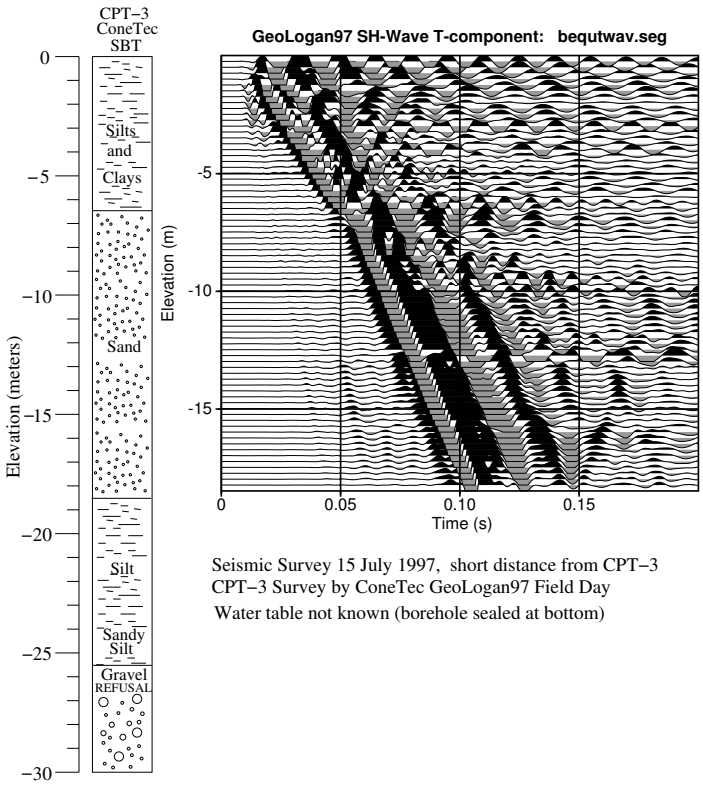


Figure 16: Difference of Source Polarizations, T-Component (*bequ* applied to twav.seg)

arrives just before the 0.1 second timing line at the bottom of the survey. A faint, unidentified wave arrives at about .05 seconds (between what may be the P- and SH-waves).

Figure 17 shows the *pwav.seg* file, also processed by *bequ*. In this case, the scaling was based on the data from 0 to .025 seconds. What may be the P-wave arrives at about .02 seconds at the bottom of the survey. The file named *pwav.seg* **is in fact the sum of the two source polarizations, vertical component**. This process will also enhance **Rayleigh** and **Near Field waves**, so one should be cautious in identifying candidate **P-waves**. **The shallower the depth, the more uncertain the wave field identification due to a superposition of different motions.**

The plotting script which generated these figures, *psPlot*, makes use of the Seismic Unix program, *segyclean,* to prepare the headers before piping it to *supswigp* which does the actual plotting.

6.8 Down-hole Seismic Analysis

6.8.1 Picking First Arrivals

BSU is distributed with a Octave procedure, *segpic.m*, which can be used to pick first arrivals. Copy the procedure into your working directory that has the files you wish to pick. Start Octave, and then run the procedure from within the Octave text window

```
segpic;
```

The procedure will prompt you for a file name and a maximum time to display. For example, you might answer **twave.seg** for the file, and **.20** for the maximum time to pick the first arrival SH-waves. Then, a graphic window will be displayed showing the first trace. Position your mouse at the point where you feel the first arrival is, and click with your left mouse button. Only the lateral position is read from the mouse click, so you don't have to be concerned about the vertical placement of the mouse, other than it be somewhere inside the plot window. The procedure moves on to the next trace after the click. This is a rather simple routine, and you can't go backwards. After all the traces have been picked, the procedure writes the picks to a file, (trace #, pick time) pairs.

The picks can be written to the file trace headers by running *bpic* in your X-term window. For demonstration, the I have picked file *twave.seg*, and in the interest of better signal to noise, picked not the first motion, but the zero

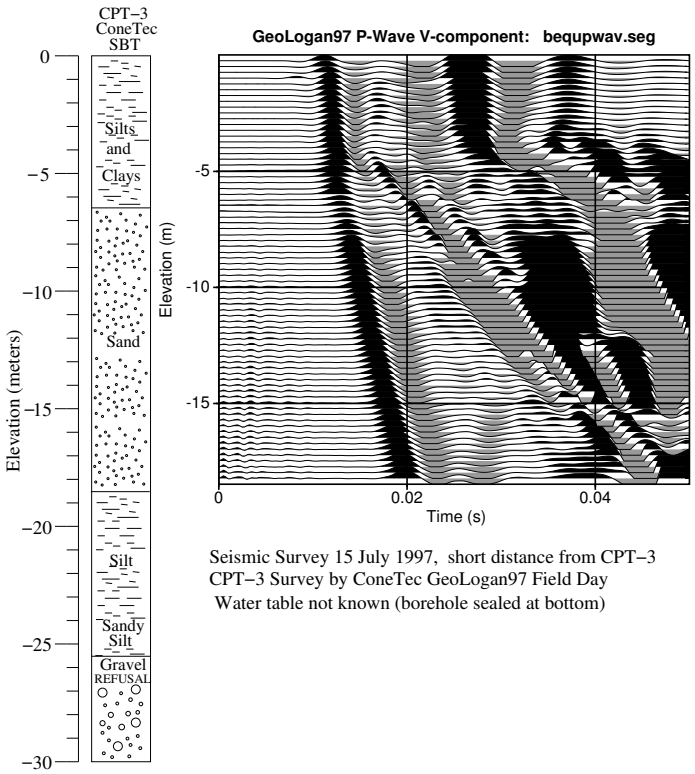


Figure 17: Sum of Source Polarizations, V-Component (*bequ* applied to pwav.seg)

crossing from the first peak to trough of the first arrival SH-wave. This occurs about .005 seconds after the first motion. The picks are inserted into headers with the command,

```
bpic twave.seg 1 twave.pic -.005
```

The last argument, -.005, shifts the value in the *twave.pic* file to a point earlier in time, where the first motion is located. The output file is called, *bpictwav.seg*. You can then move this to replace the original file, *twave.seg,* thus completing the operation. Now *twave.seg* will have the picks in the headers.

6.8.1.1 Quality control of picks The easy way to QC picks is to align the data on a constant time line using the picks in the headers for static shifts. The program, *bshf*, will do this. For example, type the command

```
bshf twave.seg 0 1 .02
```

This will align the data on the .02 second timing line by applying static shifts equal to the negative of the pick value. Figure 18 shows this process (picking, insertion to headers, and alignment in time for QC) at the final alignment step. If a pick were bad, it would be evident by a misalignment of the corresponding trace, relative to its neighbors. In Figure 18, all the picks look good.

It is evident that the propagating wavelet is changing shape (first motion peak stretches in time with increasing distance from the source). Such wavelet stretch may be interpreted as inelastic wave propagation, perhaps consistent with a viscoelastic, Kelvin-Voigt, constitutive model. Despite the expected velocity dispersion, one can determine a group velocity for the frequencies which dominate the amplitude spectrum of the signal.

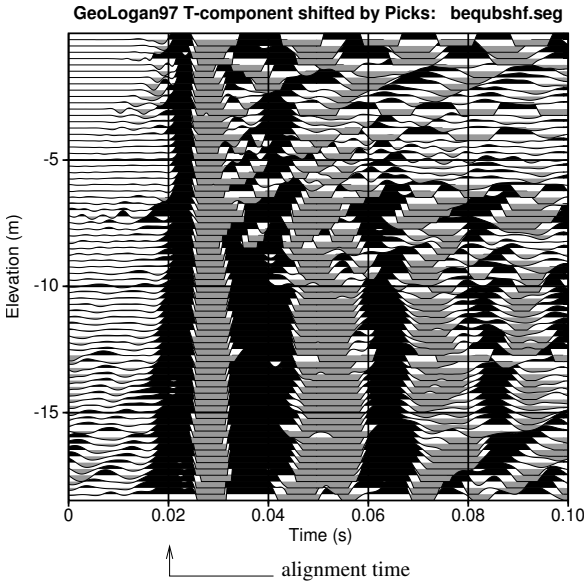


Figure 18: Alignment T-component data by first break picks for QC

6.8.2 Vertical Time and Observed Travel Time Inversion (*vfitw.m, vplot.m, bvsp*)

The BSU distribution includes a procedure, *vfitw.m*, which reads the picks in the trace headers of a BSEGY data set. The procedure projects the picks to the vertical, and prompts the user to select the top and bottom of an interval for a least squares linear fit. The fit curve is drawn and the user clicks at a point where the interval velocity solution will be printed (on the figure). This velocity is an estimate of the vertical group propagation velocity. If the results of *vfitw.m* are written to disk, a second program, *vplot.m*, may be used to plot the solutions with more user control over scales. Exporting the plot to *xfig* format will allow one to draft on the final figure.

Figure (19A) is the *vplot.m* generated plot of a two interval analysis of the T-component data. A significant increase in group velocity occurs at the transition from silt-clay soil to the sand (soil types determined by Soil Behavior Type (SBT) of the ConeTec survey made available at GeoLogan97). An alternative to working with vertical times is program *bvsp* which does an inversion based on 2 layers over a half-space. *Bvsp* works with the observed times, and solves for refracted and direct raypaths in a non-linear ray tracing scheme (horizontal boundaries). The *bvsp* result is shown in Figure (19B) (after 40 iterations). Program *bvsp* fits the observed arrival times by ray tracing, and is hardwired to do a 3 layer fit.

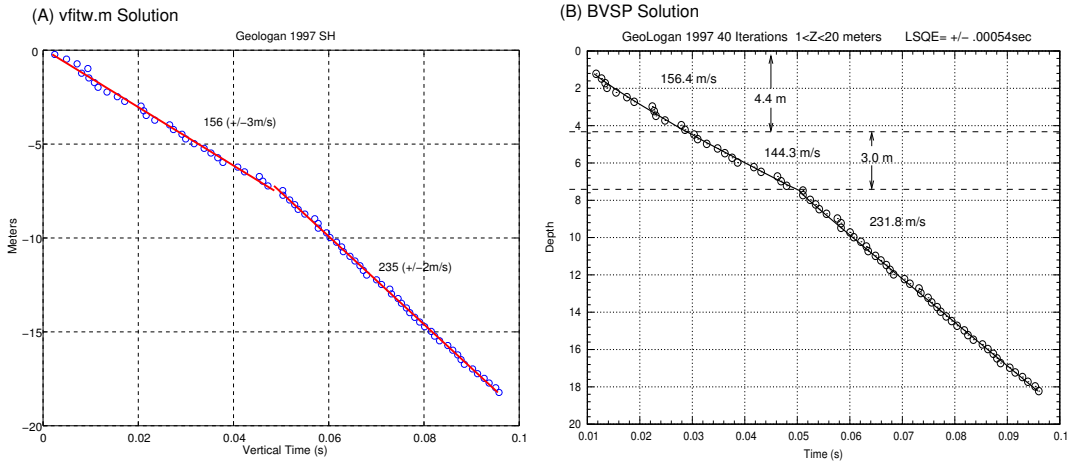


Figure 19: T-component Data Travel Time Inversions (a) Vertical Time (b) Observed Time

It should be remembered that the ConeTec and seismic survey were done at different times of year (ConeTec in November 1996, seismic in July 1997). Further, the two surveys were in the same general area, but not at exactly the same location. The lateral separation of the two could easily be 50 meters or more. For that reason, exact depth correlations are not possible.

6.8.3 Determination of Stiffness and Damping

The programs *bvas* and *bamp* are used to measure the velocity dispersion and amplitude decay as a function of frequency. These measurements are inverted for material stiffness and damping by Octave procedure *cainv3.m* according to a method described by Michaels [10]. The constitutive model is often referred to as the Kelvin-Voigt solid.

6.8.3.1 Governing Differential Equation The PDE relevant to 1-D wave propagation is given by

$$\frac{\partial^2 u}{\partial t^2} = C_1 \frac{\partial^2 u}{\partial x^2} + C_2 \frac{\partial^3 u}{\partial t \partial x^2} \tag{1}$$

where "u" is particle displacement, "t" is time, "x" is the coordinate in the direction of wave propagation, C_1 is the stiffness $\left(\frac{m^2}{s^2}\right)$, and C_2 is the damping $\left(\frac{m^2}{s}\right)$. Equation (1) reduces to the elastic wave equation when the damping value, $C_2 = 0$. In that case, the phase velocity is constant for all frequencies, and the wave does not experience any decay (for a 1-D plane wave). In the elastic case, the phase velocity will be $\sqrt{C_1}$.

In the more general case, $C_2 \neq 0$, and there will be both velocity dispersion and exponential, inelastic amplitude decay. A solution of equation (1) is

$$u(x,t) = \exp(-\alpha x) \cdot \cos(\beta x - \omega t),$$

where the wavenumber is complex and given by $\beta + i\alpha$.

Michaels [10] shows that the inelastic decay of a plane wave will be given by

$$\alpha = \frac{4\sqrt{D}\omega^2 C_2}{(2\omega C_2)^2 + D^2}$$

where ω is angular frequency (rad/s) and the quantity, D, is given by

$$D = 2\left(C_1 + \sqrt{C_1^2 + \omega^2 C_2^2}\right). \tag{2}$$

The phase velocity, c, varies with frequency according to the following relationship

$$c = \frac{2\omega^2 C_2}{D\alpha}. \tag{3}$$

The values for C_1 and C_2 can be expressed in terms of the following :

$$C_1 = \frac{\left(\beta^2 - \alpha^2\right)\omega^2}{\left(\beta^2 + \alpha^2\right)^2}, \tag{4}$$

and

$$C_2 = \frac{2\alpha\beta\omega}{\left(\beta^2 + \alpha^2\right)^2}. \tag{5}$$

Determination of C_1 and C_2 is by nonlinear joint inversion of the phase velocity, c, and inelastic decay, α , over a range of frequencies. The inversion is currently performed in the Octave procedure, *cainv3.m*. Initial estimates of stiffness and damping are obtained at the frequency corresponding to the largest α measured by *bamp*. First, C_1 is found by evaluation of equation (4). In that computation, $\beta = \frac{\omega}{c}$. Then, C_2 is estimated from equation (5).

6.8.3.2 Measurement of Velocity Dispersion (*bvas*) The program, *bvas*, measures phase velocity as a function of frequency for a given subsurface interval. The user specifies an interval in the subsurface by the top and bottom elevation. The data are filtered by very narrow band-pass filters. At each frequency, the data are shifted by a sequence of trial velocities (from Vmin to Vmax). For each trial alignment, a semblance value (Sheriff [20]) is calculated which measures the degree to which the data were aligned. The larger the semblance, the better the alignment. A Golden Section search solves for the best alignment velocity in the bracketed interval. The solution is written to a file, *bvas.his*, which is later read by the Octave procedure, *cainv3.m*.

Postscript plots *bvasqc.ps* generated by *bvas*. This is a multi-page file, one page for each frequency, which shows the quality of the trace alignment for the velocity solution. All the traces in the analysis interval are shifted into alignment at the solution velocity. A reference trace is taken as the first trace in the interval of analysis. A relative time shift is computed from the angle between each trace and the reference trace. That is, if the reference trace is U, and another trace is S, then the angle between S and the reference trace, U, is given by

$$\theta = \arccos\left(\frac{\sum(U_i S_i)}{\sqrt{\sum U_i^2}\sqrt{\sum S_i^2}}\right), \tag{6}$$

and the relative time shift at that frequency is given by

$$t_s = \frac{\theta}{\omega}. \tag{7}$$

This relative time shift is plotted about a horizontal line, the mean time shift for all the traces in the interval. The error bar for the velocity analysis is derived from the scatter about this line. A good quality solution is evident when the mean scatter is close to zero, and the points are tightly clustered about the line. A large mean, and large scatter indicate a poor quality solution. One should also examine the nature of the scatter. If it is random, then the interval is probably well chosen. However, if there appears to be two or more linear trends in the points, then one may wish to re-evaluate the choice of interval, as it may include more than one soil type. Figure 20 shows a typical QC plot with t_s plotted against geophone elevation (analysis interval was -13 to -8 meters elevation).

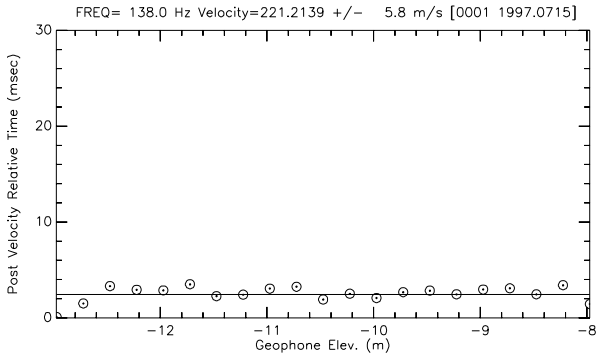


Figure 20: Velocity analysis QC plot from file *bvasqc.ps*

Postscript plot *bvas.ps* generated by *bvas*. A summary, single page, plot of velocity and semblance as a function of frequency is given by *bvas.ps*. In general, there will be a sensible agreement between high semblance values, small error bars, and good quality for that frequency in the corresponding *bvasqc.ps* plot. Zones with poor signal will have low semblance values, as will also be the case if the interval of analysis is too large for a single velocity solution (ie. an analysis zone with more than one soil velocity). Figure 21 shows a typical *bvas.ps* plot.

Computing error bars for velocity. The computation of error bars is based on the scatter of the points plotted in the *bvasqc.ps* file. Consider an alternative X-Y plot of points about a linear trend, where the "x" values

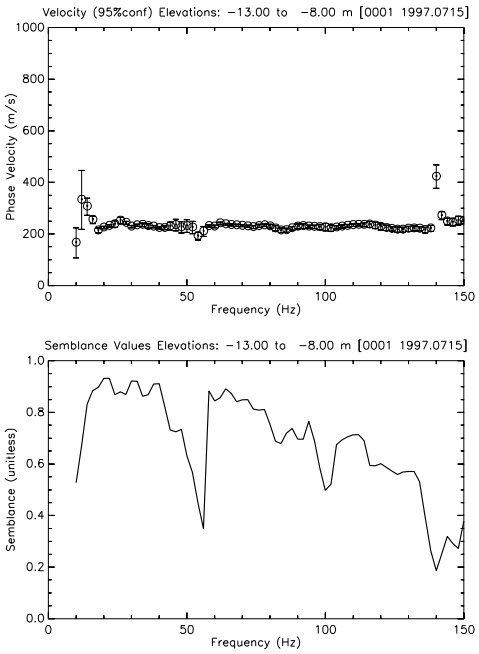


Figure 21: Summary plot showing velocity and semblance.

are the depths, z_i , of each geophone station, and the "y" values are pseudo arrival times computed from the *bvas* solution velocity and relative time shifts, t_s , about the mean shift, $\bar{t}$,

$$y_i = \frac{z_i}{V} + (t_s - \bar{t}) \,. \tag{8}$$

The slope of a least squares linear fit to these pseudo times, y_i , would correspond to the slowness, 1/V. The problem is to then estimate the variance in the reciprocal of the slope of linear solution. That is, if $y = mx + b$, then we solve for the variance, $\sigma^2_{1/m}$, assuming a least squares solution to the problem. For N pairs of (x,y), we can write the velocity as the reciprocal of the least squares solution for the slope of the line as

$$V = \frac{1}{m} = \frac{\left[(\sum x_i)^2 - N \sum \left(x_i^2 \right) \right]}{\left[(\sum x_i) (\sum y_i) - N \sum (x_i y_i) \right]}. \tag{9}$$

The variance of the velocity, $\sigma^2_V = \sigma^2_{\frac{1}{m}}$, is given by

$$\sigma^2_V = \sigma^2_{\frac{1}{m}} = \sum \left(\frac{\partial V}{\partial y_i} \right)^2 \sigma^2_{y_i}, \tag{10}$$

where σ^2_y is assumed a constant for all y_i and is estimated by the scatter around the mean t_s . In other words, σ^2_y is given by

$$\sigma^2_y = \sigma^2_{t_s} = \frac{\sum (t_s - \bar{t})^2}{N-1}. \tag{11}$$

After some algebra, we find that,

$$\sigma^2_V = \sigma^2_{\frac{1}{m}} = \sigma^2_{t_s} \cdot \frac{N \left[N \sum \left(x_i^2 \right) - (\sum x_i)^2 \right]^3}{\left[(\sum x_i) (\sum y_i) - N \sum (x_i y_i) \right]^4}. \tag{12}$$

This permits us to treat the semblance determined velocity, V, as though it were the result of a least squares fit to picked arrival times, and thus obtain an estimate of the uncertainty in the phase velocity determination. The

velocity error bars are computed as the square root of σ_V^2 (units of m/s). These error bars may be scaled by 1.96 to obtain an estimate of the 95% confidence interval (assuming normally distributed errors). The unscaled values are output to the file *bvas.his*, and then later used in the Octave joint inversion, *cainv3.m*, to obtain confidence limits on both stiffness, C_1 and damping, C_2 .

6.8.3.3 Measurement of Inelastic Amplitude Decay (*bamp*) Because there will be some beam divergence from any real, finite source, the program, *bamp*, makes a correction for beam divergence before measuring inelastic decay. Currently, the only option is for a spherical divergence correction. Under this model, the amplitude, A, of a spherically divergent wave at distance, r, from the source is given by

$$A = \left(\frac{A_o r_o}{r}\right) \cdot \exp\left(-\alpha\left(r - r_o\right)\right), \tag{13}$$

where A_o is the amplitude at a reference offset r_o . Here, amplitude is the particle velocity as measured by a moving coil velocity phone (dynamic stress/impedance). The decay at any frequency may be expressed in terms of α (1/m, also sometimes referred to as nepers/m). Alternatively, decay may be expressed in decibels as

$$dB = 20\log_{10}\left(\frac{Ar}{A_o r_o}\right) = -20\left(\log_{10} e\right)\alpha(r - r_o). \tag{14}$$

Program, *bamp*, writes the measured inelastic decay, α , at each frequency and writes the results to a file, *bamp.his*, which is later read by *cainv3.m*.

Postscript plots *bampqc.ps* generated by *bamp*. The program, *bamp*, measures amplitude decay as a function of frequency for a given subsurface interval. The user specifies an interval in the subsurface by the top and bottom elevation. The data are filtered by very narrow band-pass filters. A reference offset is computed from the distance between the source and shallowest geophone station (typically closest to the source) which still lies within the subsurface interval of interest. A peak amplitude and range is calculated for each trace in the interval, and then a decibel value is computed according to equation (14). A least squares linear solution of the form, $y = mx + b$ is performed, where "x" is the range beyond the reference, $(r - r_o)$, and "y" is the decibel value for that particular range. The slope, "m", is in units of dB/meter, and can also be converted to α in units of 1/m. Figure 22 shows a typical *bampqc.ps* plot page.

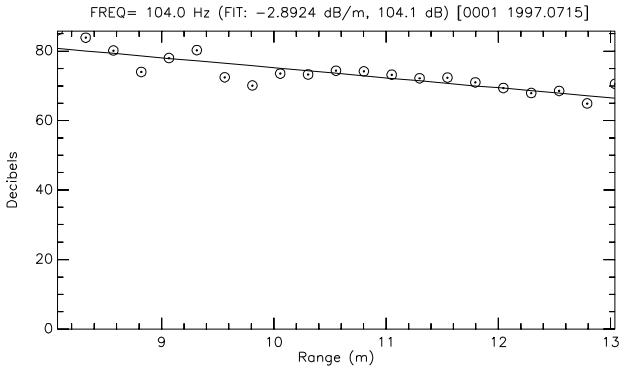


Figure 22: Amplitude decay analysis QC plot from file *bampqc.ps*

This plot serves the same role as those from bvasqc.ps discussed above. Random scatter about the trend are a measure of the uncertainty in the decay value, and the steepness of the slope is a visual representation of the decay. The slope should always dip down to the right (with increasing offset). To do otherwise would suggest an increase in amplitude with propagation, and be counter to the data model. When the slope is zero or dips upward, the inconsistency may be due to a combination of several factors. These include but are not limited to:

a). constructive interference from reflected or diffracted waves at certain frequencies

b). beam divergence which is not spherical

c). an interval which spans more than one soil type

d). poor coupling at certain frequencies, and possibly voids behind the casing
The output file, *bamp.his*, will have zeros inserted for decay values which are of the wrong sign. That is, according to the formulation shown in equation (13), α should never be negative.

Postscript plot *bamp.ps* generated from *bamp*. A summary plot of all the decay determinations as a function of frequency is provided in file *bamp.ps*. In general, there will be a sensible agreement between small error bars and good quality for that frequency in the corresponding *bampqc.ps* plot. Figure 23 shows an example of the *bamp.ps* plot.

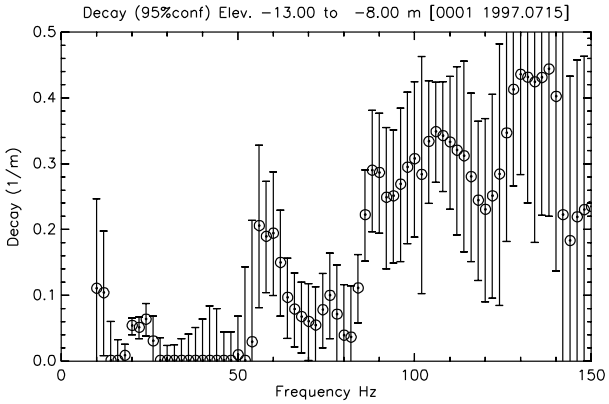


Figure 23: Summary plot showing decay as a function of frequency

Computing error bars for decay. The decay error bars are based on the scatter of the points about the least squares fit plotted in the *bampqc.ps* file. The variance in decay is computed from the deviations about the line. In general, for any least squares solution of the form $y = mx + b$, the slope, m, is given by,

$$m = \frac{(\sum x_i)(\sum y_i) - N\sum(x_i y_i)}{(\sum x_i)^2 - N\sum(x_i^2)}. \tag{15}$$

Here, N is the number of (x,y) pairs. We can estimate the variance in y from

$$\sigma_y^2 = \frac{\sum(y_i - (mx_i + b))^2}{N-1}. \tag{16}$$

The variance in the slope, m, follows from the equation used to compute m, equation(15), and the variance in the y values (which will be taken to be a constant given by equation(16)). For uncorrelated errors, the variance in the slope is given by

$$\sigma_m^2 = \sum\left(\frac{\partial m}{\partial y_i}\right)^2 \sigma_{y_i}^2, \tag{17}$$

which reduces to the following

$$\sigma_m^2 = \frac{N\sigma_y^2}{\left[N\sum(x_i^2) - (\sum x_i)^2\right]}. \tag{18}$$

The slope, m, for our problem is of course the decay factor (dB/m). Conversion to 1/m units follows by dividing by 8.68589 in the usual way.

6.8.3.4 Recording Aperture and the Selection of Filter Bandwidth for *bvas* and *bamp* As one might expect, there is a limit to how finely the spectrum may be investigated. This depends on the available aperture, which follows from the choice the user made in recording the field data. If there are N samples in each seismic trace, and if the sample interval is Δt , then the filter bandwidth should not be less than $\frac{1}{N\Delta t}$. Further, the frequency step size

(independent of the bandwidth specification) should be no smaller than the bandwidth. Both programs, *bvas* and *bamp*, should be run with the same spectral bandwidth. However, the program *cainv3.m* is able to handle different total numbers of amplitude and velocity samples in the frequency domain. Thus, it is OK to be missing either some decay or velocity samples. Further, the sampled frequencies do not have to be identical for both velocity and decay. What should be the same is the bandwidth of the filters.

It is important to record enough data (large enough aperture), if the velocity measurements are to be considered phase velocity, and not group velocity. With hammer sources on soil, I recommend recording no less than 0.5 seconds of data (translates to a bandwidth of 2 Hz).

6.8.3.5 Inversion for Stiffness and Damping (*cainv3.m*) Programs *bvas* and *bamp* store their results in *bvas.his* and *bamp.his*. The Octave procedure, *cainv3.m*, reads these files and performs the joint inversion for soil stiffness and damping. The procedure, *caplot3.m* is available to produce journal quality plots of the inversion solution. The Octave procedures should be copied to the same directory where **.his* files are located. Each depth interval will require a separate set of **.his* files, and a separate *cainv3.m* run.

Start Octave, and then from the Octave text window, type the command

```
cainv3
```

Using a GUI interface, *cainv3.m* will prompt you as follows:

DIALOG WITH *CAINV3.m*

1. **Input Files Hardwired**. The program expects the input files to have been generated by programs `bvas` and `bamp` (*bvas.his* and *bamp.his)*).

2. **Use mouse to pick min and max frequencies**. Click OK, then in the graphic display, move mouse to velocity panel and click with left mouse button on the lowest frequency desired. Then move the mouse to the highest frequency desired and click again.

3. **Select initial C1, C2 and number of iterations.** The default is computed from equations (4) and (5). This is usually a good choice. But you can enter different values if you wish. The maximum number of iterations is set to 10, and that is usually enough. See item (7) below.

4. **Choose Weighting**. Choose one of 3 options. The weighting reduces the influence of data points which have large uncertainties. One can weight by the reciprocal of the variance or the square root of the variance estimate (stdv). If you want all data points to have an equal influence on the solution, regardless of the associated measurement uncertainties, select the no weighting option.

5. **Set Block Weighing: Velocity to Decay**. This is a joint inversion which includes both velocity dispersion and decay measurements. A value near 1 will emphasize the velocity data, and largely ignore the decay information. A small value will emphasize the decay data, and discount the velocity information. Usually, one should refrain from extremes here (don't use 1 or 0), but stay in the range of 0.9 to 0.1 for useful results. This weighting is superimposed on a data type weighting (the least squares error has to be scaled to take into account the different data units). If unsure, use the default, 0.5, for generally equal weighting of velocity and decay.

6. **Continue to LSQE Plot**. The procedure halts to let you view the **last iteration results**. When ready to move on, click on the YES button.

7. **Continue to Chi Square Plot**. View the **weighted least squares error** (this is a scaled combination error of both data types). It should decrease with iteration number, and give you an evaluation of the number of iterations needed. When ready to move on, click on the YES button.

8. **Save Results to Disk**. **Chi-Square plots** for velocity and decay are shown. These are based on average estimates of the data uncertainties as expressed in the average error bars. It gives you an idea of the relative balance between the residual error and the data uncertainty. In general, there is no point in obtaining a solution with Chi-square less than unity. More iterations may help if Chi-square is above unity, and can be

further reduced. However, if one only focuses on one data type, the other will suffer. See Menke [7] for more. When ready to move on, click on the YES button. The program ends displaying the relaxation time for the solution, $T_r = \frac{C_2}{C_1}$

6.8.4 Plotting Inversion Results (*caplot3.m*)

While the graphics produced during a cainv3.m run are informative, you may wish to produce cleaner plots for a publication or document. Further, you may wish to draft on the figures using *xfig*. A number of files are written by *cainv3.m* which capture the solution and data from the inversion. These files are, *casol.dat, caoutc.dat, caouta.dat, calsqe.dat, cadeld.dat,* and *cachi2.dat.* These files are automatically read by *caplot3.m* for a user specified plotting of the results.

From the Octave text window, type the command

```
caplot3
```

Using a GUI interface, *caplot3.m* will prompt you as follows:

DIALOG WITH *CAPLOT3.m*

1. **Show Grid on Plots** Click on Yes or No for final plots.
2. **Edit Axes Limits** Either use defaults (and click OK), or use entry boxes to set new values.
3. **Select range of observations to plot.** During the inversion, you selected a subset of the frequencies that could be inverted. Do you want your final plot to show all the data, or just the range of frequencies you selected in the inversion?
4. **Continue on to Decay Plot?** Click YES to continue.
5. **Set horizontal and vertical axis limits for Decay Plot**. The defaults are for everything. However, you may want to focus on a portion of the data. The procedure will automatically write an *xfig* file, *aplot.fig*. Edit limits in the GUI and click OK.
6. **Continue to Chi-Square Plot?** Click YES to continue.
7. **Select axes limits.** Defaults are usually OK. Click OK when ready to continue.

6.8.4.1 Post *caplot3.m* processing.

At the conclusion of plotting, you may wish to export plots in XFIG format. You may want to edit the plot files in *xfig*. You may wish to add further annotations or edit fonts. You can also merge two plots together, and even rescale the combination. Once you have it looking the way you wish, export the plot to a Postscript or PDF file for plotting or inclusion in a document. An example with both velocity and decay merged together is shown in Figure 24.

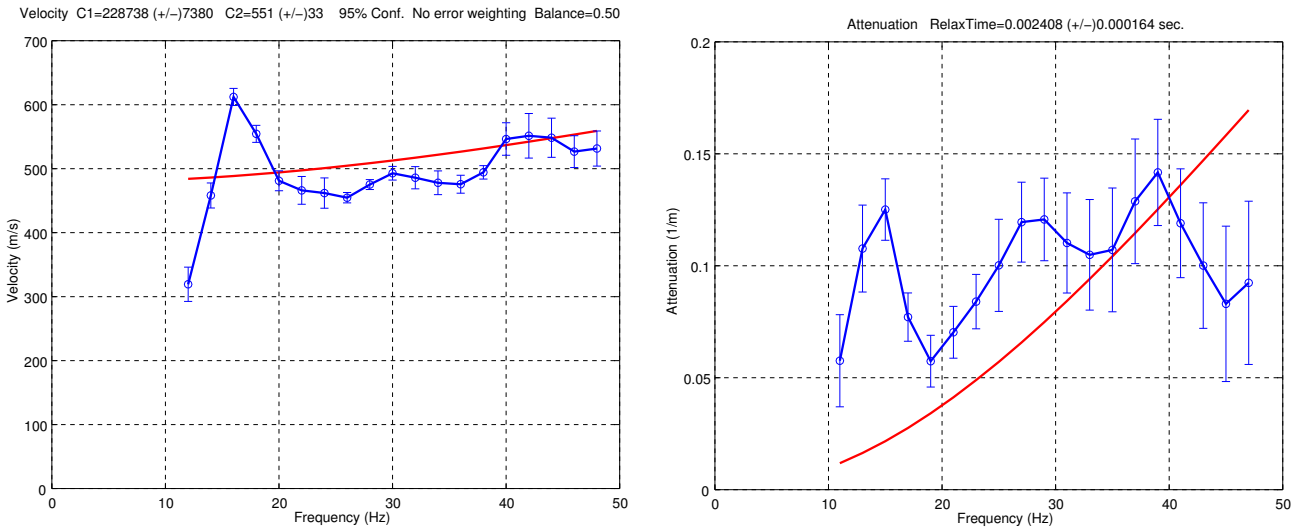


Figure 24: Merged figure showing both velocity and decay

6.8.4.2 Kelvin-Voigt Modeling with *cafwd3.m* Program *cafwd3.m* is a Octave program which permits one to compute the velocity dispersion and amplitude decay as a function of frequency. Given user supplied values for stiffness (C_1 in m^2/s^2) and damping (C_2 in m^2/s), the program computes frequency dependent phase velocity dispersion (m/s) and decay ($1/m$). A second plot re-expresses the amplitude decay as a quality factor, Q. This is computed by (Toksöz [22])

$$Q = \frac{\pi f}{\alpha V} \tag{19}$$

where α is the decay in units of $1/m$, f is frequency, and V is velocity at that frequency. There are two ways to use the program.

- Without data. The first GUI switches the code to do a forward computation only.
- With data. The first GUI switches the code to compare the forward calculations with data from a previous *cainv3.m* run. This requires that files *bvas.his* and *bamp.his* be in the same directory as the initiation of the Octave session.

To run the program, start a Octave session. Then, inside the interactive Octave text window, type:

```
cafwd3
```

With the first GUI, choose if data are to be included in the plot. If data are included, you will be prompted to use the mouse to select a frequency range to model. For the non data case, the frequency range is hardwired, and can be changed if needed by editing the program (look for fmin and fmax variables at the top). The program will do an initial case on its own. If there are data, it will look at the data and make a reasonable first guess. If there are no data, a default GUI will come up. One can then change the values for stiffness and damping and recalculate a new set of plots. Figure 25 shows examples of the two ways dispersion can be presented (ie. with or without data plots). One can keep revising the values and clicking OK as much as one wants. When done, click the "cancel" button. An example of the second Q-plot is shown in Figure 26. This corresponds to the run shown in Figure 25A. Note that the example values for C_1 and C_2 are different for the *no data* and *with data* cases.

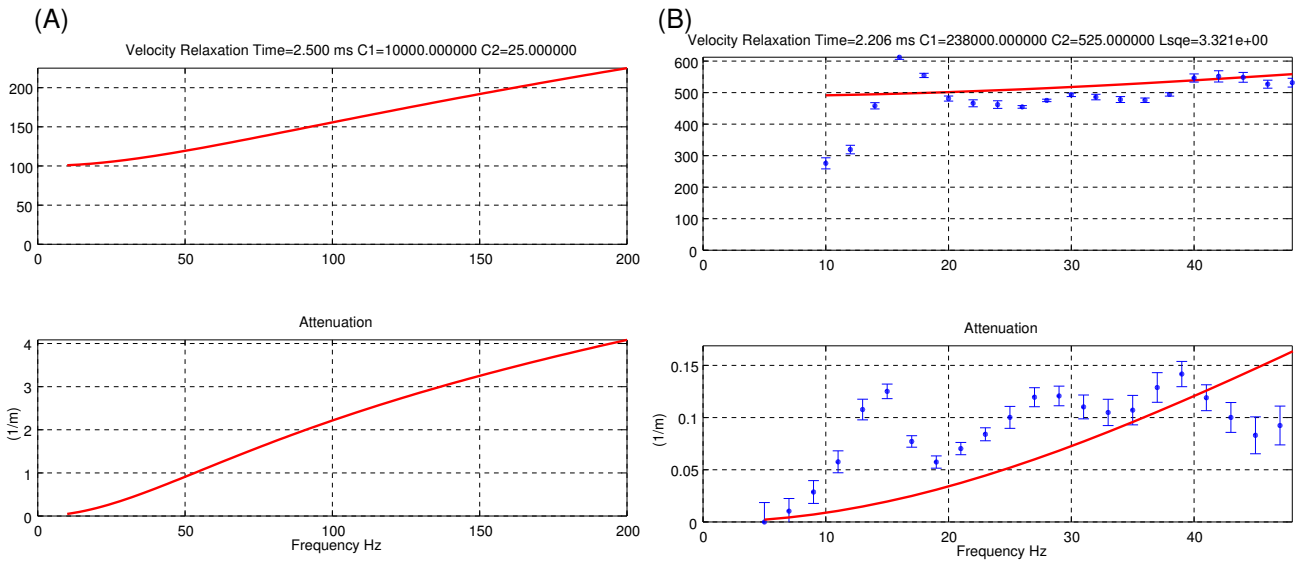


Figure 25: Sample of *cafwd3* calculations. (A) run without data (B) run with data for comparison

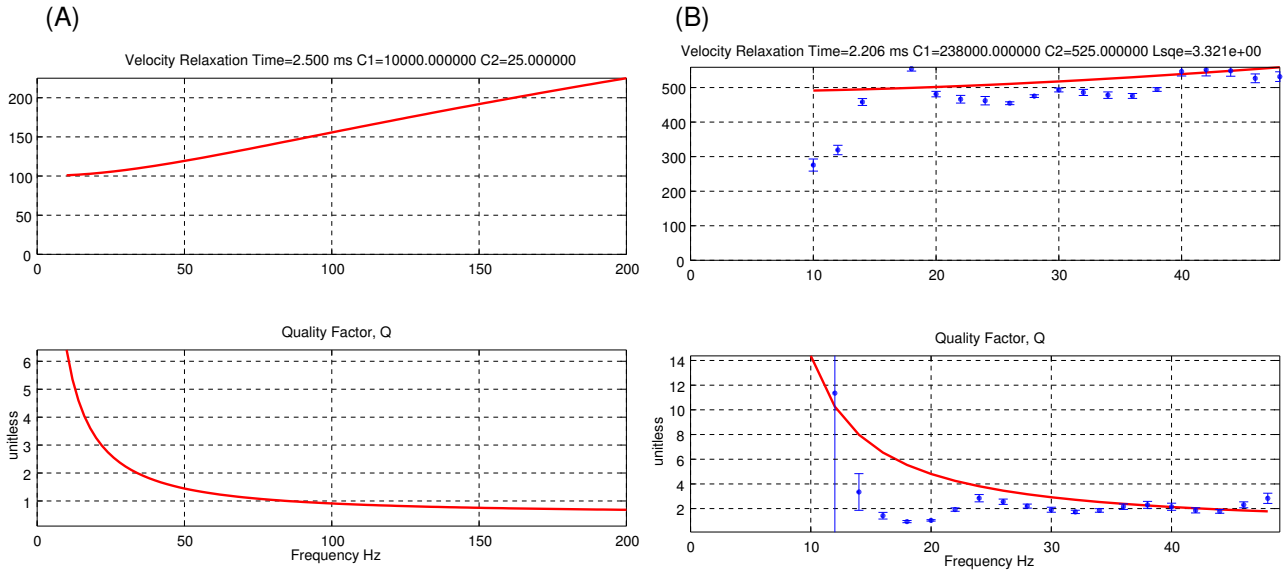


Figure 26: Sample of *cafwd3* calculations. Quality factor varies with frequency.

6.9 Seismic Refraction Processing

The BSU package includes some software for the refraction delay time method. This software is suitable for a simple problem like mapping the depth to bedrock with a soil overburden. Engineering applications include slope stability analysis, assessment for the need to blast or excavate in construction, and mapping the extent of a soil deposit (like a gravel barrow). There are no assumptions about the recording surface, or the layout of the geophones and shots. The ground topography can be irregular, and geophones do not always need to be deployed in a line. The seismic source can be buried, or on the surface of the ground.

This section will be illustrated with a data sample recorded along a highway where the issue was slope stability (Michaels [11]). The Idaho Transportation Department was considering adding a passing lane, and the amount of overburden in the up dip direction was of concern. The sample file may be downloaded from the BSU Database web page.
Go to https://cgiss.boisestate.edu/ pm/BSU/. Click on the check boxes for 1995, Idaho, and then on submit. It will list 3 download archives. ID-102.zip is the raw bison files and SEGY with some scripts.

6.9.1 Converting from SEGY to BSEGY Format

This first example is for a conventional reverse profile line shot along the road shoulder. It is used to determine the refractor velocity, overburden velocity, and delay times at locations needed later for the line which runs up the steep slope. Assuming you have downloaded the `ID-102` archive and unpacked it in a convenient location, you will note that it has the following directory structure:

```
ID-102
|--- bison
|     |--- basemaps
|--- seg
|     |---shoulderLine
|---  |   |--- directWave
|---  |   `--- headWave
|---  `---slopeLine
`--- sgy
```

Change to the `sgy` directory. These are in the data exchange format defined by the Society of Exploration Geophyicists, SEGY (Barry et al. [2]). The data format includes a reel header which goes back to the days of digital tape. It has been adapted for disk use here. The BSEGY used by BSU differs in that it does not retain the reel header, and there are some special uses for the optional headers (see section 6.1 for more). Convert these files to BSEGY with a command sequence as follows:

```
bcnv k004.sgy  0   1   1
mv bcnvk004.seg k004.seg
```

The move "mv" command renames `bcnvk004.seg` with `k004.seg`. Repeat this for files `k008.sgy` and `k009.sgy`. Then move the BSEGY data files to the "shoulderLine" directory,

```
mv k0*.seg ../seg/shoulderLine
```

The files are as follows:

- **k004.seg** is a hammer source recording with the blow near the middle of the line. It will be used with the other two files to establish an overburden velocity.
- **k008.seg** is an explosive source line, shot near geophone station 1.
- **k009.seg** is an explosive source line, shot near geophone station 48.

6.9.1.1 Creating a Base Map from BSEGY Headers

Since the shot and receiver locations are in the BSEGY headers, we can use program *bcad* to build a basemap by extracting this information and generating a CAD data exchange format file, **.dxf*. For example,

```
bcad k008.seg 1   1  2.0
```

creates file `bcadk008.dxf`. It is a simple "point" file with 2.0 meter labels. All three recording files use the same geophone stations in this example, so the only difference in running *bcad* on the 3 files will be the source location. If you download the open source program, *dxf2fig*, you can convert the **.dxf* files to *xfig* format files. These can then be edited in *xfig* to build a base map. I downloaded *dxf2fig* from http://homepage.usask.ca/ ijm451/fig/. (version 2.13). There are other versions at other websites, use a search engine if the location changes after this writing. I have included an edited version based on the two files `k008.seg` and `k009.seg`. the map is shown in Figure 27. Alternatively, I recommend Qcad to work with *.dxf files directly. Qcad is available from https://qcad.org/en/.

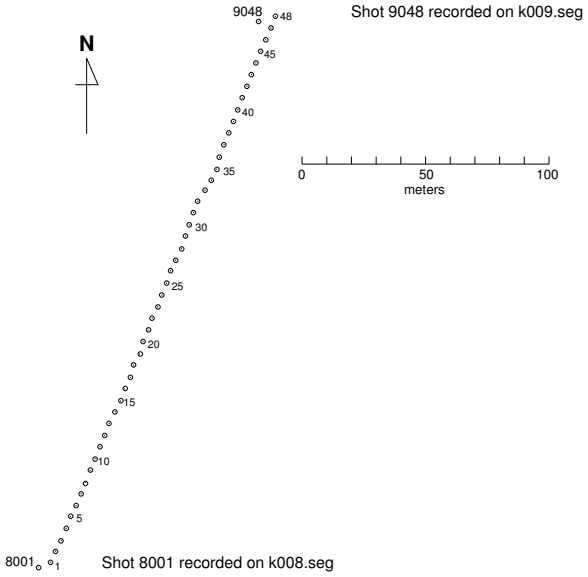


Figure 27: Base map for refraction survey along road shoulder.

6.9.2 Using *refplot.m* for first look

The data were picked for first arrivals, and these picks are already in the headers along with the source and receiver geometry. Other sections of this document describe how that is done (see section 6.8.1). We can start by using program *refplot2.m* to examine the picks and get an idea of velocities, offsets, and a preliminary sense of the data. The program offers two ways to view the data, by offset and by geophone station. Here, we will use the geophone station view. Least squares solutions still use actual distance headers, regardless of our choice here. Begin by starting a Octave session and type the following in the Octave text window:

```
refplot
```

You will enter a file name, `k008.seg`, for example. Next, choose "Stations" for the type of display. I use *xfig* for the scaling, and make sensible choices for axes limits. When the data appear in a plot, choose *DO AN INTERVAL* and then use the mouse to click on a starting and ending point for a linear (by offset) fit. One more click positions the computed velocity for the line segment solution. Since we chose *Stations* rather than *Offset*, a linear fit by offset may produce a somewhat undulating plot on a *Station'* display. The default is to use the "vp" label which is not in meters. Also, the line is not straight, and the ground is not flat. The least squares fit uses actual header distances to compute velocities and calculated times. Figure 28 shows what it should look like for a simple single fit for all the stations.

Clearly, we seem to be seeing the granitic bedrock with these high refractor velocities. If we look at the waveforms for the data (see Figure 29) we can form a view about the minimum offset needed for the granitic refraction to arrive before a direct arrival. I picked an offset of 30 meters for this cross-over distance.

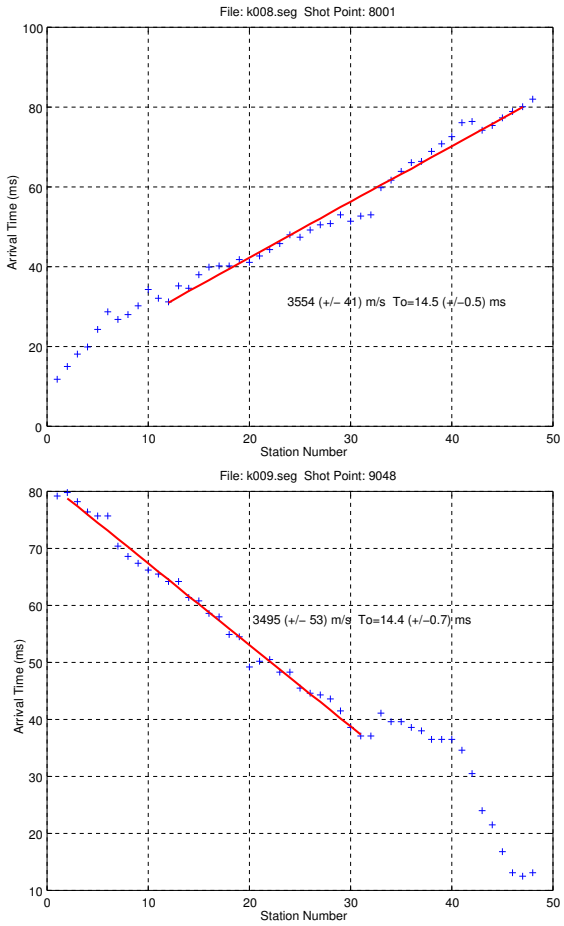


Figure 28: Plots generated with *refplot.m*.

6.9.3 Direct Wave Method

Before we analyze refractions, we can examine the direct wave to determine the overburden velocity. The program, *direct.m* packaged with BSU uses the least squares method to simultaneously solve for an overburden velocity using several shot records and arrival time picks on near offsets (less than 30 meters in our data). Figure 30 shows the general setup in a simplified case. Only data from near the shots are used. The basic travel time equation for the direct wave between shot A and geophone 1 is

$$X_{a1} \cdot \frac{1}{V_1} = t_{a1} \tag{20}$$

where X_{a1} is the distance between the shot A and geophone 1. The overburden velocity is given by V_1 and the observed first arrival time is t_{a1}.

We set up a matrix problem in the form

$$G \cdot m = d \tag{21}$$

which expands to

$$\begin{bmatrix} X_{a1} \\ X_{a2} \\ X_{b8} \\ X_{b9} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{V_1} \end{bmatrix} = \begin{bmatrix} t_{a1} \\ t_{a2} \\ t_{b8} \\ t_{b9} \end{bmatrix} \tag{22}$$

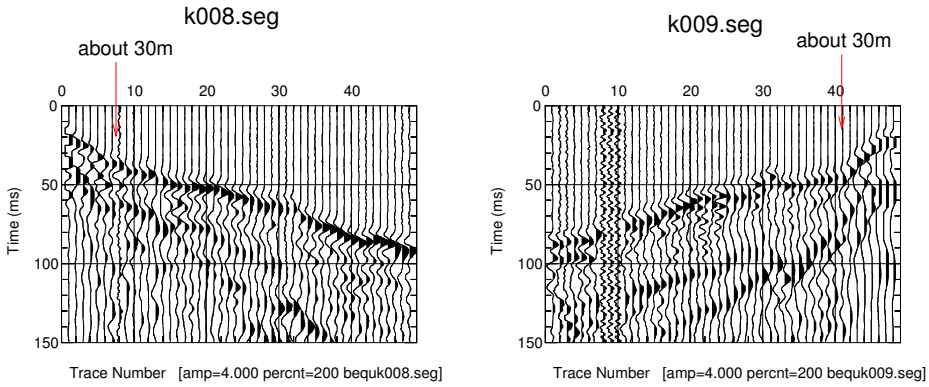


Figure 29: Choosing an estimate of the cross-over distance at 30 meters.

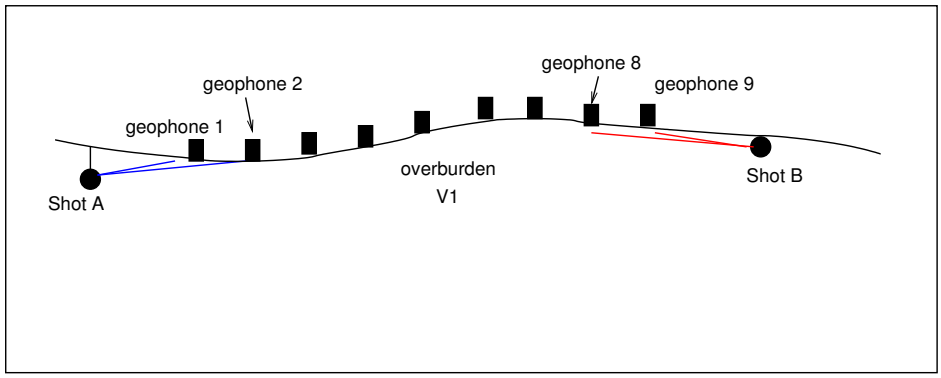


Figure 30: Direct wave raypaths used by program *direct.m*

The ordinary least squares (OLS) solution is given by (Menke [7]

$$m = \left[G^T G\right]^{-1} G^T \cdot d \tag{23}$$

It follows that the overburden velocity determination is $V_1 = \frac{1}{m}$.

6.9.4 Determination of Overburden Velocity

A Octave program, *direct.m* is included with BSU for this purpose. Having established 30 meters for the cross-over distance, we can run *bref* using the direct arrival option. We issue the command,

```
bref 0001 3 0 30 1 0 k004.seg k008.seg k009.seg
```

which uses 3 shots (the hammer and 2 explosives) to generate the required matrices and vectors. The command restricts offsets to less than 30 meters. The files created are

- **G0001** System matrix for direct arrivals
- **D0001** Data vector with pick times
- **E0001** Elevation and station vector for plotting

We start a Octave session, then execute the following command from within the Octave text window,

```
direct
```

The program prompts for the above 3 files, and then prompts for the number of shots (3 in this example). The

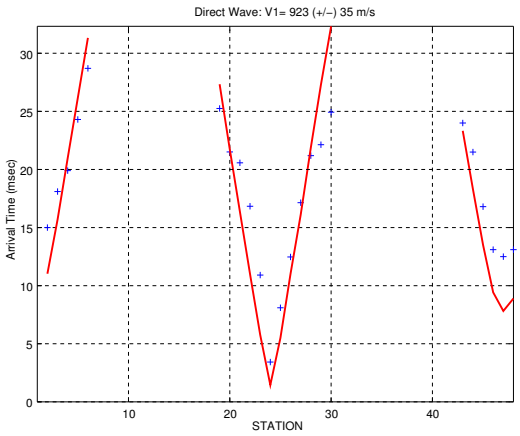


Figure 31: Solution for overburden velocity is 923 m/s based on k004.seg, k008.seg, and k009.seg.

program solves for the overburden velocity and generates a plot. Figure 31 shows the result. The title includes the simultaneous solution for the direct wave velocity.

In the actual project, additional shots were available to obtain increased confidence in direct and refracted wave solutions. The object here is too keep it simple for the reader to follow.

6.9.5 Delay Time Method

The basic theory can be found in many engineering seismology text books. The following is a brief summary of how BSU implements the method. A highly simplified case is shown in Figure 32

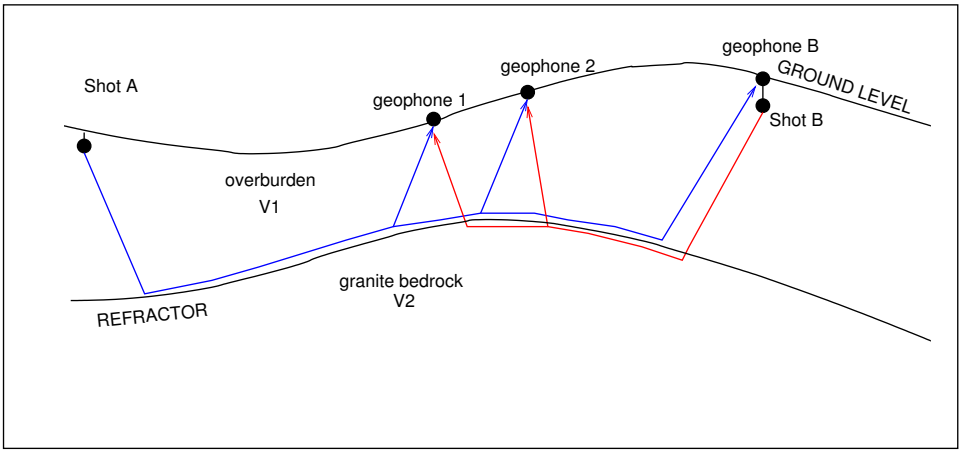


Figure 32: Simplified delay time setup. Shots A and B shoot into geophones 1 and 2.

The delay time equation for Shot A to geophone 1 is given by

$$T_a + T_1 + \frac{X_{a1}}{V_2} = t_{a1} \tag{24}$$

where T_a is the delay time at shot A, T_1 is the delay time at geophone 1, X_{a1}, is the horizontal distance between shot A and geophone 1, and t_{a1} is the observed travel time from shot A to geophone 1. The refractor velocity is V_2. A complete system becomes, in matrix form, the following:

$$G \cdot m = d \tag{25}$$

or

$$\begin{bmatrix} 1 & 0 & 1 & 0 & X_{a1} \\ 1 & 0 & 0 & 1 & X_{a2} \\ 1 & 1 & 0 & 0 & X_{ab} \\ 0 & 1 & 1 & 0 & X_{b1} \\ 0 & 1 & 0 & 1 & X_{b2} \end{bmatrix} \cdot \begin{bmatrix} T_a \\ T_b \\ T_1 \\ T_2 \\ \frac{1}{V_2} \end{bmatrix} = \begin{bmatrix} t_{a1} \\ t_{a2} \\ t_{ab} \\ t_{b1} \\ t_{b2} \end{bmatrix} \tag{26}$$

Equation 24 is the first row of equation 26. Matrix G is constructed by a program, *bref*, such that the first columns correspond to the shots, the other columns the geophones, ending in a last column giving the distance between a shot and receiver.

6.9.5.1 Adding Constraint Equations In practice, one may not have a geophone at shot B, or there may be geophones without observed first arrival times. In those cases, one must add constraint equations to the bottom of the matrix, G. For example, if we were unable to pick a time for the signal from Shot A to geophone 2, we might add a constraint that sets delay time at geophone 1 equal to that at geophone 2 (ie. $T_1 \simeq T_2$). That system would look like this:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & X_{a1} \\ 1 & 1 & 0 & 0 & X_{ab} \\ 0 & 1 & 1 & 0 & X_{b1} \\ 0 & 1 & 0 & 1 & X_{b2} \\ 0 & 0 & 9 & -9 & 0 \end{bmatrix} \cdot \begin{bmatrix} T_a \\ T_b \\ T_1 \\ T_2 \\ \frac{1}{V_2} \end{bmatrix} = \begin{bmatrix} t_{a1} \\ t_{ab} \\ t_{b1} \\ t_{b2} \\ 0.0 \end{bmatrix} \tag{27}$$

In this example, we set T_1 equal to T_2 with a weight of 9 fold in the least squares solution. This is the last row of matrix G in equation 27. It is like the following equation 28 being repeated 9 times in the matrix equation 27:

$$T_1 - T_2 = 0 \tag{28}$$

The effect is to strongly weight these two delay times equal. The OLS solution to Equation 27 is again given by

$$m = \left[G^T G\right]^{-1} G^T \cdot d \tag{29}$$

where, in the case of Equation 27, the vector, m, contains the delay times for the shots, receivers, and the refractor slowness.

$$m = \begin{bmatrix} T_a \\ T_b \\ T_1 \\ T_2 \\ \frac{1}{V_2} \end{bmatrix} \tag{30}$$

If we want to constrain the refractor velocity there is a way to do that too. Perhaps we lack reverse profiles or a second shot with at least a different angle between the shots and geophones which would yield more than one apparent refractor velocity. Modifying the above situation to include only shot A with a constrained refractor velocity, and a constrained delay time, T_a, we could set up a system like this:

$$\begin{bmatrix} 1 & 1 & 0 & X_{a1} \\ 1 & 0 & 1 & X_{a2} \\ 0 & 0 & 0 & 4000 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} T_a \\ T_1 \\ T_2 \\ \frac{1}{V_2} \end{bmatrix} = \begin{bmatrix} t_{a1} \\ t_{ab} \\ 1.0 \\ .005 \end{bmatrix} \tag{31}$$

Here, we set the velocity of the refractor at 4000 m/s and constrain the shot A delay time to .005 seconds. Both of these values would need to come from some other source, like an intersecting survey that was absent the defects here.

6.9.6 Delaytime Solution for Shoulder Line

This example is an introduction to the delaytime package included with BSU. We have a classic reverse profile situation with records k008.seg and k009.seg. We begin with executing *bref* for a normal refraction survey.

```
bref 0002 2 30 300 0 0 k008.seg k009.seg
```

This will restrict offsets to the interval of 30 to 300 meters (hopefully excluding direct arrivals). The files created are

- **G0002** System matrix for direct arrivals
- **D0002** Data vector with pick times
- **E0002** Elevation and station vector for plotting

The procedure is then to start a Octave session and execute the program *delaytm.m* from within the Octave text window. If we do this, we will get a faulty solution (negative delay times which would put the refractor above the ground level). The problem is under-determined due to missing data from the offsets less than 30 meters. We have to edit files **G0002** and **D0002**, adding constraint equations. There are 48 channels in these data, so a "G" matrix less than $2x48$ rows will spell trouble. The same is true for the "D" matrix. In fact, we note that there are only 83 rows, not 96. An additional problem (beyond no picks for less than 30 meter offset) is that the shots are offset from the line, beyond the default distance to constrain the delay times to a geophone location. We can obtain a solution by editing the G0002 and D0002 files. We add constraint equations that

1. Weight the shot delay time to match that of the closest geophone. Thus, the k008.seg shot will match geophone station 1, and the k009.seg shot will match geophone station 48 (check a header dump).
2. Weight the geophone stations without picks to match the closest station with a pick. Thus, for shot k008.seg we will need 6 constraints, and for shot k009.seg, we will need 7 constraints.

All together, we require 15 constraints (2 for shots, 13 for receivers). We add the constraints at the bottom of files G0002 and D0002. Included in directory *headWave* you will find the original and modified (with constraints) needed to obtain a solution. Weighting these constraints with a factor of 9 seems to work well. Thus, the geophone constraints will have a +9 and -9 to form an equation which will make two delaytimes equal. The corresponding "data" will be zero. Here is a partial listing.

```
9  0 -9  0  0  0  0  0  0  0 . . .
0  0  9  0  0  0  0  0 -9  0 . . .
0  0  0  9  0  0  0  0 -9  0 . . .
0  0  0  0  9  0  0  0 -9  0 . . .
0  0  0  0  0  9  0  0 -9  0 . . .
0  0  0  0  0  0  9  0 -9  0 . . .
0  0  0  0  0  0  0  9 -9  0 . . .
```

The first two columns are for the shots, the rest for receivers, up to an offset column. The first row equates the shot 8 delay time with geophone 1 delay time (their weighted by 9 sum equals zero, making them equal). The other rows set each geophone delay time equal to that of station 7. Again, the weighted sum of each delay time with that of station 7 will equal zero, making the delaytimes equal in a least squares sense. The match is not exact, since there are many other rows in the "G" matrix. For each constraint row added to the "G" matrix, there will be a an additional row added to the "D" vector, and the value for the data will be zero (to make the equation an equality statement). See section 6.9.5.1 for details on adding constraints.

To obtain a solution, refractor velocity, shot delay times, and geophone delay time constraints were added and can be compared to the original *bref* generated matrices. Using the *.mod versions, we obtained a solution shown in Figure 33. For details on running program *delaytm.m*, see section 6.9.11.

6.9.7 Broadside Shooting: Slope Line

Figure 34 is a base map showing a portion of the survey area. Data acquisition included a long line (5 meter geophone spacings) with reverse profiles along the shoulder of the highway. Shorter, cross lines were shot in the upslope direction with nominally 1 meter geophone spacings. The narrow right-of-way and steep slope conditions

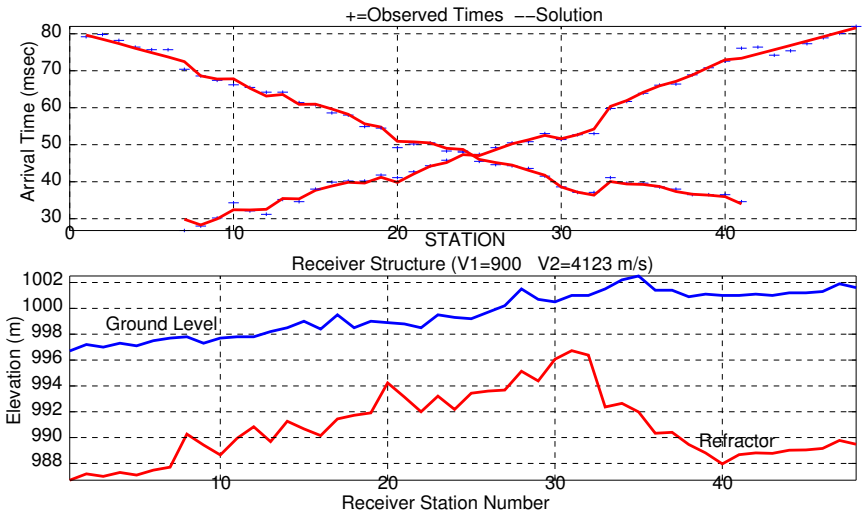


Figure 33: Delay time solution for line along road shoulder. The structure plot has been squished vertically to remove most of the vertical exaggeration in a simple figure.

prevented locating source points up the slope, and provided a severe restraint on the geophone locations in the zone of most interest. A concern was that not enough offset could be obtained in the up-slope direction.

The problem was solved in two stages. First, we did conventional reverse profile interpretations of the longer, line which paralleled the roadway (shoulder line above). Second, we used the first solution to constrain the delay time at source points used to acquire the cross-lines. In this example, we will work with just one of the cross-lines, line number 3. We will use two Bison records which were acquired with a hammer (15 fold vertical stack).

In Figure 34 we see line 1 (along the roadway) and the cross-line (line 3). The source point for shot record 10 was located near station 24 on line 1. The source point for shot record 11 was located near station 1 on line 1. We will invoke constraints to obtain a solution by using results from the line 1 solution.

6.9.7.1 Delay time Constraints The line 1 solution yielded a delay time of 0.0089 seconds at the hammer location for shot 10, and a delay time of 0.0150 seconds for the hammer location for shot 11. These values will be used to constrain the line 3 solution.

6.9.7.2 Refractor Velocity Constraint The other major constraint will be the refractor velocity, V2=4187 m/s determined from reverse profiles on line 1. This constraint is necessary since reverse profiles were not possible for line 3. As can be seen on Figure 34, the hammer blows fired broadside into line 3. The location for the source was not arbitrary, but chosen to be far enough away from all the geophone stations on line 3 (beyond the cross-over distance for the refraction). The bedrock refractor was granite, the overburden was silty sand with gravel (Unified classification SM). Soil testing indicated an internal angle of friction of 31.5^o . Given that the topographic slope was about 30^o , the in place overburden was judged to be at the angle of repose.

6.9.8 Converting the Bison File to BSEGY, Setting Geometry (*topcon, bis2seg, bhed*)

If you have unzipped the sample data set, you will find 4 files included to illustrate this example. In the bison subdirectory, **/ID-102/bison**, are the files:

- line3.nez [survey data from a Topcon total station, converted to NEZ format (5A12).
- bnk00010 [the Bison Engineering Seismograph file for shot 10]
- bnk00011 [the Bison Engineering Seismograph file for shot 11]
- gogeom [a script which will automate the process and is described next]

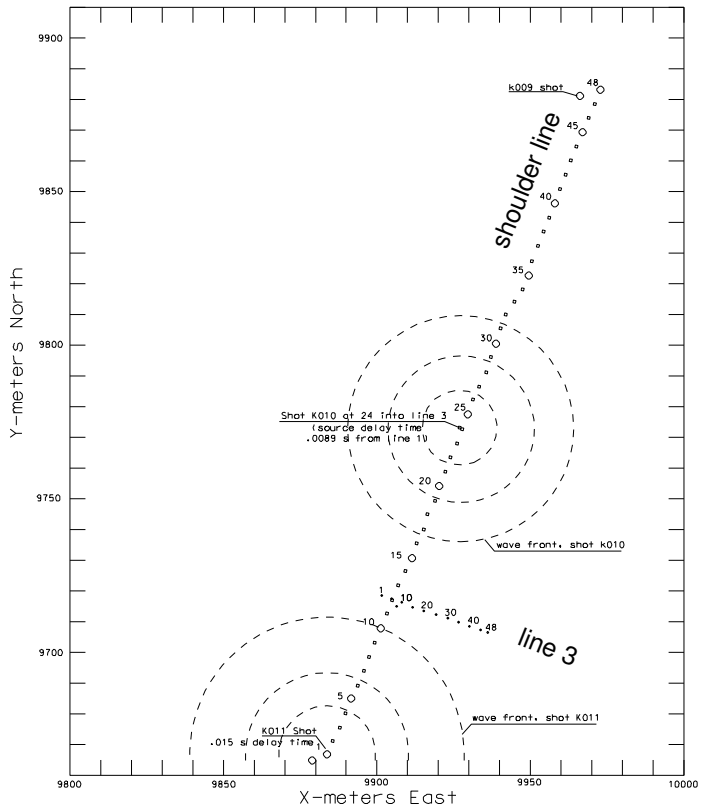


Figure 34: Base map for refraction survey (line 3 goes up hill from the roadway)

6.9.8.1 Contents of the gogeom script. The *topcon* man page and online documentation will be useful in understanding the first command. What it does is combine the survey information from *line3.nez* with header information from the Bison file, *bnk00010* to form a file, *bnk00010.xyz* which can be read by program *bhed*. The program *bis2seg* converts the Bison file to a BSEGY file with minimal headers, then bhed is run and merges the *bnk0010.xyz* information into the headers to complete the geometry setting. The Unix move command, "*mv*" , is used in places to rename the file to the 4 character BSEGY convention. The flow is replicated for shot 11. The *gogeom* script is as follows:

```
#!/bin/sh
# convert Bison to BSEGY
# shot10 (at station 24 line 1) Hammer Source
 topcon line3.nez bnk00010 0003 0. 24 48 1 48 10 0. 0  180  0  0
 bis2seg bnk00010
 mv bnk00010.seg k010.seg
 bhed k010.seg bnk00010.xyz 0
 mv bhedk010.seg k010.seg

# shot11 (at station 1 line 1) Hammer Source
 topcon line3.nez bnk00011 0003 0. 1 48 1 48 11 0. 0  180  0  0
 bis2seg bnk00011
 mv bnk00011.seg k011.seg
 bhed k011.seg bnk00011.xyz 0
 mv bhedk011.seg k011.seg
```

After running gogeom, you will have files *k010.seg* and *k011.seg* in the bison directory. Move these to the **/ID-102/seg** directory, and change to that directory for the next step in the process. Figure 35 is a plot of the k011.seg data. The data have been scaled by program *bscl*, and then plotted with Seismic Unix, clip=1.

By the way, for those who don't have Bison formatted data, there is an alternative program in BSU named

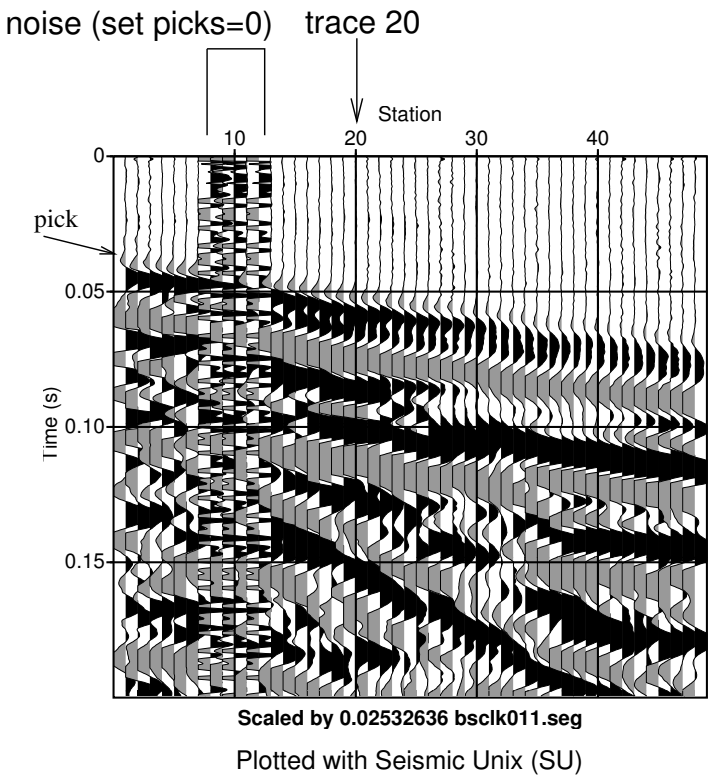


Figure 35: Scaled k011.seg refraction data

topcon2. It does essentially the same thing as *topcon*, but for SEG-2 formatted seismograph files. You will need to read the man pages on *topcon2*, since there are some differences.

6.9.9 Picking First Breaks

This topic was initially covered in the down-hole section 6.8.1. As described in that section, one copies the file *segpic.m* into the working directory, starts Octave and picks first arrivals. The program, *bpic*, is used to insert the pick times into the trace headers. **Not all traces can be picked due to some background noise. In those cases, one should set the pick time to exactly zero**, since program *bref* will use that convention to build suggested constraint equations. Figure 36 shows trace 20 of k011.seg as it appears during the execution of *segpic.m* in Octave. My picks are included in the data sample distribution as files k010.pic and k011.pic for comparison with your own picks.

6.9.10 Building the System of Delay Time Equations (*bref*)

The philosophy behind program *bref* is that the computer does computations, but the user makes the decisions. The delay time formulation is described more fully in Michaels [9] . The formulation is a linear matrix equation of the form

$$Gm = d, \tag{32}$$

where the matrix, **G,** contains information about the source and receiver geometry, **m,** is a vector of delay times and the refractor slowness, and the vector, **d,** holds the observed first arrival pick times. When multiplied out, a row of equation (32) can be written as

$$T_s + T_g + \frac{x}{V_2} = t_p, \tag{33}$$

where T_s is a shot delay time, T_g is a geophone delay time, $\frac{x}{V_2}$ is the horizontal shot to geophone separation divided by the refractor velocity, and t_p is the first break pick time, assuming that the geophone is located far enough away from the source to have the refracted head wave arrive before any other waves.

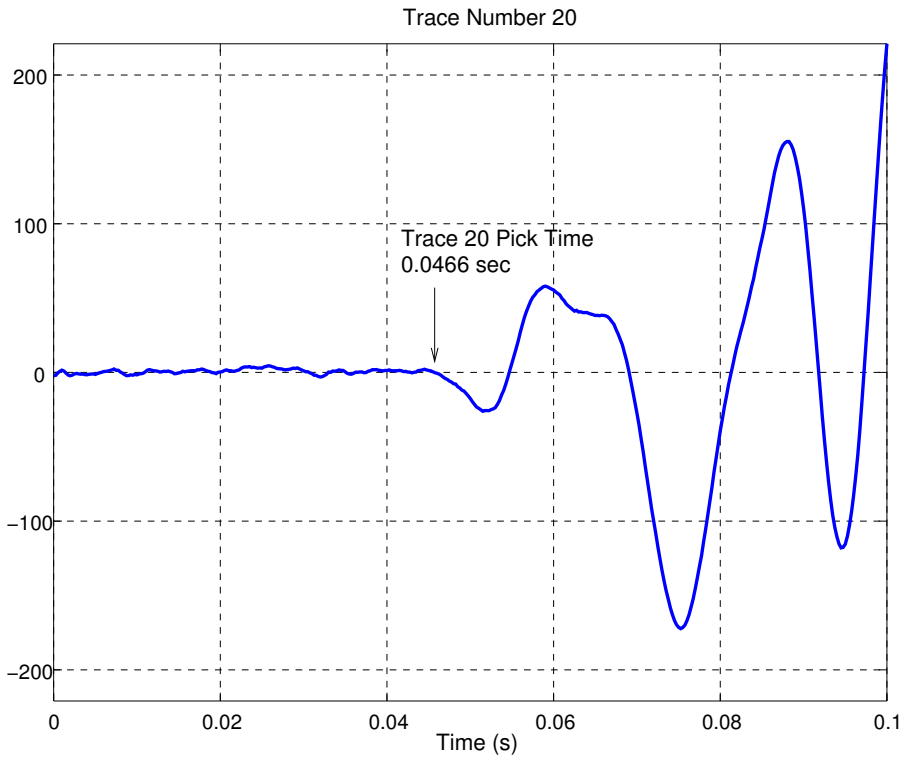


Figure 36: Trace 20 as seen in *segpic.m* run

6.9.10.1 Running *bref* This is an interactive program which prompts the user as the following dialog illustrates. The user responses are boxed.

```
  enter:  line number (4 characters)
0003
Enter:  number of shots
2
Enter:  minimum offset to include
40
Enter:  maximum offset to include
200
Is output for refraction or direct analysis?
0=refraction 1=direct
0
irecip 0=normal 1=reciprocal shooting
0
Enter:  input file name
k010.seg
Enter:  input file name
k011.seg
shot x,y= 9927.  9773.
shot x,y= 9883.  9666.
Traces Processed = 48
====> output in G0003
====> listing in brefk010.lst
```

In the above example, the program writes 3 files, the last 4 characters of which are the line number as specified in the dialog.

1. G0003 system matrix, G, of equation (32)
2. D0003 data vector, d, of equation(32)
3. E0003 elevation file (3 columns; trace#, station_name, elevation)

In general, you will have to edit the Gxxxx and Dxxxx files. The sample data set (ID-102.zip) includes examples of these files as created by *bref* (G0003.org etc.) and as modified to produce a constrained solution (G0003.mod etc., 15 constraints). Specific files created by *bref* will be highly dependent on the first break picks in the *.seg file trace headers. The convention is to flag picks *exactly equal to zero* as being unable to be picked. If none of the *.seg files has a pick at a particular geophone station, then the problem will be singular (unless you manually edit the file and add some rows to constrain the solution). *Bref* will make an attempt to add constraint rows to the bottom of the Gxxxx file, but they will be incomplete, since the user must decide what to do (don't expect this code to make critical decisions, that is your job). To replicate samples in archive, see README file in slopeLine directory.

6.9.10.2 Conventions: Structure of Gxxxx matrix The convention is to have the first *N_shots* columns correspond to source positions. The remaining *N_traces* columns correspond to the geophone stations. In our problem, there are 2 shot records, so the first two columns of G0003 are for shots 10 and 11 (because that was the order they were specified when running *bref*). The remaining columns correspond to geophone stations in the order the traces are read from the *.seg files. The last column is the horizontal distance between the source and receiver. The following matrix equation illustrates the setup for a very small problem (just for illustration, 2 shots, 3 geophones):

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & x_{11} \\ 1 & 0 & 0 & 1 & 0 & x_{12} \\ 1 & 0 & 0 & 0 & 1 & x_{13} \\ 0 & 1 & 1 & 0 & 0 & x_{21} \\ 0 & 1 & 0 & 1 & 0 & x_{22} \\ 0 & 1 & 0 & 0 & 1 & x_{23} \end{bmatrix} \cdot \begin{bmatrix} T_{s_1} \\ T_{s_2} \\ T_{g_1} \\ T_{g_2} \\ T_{g_3} \\ \frac{1}{V_2} \end{bmatrix} = \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \end{bmatrix}, \tag{34}$$

where the shot delay times are T_{s_j} and the geophone location delay times are T_{g_j} . The refracted, first arrival times are indicated as t_{sr} for the s-r shot and receiver pair, and the offset between the s-r shot-receiver pair is also subscripted as x_{sr} .

6.9.10.3 Conventions: Structure of Dxxxx vector The convention is for two columns. The first column is the observed first break refracted arrival time. The second column is the geophone station label. If there are constraints at the bottom, then these will be as needed to complete a constraint equation.

6.9.10.4 Editing the Gxxxx and Dxxxx files Suggested constraint equations are weighted by a factor of 9. If you inserted my picks into the *.seg data trace headers, then you will note that *bref* has returned a Gxxxx matrix with 4 rows at the bottom that need editing. One strategy is to set the delay time for an unpicked station exactly equal to the nearest neighbor which does have a pick. One does this by replacing a "0" with a "-9" at the station which has the constraining value. The idea is to form an equation which sums to zero. Thus, to constrain the delay time for geophone station 8 to equal that at station 7, we edit the file, placing a -9 in column 7 of the row with a *bref* provided 9 in column 8. The corresponding entry in the same row of Dxxxx will be zero. Thus, the product Gm=d produces a constraint equation of

$$-9T_{g_7} + 9T_{g_8} = 0. \tag{35}$$

where T_{g_7} is the delay time at geophone 7, and T_{g_8} is the delay time at geophone 8. In the least squares solution, this is given a weight of 9, which makes the constraint fairly strong.

In our example problem, one needs additional constraints, beyond what is required for unpickable data. First, to tie the line 1 solution, we need to constrain the shot delay times to match the values obtained from the line 1 solution. In this case, the decision is to give these constraints a weight of 10 (the math is simple, move the decimal over one place). Thus, one adds two rows in the G0003 file, one has a 10 in column 1, and the other has a 10 in column 2. Corresponding rows need to be added to the D0003 file. In these rows (first column) we place the delay

times from the line 1 solution (scaled by a factor of 10). Thus, for the row with a 10 in column 1 of the G0003 file, we enter a value of 0.0890 in the D0003 file, which is 10 times the delay time at station 24 of line 1, the same location where the hammer was placed for shot 10, file k010.seg. For shot 11 (column 2 has a 10 in G0003), we place a value of 0.1500 which is 10 times the delay time at station 1 of line 1. Thus, the product Gm=d produces a constraint equations

$$\begin{array}{l} 10T_{s_1} = .0890 \\ 10T_{s_2} = 0.150 \end{array} \quad , \tag{36}$$

where T_{s_1} is the delay time for shot 10 (first column), and T_{s_2} is the delay time for shot 11 (second column).

One last constraint equation must be added. This is to constrain the refractor velocity. We can not solve for this since both source positions are basically measuring a single apparent velocity on the refractor. We draw on the line 1 solution again (it had reverse profiles and was able to resolve the refractor velocity). The refractor velocity was found to be 4187 m/s for the granite. One way to do this is to add one more row with the only non-zero value being the refractor velocity in the last column of G0003. Normally, the source-receiver distance goes in this column, but here, we make it a constraint by adding a last row to D0003, and inserting a "data" value of 1.000 in the first column. Thus, the product GM=d produces a constraint equation

$$\frac{4187}{V_2} = 1.000 \tag{37}$$

from this last row. The files *G0003.mod, D0003.mod* and *E0003.mod* included with the data sample are modified as described above.

6.9.11 Running the Delay Time Inversion (*delaytm.m)*

The actual solution is computed in a Octave procedure, *delaytm.m*, found in the BSU distribution. The overburden velocity was determined to be 664 m/s from near offset and up hole analysis (line 1 was recorded with buried explosives). The procedure writes a number of *.dat files which are read for additional plotting.

Dialog when running *delaytm.m* A number of GUI prompts require responses from the user. These are described as follows:

- Choose a G matrix file. Select G0003.mod, or perhaps your own edited version. Click OK
- Choose a D vector file. Select D0003.mod, or your own version. Click OK.
- Choose an E vector file. Select E0003.mod, or your own version. Click OK.
- Number of Shots? Enter 2 and click OK.
- Number of Channels? Enter 48 and click OK.
- Enter smoothness constraint. Enter a value of 0.1. What this does is increase the least squares error slightly, but adds a built in change to the objective function (the spatial derivative of the delay time solution). A value of zero here will produce the smallest error in the solution, but may map picking noise into false structure).
- The text window will show the shot delay times. They should agree with those specified in the constraint equations.
- The graphic window will have the refractor velocity in the title, and it should agree with the constraint applied in our problem. The plotted curve is a graph of the delay time solution, geophones.
- Enter overburden velocity. Here, type in a value of 664 as discussed above.
- The graphic will show a plot of the ground topography and the refractor structure. This is one end-limit solution that resolves the delay times completely as variations in the refractor structure. This is the preferred solution for this geologic setting.

- Enter a constant refractor depth. This is an alternative solution which resolves the delay times as variations in the overburden velocity (perhaps due to variations in water content). If you use the default, 10 m, you will see that the overburden velocity would have to vary from 1265 to 300 m/s to explain the data. This end-limit solution is not preferred in this geologic setting.

- Number of constraint equations? Enter 7 here, corresponding to the number of rows at the bottom of the G0003 file that we edited. This helps avoid a completely trashed plot where constraints become overprinted with the time plot. With more than one shot record, there may be some over plot of the computed times in this last graphic. You can fix that in Xfig.

Resolving Delay Times The *delaytm.m* procedure solves for the delay times, and then resolves them into two end-limit solutions. Variations in delay times are resolved totally into a variation in the refractor structure with the formula

$$\begin{aligned} h &= \frac{V_1}{\cos(\theta)} \cdot T_j \\ \theta &= \arcsin\left(\frac{V_1}{V_2}\right) \end{aligned} \quad , \tag{38}$$

where T_j is the j-th delay time and "h" is the distance from the source or geophone to the refractor. This distance, h, is a radius specifying a circle, somewhere on which the refractor may be found. In delaytm.m, the refractor position is plotted a distance, h, directly below the source or geophone (unmigrated position). For the purpose of most engineering surveys, migration of the refractor point is not a significant issue (the distances are quite small).

The alternative end-limit resolution of delay times is as a variation in overburden velocity. The user provides a distance from the recording surface to the refractor (held constant), and an overburden velocity is found from the formula

$$V_1 = \frac{V_2}{\sqrt{1+\left(\frac{T_j V_2}{h}\right)^2}}, \tag{39}$$

where T_j is the j-th delay time and "h" is the constant distance from the recording surface to the refractor. This assumes that the refractor velocity, V_2 , is constant, and the only variation is in V_1 , the overburden velocity. This type of solution makes sense when the water content of the overburden soil is known to vary, and the overburden thickness is relatively constant. In reality, the truth will be somewhere between these two limiting cases. See Michaels [9] for a discussion on this topic.

One should probably produce *xfig* scaled plots, as some CAD work is usually required to clean things up. Figure 37 shows how a final merging of the exported *.fig files will look with a little CAD effort. In this case, the structural solution, Figure 37B is preferred because of our knowledge of the geology from trenching and surface observations.

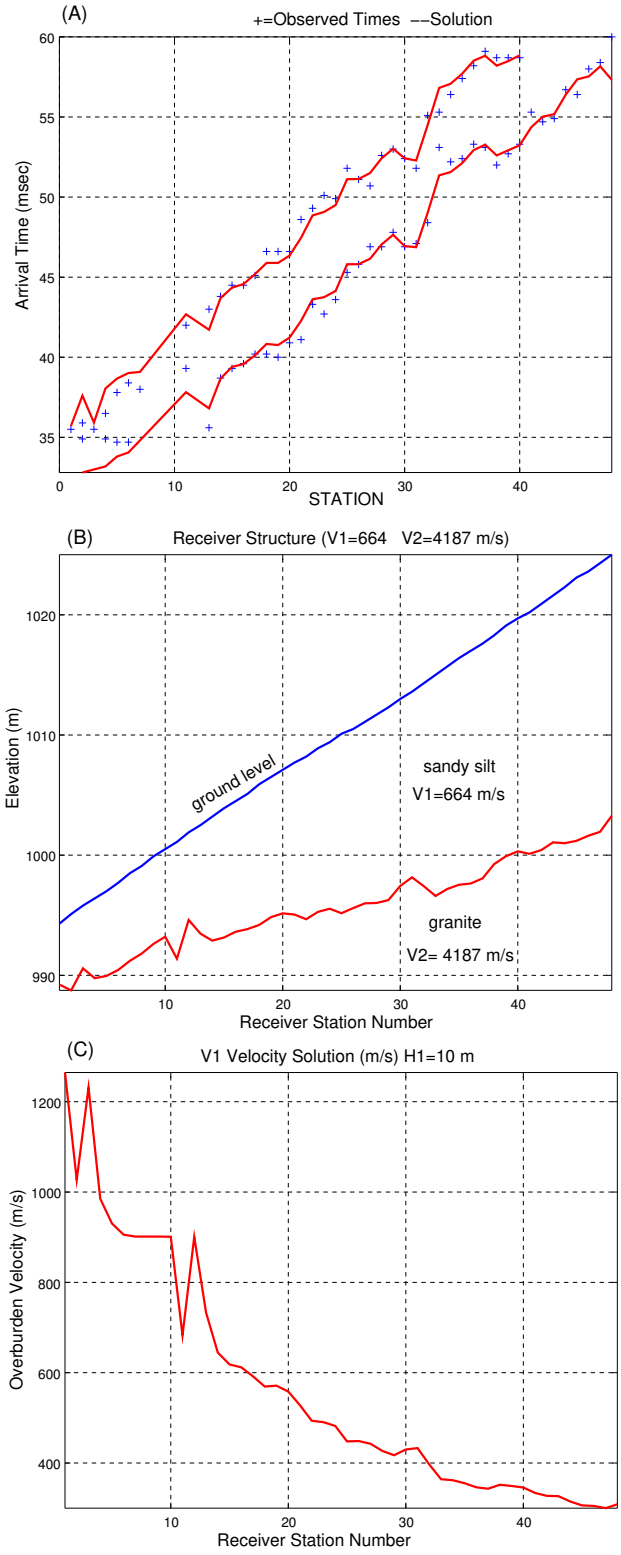


Figure 37: Line 3 solution, merged *xfig* plots. A). Arrival times and fit, B). Structural Solution (accepted), C). Overburden velocity solution (rejected)

6.10 Reciprocal Refraction

When shooting a refraction profile across a river, it is advantageous to exchange the conventional shot and geophone locations for the survey. The river environment is typically very noisy due to the river current and its disturbing effects on the geophone. Placing the geophones on the river bank, and deploying an airgun source off an existing bridge can be very successful in developing a refraction profile. An example is shown in Figure 38. In theory, one could use just 2 geophones, one on the north bank, and one on the south bank. When sorted to common geophone gathers, this would provide reverse profiles for refraction analysis. However, deploying a grid of geophones costs very little more in effort, and opens up the opportunity for array forming to further increase the signal to noise. Selected files from this project may be downloaded from the BSU Database web page.
Go to https://cgiss.boisestate.edu/ pm/BSU/. Click on the check boxe for *Horse Shoe Bend* and then on submit. It will list the download archive. ID-103.zip contains the raw EGG SEG2 files, processed BSEGY and SEGY files with some scripts. Assuming you have the `ID-103` archive and unpacked it in a convenient location, you will note that it has the following directory structure:

```
ID-103
|--- bref
|--- seg
|     |---temp
|--- seg2
‘--- sgy
```

Packaged with BSU is the Octave program *delaytmR.m*. The “R” in the name indicates it is for use with reciprocal shooting. This program is nearly identical to the previously described program without the “R” in the name. There is one significant difference, and that is the requirement for an additional water depth file, *wds.dat*. This file consists of 2 columns, *(water depth, shot station number)*.

The reciprocal nature of the problem is done entirely with BSU programs which resort to common geophone gathers. The roll of shot and receiver are reversed, so pretend that when `bref` is run, *the question of number of shots is answered by the number of receiver gathers*. The Octave processing is essentially blind to this reversal in roles, since it merely reads the *bref* generated matrices and treats the picks as though they were from common shot gathers, with the additional processing of a water layer.

6.10.1 Sorting to Common Receiver Gathers

In this example, we have 21 shot records taken with shot positions moving across the river. Each shot record contains two blocks of channels with a total of 64 channels (some of which were disconnected since the cable take outs would have been on the bridge). One block is from the North bank, the other from the South bank of the river. A partial listing of the header dump shows the south block of channels for the northern most shot, 1001. To save space, not all channels are shown. This type of survey can be very challenging, since good record keeping is essential. Here, a significant number of cable take outs are left unconnected. Getting off by even one channel can lead to chaos. The actual shots were in the river, suspended from the bridge deck by the compressed air hose. The coordinates in the headers are for the airgun position in the river, NOT the suspension point on the bridge. The airguns drift somewhat west of the bridge due to the current.

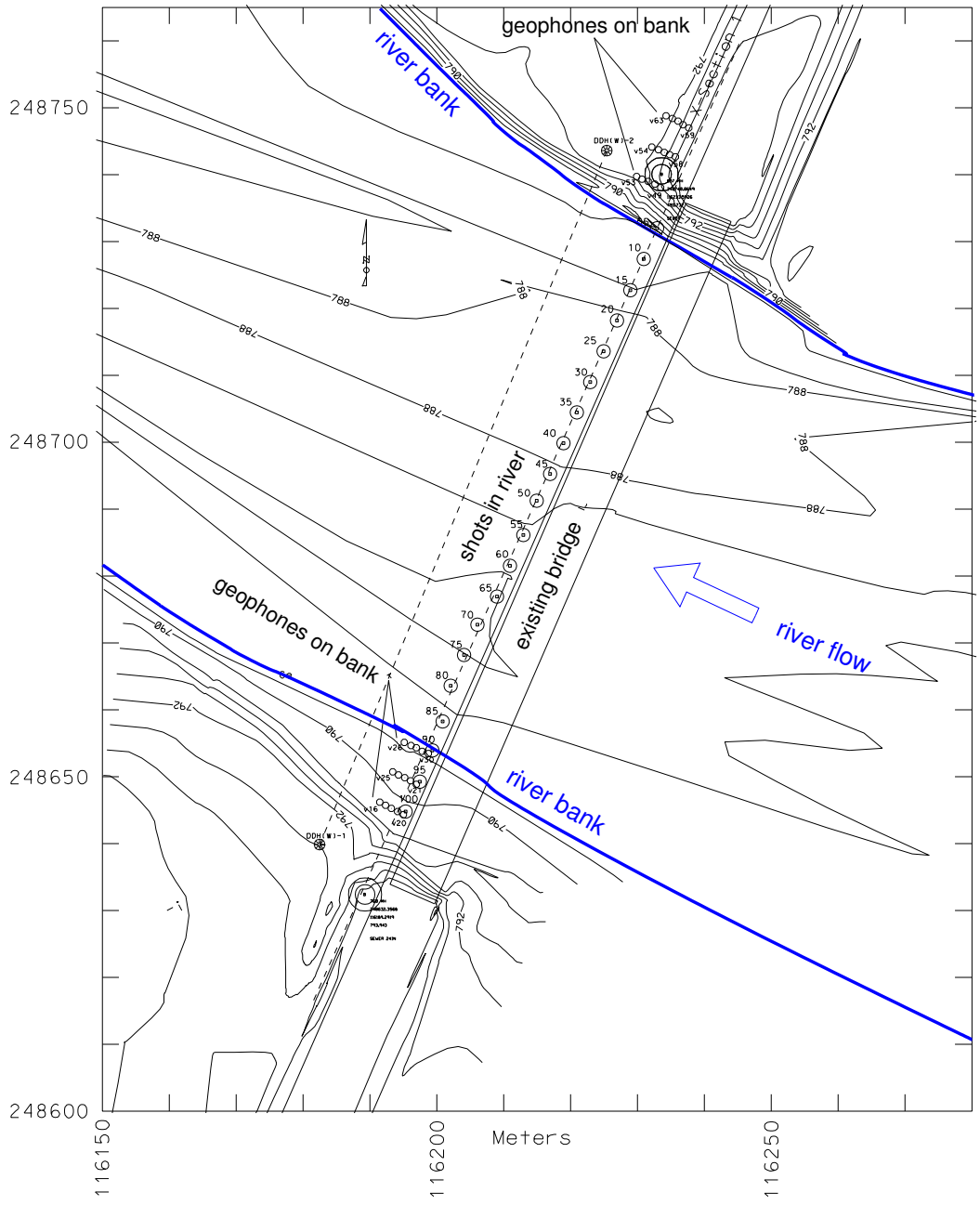


Figure 38: Reciprocal shooting for refraction surveys across rivers. Bridge foundation investigations benefit from placing the geophones on land, and the source suspended from the bridge in the river.

```
                    |-------------------------------|
                    |   PARTIAL SEGY HEADER DUMP    |
                    |            1001.seg           |
                    |-------------------------------|
 ------------------------------------------------------------------------------------------
   Length =  4000 samples             |  Shot Elevation =      789.3
   Sample Interval =    0.00025 sec.  |  Shot Depth =        0.0
   Delay Time =    0 msec.            |  Up Hole Time =     0 msec
   Low Cut Filter  =   10 Hz.         |  Shot X-COORD =    1052.01
   High Cut Filter = 1000 Hz.         |  Shot Y-COORD =    1101.10
   Line ID:                           |  Shot Date (year.day) =    0.0000
   Shot Orientation:                  |  Shot Time (hr:min)   = 00:00
   Azimuth=  0 Deg.  Vertical=180 Deg.|  Charge Size (grams)=     0
 ------------------------------------------------------------------------------------------
 TRACE|SHOT| STATION | OFFSET|        RECEIVER          |VERT|1STBRK|K-GAIN|AZI|VER|
   #  |REC.|SHOT  REC|       |  ELEV.  X-COORD   Y-COORD|FOLD|(SEC.)| (dB) |   |   |
 -----|----|---------|-------|--------------------------|--|------|------|---|---|
    1 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    2 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    3 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    4 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    5 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    6 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    7 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    8 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
    9 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
   10 |1001|  00   30|  90.56|  789.53   1010.76   1020.48| 7|0.0000|   24 |  0|  0|
   11 |1001|  00   29|  90.62|  789.48   1009.83   1020.89| 7|0.0000|   24 |  0|  0|
   12 |1001|  00   28|  90.54|  789.48   1008.99   1021.43| 7|0.0000|   24 |  0|  0|
   13 |1001|  00   27|  90.56|  789.44   1008.22   1021.83| 7|0.0000|   24 |  0|  0|
   14 |1001|  00   26|  90.58|  789.37   1007.26   1022.35| 7|0.0000|   24 |  0|  0|
   15 |1001|  00   25|  95.28|  789.86   1005.26   1018.09| 7|0.0000|   24 |  0|  0|
   16 |1001|  00   24|  95.34|  789.90   1006.10   1017.55| 7|0.0000|   24 |  0|  0|
   17 |1001|  00   23|  95.32|  789.90   1006.98   1017.09| 7|0.0000|   24 |  0|  0|
   18 |1001|  00   22|  95.41|  789.93   1007.81   1016.55| 7|0.0000|   24 |  0|  0|
   19 |1001|  00   21|  95.51|  789.98   1008.65   1016.01| 7|0.0000|   24 |  0|  0|
   20 |1001|  00   20| 100.45|  790.21   1006.44   1011.59| 7|0.0000|   24 |  0|  0|
   21 |1001|  00   19| 100.38|  790.23   1005.57   1012.11| 7|0.0000|   24 |  0|  0|
   22 |1001|  00   18| 100.31|  790.22   1004.69   1012.65| 7|0.0000|   24 |  0|  0|
   23 |1001|  00   17| 100.29|  790.23   1003.86   1013.13| 7|0.0000|   24 |  0|  0|
   24 |1001|  00   16| 100.25|  790.28   1003.02   1013.64| 7|0.0000|   24 |  0|  0|
   25 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
   26 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
   27 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
   28 |1001|  00   00|1715.28|    0.00      0.00      0.00| 7|0.0000|   24 |  0|  0|
   . . . continues for total of 64 channels . . .
```

Note that channels 1-9 are disconnected from any geophones, as are 25-28. The live channels in the portion of the dump shown are 10-24. We are particularly interested in the "STATION REC" label, as this will be used to select a single geophone signal in our gather. We begin by concatenating all the shot records into a single file using simple Unix commands in a bash script.

```
#!/bin/bash
# script to concatenate all shot records into file 0000.seg
F="1021.seg 1020.seg 1019.seg 1018.seg 1017.seg 1016.seg 1015.seg 1014.seg 1013.seg 1012.seg 1011.seg
1010.seg 1009.seg 1008.seg 1007.seg 1006.seg 1005.seg 1004.seg 1003.seg 1002.seg 1001.seg"
for f in $F; do
cat $f >> 0000.seg
done
```

We then use program *bext* to extract those signals who have a "STATION REC" header of 26.

```
#!/bin/bash
# Note: 4char name space,space,2,6
bext 0000.seg r "  26"
mv bext0000.seg VP26.seg
```

The result can be viewed by examining the header dump for file *VP26.seg*. Because the shots were concatenated in order from largest file number to smallest, our extraction of geophone 26 signals retains that ordering. Here is the partial header dump of the new file. Note that the first trace corresponds to SHOT RECORD, 1021.seg. This is labeled as SP100 on the map (SP=shot point). The last trace corresponds to the Northern most shot, SP00 on the map. All the traces are for geophone STATION REC = 26. We named the file *VP26.seg* following the convention of VP=voltage point (a historic reference to the take-out where a geophone is plugged in). On Figure 38, VP26 is labeled V26 to save space, and the SP is dropped for the similar reason, to avoid clutter.

```
                    |-------------------------------|
                    |   PARTIAL SEGY HEADER DUMP    |
                    |           VP26.seg            |
                    |-------------------------------|
-------------------------------------------------------------------------------------------
  Length =  4000 samples             |  Shot Elevation =      789.9
  Sample Interval =    0.00025 sec.  |  Shot Depth =        0.0
  Delay Time =    0 msec.            |  Up Hole Time =     0 msec
  Low Cut Filter  =   10 Hz.         |  Shot X-COORD =    1006.80
  High Cut Filter = 1000 Hz.         |  Shot Y-COORD =    1012.04
  Line ID:                           |  Shot Date (year.day) =    0.0000
  Shot Orientation:                  |  Shot Time (hr:min)   = 00:00
  Azimuth=  0 Deg.  Vertical=180 Deg.|  Charge Size (grams)=     0
-------------------------------------------------------------------------------------------
TRACE|SHOT| STATION | OFFSET|        RECEIVER          |VERT|1STBRK|K-GAIN|AZI|VER|
  #  |REC.|SHOT  REC|       |  ELEV.  X-COORD   Y-COORD|FOLD|(SEC.)| (dB) |   |   |
-----|----|---------|-------|--------------------------|--|------|------|---|---|
   1 |1021| 100   26|  10.33|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   2 |1020|  95   26|   6.30|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   3 |1019|  90   26|   4.33|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   4 |1018|  85   26|   6.51|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   5 |1017|  80   26|  10.92|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   6 |1016|  75   26|  15.79|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   7 |1015|  70   26|  20.72|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   8 |1014|  65   26|  25.83|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
   9 |1013|  60   26|  30.77|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  10 |1012|  55   26|  35.74|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  11 |1011|  50   26|  41.20|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  12 |1010|  45   26|  45.68|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  13 |1009|  40   26|  50.67|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  14 |1008|  35   26|  55.63|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  15 |1007|  30   26|  60.64|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  16 |1006|  25   26|  65.61|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  17 |1005|  20   26|  70.62|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  18 |1004|  00   26|  90.58|  789.37   1007.26   1022.35| 9|0.0000|   24 |  0|  0|
  19 |1003|  15   26|  75.61|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  20 |1002|  10   26|  80.62|  789.37   1007.26   1022.35| 1|0.0000|   24 |  0|  0|
  21 |1001|  00   26|  90.58|  789.37   1007.26   1022.35| 7|0.0000|   24 |  0|  0|
```

A single shot record collected with an airgun in the river can produce a large amplitude surface wave train along the river bottom. That was the case with this survey, and so the P-wave refractions required enhancement to be picked. This included array forming by summation of geophone signals from geophones in the grid, followed by filtering to enhance the higher frequency content. Figure 39 illustrates the benefit of additional processing. The enhancement processing with BSU included:

- **Array Summing** with program *bsum*. Geophones in a row were summed, not to cancel the surface waves, but rather to enhance the P-wave refraction content.
- **Profile Deconvolution** with program *bdcn*. The operator length was 235 samples. The design gate was 0 to .06 seconds. Stability factor was set at 1% of the zero lag autocorrelation amplitude.
- **Band-pass Filter (zero phase)** Pass band was from 50 to 100 Hz. Here, I used Seismic Unix (SU), because *bfil* had not yet been written.

6.10.2 Running *delaytmR.m*

The P-wave enhancement was used to produce 3 North bank and 3 South bank enhanced data sets. The following listing from the *bref* run documents the choice of parameters. The key parameter was to choose reciprocal shooting (changes the header search from shots to receivers as can be seen in the listing).

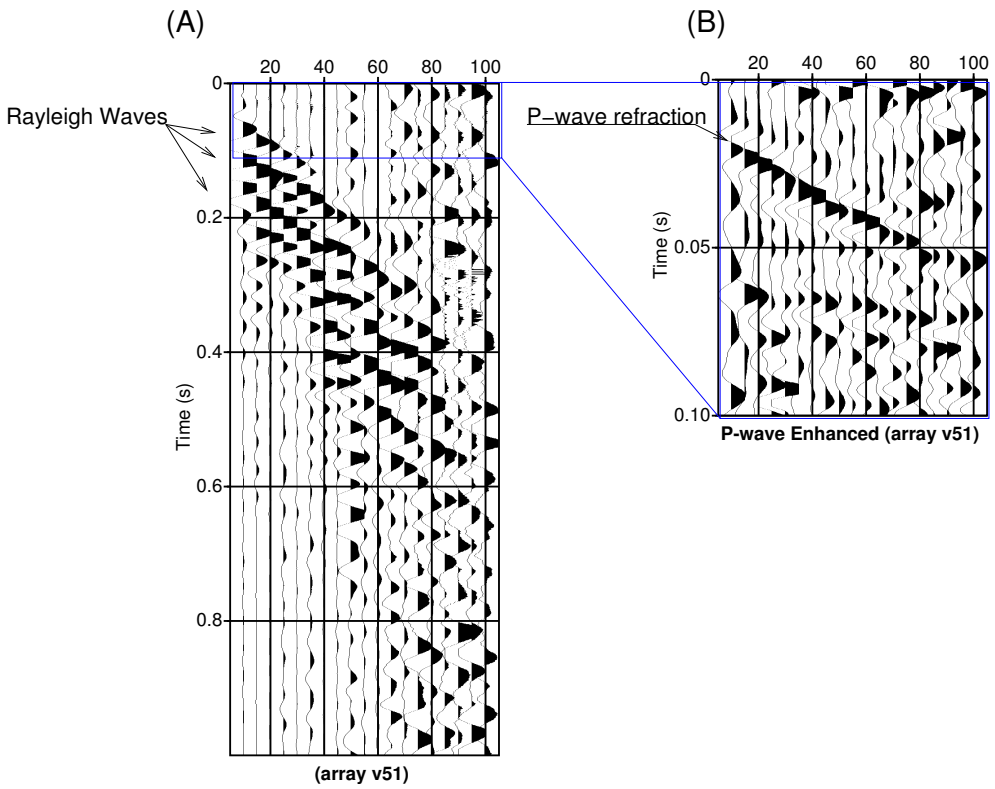


Figure 39: Array forming and filtering to enhance higher frequencies were needed to pick refractions. (A) shows an array formed record with strong Rayleigh and SV wave content. (B) is a blowup of the shallow data enhanced for P-waves by filtering.

```
Program bref
 Line Number: 0001
 System matrix output File: G0001
 Data vector output File: D0001
   receiver x,y=     1049.       1104.
   receiver x,y=     1009.       1021.
   receiver x,y=     1051.       1108.
   receiver x,y=     1007.       1017.
   receiver x,y=     1054.       1113.
   receiver x,y=     1005.       1013.
  Number of traces=   19
  Parameters:
     Number of shots=    6
     Minimum offset =      10.0
     Maximum offset =     200.0
     Refraction analysis
     Average Receiver Spacing=    5.0079
     Ground Consistent Zone=    3.7559
     Input File Names:
          N000.seg
          S000.seg
          N001.seg
          S001.seg
          N002.seg
          S002.seg
```

The problem required 3 constraint equations. These were to set the 3 common geophone points and the neighboring shot points to have the same delaytime.

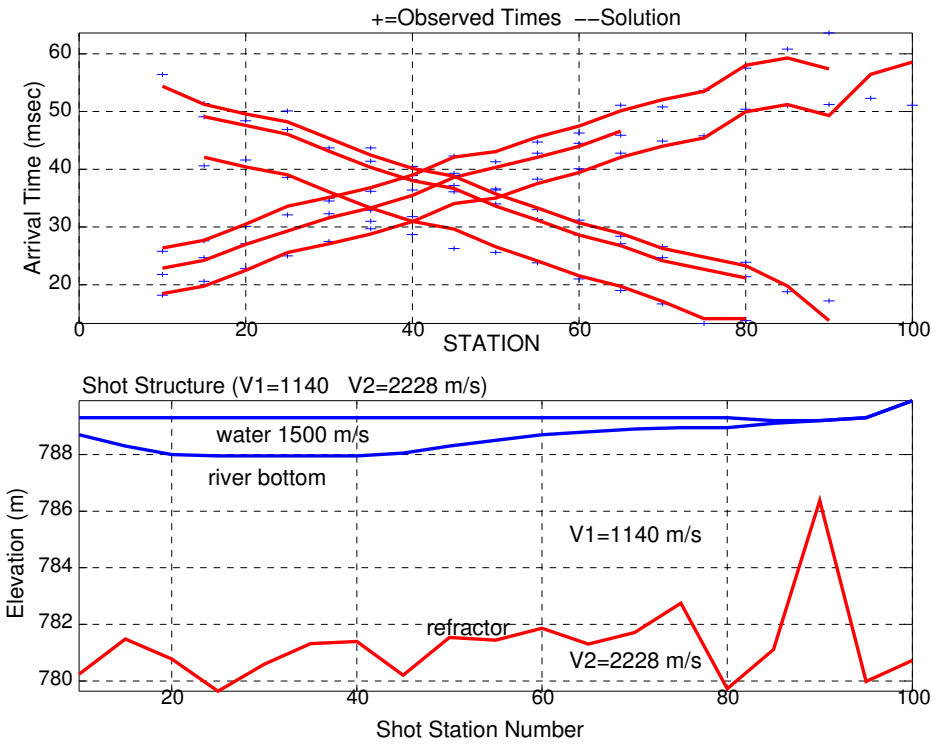


Figure 40: Solution from *delaytmR.m* analysis of 6 common geophone records and 3 constraints. Note, even after squishing the plot, there is about 12:1 vertical exaggeration on the structure.

6.11 Surface Wave Processing

The BSU package includes some software for Rayleigh wave analysis. These programs are as follows:

- **bvax.f** A fortran program which extracts a dispersion curve from a *bsegy* data set.
- **FwdR1.m** A Octave program which does the forward problem and displays a measured dispersion curve. Where *invR1.m* is automated, this program is interactive, requiring the user to make changes to the model with each iteration.
- **invR1.m** An automated Octave inversion program which reads the dispersion curve from a *bvax* run. The input file to the Octave program is `bvax.his`. If multiple runs of *bvax* are to be made, the file `bvax.his` should be deleted before each new run, since it is appended to during a *bvax* execution.

Example programs:

- **rayleigh.m** This is an example program which shows how to link the dispersion computing module *disper.oct (shared object)* with the octave session. The fortran subroutine in file `rwv.f` is based on the fortran program *disper.f* included in BSU. When running *rayleigh.m*, the subroutine file, *rwv.f* must be in the directory where one is launching Octave. On execution of *rayleigh.m*, a check is made to see if a compiled version exists, and if not a compilation step occurs. The object module is dynamically linked to the Octave procedure to produce high speed computations. In addition to the Fortran file, *rwv.f*, two other files should also be in the path or directory of computation. These are *wrapper.cpp* and *build_disper_oct*.
- **moho.m** Similar to *rayleigh.m*, but computes dispersion as a function of period rather than frequency. Also an example program.

6.11.1 Example Rayleigh Wave Processing: Measuring Dispersion

The next step is to use the program *bvax* to measure a dispersion curve from the waveform data. Here, we are using the synthetic data of Figure 45. The input parameters for a command line execution are show below:

```
     bvax infile xmin xmax vmin vmax nvel . . .
          fmin fmax delf bwd iskp ivscn

infile  =input file name
xmin    =minimum offset (float)
xmax    =maximum offset (float)
vmin    =minimum velocity
vmax    =maximum velocity
nvel    =number of velocity increments
fmin    =minimum frequency Hz
fmax    =maximum frequency Hz
delf    =frequency increment Hz
bwd     =filter bandwidth Hz
iskp    =skip filtering (if files already exist)
          1=YES  0=NO (-1=NO and delete when done)
ivscn   =output velocity scan data sets
          1=YES  0=NO
```

```
bvax wavV.seg .5 50.  50.  500.  200 6.  50.  1 .25 -1 0
```

The choice of frequency increment and bandwidth will depend on your data record length. For example, for 0.5 second signals, set $delf = 2$ Hz increments. For 2.0 second signals, set $delf = 0.5$ Hz. The longer the temporal aperture, the finer the spectral resolution possible. The program determines phase velocity using cross correlations of extremely narrow band signals (one can't pick arrivals, that would be appropriate for a group velocity procedure). At each frequency selected, trial velocities are applied as static shifts, and semblance computed to determine the degree of alignment. A peak in the semblance is picked by a Golden Section search, and the hope is that the picked peak is for the fundamental mode. The number of trial velocities needs to be large enough to provide good image files, but need not be large if your interest is primarily in the picks of semblance peaks (output file *bvax.his*).

6.11.2 Running BVAX

This program measures surface wave dispersion. The *bvax* code has a number of outputs:

- **bvaxnnnn.lst** Output listing, `nnnn.seg` would be the input file.
- **bvax.ps** Post script plot file of measured dispersion.
- **bvaxqc.ps** QC multipage plot, one page for each frequency.
- **bvax.his** Text file with 5 columns: [Frequency, Phase Velocity, Velocity Uncertainty (1 Stdv), Semblance, Tbar (mean), Tvar (variance)]
- **semblance.dat** Text file, 3 columns: `[Frequency, Phase Velocity, Semblance]` used to generate Gnuplot files.
- **CLRplot.gp** This is a Gnuplot file that generates a color map of dispersion semblance values.
- **clrplot.png** BVAX automatically generated image from CLRplot.gp
- **CNTplot.gp** This is a Gnuplot file that generates a contour plot of dispersion semblance values.
- **cntplot.png** BVAX automatically generated image from CNTplot.gp
- **MSHplot.gp** This is a Gnuplot file that generates a mesh image of semblance values.
- **GRYplot.gp** This is a Gnuplot file that generates a grey plot image of semblance values.

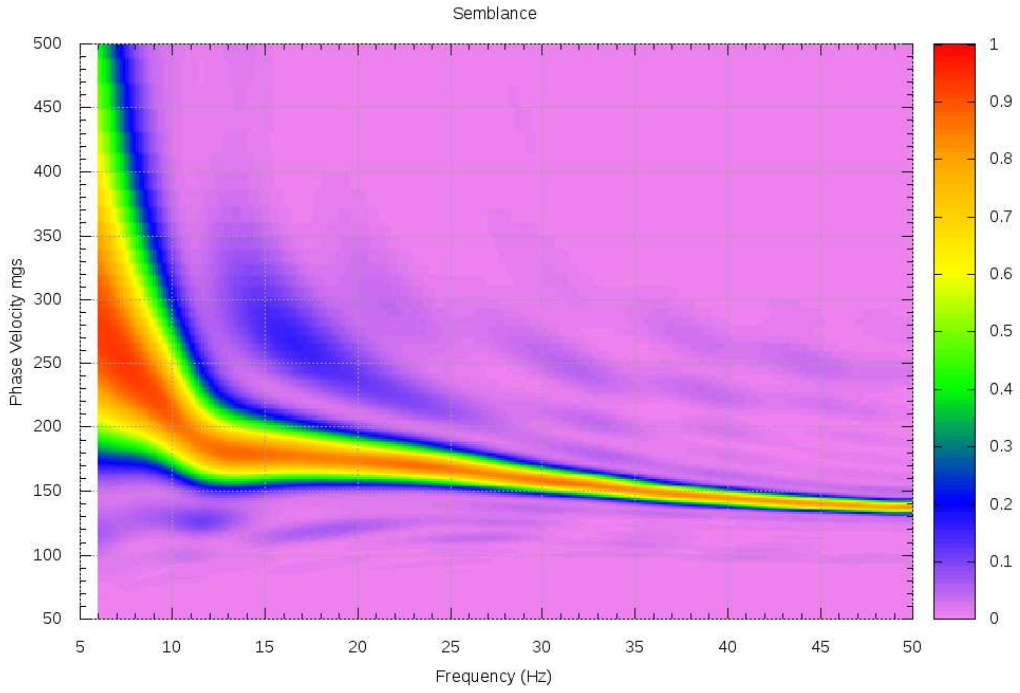


Figure 41: Color plot of semblance for example soil profile of Figure 42. The fundamental mode appears as red. A weaker higher mode is also visible as a lighter shade of blue.

6.11.3 Example Rayleigh Wave Processing: Synthetic Seismogram

A synthetic Rayleigh wave vertical component seismogram was created for the following 1D-layered model.

point	depth (m)	Vs (m/s)	Vp (m/s)	Density
1	0.0	125.	312.	2000.0
2	1.6	153.	382.	2000.0
3	3.2	217.	542.	2000.0
4	6.6	165.	412.	2000.0
5	16.0	387.	967.	2000.0

The above model specifies points of control. Between the control, the elastic moduli are linearly interpolated in steps set by a incremental layer thickness. Here, the step size was 0.1 meters. A graphical display of this model is shown in Figure 42. Note the curved interpolation of velocity when elastic constants are linearly interpolated. One starts the process with program *gendis* which interactively prompts the user for the depths, velocities, and densities. The result is a namelist file, *disper.d*, that is then read by program *disper*. Program *disper* computes the phase velocity dispersion, creating as an output a dispersion file, (earth.crv in this case) and a plot of phase velocity as shown in Figure 43

We compute a synthetic seismogram using the program, *waves*. Program *genwav* prompts the user for parameters like trace spacing, maximum time, component of motion, and produces output file *waves.d*. Program *waves* reads *waves.d* and the dispersion curve saved in output file *earth.crv* from the *disper* run. Here we will use a source moment tensor for a vertical impact source, a spectral band from 1 to 50 Hz (the total available from the earth.crv file). Offsets will range from 1 to 48 meters, with 1 meter trace spacing. The program *waves* employs a minimum phase wavelet with bandwidth reduced somewhat from the range of frequencies specified (to avoid an abrupt transition from the available source spectrum). The source wavelet used in the synthetic seismogram is shown in Figure 44.

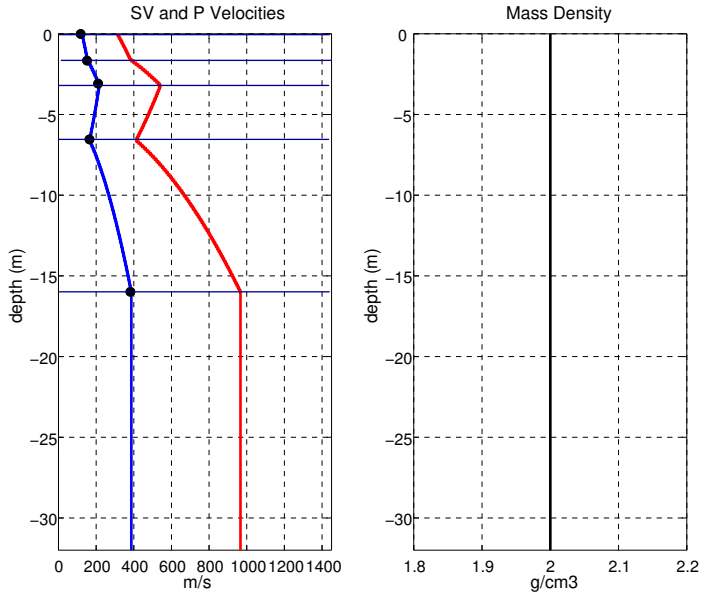


Figure 42: Example Rayleigh wave model with 0.1 meter step interpolation between control. The interpolation is linear in elastic modulus or density. See section 7.3.2 for additional details.

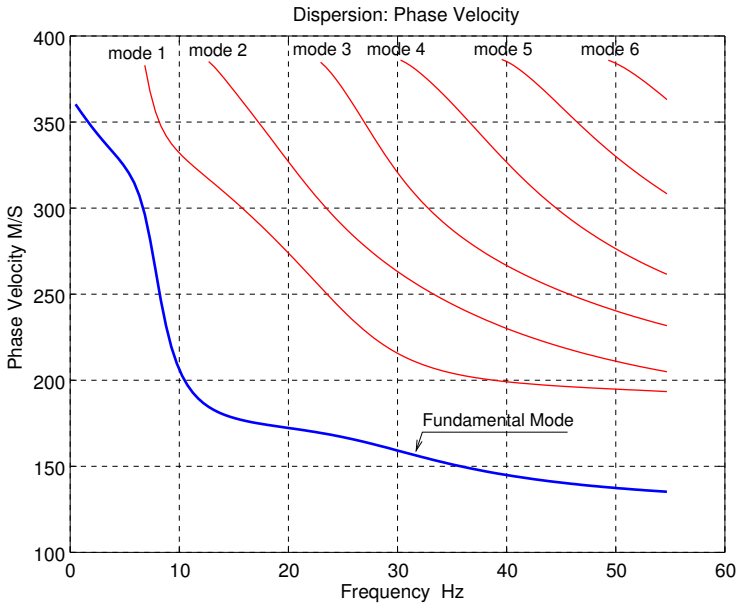


Figure 43: Phase velocity computed by program *disper* for the model of Figure 42.

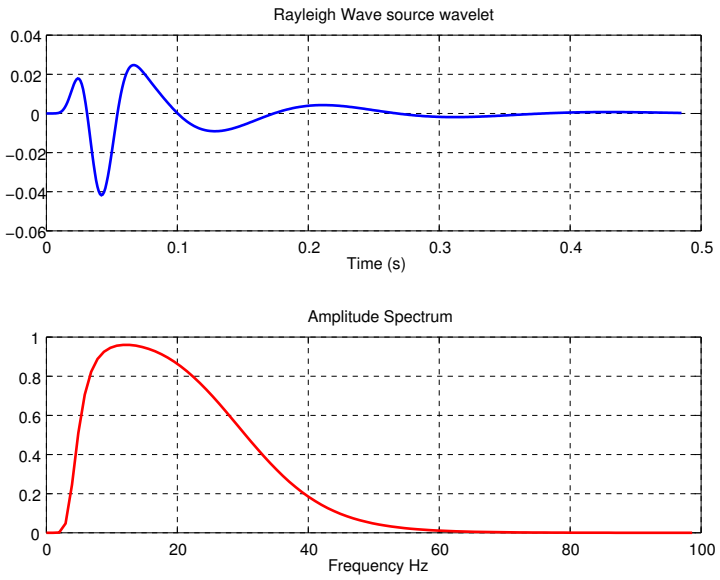


Figure 44: Source wavelet for synthetic Rayleigh wave seismogram, model of Figure 42.

Program *waves* (see section 7.3.6) computes the actual waveforms for a synthetic seismogram. Signals of particle displacement, vertical component, are shown in Figure 45.

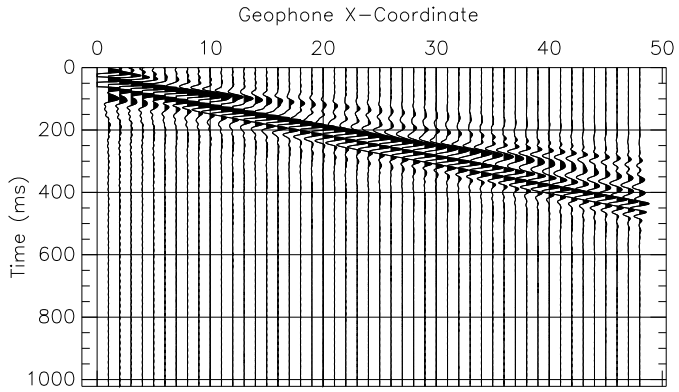


Figure 45: Synthetic vertical component Rayleigh wave seismogram, model of Figure 42. See section 7.3.6 for further details.

6.11.4 Example Rayleigh Wave Processing: Manual Interpretation (FwdR1.m)

The output dispersion file *bvax.his* can be read by program *FwdR1.m* to manually invert the Rayleigh wave dispersion problem. The default install location for the Octave program is `/usr/local/share/octave/site-m`. The initial model sets the number of control points in the model, shear wave velocities, and depth points. The initial model is placed in a text file, for example call it *model.txt*.

Below is an example for a 5 point Vs profile. The first row is just the number of control points, 5 in this case. The next row are the shear wave velocities at each depth point given in the 3rd row. The units are meters/second and meters.

```
5
125. 153. 217. 165. 387.
0 1.6 3.2 6.6 16.
```

The Octave procedure requires Fortran subroutine, *rwv.f* and files *wrapper.cpp*, *build_disper_oct*, to be located in the working directory or path. The Octave program will check to see if the ELF shared object, *disper.oct*, is compiled and in the directory or path. If not, it will automatically compile and link this subroutine which speeds up the computations. The script, *build_disper_oct* must be executable. The program *FwdR1.m* prompts the user with a number of setup GUI's after the initial question for the name of the model text file. The first asks for the compressional velocity setting (constant ratio or fixed value). If you choose constant ration of Vp/Vs, then you are prompted for Poisson Ratio, Grain Density, Porosity and Degree of water saturation. Edit if needed and then click OK. A message will display the Vp/Vs ratio and density that will be used.

If data are present in the form of a *bvax.his* file, those will be plotted blue with error bars and the model dispersion will be plotted in red. One is then prompted to continue or quit. If you continue, click Yes and the current model will be displayed as a list of entry boxes. Make changes as desired to bring the red curve closer to the blue curve. The Nlayer value is displayed, but if you change that, it will be ignored. To change the number of control points, you will need to start fresh with a new *model.txt* file. Once you have a satisfying fit, click No to exit the loop and then save your plots. The title of the Figure 1 plot has the current model. *NOTE: Rayleigh wave dispersion is primarily sensitive to shear velocity, Vs, but compressional velocity Vp and density will have a noticeable effect. If you change the Vp and Density assumption, then you will likely make an adjustment to Vs and depth.*

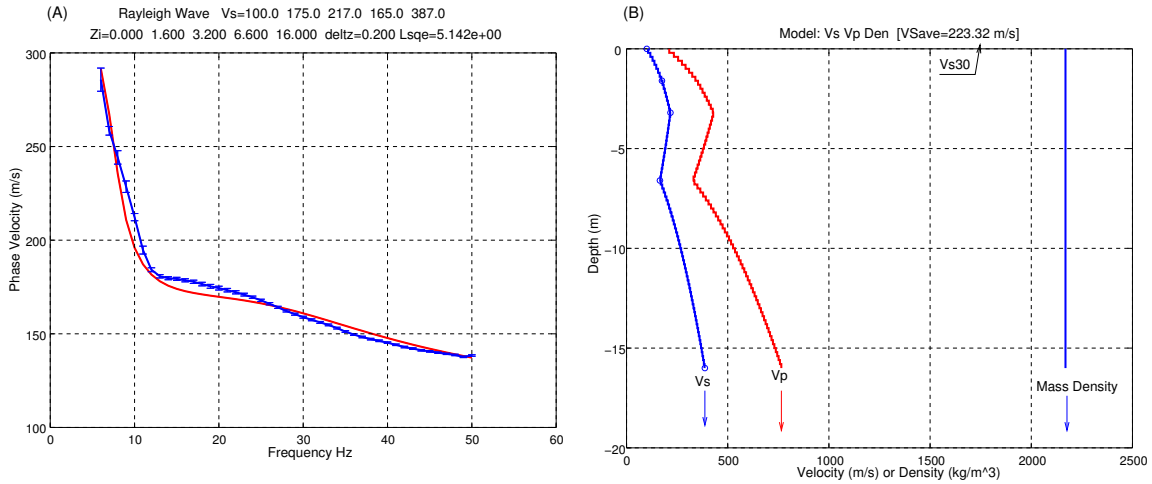


Figure 46: Manual modeling with *FwdR1.m*, final trial (A) dispersion and (B) soil profile. Vs30 is in the title bar of (B) assuming parameters remain constant down to 30 meters.

6.11.5 Example Rayleigh Wave Processing: Automated Inversion (invR1.m)

The program, *invR1.m*, will attempt to determine a layered model which matches a measured phase velocity dispersion profile. As in program, *FwdR1.m*, the dispersion curve is read from a text file, *bvax.his*. The Fortran subroutine, *rwv.f*, must also be in the directory. The inversion ran for 3 iterations. The solution is shown graphically in Figure 47. Only 3 singular values were used (see Figure 48).

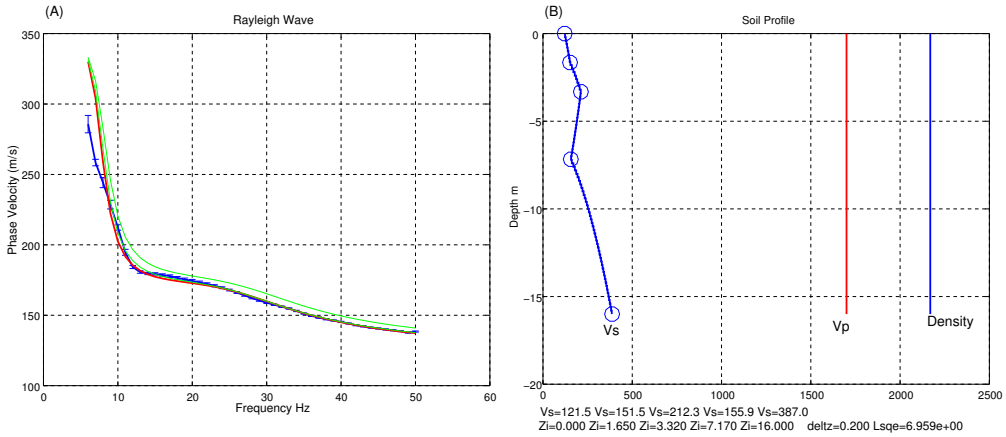


Figure 47: Automated modeling with *invR1.m*. Initial model and intermediate models are shown in cyan. The 3rd, terminating iteration, is shown in red. The fit can be compared to that achieved in Figure 46. The model is shown for the 3rd iteration and is tabulated in the caption of (B). Note that both velocity and depth of control points were free to vary.

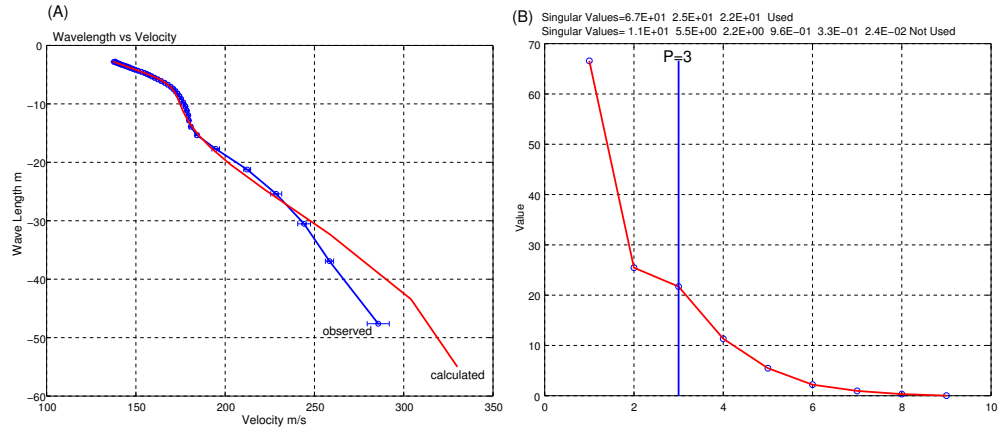


Figure 48: Automated modeling with *invR1.m*. (A) Dispersion as a function of wavelength. (B) Singular values sorted by size. Only the 3 largest singular values were used (P=3).

6.11.6 Spectral Analysis of Surface Waves SASW (SASW.m, saswv.m)

Prior to the multi-channel recording of surface waves, the SASW method was developed to employ just two geophones and spectrum analyzer instrumentation. The method has also be used with geophone signals processed with the Fast Fourier Transform (FFT) [21]. A typical survey involves taking a source effort on both sides of the geophones (only left side shown in Figure 49). Sounding is performed by expanding the geophone and source separation, X, often changing the source so that low frequencies are radiated at larger spacings, X. Geophones are placed symmetrically about the center line for all expansions. In this example, we take the first two traces of our synthetic seismogram shown in Figure 45. Shown in Figure 49 are blow-ups of the first two traces. Octave program, *SASW.m*, is used to compute the phase velocity dispersion.

Program *SASW.m* requires that function files *segyinfo.m* and *bsegin.m* be located in the same directory as the data file and *SASW.m*. Upon execution, the program asks the user to select a *.seg file and two signals from the *.seg data file. The FFT is then computed for both signals, $S_j(f)$ and $S_k(f)$. The cross-spectrum is computed in the frequency domain by multiplying one spectrum with the complex conjugate of the other spectrum:

$$G_{jk}(f) = S_j(f) \cdot \tilde{S}_k(f) \quad . \tag{40}$$

The cross spectrum, $G_{jk}(f)$ is complex. In polar coordinates, the phase is given by $\phi(f)$. A time delay is computed

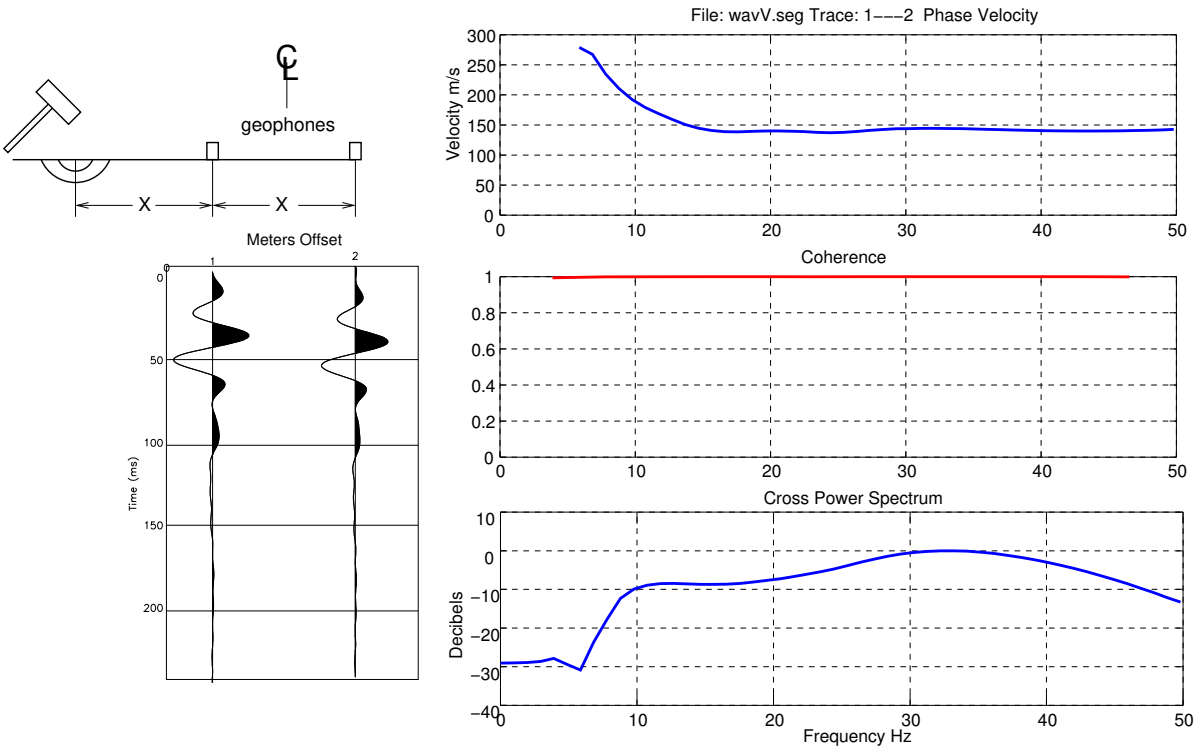


Figure 49: SASW recording places two geophones about a center line. The FFT is used to perform a cross correlation between the two signals in the frequency domain. The phase velocity dispersion curve is computed from the phase of the cross correlation and knowledge of the geophone spacing. Unwrapping of phase is required to compute dispersion beyond the spatial Nyquist frequency.

at each frequency by

$$t = T \cdot \phi(f)/2\pi \quad , \tag{41}$$

where T is the period ($T = 1/f$) for frequency, f. The phase velocity at each frequency, f, is computed by

$$V = \frac{X}{t} \quad , \tag{42}$$

where X is the spacing between the two geophones. The coherence, C, is computed from the ratio of the square of the cross-spectrum amplitude squared to the product of the auto-spectra of each signal,

$$C = \frac{|G_{jk}(f)|^2}{G_{jj}(f) \cdot G_{kk}(f)} \quad . \tag{43}$$

The coherence is computed using the *pwelch* function in Octave. This function divides a signal into a number of overlapping intervals and averages the resulting spectrum (ie. the modified peridogram method).

Some cautions are warranted in using the SASW method. First, it is assumed that only the fundamental mode is in the analysis window. Higher modes will have an adverse affect on the computations. This can be avoided by using sources which radiate low frequencies at large geophone separations. Using a broad-band source at large geophone separations increases the risk that higher modes will enter the analysis window. The dispersion curve computed by *SASW.m* can then be inverted using the Octave program *invR1.m*, as was illustrated in the example above. Thus, *SASW.m* is an alternative to *bvax* for computing a phase velocity dispersion curve.

Program *saswv.m* is for non-impulses sources. A sample data set is included in */usr/local/octave/site-m/dx32f.txt* (T-Rex Shaker data shared in an ASCE Geophysical Engineering Committee project). See file *Dataformat.pdf* in the same directory that documents this included sample.

ll

6.12 Spectral Analysis

There are a number of ways to compute amplitude spectra of signals in BSU. The following are just some examples using both Octave and compiled BSU programs.

6.12.1 Yule-Walker All Pole Spectra

6.12.1.1 Using *yulewalker.m* The Octave program, *yulewalker.m* can be used to read a single seismic trace from a BSEGY file and compute an all pole spectrum. The order of the process is set by the number of terms to include in the computation of an autocorrelation. Figure 50 shows the Geologan down-hole data.

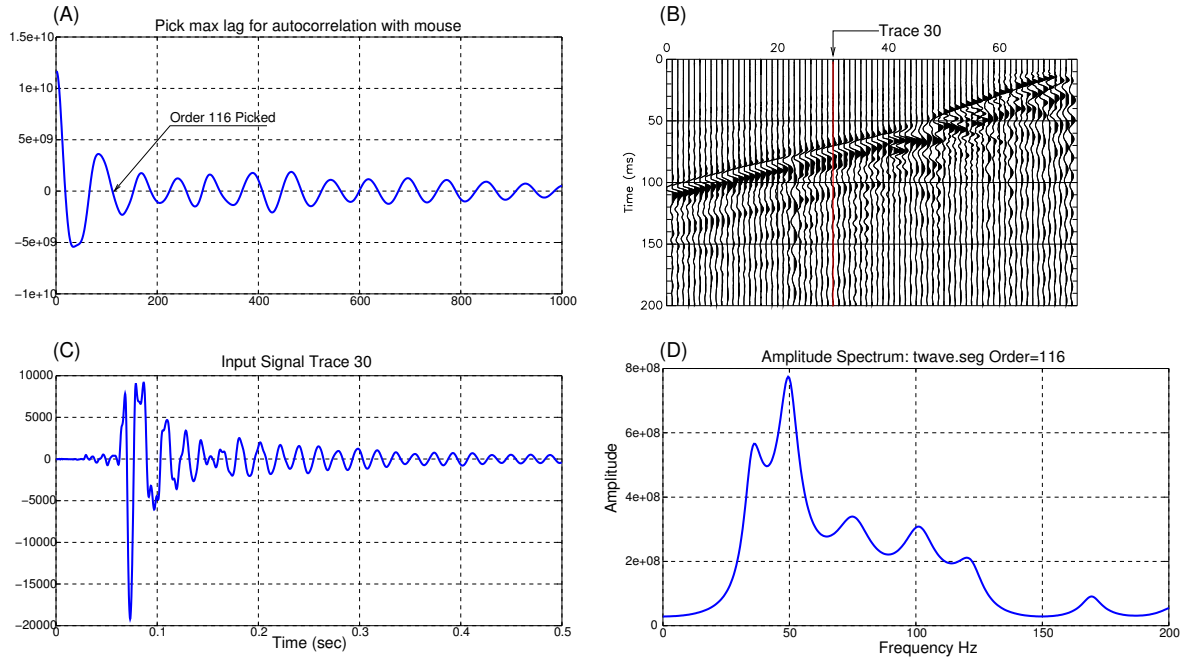


Figure 50: (C). Geologan down-hole data. Octave program *yulewalker.m* is used to select trace 30. (A) Picking a length of the autocorrelation (nlag=116), (B) Downhole data, (C) Selected signal trace 30, (D) Yule Walker all pole spectral estimate.

To run *yulewalker.m*, you need to have the following files in the same directory as the BSEGY data (in this case, a file named twav.seg). Start Octave from within the directory with these files:

- segyinfo.m
- bsegin.m
- yulewalker.m
- twave.seg (or whatever *.seg file you want to analyze)

From within the Octave text window, issue the command,

```
yulewalker;
```

The program will prompt you for the file name in the Octave text window. Here, we enter *twav.seg*. Then a GUI dialog panel will pop up and ask for the following input:

- **Input? Data or Autocorrelation** Is the file a time signal or an autocorrelation?
  A GUI will pop up showing the number of traces, maximum recorded time, and sample interval. Click OK
- **trace number** (the signal we wish to compute a spectrum for)

- **Max Time and Frequency** Entry boxes for calculation and display.
- **Remove DC?** Remove or not to remove DC (zero frequency) content.
- **Pick Order of Spectrum** Click with mouse on autocorrelation.
  Yule Walker spectrum both in amplitude and Decibel displays will pop up.

The program then generates the plots as shown in Figure 50 A, B, C, and D. The Decibel version of the spectrum is not shown here.

6.12.1.2 Using *yulewalker.m* with Autocorrelation Input

The difference between time signals and autocorrelation in section 6.12.1.1 and here is that the input would be an autocorrelation instead of a seismic signal. We use the BSU program *bxcr* to compute an autocorrelation data set in BSEGY format. Fore example, using the same data as above, we issue the following commands from an xterm within the directory with the data set of interest.

```
bxcr twave.seg twave.seg 0.0 0.5 0.1
bstk bxcrtwav.seg
```

The first command cross correlates the data set *twave.seg* with itself (ie. an autocorrelation). The gate is 0 to 0.5 seconds, for a maximum correlation lag of 0.1 seconds. This is followed by a stacking or averaging of all the autocorrelations into a single signal with program *bstk*. This averaged autocorrelation is then replicated so that the input and output BSEGY files have the same number of traces. In generating a spectrum, it does not matter which one of the traces from file *bstkbxcr.seg* we use. They are all the same. In Figure 51 we again use trace 30, but could have used any other equally well.

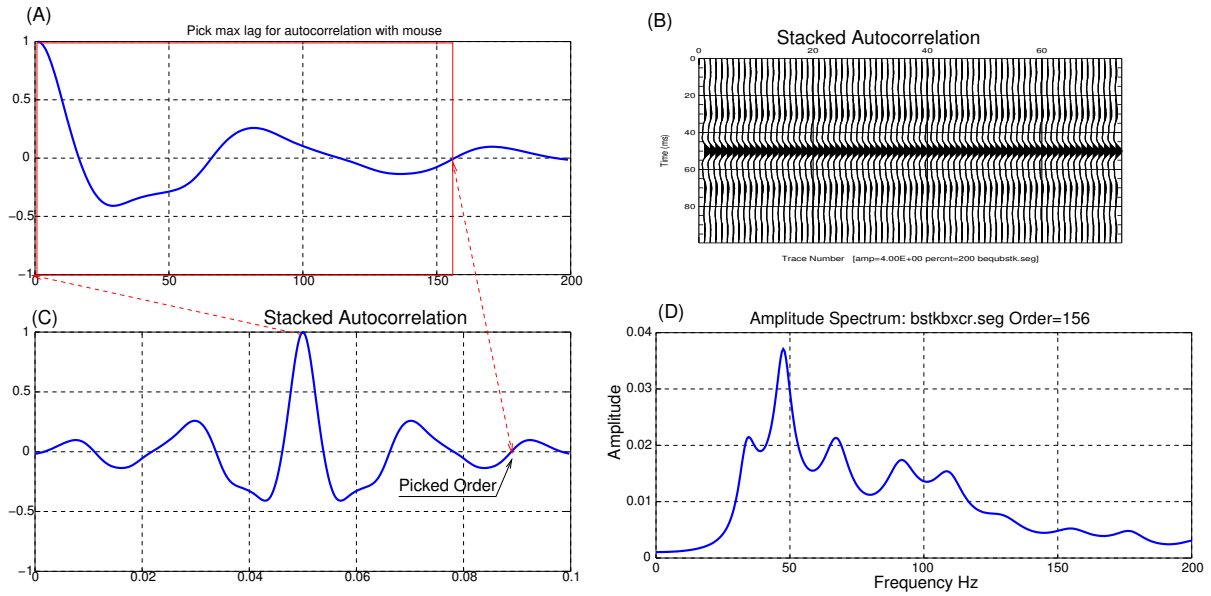


Figure 51: (A) Picked portion of autocorrelation. Sets spectrum order at 156. (B) Input file from bstk of bxcr. (C) Plot of the selected trace 30. (D) All pole amplitude spectrum.

In both of these examples of Yule-Walker spectra, the Octave programs are computing amplitude spectra (not power spectra). The square root is taken of the power spectrum before plotting.

7 Seismic Modeling with BSU

7.1 Solution to Lamb's Problem (*lamb*)

The BSU distribution includes some synthetic seismogram capabilities. The solution of Lamb's problem (Lamb [6]) is computed by *lamb*. The program, *lamb*, is a recreation of a program originally described by Mooney [15] .

The algorithm closely follows that described in Mooney's paper, with the exception that an autoregressive operator is used to generate a minimum phase wavelet to filter the data (Mooney convolved with a zero phase wavelet). Mooney's work cites an earlier solution as the basis for his program (Pekeris [16]), and that paper is well worth examining for clarification on any points not entirely clear in Mooney's publication. The current BSU version only implements the case for a Poisson's ratio of $\frac{1}{4}$. Modification to other cases is relatively straight forward, and will probably be done at some point in the future.

Briefly, Lamb's problem is the solution for the inhomogeneous waves that travel on the surface of an elastic half-space due to a vertical, point impact source. Lamb referred to these waves as major and minor tremors. Today, we would call these the Rayleigh (major), P- and S-waves (minor). A test of the *lamb* program was to recreate previously published solutions plotted in unit less time. The unit less time coordinate is given by

$$\tau = \frac{V_s \cdot t}{R}, \tag{44}$$

where R is the range (m) from the source, t is time (s), and V_s is the shear wave velocity (m/s). This solution is shown in Figure 52, and compares favorably with previously published solutions (Mooney [15] figure2; Pekeris [16] figures 3 and 4; and Richards [18] figures 2 and 3, integrals I_3 and I_4 respectively). The GSL library provides the necessary elliptical integrals.

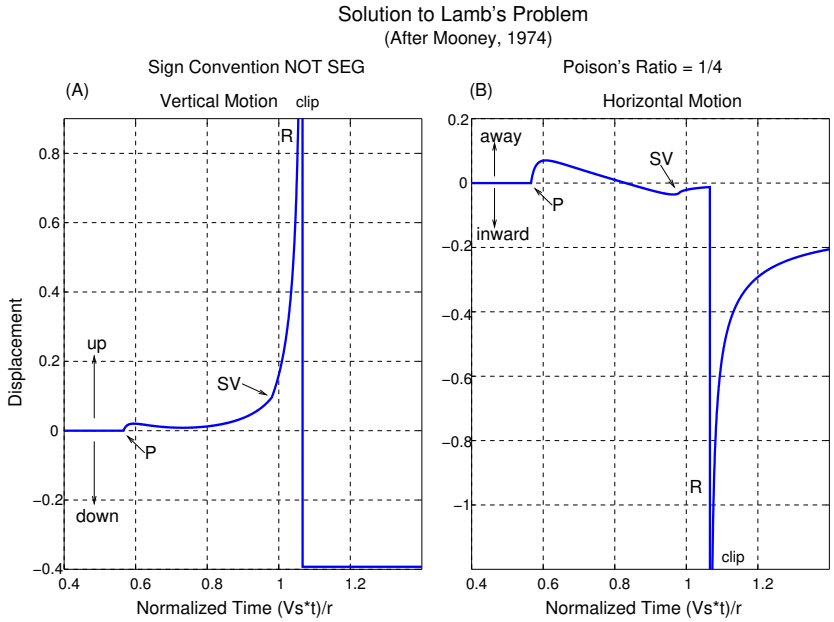


Figure 52: Solution to Lamb's Problem (after Mooney, 1974 [15]). Step function source.

7.1.1 Running Program *lamb*

This program has 14 input arguments. The user is encouraged to review the man page and the online documentation for *lamb*. For the man page, type

```
man lamb
```

The online help is obtained by typing the following in an xterm,

```
lamb -h
```

.

Most arguments are straight forward, and obviously necessary to computing a synthetic seismogram. Since the code only permits a Poisson's ratio of $\frac{1}{4}$, the program only asks for a S-wave velocity, V_s . No P-wave velocity can be entered since this limitation sets $\frac{V_p}{V_s} = \sqrt{3}$. The input argument, *itype*, is used to select the type of signal which will be output.

7.1.1.1 The *itype* argument in *lamb*.

- itype=1: The vertical and horizontal ground displacement will be computed and output in files *lambv.seg* and *lambh.seg*, respectively. The source function is a displacement step function.
- itype=2: This may be viewed in either of two ways. The derivative of a step function is an impulse function, and itype=2 is the derivative with respect to time of the itype=1 case. Thus, this may be viewed as the ground displacement for an impulse source function. Alternatively, this can also be viewed as the ground particle velocity for a source step function.
- itype=3: The source function is a single degree of freedom dampened resonator (exponentially decaying sinusoid wavelet, ground displacement function). The source is described by the *sfreq* and *sdamp* parameters. The output signals are ground displacements.
- itype=4: The source function is the same as itype=3, but the signals have been differentiated with respect to time. Thus, the output signals may be viewed as ground particle velocity for the itype=3 source.
- itype=5: This option switches the output signal from the actual ground motion to the motion of a geophone element (which is modeled as a single degree of freedom dampened oscillator bonded to the half-space at the surface). The geophone is described by the *gfreq* and *gdamp* parameters.
- itype=6: The itype=5 option with the output signal differentiated with respect to time. The output is the geophone element velocity. This case corresponds to the voltage that would be recorded by a velocity geophone.
- itype=7: The output signal is the source displacement function corresponding to itype=3, at the point of source application. If you have run a case with itype=3, this is the source wavelet.
- itype=8: The particle velocity of the source function at the point of application. This corresponds to itype=4 source wavelet.
- itype=9: This is like itype=7, with a geophone bonded at the source point. This is the displacement source wavelet as seen from the motion of a modeled geophone element.
- itype=10: This is like itype=8, with a geophone bonded at the source point. This is the particle velocity of the source wavelet as seen from the particle velocity of a geophone element bonded to the source point.

7.1.1.2 The *pol* argument in *lamb*. This selects a sign convention. In addition, if *pol*=0 (Test Mode), then the 1/R amplitude decay is also removed (as was the case in computing Figure 52). If *pol*=-1, then an upward motion produces a negative value (the SEG sign convention). Setting *pol*=+1 reverses the sign convention (upward motion produces a positive number). Interpreting *pol* for horizontal motion is much more confusing. In real life, one must keep track of the direction to the source, and the relative orientation of a geophone element, and how it is wired. Basically, if *pol=0* or *pol=+1*, then the sign convention of the motion is as shown in Figure 52. If *pol=-1*, then it is the reverse of what is shown in Figure 52. In that figure, the word "away" means motion away from the source location (in the direction of increasing distance away from the source).

7.1.1.3 The *stab* argument in *lamb*. There really is no need for a stability factor, unless you are running itype=8 which takes the derivative right at the source point, R=0. With all other itype values, you can set *stab*=0. The derivative operator used in *lamb* is a bilinear transform realization of the derivative with respect to time. In terms of the Z-transform,

$$Y(z) = \frac{2}{\Delta t} \cdot \frac{(1-z)}{((1+stab)+z)} \cdot X(z), \tag{45}$$

where Y(z) is the output signal Z-transform, and X(z) is the input signal Z-transform. The *stab* factor nudges the pole at the Nyquist frequency slightly off the unit circle to prevent extreme blow up of any Nyquist amplitudes in the input signal. The derivative is realized by an ARMA operator. The formula which realizes equation (45) is

$$y_j = \frac{\frac{2}{\Delta t}\left(x_j - x_{j-1}\right) - y_{j-1}}{(1+stab)}, \tag{46}$$

where y_j is the j-th sample corresponding to the temporal derivative of x_j , the j-th sample of the input signal being differentiated.

7.1.1.4 Examples of *lamb* The signals shown in Figure 52 were computed by lamb in Test Mode (*pol*=0). The command to generate the Figure 18 signals is

```
lamb 150.  1.0 1 2.  .0005 150.  1700.  1 100.  .3 100.  .3 0 0.
```

Not all of the parameters matter (density won't affect the result since it is test mode). There are many combinations of S-wave velocity and range that will work (see equation (44) for normalized time scale used here). You can generate the plots of Figure 52 to confirm correct operation of the code. This is done in Octave, execute the procedure *traplt.m* from within the Octave text window. Make sure you also have the functions *bsegin.m and segyinfo.m,* in the working directory as they are called by *traplt.m* . Also note that you will have to specify the axes limits to clip the signals as was done in Figure 52. After finishing execution, delete the second figure of the spectrum and then issue an axis command.

```
axis([0.4,1.4,-0.4,0.9]) % for the vertical motion
axis([0.4, 1.4, -1.2, 0.2]) % for the horizontal motion
```

As a second example of what *lamb* can do, we can generate 48 channel records for vertical and inline radial motion. The half-space medium has a mass density of 1700 $\frac{kg}{m^3}$, a $V_s = 250$ m/s and $V_p = 433$ m/s. The command generating the synthetic seismograms is

```
lamb 1.  1.0 48 .5 .0005 250.  1700.  6 50.  .7 10.  .7 -1 0.
```

Figure 53 shows the resulting synthetics. Note that this is an elastic medium assumption. **Actual soils may differ significantly from an elastic assumption**. So, take it for what it is. The data have been rescaled to show waveforms. Each signal has been scaled by the L2 norm for that trace, thus removing the fading of amplitude with increasing offset.

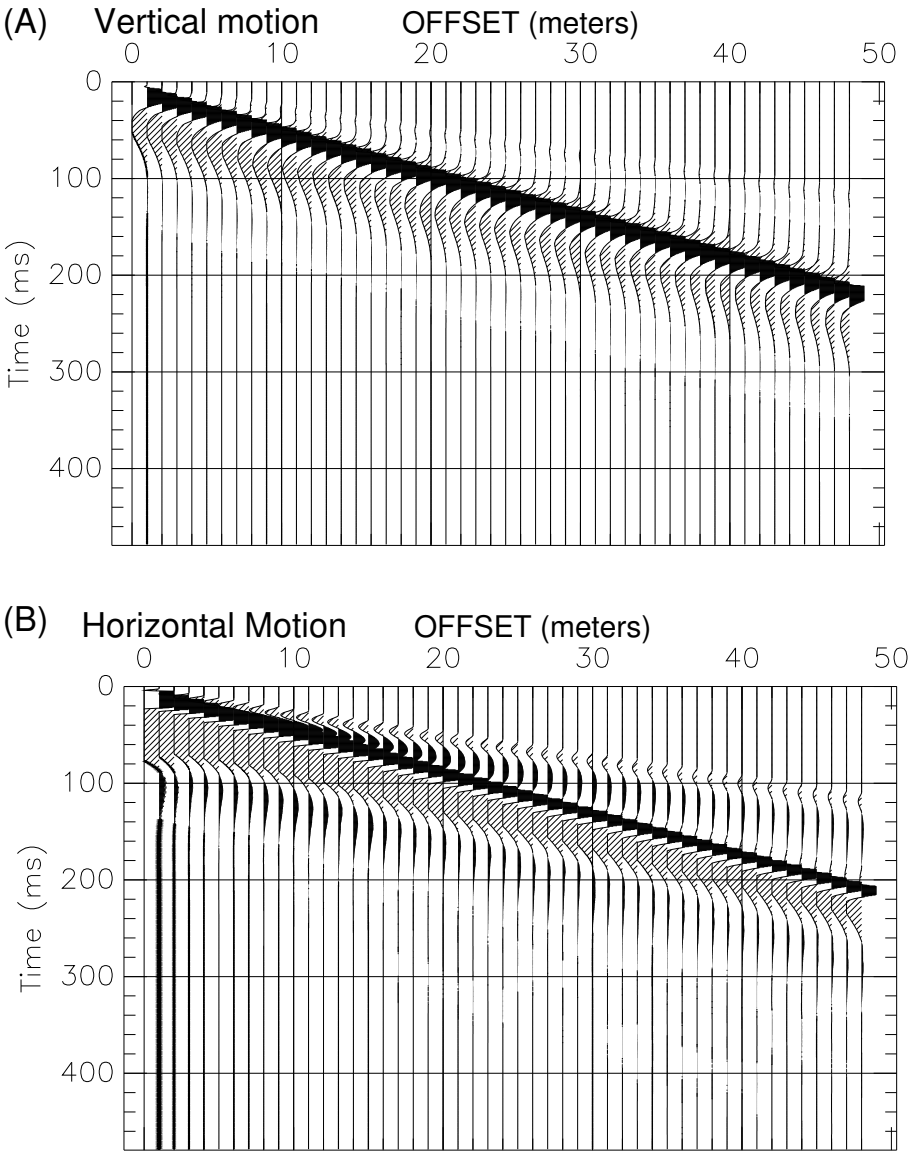


Figure 53: Synthetic seismograms generated by *lamb* (see text for model)

7.2 Elastodynamic Solution Near and Far Field (*bnfd*)

The program *bnfd* computes synthetic seismograms for a point source in a whole space. While limited in terms of practical applications (most problems of interest are in more complicated media), the solution does provide insight into radiation patterns and issues related to the transition from near to far field. The computation is taken directly from Aki and Richards [1] (equation 4.23, page 73). The program has 9 command line arguments which are documented in the man pages and the online help. Users are encouraged to review the documentation. The man page is viewed by typing

```
man bnfd
```

and the online help is viewed with the command,

```
bnfd -h
```

.

The program allows the user to restrict the terms included in the computation, as well as select a specific component of motion to display. The source is represented by a single equivalent force applied within the medium. The waves are computed relative to that point where the source is applied. The wavelet (scalar source moment) is an exponentially decaying sinusoid

$$X_o(t) = \exp(-\alpha t) \cdot \sin(2\pi f_c), \tag{47}$$

where f_c is the center frequency of the source spectrum. The near-field integral is evaluated by Simpson's rule. For the convenience of the reader, equation 4.23 is repeated below

$$u_i(x,t) = \begin{array}{c} \frac{1}{4\pi\rho}(3\gamma_i\gamma_j - \delta_{ij})\frac{1}{r^3}\int_{\frac{r}{\alpha}}^{\frac{r}{\beta}} \tau X_o(t-\tau)\,d\tau \\ +\frac{1}{4\pi\rho\alpha^2}\gamma_i\gamma_j\frac{1}{r}X_o\left(t-\frac{r}{\alpha}\right) \\ -\frac{1}{4\pi\beta^2}(\gamma_i\gamma_j - \delta_{ij})\frac{1}{r}X_o\left(t-\frac{r}{\beta}\right) \end{array}, \tag{48}$$

where the P-wave velocity is α , the S-wave velocity is β , the distance from the source point to the observer is r, the direction cosines are the $\gamma_i = \frac{x_i}{r}$, density is ρ , time is t, particle displacement in the i-direction is u_i , x is the position vector, and δ_{ij} is the delta function. The first term in equation (48) is the near-field term, the second, the far-field P-wave motion, and the last, the far-field S-wave.

7.2.0.1 Example of *bnfd* The program *bnfd* uses a design BSEGY data file to set up the details of the synthetic data set (number of traces, sample interval, layout of source relative to receivers, etc). In this example, we will use the *lambv.seg* file from the second *lamb* example (**section 7.1.1.4** above). It does not matter what components are specified in the design file, since this will be ignored in *bnfd* (which uses command line arguments to specify the source and receiver polarizations). The sample shown in Figure 54 is for a horizontal force (in x_1 direction) and observer stations located at the same elevation as the source, extending in a line out to 48 meters. The motion captured at the observer point is also in the x_1 direction (we would not expect any motion in the other directions). The command used to generate this sample is

```
bnfd lambv.seg 1 433.  250.  1700.  100.  50.  1 7
```

The material properties are the same as for the Lamb's problem example. However, this solution is in a whole space, and we are observing the computation of body waves, free from any boundary.

The near field dominates at the near offsets, and then declines in amplitude with increasing offset. The listing file produced by *bnfd* (in this case *bnfdlamb.lst*) provides a listing of the relative amplitudes. The S-wave amplitude is zero in this case, and we are only looking at P- and Near-field waves. A sample of the amplitudes taken from the listing are:

- Offset=1 meter, Near Field=9.362e-05 P-wave= 2.497e-10
- Offset=24 meters, Near Field=6.772e-09 P-wave=1.040e-11
- Offset=48 meters, Near Field=8.465e-10 P-wave= 5.201e-12

So we can see that we have not yet gone far enough out for the P-wave to match the near field wave amplitude. The source wavelet in this case had a center frequency of 50 Hz, and a decay rate of 100 per second.

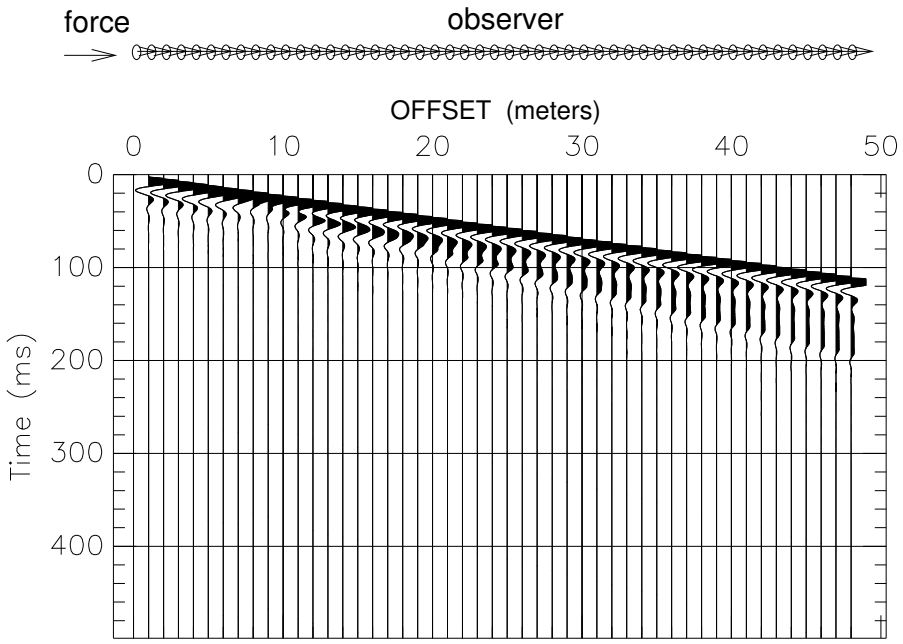


Figure 54: Near and Far Field computations (source in x1, motion in x1 directions). The data have been trace qualized by the L2 norm of each offset signal to prevent fading of the motion due to amplitude decay.

7.3 Elastic Rayleigh Wave Modeling

7.3.1 Program halfsp

The basic computation is for a Rayleigh wave in a half-space medium. The motion-stress vectors for this type of problem are easily computed with the BSU program *halfsp*. Man pages for this program can be viewed in an x-term with the command,

```
man halfsp
```

This program is very basic with only a man page for documentation. The output is a text file listing the half-space properties at the top followed by 4 columns of the motion stress vector values with depth. To read more on this topic, see Aki and Richards ([1], chapter 7). The program is run from an x-terminal with a single command:

```
halsfp
```

The user is then prompted for the half-space properties (density, P-velocity, S-velocity), and the depth sampling for the motion-stress vector computations. A typical dialog is as follows:

```
pm@penguin:  halfsp
ENTER RHO,VP,VS
1600,800,100
ENTER FREQ,NZ,Z0,ZEND
40,50,0,20
PHASE VEL= 95.4331
```

This produces a file, halfsp.tmp, a portion of which is shown below:

```
 HALFSP.F OUTPUT:
RHO=0.1600E+04
MU=0.1600E+08
LAME=0.9920E+09
FREQ=0.4000E+02
P-WAVE VELOCITY=0.8000E+03
S-WAVE VELOCITY=0.1000E+03
RAYLEIGH WAVE PHASE VEL=   95.4331
```

```
R1=Horiz. Displacement R2=Vertical Displacement
R3=Horiz. Stress       R4=Vertical Stress


  DEPTH      R1         R2         R3         R4

    0.0  0.120E+01  -.219E+01  0.804E+02  0.000E+00
    0.4  -.122E+00  -.259E+01  0.834E+08  0.458E+08
    0.8  -.439E+00  -.224E+01  0.902E+08  0.495E+08
    1.2  -.444E+00  -.175E+01  0.762E+08  0.418E+08
    1.6  -.367E+00  -.132E+01  0.592E+08  0.325E+08
    2.0  -.283E+00  -.981E+00  0.445E+08  0.244E+08
    2.4  -.212E+00  -.722E+00  0.329E+08  0.181E+08
    2.8  -.157E+00  -.529E+00  0.242E+08  0.133E+08
    3.2  -.115E+00  -.387E+00  0.177E+08  0.972E+07
    3.6  -.842E-01  -.282E+00  0.130E+08  0.711E+07
    4.0  -.616E-01  -.206E+00  0.946E+07  0.519E+07
    4.4  -.450E-01  -.151E+00  0.691E+07  0.379E+07
    4.8  -.328E-01  -.110E+00  0.505E+07  0.277E+07
    5.2  -.240E-01  -.803E-01  0.368E+07  0.202E+07
    5.6  -.175E-01  -.586E-01  0.269E+07  0.148E+07
    6.0  -.128E-01  -.428E-01  0.196E+07  0.108E+07
    6.4  -.933E-02  -.312E-01  0.143E+07  0.786E+06
    6.8  -.681E-02  -.228E-01  0.105E+07  0.574E+06
    7.2  -.497E-02  -.166E-01  0.764E+06  0.419E+06
    7.6  -.363E-02  -.121E-01  0.558E+06  0.306E+06
    8.0  -.265E-02  -.887E-02  0.407E+06  0.223E+06
    .
    .
    .
   18.0  -.101E-05  -.340E-05  0.156E+03  0.855E+02
   18.4  -.740E-06  -.248E-05  0.114E+03  0.624E+02
   18.8  -.541E-06  -.181E-05  0.830E+02  0.456E+02
   19.2  -.395E-06  -.132E-05  0.606E+02  0.333E+02
   19.6  -.288E-06  -.964E-06  0.443E+02  0.243E+02
   20.0  -.210E-06  -.704E-06  0.323E+02  0.177E+02
```

7.3.2 Rayleigh Wave Dispersion (programs gendis and disper)

The program *disper* computes Rayleigh wave dispersion curves which can then be input to program *waves* for synthetic seismogram computation. The computation is taken directly from Aki and Richards [1] (chapter 7 on surface waves). Because there are a large number of input variables, the program *disper* uses a file with namelists. Since velocity and density are easier values to input than elastic moduli, a helper program is also available, *gendis*. Man pages exist for both *disper* and *gendis*. From an x-term, type

```
man gendis
```

or

```
man disper
```

.

There is no online help as with other BSU programs. That is **DO NOT TYPE**:

```
disper -h
```

All this will do is create an empty file named *-h*, and that will be difficult to remove unless you know how to do it (the problem is the dash). *If you have already made this mistake, you can remove the empty file as follows:*

```
rm ./-h
```

7.3.2.1 Two ways to run disper Program *disper* is usually run to compute Rayleigh wave dispersion for a layered earth model. The resulting dispersion curves are written to a file which can be read by program *waves* to compute a synthetic seismogram. There is also a listing file, *disper.tmp* which can be examined for phase velocities computed at each frequency, for all possible modes up to 9 modes. Octave programs are also generated to plot the phase velocity dispersion, and even the layered earth model.

The other way to run disper is to examine the above type of run listing file, and choose a phase velocity at a frequency of interest for a mode of interest. These values can be inserted into the *disper.d* namelist file for

computation of the motion-stress vectors. In that case, a Octave program will be generated for plotting the vectors. See section 7.3.5.1 below.

7.3.3 gendis

The program is invoked from the command line

```
gendis
```

You will be prompted for the following inputs:

1. **Name of output file:** This is the file which will be read by *disper*. Recommend using *disper.d* for a name. Limit to 40 characters.

2. **Sample rate:** Actually, not really a rate, this is the sample interval in seconds when following *disper* with a synthetic seismogram computation in *waves*. It is relevant to the frequency step size (see next parameter).

3. **Tmax:** This is the maximum time that will be used in the *waves* synthetic seismogram program. It will affect the step size in the frequency domain, and will be adjusted slightly so that a radix 2 FFT can be employed. That is, in the time domain, the number of samples will be a power of 2. The frequency increment will be $\Delta f = \frac{1}{N\Delta t}$, where N=number of samples, and Δt=the sample interval specified by the above parameter. Also, if you plan on computing a synthetic seismogram, you probably don't want FFT wrap a round. Thus, consider your slowest velocity in your model, and the maximum offset of a synthetic which will be computed in *waves*. Use a safety factor of 2 (make tmax 2 times larger than the latest arrival on the far offset). What ever you decide here, USE THE SAME VALUE LATER IN WAVES.

4. **Minimum frequency:** This is the minimum in Hz. Recommended value is 1, even if you have a geophone with a higher cut-off frequency in mind. Choose this and the next parameter to give a wider bandwidth than you think you will need (unless you are not going to compute a synthetic, and only want a plot of the dispersion curves). The reason is that the *waves* program computes a scaler source moment (wavelet) based on this and the next parameter. The wavelet will be minimum phase. *Too narrow a bandwidth will give a very ringy synthetic seismogram.*

5. **Maximum frequency:** Specify in Hz also. Too high a value may lead to an unstable computation, where too high depends on the model parameters and the inherent difficulty in integrating a stiff equation. Too low a value will lead to a ringy wavelet if you go on to compute a synthetic with program *waves*.

6. **Maximum mode number, modemx:** This is the maximum mode to compute. The limit is 9 modes. For a fundamental only computation, set to 1.

7. **Step size in depth, deltz:** Since the propagator matrix method is used, there is little value in using too small a step size in computing the motion-stress vectors. Too small is defined as being a lot of computations in each layer. For a layer over a half-space, use the upper layer thickness. For a many layered case, use the smallest layer thickness. The integration is automatically adjusted to layer boundaries.

   There is an option in *disper* to compute the motion-stress vectors at a single frequency and plot these with depth. If you are doing that, then a small step size would be appropriate. This type of computation is usually done after a dispersion curve computation, since you will need to provide a phase velocity for the specific frequency of interest.

8. **Number of control points:** This is the number of points in depth where velocities of P-wave, S-wave and mass density are specified. Linear interpolation based on the choice of depth step size above will create a layered model. To create layers larger than the step size, more control points are needed. For example, consider specifying a single thick layer over a half-space. This would require 3 points (as shown in the *gendis* manpage). One always starts at depth 0, surface of the earth. For a thick layer over a half-space, the second control point would be at the top of the next layer, and have material properties equal to the first control point. The third point would be a very small distance below the second control point, and give the half-space properties. See the man pages for this example, or later sections below.

9. **Layer properties:** Actually, these are comma separated quadruplet entries giving the shear velocity (beta), P-wave velocity (alpha), mass density (rho), and control depth (tops of layers). You will be prompted until all the control points have been entered (it loops, and you can't go back).

7.3.3.1 gendis The program gendis is run interactively from an X-terminal. The following is an example and corresponds to the soil profile shown in figure 55:

```
   enter: name of output file ( < 40 char)
disper.d
   enter: sample rate
.001
   enter: tmax for trace
2.0
   enter: minimum frequency
1
   enter: maximum frequency
150.
   enter: maximum mode #
9
   enter: deltz step size
2.
   enter: number of control
3
  layer( 1) enter: beta,alpha,rho,depth(top)
100, 800, 1600, 0
  layer( 2) enter: beta,alpha,rho,depth(top)
100, 800, 1600, 2
  layer( 3) enter: beta,alpha,rho,depth(top)
400, 2000, 1700, 2.001
 twice npts= 4096
 twice tmax=    4.0960
  output====>disper.d
```

7.3.4 showmdl

To illustrate programs in this section, we will continue with the *gendis* example. Again, a graphical sketch of that simple layer over a half space model is shown in Figure 55.

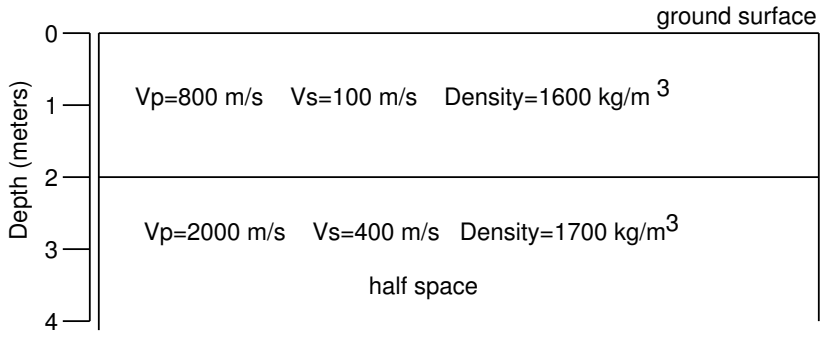


Figure 55: Simple layer over a half space model used in the *gendis* man page.

After you have created a namelist file with *gendis*, you may want to confirm that you have entered the correct model. Let's assume the file name is *disper.d*. The model illustrated in Figure 55 has been used in this example. From the command line, you would type:

```
showmdl disper.d
```

This would produce the following listing. The last 3 lines are not part of the *disper.d* file, but echo back what the Lame's constant and shear modulus compute to in terms of the velocities and densities.

```
pm@penguin:~$ showmdl disper.d
file: disper.d
       &disper
       nlay=    3,
     rho= 0.1600E+04, 0.1600E+04, 0.1700E+04,
     mu= 0.1600E+08, 0.1600E+08, 0.2720E+09,
     lame= 0.9920E+09, 0.9920E+09, 0.6256E+10,
     zi= 0.0000E+00, 0.2000E+01, 0.2001E+01,
       deltz=    2.0000,
       modemx=9,
       nfreq=610, flo=  0.1000000E+01, delf= 0.24414061E+00,   jsmax=300, ksw=0
        pvlcty=0.0, pfreq=0.0, zend=100.0,
       ofile='disper.tmp',
       octav1='phase.m', octav2='mat2.m',
       curve='earth.crv', /
      trimmed deltz=   2.00100


     point       depth       beta      alpha      rho

         1        0.000      100.00      800.00   1600.0
         2        2.000      100.00      800.00   1600.0
         3        2.001      400.00     2000.00   1700.0
```

7.3.5 disper

We run *disper* with the following command,

```
disper disper.d
```

The output includes a file, *disper.tmp*, which lists all the phase velocities for all the modes in the frequency range requested. In this example, the name of the dispersion curve file is *earth.crv*, and a Octave file is generated, *phase.m* which can be used to plot the dispersion curves. Start a Octave session and type the following in the Octave text window:

```
phase;
```

The resulting plot is shown here in Figure 56. The modes have been annotated in Xfig.

7.3.5.1 Editing the namelist file, disper.d You can edit the *disper.d* file to make changes, should you wish to change something, or run a motion-stress vector plot. The *showmdl* listing augments the contents of the *disper.d* file with the layer control points at the bottom of the listing. The shear modulus (mu) and Lame's constant (lame) are computed from your velocities and densities.

Note that there are some additional parameters set to zero, *pvlcty* and *pfreq*. Also, parameter *zend* is set to 100 meters maximum depth, arbitrarily. If the first two are set to zero, then a dispersion curve is calculated over the specified frequency range. You can compute motion-stress vectors at a single frequency by assigning the correct values to these two parameters. When the namelist file is input to *disper*, the computed dispersion curve is captured in a listing, *disper.tmp*, as well as in a phase velocity Octave program, output as *phase.m* by default (unless you change it by editing *disper.d*).

You compute a dispersion curve by executing the following command at the command line:

```
disper disper.d
```

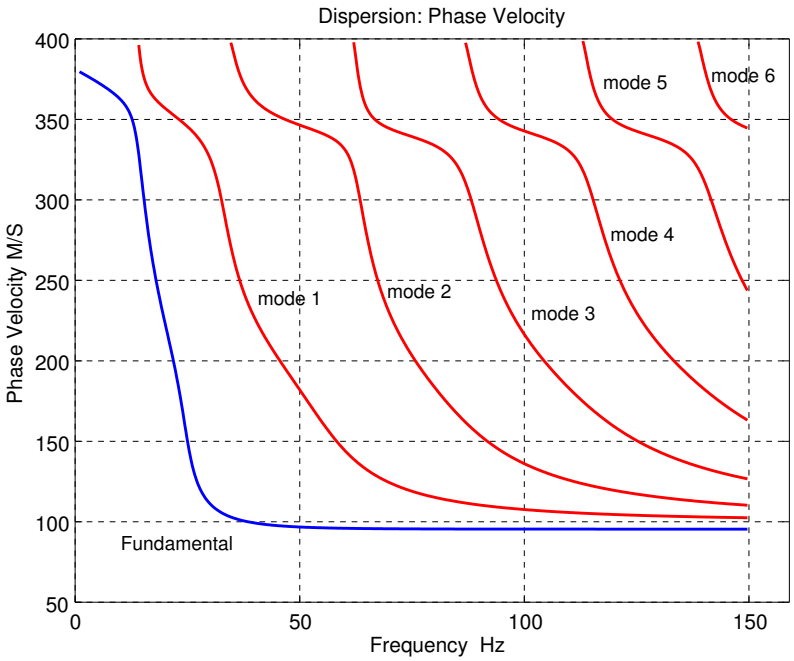


Figure 56: Phase velocity curves computed for model in Figure 55.

Examination of the output listing file, *disper.tmp* at a frequency of 40.0390600 Hz will show the following:

```
 40.0390600 | 99.2208411 226.1749195 361.9392410
```

On this line of output, we see that there are 3 modes possible at this frequency. The phase velocity of the fundamental mode is 99.2208411 m/s. The highest mode has a phase velocity of 361.9392410 m/s.

We copy *disper.d* to *disper1.d* and edit the copied file to compute the fundamental mode motion-stress vectors. The edited file might look like this:

```
   &disper
  nlay=    3,
rho= 0.1600E+04, 0.1600E+04, 0.1700E+04,
mu= 0.1600E+08, 0.1600E+08, 0.2720E+09,
lame= 0.9920E+09, 0.9920E+09, 0.6256E+10,
zi= 0.0000E+00, 0.2000E+01, 0.2001E+01,
  deltz=   0.0500,
  modemx=9,
  nfreq=610, flo=  0.1000000E+01, delf= 0.24414061E+00,   jsmax=300, ksw=0,
   pvlcty=99.2208411   , pfreq=40.0390600, zend=5.,
  ofile='disper.tmp',
  octav1='phase.m', octav2='mat2.m',
  curve='earth.crv', /
```

Note that the line beginning with *pvlcy* has been edited, and the *deltz* value has been decreased to make a better sampled motion-stress vector plot. With these changes, we re-run disper with this file, and the Octave program file, *mat2.m* is created instead of *phase.m*. We can execute that program by starting a Octave session and executing *mat2.m* . From within Octave, we type:

```
mat2;
```

Figure 57 shows the plot generated by Octave and the *mat2.m* program. There have been some additional annotations drafted to show key points (vanishing stress boundary condition at the surface, top of half space, and identification of the vertical and horizontal components of motion).

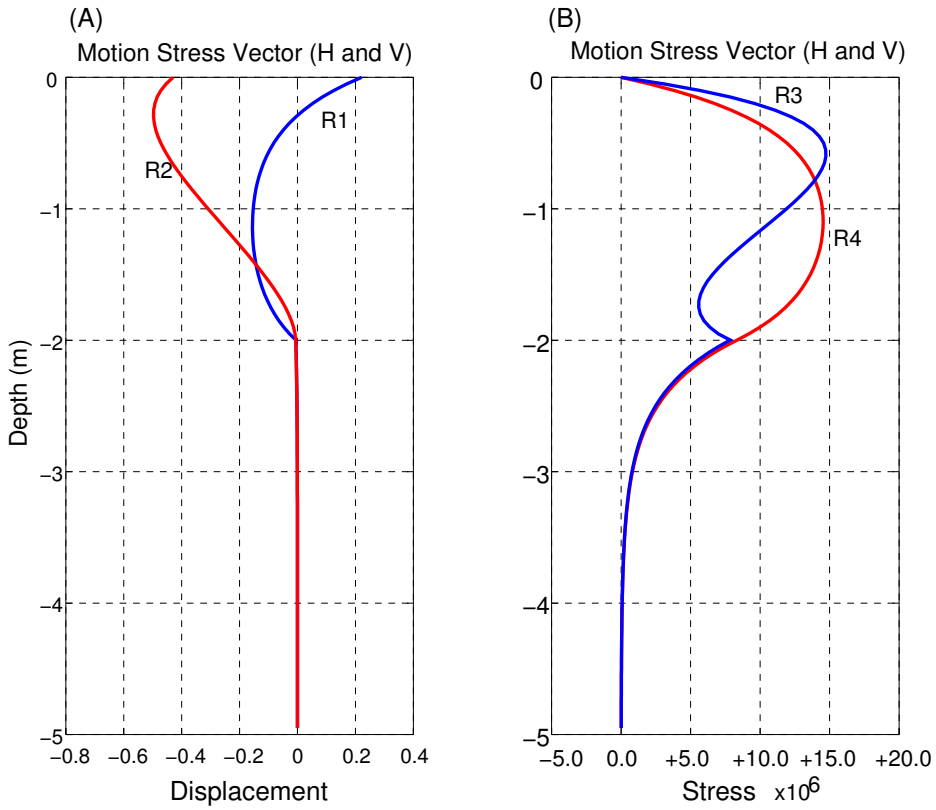


Figure 57: Motion-stress vectors for simple layer over a half space model of Figure 55. A) Displacement vectors, B) Stress vectors. Horizontal motion is R1, vertical motion is R2. Horizontal stress is R3, vertical stress is R4.

7.3.6 Synthetic Rayleigh Wave Seismograms (waves)

The program *waves* can be used to generate either a vertical or horizontal in-line component synthetic seismogram. Only the Rayleigh waves are computed (no body waves). The program *waves* is a programming of the equations found in Aki and Richards ([1] chapter 7). Details on the programming are also found in Michaels [8]. The use of synthetic Rayleigh wave seismograms is presented in Michaels [14].

IMPORTANT You need to run program *disper* before running this program since it needs the results of the dispersion computation. Program *waves* will compute group velocities and seismic traces in BSU's BSEGY format.

7.3.6.1 genwav Program *genwav* is a helper program that generates a namelist file to be input to program *waves*. Run from an x-term, the program prompts the user for inputs which include descriptions of the simulated source (spectrum, moment tensor), and the simulated receiver array. Other documentation beyond what follows can be found in the *wave* man page. A description of the dialog follows:

1. **Name of namelist file:** This is limited to 40 characters, suggest using *waves.d*.
2. **Name of dispersion curves file:** The default from *disper* is *earth.crv*. This name is defined in the *disper.d* file input to program *disper*.
3. **Near offset:** This is the nearest geophone offset from the source in meters.
4. **Number of receivers (nrec):** This is the number of geophones deployed in the simulation.
5. **Minimum group velocity expected:** This is used to give a suggestion for the next parameter, *tmax*. It is more of a warning, since you must choose a sample interval and *tmax* that produce the exact same frequency spacing as was used in *disper*. The warning is an attempt to avoid a synthetic with FFT wrap-a-round which would result if the combination of maximum offset and record time would result in a wave running off the bottom of the record.
6. **Sample rate (fsamin):** This really is the sample interval in seconds.
7. **Frequencies Hz (fmin, fmax):** This defines the bandwidth of the simulated source. Should be the same as used in the *disper* run. Can be narrower than *disper*, but not broader (the curves file has only as many frequencies as were generated in *disper*.
8. **Maximum mode:** This sets the maximum mode number to be included. You can edit the namelist file later to change this, even doing specific mode simulations (as long as they were computed in *disper*).
9. **Plot switch, ksw:** This program selects the type of dispersion plots to be generated for dispersion. A value of zero gives velocity vs. frequency plots (phase and group velocity). A value of unity will produce wavenumber vs. frequency plots.
10. **Plot format:** A value of zero outputs Octave programs for dispersion plots (RECOMMENDED). The alternative is Maple. If you choose Maple, your mileage may vary depending on changes that may have occurred to Maple.
11. **Component of motion (irvsel):** The output options are 0=Vertical, 1=Radial component of motion in the output seismic traces. This is meant to simulate either vertical component or in-line radial component geophones.
12. **Source depth:** This is the depth of the simulated source below the assumed horizontal ground surface.
13. **Diagonal elements of moment tensor:** If you want to simulate a non-diagonal tensor, you can edit the namelist file, *waves.d*. A buried explosive source would use 1,1,1 here. A vertical impact source at the surface would use 0,0,1. This tensor controls the radiation pattern of the source, and combined with the source frequencies defines your source. The source wavelet (scalar source moment) is computed as a minimum phase wavelet, and is available in file *m0.mat* generated from the *waves* run. The wavelet at the source in file *m0.mat* is a text file, two columns, sample time and wavelet amplitude.

The example presented in this section will use the model of Figure 55 and the dispersion curves that were produced (*earth.crv*). The source will be a simulated vertical impact at the surface. The following is a log of the interactive run of *genwav*:

```
  genwav
   enter name of namelist file (40 char)
   Example: waves.d
waves.d
   enter name of dispersion curve file
   (this is file from disper.f)
   Example: earth.crv
earth.crv
   enter near offset:  xnear
1
   enter group interval:  delx
1
   enter number of receivers:  nrec
48
   enter minimum group velocity expected
100
  RECOMMENDED minimum tmax=    0.9600
   enter: maximum trace time, tmax
2.0
   enter: sample interval (seconds), fsamin
.001
   enter frequencies: fmin, fmax
1, 100
   enter maximum mode to include
9
   enter ksw switch 0=c plot, 1=k plot
0
   enter type of plot format, mapmat
   0=octave (Matlab)  1=scilab
0
   enter Output option 0=Vertical  1=Radial
0
   enter source depth
0
  enter (3) diagonal elements, moment tensor
0,0,1
  Padded Radix 2 tmax=    4.0960
  Number of points in signal= 4096
-------------------------------------------------
  .......Frequency interval=  0.24414061
  NOTE: Frequency Interval MUST MATCH DISPER OUTPUT
  WAVES will output signal length = 1.0/delf
  IF MISMATCHED: CHANGE sample rate in WAVES
                 or RERUN DISPER
-------------------------------------------------
  Number of frequencies=  409
  output in =====>waves.d
```

After running *genwav*, we have the following *waves.d* namelist file.

```
  &waves
  ksw= 0, stepz=20,
  modes=1,2,3,4,5,6,7,8,9,
  fmin=    10.0000, fmax=  100.0000,
  fsamin=   0.00100,
  curve='earth.crv',
  mapmat=0,
  matlb1='matc.m',   scilb1='matc.sci',
  matlb2='matu.m',   scilb2='matu.sci',
  irvsel=0,
  ofile='waves.tmp', /
  &source
  tm= 0.0, 0.0, 0.0,
     0.0, 0.0, 0.0,
     0.0, 0.0, 1.0, /
  sz=    0.00, sy=0.00, sx=0.00, /
  &recvr
  nrec=48,
  rz=48*0.0,
  ry=48*0.0,
rx=      1.000,     2.000,     3.000,     4.000,     5.000,
     6.000,     7.000,     8.000,     9.000,    10.000,
    11.000,    12.000,    13.000,    14.000,    15.000,
    16.000,    17.000,    18.000,    19.000,    20.000,
    21.000,    22.000,    23.000,    24.000,    25.000,
    26.000,    27.000,    28.000,    29.000,    30.000,
    31.000,    32.000,    33.000,    34.000,    35.000,
    36.000,    37.000,    38.000,    39.000,    40.000,
    41.000,    42.000,    43.000,    44.000,    45.000,
    46.000,    47.000,    48.000,
  /
```

Note that we have selected up to 9 modes, used a slightly narrower bandwidth than is available from the *disper* run, chosen a vertical impact source (matrix tm=) at the surface (sz=0.0), and a sample interval of .001 seconds. There are 48 geophones at the surface (rz=0.0) that extend in the x-axis direction along which the simulated Rayleigh wave will propagate. A vertical component geophone response is selected with irvsel=0. A listing file logging the run will be called *waves.tmp*. Plot programs will be in Octave format (mapmat=0). These plots will be in terms of velocity vs. frequency (ksw=0). There will be two dispersion plot programs, *matu.m* (group velocity) and *matc.m* (phase velocity, as from the *disper* run). A trace equalized plot of the synthetic seismogram is shown in Figure 58

The group velocity plot is generated in a Octave session. Start Octave, and then in the Octave window, one types:

```
matu;
```

Figure 59 shows the group velocity plot. The corresponding phase velocity curves can be plotted by executing *matc.m* from within Octave. This will produce a plot similar to the plot in Figure 56, but limited to match the frequency range in the actual *waves* run that produced group velocities of Figure 59.

7.3.6.2 Editing the *waves.d* file One can edit the *waves.d* file, perhaps after copying it to a new file with a different name. For example, if you wish to compute only the fundamental, you would change the modes line to the following

```
modes=1,0,0,0,0,0,0,0,0,
```

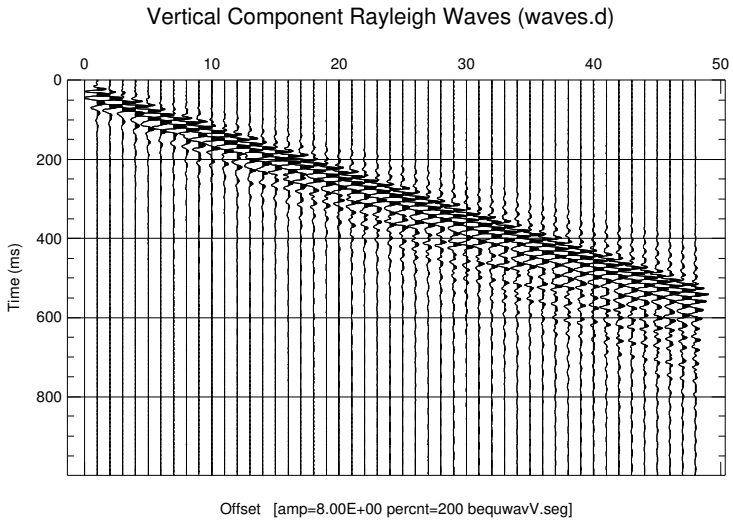


Figure 58: Plot of vertical component motion, trace equalized to remove amplitude decay with offset. This permits viewing the waveform changes with offset. Compare this to the horizontal motion in Figure 60.

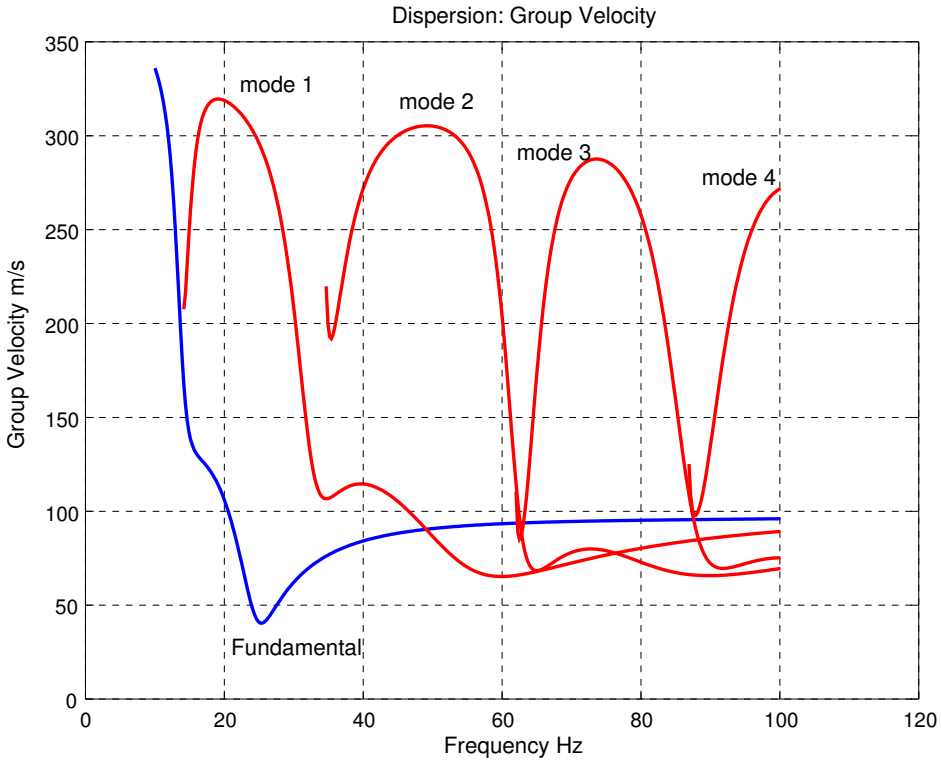


Figure 59: Group velocities are available by plotting *matu.m* from within Octave.

To compute only the second mode above the fundamental, the modes line would be as follows:

```
modes=0,0,3,0,0,0,0,0,0,
```

You can change the source moment tensor, the location or number of geophones, and their orientation. Just about anything can be edited. However, you can not ask for more frequencies than were computed in the *disper* run, since you are limited to those frequencies in the *earth.crv* file. For example, let's just edit the file to compute the in-line horizontal component of motion. This would be like deploying horizontal phones aligned with the x-axis (axis of propagation). The new file, *wavesR.d* might look like this:

```
  &waves
 ksw= 0, stepz=20,
 modes=1,2,3,4,5,6,7,8,9,
 fmin=    10.0000, fmax=  100.0000,
 fsamin=   0.00100,
 curve='earth.crv',
 mapmat=0,
 matlb1='matc.m',   scilb1='matc.sci',
 matlb2='matu.m',   scilb2='matu.sci',
 irvsel=1,
 ofile='waves.tmp', /
 &source
 tm= 0.0, 0.0, 0.0,
    0.0, 0.0, 0.0,
    0.0, 0.0, 1.0, /
 sz=    0.00, sy=0.00, sx=0.00, /
 &recvr
 nrec=48,
 rz=48*0.0,
 ry=48*0.0,
rx=      1.000,     2.000,     3.000,     4.000,     5.000,
     6.000,     7.000,     8.000,     9.000,    10.000,
    11.000,    12.000,    13.000,    14.000,    15.000,
    16.000,    17.000,    18.000,    19.000,    20.000,
    21.000,    22.000,    23.000,    24.000,    25.000,
    26.000,    27.000,    28.000,    29.000,    30.000,
    31.000,    32.000,    33.000,    34.000,    35.000,
    36.000,    37.000,    38.000,    39.000,    40.000,
    41.000,    42.000,    43.000,    44.000,    45.000,
    46.000,    47.000,    48.000,
 /
```

What has changed? We change the orientation of the receivers by setting *irvsel=1*. We can also change the name of the listing file to *wavesR.tmp*. All else remains the same. The program will automatically change the name of the output BSEGY file. These names are hardwired. For vertical component synthetics, the name is *wavV.seg*, and for horizontal component data *wavR.seg*. A trace equalized plot of the synthetic seismogram is shown in Figure 60.

7.3.6.3 Signal Amplitudes The computed synthetic seismograms are for particle displacement. Thus, they are not what one would see with a velocity geophone (which measures particle velocity). The source wavelet is captured in a text file from the *waves* run. The file is named *m0.mat*. It consists of 2 columns, record time and signal amplitude. This wavelet is filtered by the computed Rayleigh wave earth response and computed radiation pattern of the source moment tensor to produce the synthetic.

The actual spectral bandwidth will be less than is commonly defined (common definitions include the -3dB or -6dB point for example). We can plot the wavelet and its amplitude spectrum by running the Octave procedure, *m0.m*, generated by *waves*. For example, a brief listing of the generated *m0.m* file is shown below:

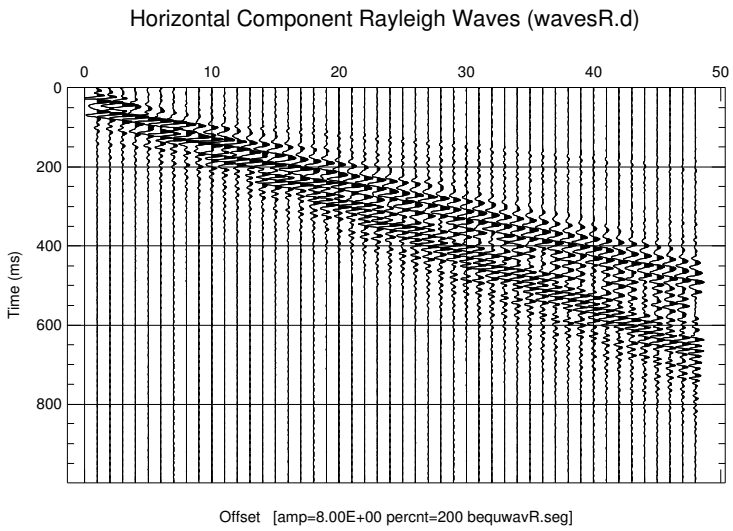


Figure 60: Plot of horizontal component motion, trace equalized to remove amplitude decay with offset. This permits viewing the waveform changes with offset. Compare this to the vertical motion in Figure 58.

```
clear
 // generated by waves.f <pm@cgiss.boisestate.edu>
 // scalar source moment, plot wavelet

 data=[...
    0.00000   0.1145539E-06 ;...
    0.00100   0.1769859E-05 ;...
    0.00200   0.1318357E-04 ;...
    0.00300   0.6318095E-04 ;...
       *           *
       *           *
       *           *
    1.02100  -0.1863727E-42 ;...
    1.02200  -0.2017870E-42 ;...
    1.02300  -0.2087935E-42 ;...
 ];
 tm=data(:,1);
 mo=data(:,2);
 Mo=fft(mo);
 npts=length(mo);
 k=npts/2;
 dt=tm(2)-tm(1);
 frq=0:npts-1;
 frq=frq/(npts*dt);
 subplot(211)
 plot(tm,mo,'-b','linewidth',2);
 title('Rayleigh Wave source wavelet');
 xlabel('Time (s)');
 grid on;
 subplot(212)
 plot(frq(1:k),abs(Mo(1:k)),'-r','linewidth',2)
 title('Amplitude Spectrum');
 xlabel('Frequency Hz');
 grid on;
```

The resulting plot is shown in Figure 61. Note, that the bandwidth (by common definitions) is less than the values specified by *fmin* and *fmax*.

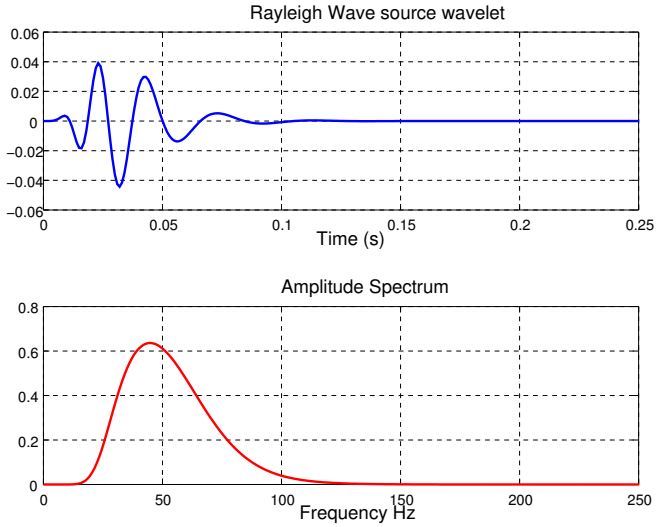


Figure 61: Wavelet plot from Octave program *m0.m*. Note that the bandwidth is less than conventional definitions would imply. When you set *(fmin,fmax)* in *waves.d*, you are basically setting nearly the complete limit of frequencies. The program reduces the bandwidth to approximately $4fmin$ and $fmax/2$. This figure has been enlarged to show detail with the axis command.

7.3.6.4 From Displacement to Velocity If you wish to see waveforms similar to what a velocity geophone would produce, you can use BSU to differentiate the seismic signals. The program to use would be *bdif*. For example, one could execute the following:

```
bdif wavV.seg .1
```

This would produce the result shown in Figure 62. It is a bit hard to compare the displacement to its derivative at this scale, so we can plot the first (near offset) signal from both data sets (*wavV.seg* and *bdifwavV.seg*). We issue the following commands:

```
bplt bdifwavV.seg 1 1 0 1 1 0 .2 1 .4 200 3.  3.
bplt wavV.seg 1 1 0 1 1 0 .2 1 .001 200 3.  3.
```

Figure 63 shows a comparison between the differentiated and original vertical component near offset signals. The differentiation computed in program *bdif* permits a stability factor since the program uses a bi-linear transform method that places a pole at the Nyquist frequency. In the Z-transform plane, the pole moves from $Z = -1$ to $Z = -(1+stab)$, where *stab* is the stability factor. In the example above, we set $stab = 0.1$. If the data have been filtered to remove any signal at and near the Nyquist frequency, then the *stab* factor can be small or zero. However, even the slightest amount of noise at the Nyquist frequency will be magnified without a *stab* factor. Typically, I choose $0.1 < stab < .5$.

Bilinear Transform derivative A reference for the theory of the bi-linear method of filter design can be found in Gold and Rader [5]. Mathematically, the Z-transform for the derivative is given as

$$Y(Z) = \frac{2}{\Delta t} \cdot \frac{(1-Z)}{((1+stab)+Z)} \cdot X(Z) \tag{49}$$

where Δt is the sample interval, $X(Z)$, is the input signal Z-transform, and $Y(Z)$ is the output (ie. derivative of X),

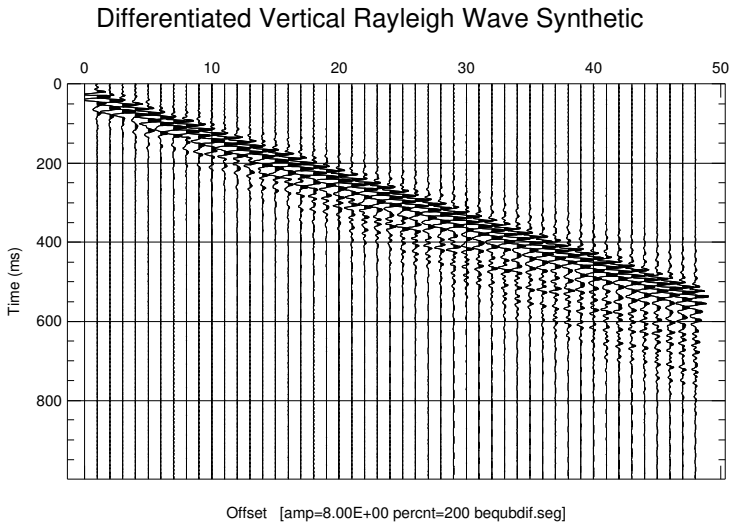


Figure 62: Plot of file *bdifwavV.seg*, differentiated *wavV.seg* simulates what a velocity geophone might see. Compare to Figure 58.

Z-transform. Feedback is used in an auto-regressive-moving-average (ARMA) operator to realize this equation in the time domain:

$$y_j = \left(\frac{2}{\Delta t} \cdot \left(x_j - x_{j-1} \right) - y_{j-1} \right) / \left(1 + stab \right) \quad . \tag{50}$$

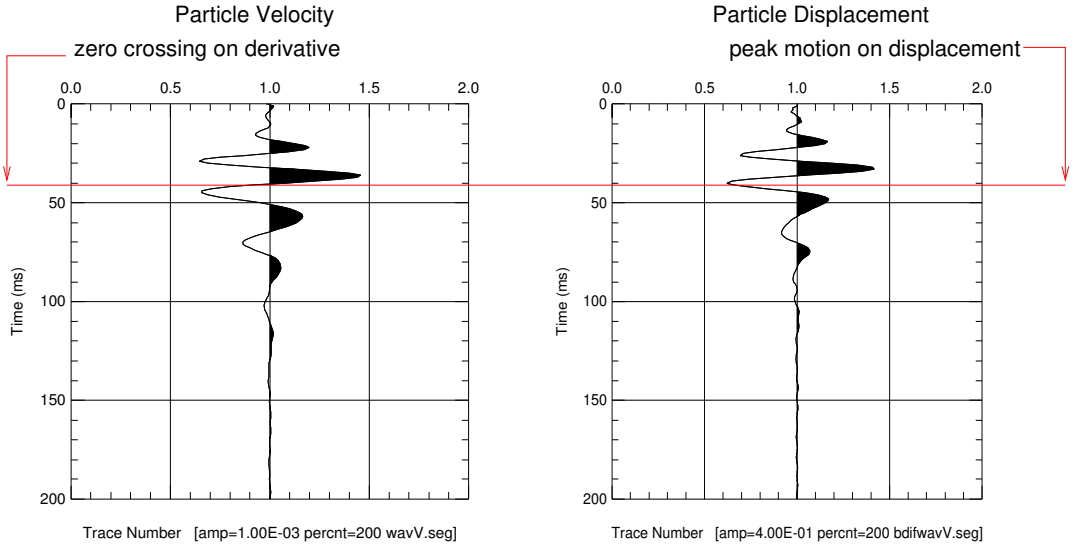


Figure 63: Plot of file *bdifwavV.seg*, differentiated *wavV.seg* simulates what a velocity geophone might see. Only near offset signals are shown for easier comparison.

7.3.6.5 Pitfalls in setting parameters The output dispersion curves computed by *disper*, typically saved in a file named *earth.crv*, contains rows, each row for a computed frequency. The other elements in the row are wavenumbers for each mode. This frequency increment is set by the aperture implied by file *disper.d*. In running *gendis*, the aperture questions are the first two (sample interval and tmax). The frequency step size is $\Delta f =$

$1/(N\Delta t) = 1/tmax$, where Δt is the sample interval. The program *waves* must use the exact same frequency step size. When running *genwav*, answer the question for maximum trace time, *tmax*, using the same value as when running *gendis*. The sample rate question (really sample interval) should also be answered using the same value as with *gendis*. This will guarantee that the two programs are consistent.

However, even if one does specify the same aperture, there still may be a problem with the implied model. Program *disper* uses parameter *deltz* to subdivide intervals between control points with layers. The smaller *deltz*, the more layers. In the above example, a single layer over a half space is forced using three control points. The first two (depth=0 and depth=2.0) have the same material properties, making that entire interval produce the same result regardless of the *deltz*. The discontinuity is represented by a third control point just slightly below the second (at depth=2.001). While abrupt, a finite interval exists between 2.000 and 2.001 meters depth. When running *waves*, a smaller interval than *deltz* is required to compute the energy integrals which yield group velocity and the lagrangian. The smaller interval is set by a computed "minimum wavelength", subdivided by namelist parameter, *stepz*. If *stepz* is too large, the implied depth interval (analogous to *deltz*) may actually subdivide the discontinuity into additional layers. The minimum wavelength is given by

$$\lambda = \frac{\beta_{minimum}}{f_{max}} \tag{51}$$

This means that changes, like extending the maximum frequency computed, may lead to a resampling of the discontinuity into layers. This problem presents itself as a glich in the group velocities and an increase in the lagrangian. The corrective action is to edit the *waves.d* namelist file, decreasing the *stepz* parameter. Alternatively, one may wish to re-run disper making the discontinuity thinner. Figure 64 illustrates the problem in (B).

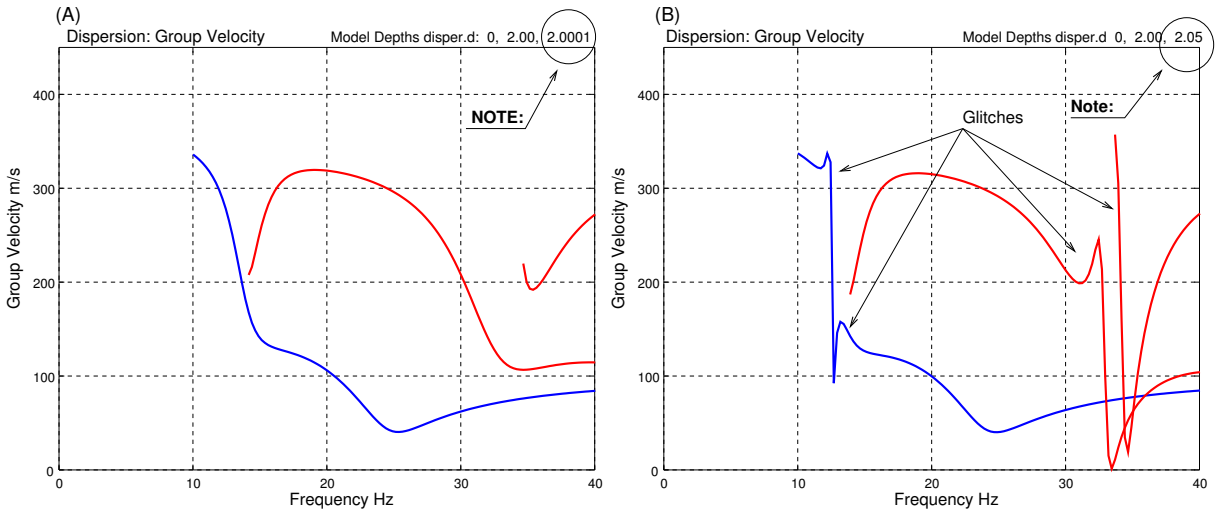


Figure 64: (A). Correct *waves* computation of dispersion. (B). Illustrates too large a depth difference between top and bottom of the discontinuity. The solution is to make the discontinuity more abrupt in *disper.d* or decreasing *stepz* in *waves.d* to remove the glitches.

8 Hydraulic Conductivity from Seismic Damping

The use of the Kelvin-Voigt (KV) representation has long been a standard in geotechnical engineering and soil dynamics. In the vibrator perspective, it consists of a mass, spring, and dashpot with the spring and dashpot in parallel configuration. Figure 65-A shows both the vibrator and wave perspectives of the KV model. The KV model is limited by the single mass, making it unable to represent multi-phase media like a water saturated sand. In the case of a water saturated sand, there are two masses (solid frame, pore fluid).

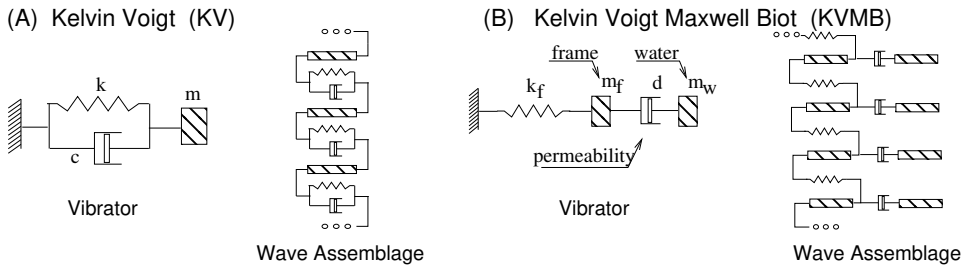


Figure 65: (A). Kelvin-Voigt (KV) representation for both vibrator and wave assemblage. (B) Kelvin-Voigt-Maxwell-Biot (KVMB) representation.

Building on the work of others, it became apparent that the ideas of Maxwell and Biot could be combined to produce an alternative model with two masses that would produce vibratory behavior close to from the KV model (Michaels [13]). This alternative representation is the Kelvin-Voigt-Maxwell-Biot (KVMB) model shown in 65-B. The two masses represent the solid frame and the fluid (typically water, but could be any fluid with a viscosity). The dashpot can be represented by a collection of conduits for the fluid to move relative to the solid frame. While the model based on parallel tubular pores within a solid mass is unlikely to be a true image of a water saturated soil, it is capable of capturing the behavior of real granular water saturated media when excited by seismic forces.

Some questions can not be addressed by the KV model. Consider, for example, this question:

> Should seismic damping increase or decrease with an increase in hydraulic conductivity?

The problem is that there is no place for permeability in the KV model. It turns out that neither is likely to be the only answer. Once one introduces a mechanism to capture permeability or hydraulic conductivity for a specific fluid into the model, the question can be addressed. The KVMB model, on the other hand, introduces permeability and fluid viscosity through the dashpot placed between the two masses. The result is the ability to capture both coupled and uncoupled possible motions. This results in behavior that answers the question with both alternatives being true.

The dashpot provides viscous friction. Friction can be low when the soil is very tight (low permeability) and the motion of fluid and frame are largely coupled. That is, they move together. Low levels of friction are also possible in a highly permeable soil, one with large pore spaces that permit easy flow between frame and fluid. This is the uncoupled case. The KVMB model predicts that there will be an intermediate place between these two extremes where friction is greatest, and this produces a peaked response.

The metric chosen to represent viscous friction is the KV damping ratio. This is possible even though that representation can not explain the friction as the KVMB model does. There is a mapping between the two representations because the KVMB model predicts behavior very close to that of a true KV vibrator. The mathematics are developed in Michaels [13] and involve dropping the real eigenvalue of the KVMB system and relating the two complex KVMB eigenvalues to the only two eigenvalues of the KV system.

With this mapping, the body of KV based measurements can be related hydraulic conductivity. Be these measurements from vibratory experiments (like resonant column) or from shear wave propagation of dispersion and propagation decay, the mapping provides a way to connect them to permeability of the medium (once the fluid viscosity and porosity are known). Knowledge of porosity is required since it splits the mass into the two components, fluid and frame.

8.1 Mapping KVMB to KV

Program *kvKVMBscan.m* illustrates the effect of mass ratio between solid frame and a fluid. In Figure 66, the viscosity for water is employed. The most friction results when the masses are equal (not a likely situation given the difference in density of most grains and water). The larger the frame to fluid mass ratio, one would expect the lower the porosity.

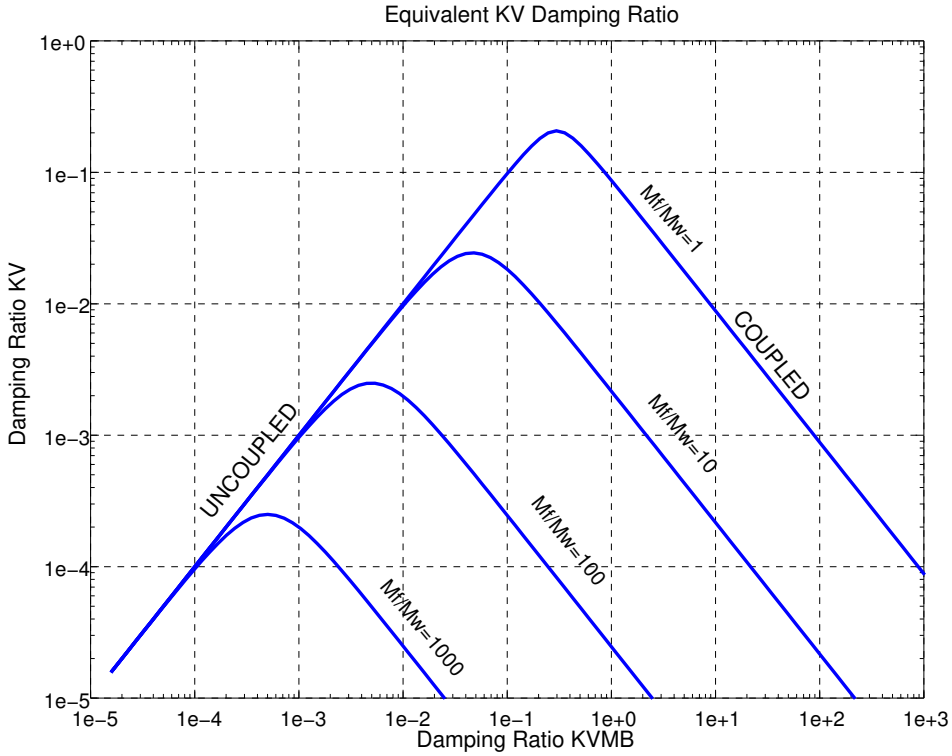


Figure 66: Octave program, *kvKVMBscan.m*, can be run to illustrate the effects which largely depend on porosity. Shown are cases for different mass ratios of solid frame and pore fluid.

The reader should note that damping ratio in the KVMB context spans a very different range than the traditional KV damping ratio. This is true despite the fact that the same formula is used for both representations. As explained in Michaels [13], there is a problem in deciding what mass to use in the formula for damping ratio when considering the KVMB case. Recall, damping ratio is given by:

$$\delta = \frac{D}{2\sqrt{K \cdot M}} \tag{52}$$

where D is damping, K is spring constant, and M is mass. The program uses the combined mass of fluid and frame since most real soils will have permeabilities on the coupled side of the curve.

8.2 KV Damping Ratio vs Hydraulic Conductivity

The Octave program, *kdKVMBscan.m*, permits a view of the mapped KV damping ratio as a function of hydraulic conductivity. The symbol, Kd, is used in this discussion to represent hydraulic conductivity (units m/s). Any good text on soil mechanics can be reviewed for the difference between absolute permeability and hydraulic conductivity. Permeability (units m^2) is a concept without context of fluid present. The concept of hydraulic conductivity adds the context of fluid viscosity. Thus, figures showing Kd on an axis are for a specific fluid present, and this is hard wired to water and it's viscosity.

Figure 67 shows two alternative ways to plot the representation. Hydraulic conductivity is the horizontal axis in (A), and an effective "pore diameter" in mm is shown in (B). The curves are plotted for 15 Hz shaking.

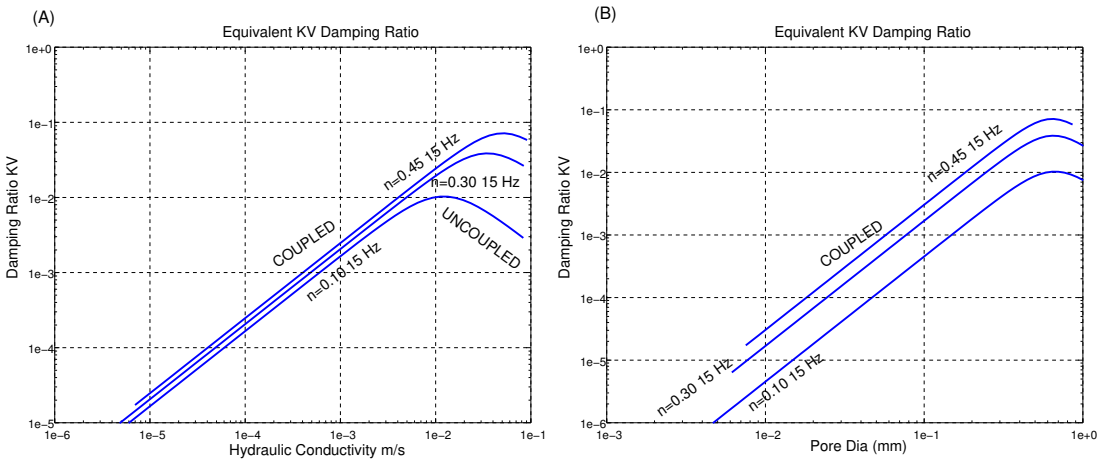


Figure 67: Octave program, *kdKVMBscan.m*, can be run to illustrate the effects which largely depend on porosity and frequency of shaking. Shown are the case for 15 Hz shaking. The user can choose a horizontal axis of either (A) hydraulic conductivity (m/s), or (B) "pore diameter (mm)"

As frequency increases, the curves would shift to the left. Frequency of shaking will be explored in the next subsection. The less permeable a soil is, the more rapidly it must be shaken to stimulate fluid motion with respect to the frame, and it is the relative motion that produces the friction. Recall that friction is measured by damping ratio. This relationship with frequency is why shear waves are dispersive in a viscous medium.

8.3 Frequency and Hydraulic Conductivity

The Octave program *fqKVMBscan.m* presents the model's representation of a soil's dynamic behavior under different shaking frequencies. For the case of 30% porosity, Figure 68 shows that increasing hydraulic conductivity (Kd) should produce more friction as represented in the KV damping ratio when the pore fluids are coupled to the frame.

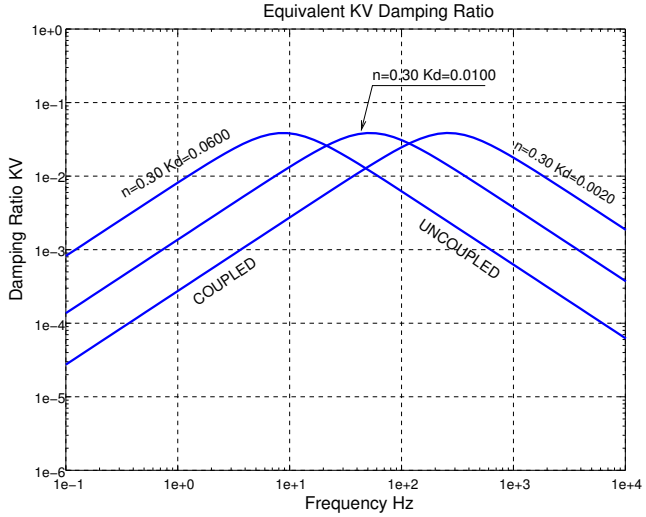


Figure 68: Octave program, *fqKVMBscan.m*, can be run to illustrate the relationships possible between hydraulic conductivity and KV damping ratio, the metric for viscous friction.

Interestingly, when one looks at uncoupled behavior, the reverse is true. Increasing permeability decreases the KV damping ratio, and there is less friction at those high frequencies.

8.4 Inverting Stiffness and Damping for Hydraulic Conductivity

Octave program *KD4kvmb.m* can be used to invert values of stiffness and damping to hydraulic conductivity. For example, in a down-hole shear wave transmission survey, one can measure body wave dispersion and amplitude decay with distance to obtain values for stiffness and damping. The procedure to invert velocity dispersion and amplitude decay is detailed in Michaels [10]. The result of the inversion are two coefficients of the viscoelastic 1D wave equation (beam divergence is corrected for in the process). The wave equation is a 3rd order PDE:

$$\frac{\partial^2 u}{\partial t^2} = C_1 \frac{\partial^2 u}{\partial z^2} + C_2 \frac{\partial^3 u}{\partial t \partial z^2} \tag{53}$$

where C_1 is stiffness (m^2/s^2) and C_2 is damping (m^2/s). The ratio C_2/C_1 yields relaxation time in seconds. In equation 53 time is represented by t and particle displacement by u. Distance in the direction of wave propagation is z. Figure 69 shows an example run assuming shaking at 12 Hz and a porosity of 25%. Note that if the value of C_2 is so large compared to C_1, that it produces a KV damping ratio that falls above the peak of the curve, no solution is possible. In such a case, one may re-evaluate all the assumptions. When deciding on a likely solution from the two possible, most soils will fall on the coupled side of the curve, making this example solution of $K_d = .0122 m/s$ the more likely one.

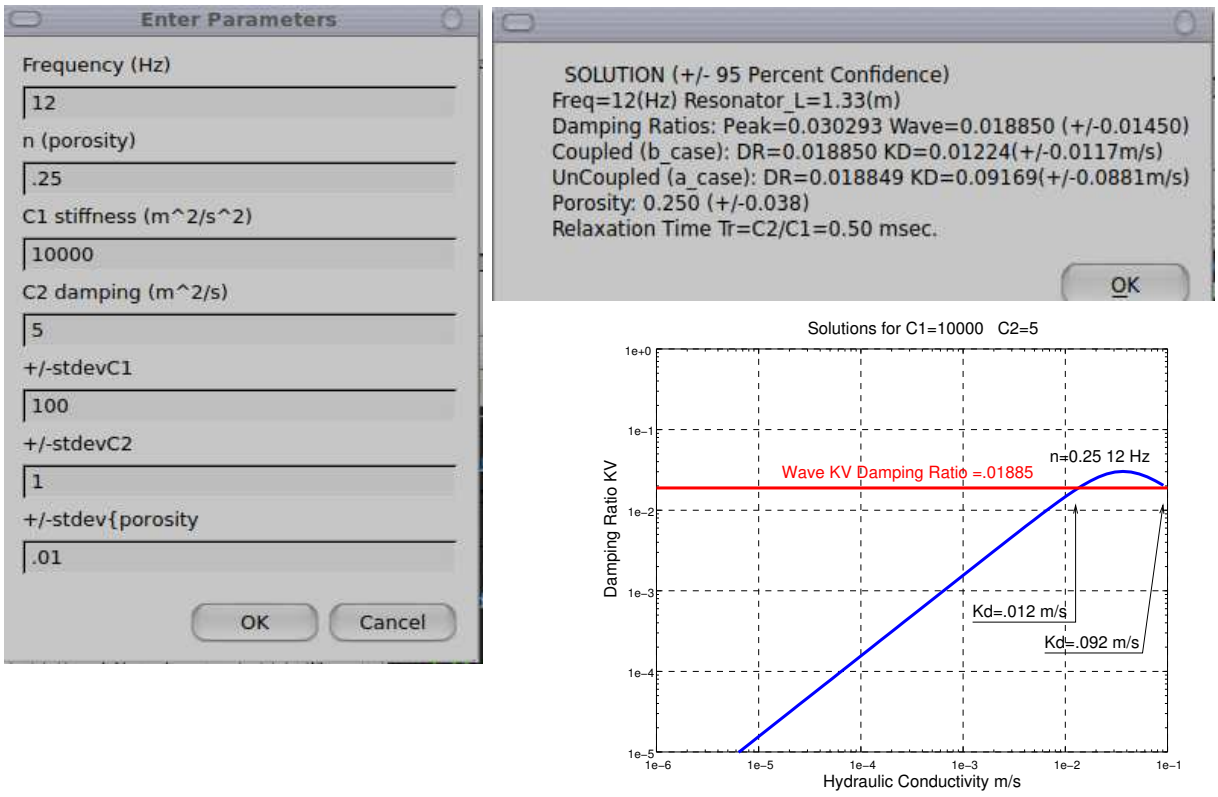


Figure 69: Octave program, *KD4kvmb.m* prompts the user for porosity (n), stiffness (C_1), damping (C_2), frequency of shaking and related uncertainties. Then when run, a display of the solution is given in a message box. Also show is the graphical image of the process. The C_1 and C_2 values produce a KV damping ratio that is represented by the horizontal line that intersects the KVMB to KV curve. The two intersections are the solution.

Are these predictions accurate? No, of course not. But they are a starting point in estimating hydraulic conductivity from shear wave measurements. While the predicted behaviors are likely correct in terms of how real soils behave, real soils are far more complex, pore spaces are not cylindrical tubes, and fluid flow is not always laminar. In the context of granular soils, these tools may be helpful in mapping soil units, their permeabilities, and predicting levels of damping that might occur when exposed to seismic waves.

References

[1] K. Aki and P.G. Richards. *Quantitative Seismology Vol. 1*, volume 1. W.H. Freeman and Co., 1980. 557p.

[2] K.M. Barry, D.A. Cavers, and C.W. Kneale. Report on recommended standard for digital tape formats. *Geophysics*, 40(2):344–352, 1975.

[3] J. K. Cohen and Jr. J. W. Stockwell. *CWP/SU: Seismic Unix release 34: a free package for seismic research and processing*. Center for Wave Phenomena, Colorado School of Mines, 2000.

[4] D. Crice. *BHG-2, BHG-3 borehole geophone operation manual*. GeoStuff, 19623 via Escuela Dr., Saratoga, Ca 95070, 1996. 16p.

[5] B. Gold, C. M. Rader, A. V. Oppenheim, and Stockham T. G. Jr. *Digital Processing of Signals*. Lincoln Laboratory Publications. McGraw-Hill, 1969. San Francisco.

[6] H. Lamb. On the propagation of tremors over the surface of an elastic solid. *Phil. Tran. Royal Society of London*, Series A(203):1–42, 1904.

[7] W. Menke. *Geophysical data analysis, discrete inverse theory*. Academic Press, 1989. San Diego 289pgs.

[8] P. Michaels. *Surface wave inversion by neural networks*. PhD thesis, University of Utah, Salt Lake City, Utah, 1993.

[9] P. Michaels. A geophysical site investigation for a bridge foundation in a narrow canyon. *Environmental & Engineering Geoscience*, 1(2):219–226, 1995.

[10] P Michaels. In situ determination of soil stiffness and damping. *Journal of Geotechnical and Geoenvironmental Engineering*, 24(8):709–719, 1998.

[11] P. Michaels. Use of engineering geophysics in the design of highway passing lanes. *Proceedings of the Symposium on the Application of Geophysics to Engineering and Environmental Problems, SAGEEP99*, pages 179–187, 1999.

[12] P. Michaels. Use of principal component analysis to determine down-hole tool orientation and enhance SH-waves. *Journal of Environmental and Engineering Geophysics*, 6(4):175–183, 2001.

[13] P. Michaels. Relating damping to soil permeability. *International Journal of Geomechanics*, 6(3):158–165, 2006.

[14] P. Michaels and R. B. Smith. Surface wave inversion by neural networks (radial basis functions) for engineering applications. *JEEG*, 2(1):65–76, 1997.

[15] H.M. Mooney. Some numerical solutions for Lamb's problem. *Bulletin of the Seismological Society of America*, 64(2):473–491, 1974.

[16] C.L. Pekeris. The seismic surface pulse. *Proceedings of the National Academy of Science*, 41:469–480, 1955.

[17] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes, The art of scientific computing Fortran version*. Cambridge University Press, 1989. 702p.

[18] P.G. Richards. Elementary solutions to Lamb 's problem for a point source and their relevance to three-dimensional studies of spontaneous crack propagation. *Bulletin of the Seismological Society of America*, 69(4):947–956, 1979.

[19] E.A. Robinson. *Multichannel Time Series Analysis with Digital Computer Programs*. Holden-Day, 1967. 298p.

[20] R.E. Sheriff. *Encyclopedic dictionary of exploration geophysics*. Society of Exploration Geophysics, 1991.

[21] K. H. Stokoe and S. M. Nazarian. Use of rayleigh waves in liquefaction studies. In R. D. Woods, editor, *Measurement and use of shear wave velocity for evaluating dynamic soil properties*, GSP, pages 1–17. ASCE, 1985.

[22] M.N. Toksöz and D.H. Johnston. *Definitions and Terminology*, pages 1–5. Geophysics reprint series No. 2. SEG, 1981.

A Appendix (bhelp listing)

```
babs.c    rectify seismic traces
bagc.c    automatic gain control of traces (scale in time and space)
ba2s.c    FORMAT CONVERSION: ASCII TEXT ---> BSEGY (no geometry setting)
bamp.F90  amplitude analysis by frequency (K-V Solid)Downhole Sph. Div.
bamx.F90  amplitude analysis by frequency (K-V Solid)SurfaceWaves Cyl.Div
bbal.c    balances two data sets to have same MAV (mean absolute value)
bcad.f    plot seismic traces as CAD (*.dxf; digital exchange file)
bcar.c    apply moving average (box car) filter as function of time
bcnv.c    FORMAT CONVERSION: BSEGY <---> SEG-Y  (BSU=*.seg, SEG-Y=*.sgy)
bcrd.f    coordinate rotation and translation, BSEGY geometry headers
bdat.c    datuming program for refraction data (easier for picking)
bdcn.f    deconvolution (profile or trace mode), prediction or error out
bdif.f    differentiates w.r.t. time using Bilinear Transform method
bdum.f    generate dummy data set with user defined impulse position
bdump.f   generate a dump of selected BSEGY header values
bedt.f    edit BSEGY seismic file (traces, time, sample interval, etc.)
bequ.c    trace equalize data by L2 norm or Maximum Absolute Value
bext.c    extract traces from a merged data set based on header values
bfil.f    ARMA FILTER of seismic traces (low-, band-, or high-pass)
bfit.f    Solves for interval velocity from times in headers (VSP)
bftr.f    FILTER traces with other *.seg traces, or namelist from bdump
bfxt.f    F-X Transform of seismic traces
bgar.c    exponential GAIN recovery, by range specification
bgaz.c    exponential GAIN recovery, by depth specification
bhed.f    up/down load selected header information from/to a text file
bhelp.c   this listing of BSU package contents
bhod.F90  hodogram by PCA to determine down-hole tool orientation
bint.f    numerical integration of seismic traces (trapezoidal rule)
bis2seg.c FORMAT CONVERSION: BISON ---> BSEGY  (no geometry setting)
bkil.f    either kill (delete) or zero seismic traces
bmed.f    median mix of seismic traces (spatial)
bmix.f    mean mix of seismic traces (spatial)
bmrg.f    merge traces from many files to a single file
bmrk.f    mark first break picks with a delta function on waveform
bmst.f    MASTER illustrates programing in BSU, FORTRAN
bnez.c    GEOMETRY: Create survey *.nez (Northing,Easting,Elevation) file
bnfd.c    MODELING:  computes near and far field in elastic whole space
bnos.f    MODELING:  generate band-limited random noise traces
boff.c    computed source to receiver offset and add to headers
bpic.f    automatic first break picker, or inserts picks from a file
bplt.c    plot seismic traces X11|PS|XFIG|JPEG|PDF formats wt,va,blk/gry
brdc.f    remove DC bias from traces
bred.f    apply linear or hyperbolic correction in time to traces
bref.f    delay time refraction analysis setup
brev.f    reverse either channel order or sample polarity
brot.f    rotate seismic traces (3-component data)
brpt.c    remove pretrigger (updates header and static shift data)
brsp.f    resample data in time (augmentation with zeros in freq domain)
bscl.f    scale data set with user provided, or data derived factor
bsdc.f    show DC level in seismic traces as %MAV (mean absolute value)
bshf.f    apply a static shift to seismic traces (shift in time)
bshp.f    Wiener least squares shaping filter
bsrt.c    sorts traces by offset
bstk.f    stacks traces in a gather (number traces out=number in)
bsum.c    weighted sum of two data sets: file_out = file1 + C*file2
bswp.c    byte swap a BSEGY or SU data set (toggle endian)
btor.f    applies geophone orientations to trace headers (from PCA)
```

```
bvas.F90  measures body wave velocity dispersion (Kelvin-Voigt solid)
caplot.F90 PostScript plots of combined bvas and bamp runs
bvax.F90  measures velocity dispersion on surface data
bvel.f    apply correctional velocity (linear with range) Down-Hole
bvsp.f    down-hole travel time inversion with ray bending
bwht.f    non-linear whitening program (agc applied to bank of filters)
bwin.f    window the data aperture with raised cosine tapers
bxcr.f    cross (or auto) correlation, output to BSEGY data set
tplt.c    plot a single trace, pipes to GNUPLOT and writes gnuplot file
qplt.c    plot all traces (scaled by max abs) pipe to GNUPLOT and write
          a gnuplot file (qgraph.gp)
gendis.f  helper program to build namelist file for program disper.f
disper.f  computes dispersion curves for Rayleigh waves (see also waves)
showmdl.f show model in disper.d file
genwav.F90 Helper program to build namelist file for program waves.f
waves.F90 computes synthetic Rayleigh wave only seismograms (see disper)
cmst.c    MASTER illustrates programing in BSU, C-LANGUAGE
egg2seg.c FORMAT CONVERSION: EEG's SEG-2 ---> BSEGY (no geometry)
seg2txt.f FORMAT CONVERSION: BSEGY ----> ASCII (Octave/Matlab)
seg2csv.c FORMAT CONVERSION: BSEGY ----> CSV (comma separated value)
          CSV files can be input into spread sheets
seg2dump.c RAW DUMP of SEG-2 data file to text listing.
genb2s.f  BASH SCRIPT GENERATOR: sets up a block of bis2seg runs
genbhod.f BASH SCRIPT GENERATOR: sets up for PCA analysis   s-wave source
genbhodV.f BASH SCRIPT GENERATOR: sets up for PCA analysis  vert. source
genbrot.f BASH SCRIPT GENERATOR: sets up for run of brot on many files
genref.f  BASH SCRIPT GENERATOR: for geometry setting on pattern, CDP
gensetg.c generates control files for geometry setting, setgeom program
genvsp.f  BASH SCRIPT GENERATOR: setting geometry for down-hole surveys
halfsp.f  computes motion-stress vectors and velocity of Rayleigh wave
lamb.c    MODELING:  computes solution to Lamb's problem (Vp/Vs=sqrt(3))
picrestore.f Restore suxpicker picks on data reduced by bred.f
setgeom.c Sets geometry for reciprocal refraction shooting (see gensetg)
genwaw.c  Interactive, Set geometry and convert SEG-2---->BSEGY
          (genwaw good use is for walk-away shooting)
top2dxf.f FORMAT CONVERSION: Survey NEZ (*.nez) ---> CAD (*.dxf)
top2nez.f FORMAT CONVERSION: RAW TOPCON  ---> Survey NEZ (*.nez)
topbcrd.f apply coordinate translation and rotation to Survey NEZ (*.nez)
topcon.f  combines NEZ with BISON file --->*.xyz file Run bhed on *.xyz
topcon2.c combines NEZ with EGG SEG-2  ---> BSEGY with geometry
traplt.f  line printer style plots of samples and spectrum


C-FUNCTIONS:
bargrid.c   draw a progress bar grid to screen (stdout)
c_boxit.c   apply a running average (box car) filter to a signal
c_bsegin.c  read a seismic trace in BSEGY format
c_bsegout.c write a seismic trace in BSEGY format
cr_labl.c   write a copyright label to screen (stdout)
exbar.c     draw progress bar to screen (stdout)
fcr_labl.c  write a copyright label to a file
findxyz.c   extract geophone and source coordinates from headers
fmax_min.c  compute maximum and minimum value in a signal, and indexes
fnorm.c     compute L2 norm of a signal
hlp_labl.c  print a help label to screen (stdout)
in_chk.c    scan a data set for number of traces, sample interval, etc.
lsqufit.c   least squares fit, straight line function: y=mx + b
mav.c       compute mean absolute value of a signal
names.c     build file names from process ID
```

```
onepole.c   perform a single pole (Z-plane) filter
xdrfloa.c   IBM functions to convert between IEEE and IBM floats


FORTRAN SUBROUTINES:
agc.f       apply Automatic Gain Control (AGC) on a signal
arma.f      apply an Autoregressive Moving Average (ARMA) filter
bnoise.f    compute band-limited random noise
boxit.f     apply a running average (box car) filter to a signal
bsegin.f    read a seismic trace in BSEGY format
bsegout.f   write a seismic trace in BSEGY format
cfilt.f     design a bank of band-pass digital filters
chktrc.f    scan a data set for number of traces, sample interval, etc.
conv.f      apply a selected filter out of bank designed by cfilt.f
conv2.f     time domain convolution of two signals
cross.f     E. A. Robinson's cross correlation subroutine
cshift.f    shift traces in frequency domain by phase rotation
czero.f     zero a complex signal
demult.f    demultiplex a complex trace into real and imag. parts
dot.f       E. A. Robinson's inner (dot) product between two vectors
drum.f      E. A. Robinson's phase unwrapping subroutine (modified by pm)
eureka.f    E. A. Robinson's solution by Levinson Recursion
findxyz.f   extract geophone and source coordinates from headers
flist.f     line printer style plot of a complex signal (spectrum)
fmed.f      bubble sort on a vector, returns median value
fold.f      E. A. Robinson's convolution subroutine
l2norm.f    rescale a seismic signal by its L2 norm
nlogn.f     E. A. Robinson's Fast Fourier Transform (Radix 2)
nrad2.f     determines the first power of two larger than a given value
pltbar.f    draw progress bar to screen (like bargrid.c and exbar.c)
polar.f     E. A. Robinson's conversion from rectangular to polar form
rand.f      CMLIB: generates uniform pseudo-random number
rect.f      convert complex numbers from polar to rectangular form
runif.f     CMLIB: calls rand.f to generate longer period random sequence
shape.f     E. A. Robinson's least squares shaping filter
tlist.f     line printer style plot of a signal (also lists values)
vshft.f     shifts a signal by linear interpolation of samples
xcor.f      cross correlates two signals
xcor1.f     one sided autocorrelation of a signal
xmax.f      find maximum and minimum values in a vector
C-INCLUDE FILES: Source Code Directory=bsu-3.0.0/src/C/include/
                 Installed   Directory=/usr/local/include/bsu-3.0.0/
c_bsegy.h           BSEGY header structure
sub4.h              prototypes of library functions

FORTRAN INCLUDE FILES: Source Code Directory=bsu-3.0.0/src/Fort/include/
                       Installed  Directory=/usr/local/include/
bsegy.inc            BSEGY header declaration and equivalence statements
bsegy.f90            BSEGY header declaration and equivalence statements

                       OCTAVE/MATLAB CODES
Octave:   Installed Directory=/usr/local/share/octave/site-m
bsegin.m         Reads seismic traces in BSEGY format
bsegout.m        Write seismic traces in BSEGY format
segyinfo.m       Scan a BSEGY file
traplt.m         Plot a signal from a BSEGY data set and its spectrum
segpic.m         Pick first breaks using mouse on waveform
profplot.m       Plot all traces in a file, simple waveform format
yulewalker.m     Plot all pole spectrum of seismic trace or from an
```

```
                autocorrelation signal as input.

                 DOWN HOLE SOFTWARE  Octave/Matlab
hodoplot.m      Plot hodogram (two channels in same file)
hodo2plot.m     Plot hodogram (channels in two different *.seg files)
seisazi.m       Plot header azimuth of receiver component
                (intended for use prior to rotation to a constant az)
vfitw.m         Least Squares fit to first break down-hole data
vplot.m         Nice plot of vfitw.m results
cainv3.m        Kelvin-Voigt inversion of down-hole (VSP) data
                (requires bvas.his and bamp.his results)
caplot3.m       Nice plot of cainv3.m results
cafwd3.m        Forward modeling, Kelvin-Voigt computes dispersion

                 REFRACTION SOFTWARE Octave/Matlab
direct.m        Direct wave analysis for overburden velocity
delaytm.m       Delay time analysis of refraction data conventional
delaytmR.m      Delay time analysis of refraction data reciprocal
refplot.m       First break analysis (apparent velocity, intercept)

                 SURFACE WAVE SOFTWARE Octave/Matlab
rayleigh.m      Example on how to dynamically link rwv.f for dispersion
                curve computation in Octave environment.
moho.m          Example similar to rayleigh.m, but plots dispersion
                as a function of period.
FwdR1.m         Forward problem, Rayleigh wave given bvax results to
                compare to.  model.txt is a file with 3 rows
                describing the soil profile.  Example (nlay=3)
                nlay
                v1  v2  v3
                z1  z2  z3
invR1.m         Inverts a dispersion curve from bvax results (bvax.his)
                Truncated Singular Value method employed.
                Starting model in model.txt as in FwdR1.m above.
SASW.m          Spectral Analysis of Surface Waves
                Data= two time domain signals from a *.seg BSEGY data
                set.  Computes required cross-spectra and coherence
saswv.m         Spectral Analysis of Surface Waves
                Data= Cross-Power Spectrum and Coherence, text file
                dx32f.txt sample data from NGES, Texas A&M Georisk 2011
Dispcurve.m     Example program computes dispersion from spectral record
                Data= Frequency spectra recorded at many offsets
                20mSourceOffset.mat sample data Christchurch New Zealand
                GeoCongress 2014 ASCE meeting, see GSP 234-235
PseudoTime.m    Example program computes time series from recorded
                same frequency spectra data set as in Dispcurve.m above
kvKVMBscan.m    Kelvin Voigt to KVMB damping ratio for frame:water mass
kdKVMBscan.m    KV damping ratio vs hydraulic conductivity for porosity
fqKVMBscan.m    KV damping ratio vs frequency for hydraulic conductivity
KD4kvmb.m       Wave stiffness,damping + porosity solution hydraulic c.

SCRIPTS:  Installed Directory=/usr/local/share/bsu/scripts
 Example scripts which can be customized as needed.
 xplot          X11 Plot waveform data using BSU program bplt
 psplot         Postscript Plot wavform data using BSU program bplt
 xPlot-su       X11 Plot waveform data using Seismic Unix (SU required)
 psPlot-su      Postscript Plot wavform data using Seismic Unix
 rename-btor    Automated renaming of files after btor execution
 mergeplots     Concatonates group of Postscript files into a single PDF
```

```
 Merge-all       Script to process all combinations of VSP data
 Merge2          VSP processing if load cell + 3 downhole + 3 ref. phones
 Merge           VSP processing to P-wave and SH-wave data sets

DOCUMENTATION: Installed Directory=/usr/local/share/doc/bsu
 README
 bsu-user-guide3-2.pdf   (pdf user guide)
 bsu-html.tar.gz       (html user guide)
 man-bsu3-html.zip     (html man pages)
```

B Appendix (*Merge-all*)

The following bash script is used in down-hole surveys to form the various resorted data sets. These include both difference and sum of opposite source polarizations, for each individual component.

```
#!/bin/sh
#  $Id: Merge-all,v 1.1.1.1 2017/04/13 21:40:01 pm Exp $
# Merge-all:  basic script to merge traces in all combinations
#             (sum or difference) by each component.
#
# Notation: component/location/sum or dif/
# Examples:
#           tddf=Tcomponent, down-hole, difference
#           trdf=Tcomponent, reference, difference
#           tdsm=Tcomponent, down-hole, sum
#           trsm=Tcomponent, reference, sum
#
# This combines all the shot gathers into receiver gathers for the different
# geophone components in a down-hole survey.  You should have run bhod first
# to determine tool orientation, btor to insert orientations into headers,
# and brot to rotate the data into a standard orientation.  If you have, then
# you can run this script.
#
#     Copyright (c) 2017 Paul Michaels
#     <pm@cgiss.boisestate.edu>
#     This program is free software; you can
#     redistribute it and/or modify it under the terms
#     of the GNU General Public License as published
#     by the Free Software Foundation; either version
#     2 of the License, or (at your option) any later
#     version.  This program 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 General Public
#     License for more details.
#     You should have received a copy of the GNU
#     General Public License along with this program;
#     if not, write to the Free Software Foundation,
#     Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

#set -x

#...define parameters  YOU MUST DEFINE THESE FOR EACH NEW SURVEY
#                      SINCE IT IS LIKELY THAT THE NUMBER OF RECORDS
#                      WILL VARY WITH EACH SURVEY !!!!
odmin=01
odmax=145
evmin=02
evmax=146
PRFX=L
#         eklmax=(odmax-1)/2
eklmax=`bc <<END
($odmax-1)/2
END`
echo $eklmax
#         oklmax=eklmax+1
oklmax=`bc <<END
($eklmax+1)
```

```
END‘
echo $oklmax

#shaping filter parameters (bshp)
tmin=0.
tmax=0.1
npf=360
stab=.0001

#polarity file definitions
az90=1
az270=2

#Scale Data by VERTICAL REF (ch4) (Absolute Scale from last, evmax)
bscl brot$PRFX$evmax.seg 4 1 3 1>/dev/null
AMP=‘gawk ’/Peak Absolute Value/ {print $4}’ bsclbrot.lst‘
echo $AMP

FILE=‘find brot*seg | sed s/\.seg/’’/g |sed s/brot/’’/g‘
for i in $FILE; do
        bscl brot$i.seg 4 1 3 1>/dev/null
        mv bsclbrot.seg bscl$i.seg
        bscl bscl$i.seg 4 1 0 $AMP 1>/dev/null
        mv bsclbscl.seg bscl$i.seg
done

#...reference phone gathers (VERTICAL=ch4)
bmrg bscl$PRFX $odmin $odmax 2 4 4 1>/dev/null
mv bmrg.seg rfv1.seg
bmrg bscl$PRFX $evmin $evmax 2 4 4 1>/dev/null
mv bmrg.seg rfv2.seg

#...reference phone gathers (RADIAL=ch5)
bmrg bscl$PRFX $odmin $odmax 2 5 5 1>/dev/null
mv bmrg.seg rfr1.seg
bmrg bscl$PRFX $evmin $evmax 2 5 5 1>/dev/null
mv bmrg.seg rfr2.seg

#...reference phone gathers (TRANSVERSE=ch6)
bmrg bscl$PRFX $odmin $odmax 2 6 6  1>/dev/null
mv bmrg.seg rft1.seg
bmrg bscl$PRFX $evmin $evmax 2 6 6  1>/dev/null
mv bmrg.seg rft2.seg

#...down hole swc phone gathers (TRANSVERSE=ch3)
bmrg bscl$PRFX $odmin $odmax 2 3 3  1>/dev/null
mv bmrg.seg swt1.seg
bmrg bscl$PRFX $evmin $evmax 2 3 3  1>/dev/null
mv bmrg.seg swt2.seg

#...down hole swc phone gathers (RADIAL=ch2)
bmrg bscl$PRFX $odmin $odmax 2 2 2  1>/dev/null
mv bmrg.seg swr1.seg
bmrg bscl$PRFX $evmin $evmax 2 2 2  1>/dev/null
mv bmrg.seg swr2.seg

#...down hole swc phone gathers (VERTICAL=ch1)
bmrg bscl$PRFX $odmin $odmax 2 1 1  1>/dev/null
mv bmrg.seg swv1.seg
```

```
bmrg bscl$PRFX $evmin $evmax 2 1 1  1>/dev/null
mv bmrg.seg swv2.seg
#-------------------------------------------

# Notation: component/location/sum or dif/
#           tddf=Tcomponent, down-hole, difference
#           trdf=Tcomponent, reference, difference
#           tdsm=Tcomponent, down-hole, sum
#           trsm=Tcomponent, reference, sum
#
#DIFFERENCE ENHANCEMENT
#    downhole (side wall clamping phone)
bsum swt$az90.seg swt$az270.seg -1.0
mv bsumswt$az90.seg tddf.seg

bsum swv$az90.seg swv$az270.seg -1.0
mv bsumswv$az90.seg vddf.seg

bsum swr$az90.seg swr$az270.seg -1.0
mv bsumswr$az90.seg rddf.seg

#    reference phone at surface
bsum rft$az90.seg rft$az270.seg -1.0
mv bsumrft$az90.seg trdf.seg

bsum rfr$az90.seg rfr$az270.seg -1.0
mv bsumrfr$az90.seg rrdf.seg

bsum rfv$az90.seg rfv$az270.seg -1.0
mv bsumrfv$az90.seg vrdf.seg
#-------------------------------------------

#SUM ENHANCEMENT
#    downhole (side wall clamping phone)
bsum swt$az90.seg swt$az270.seg +1.0
mv bsumswt$az90.seg tdsm.seg

bsum swv$az90.seg swv$az270.seg +1.0
mv bsumswv$az90.seg vdsm.seg

bsum swr$az90.seg swr$az270.seg +1.0
mv bsumswr$az90.seg rdsm.seg

#    reference phone at surface
bsum rft$az90.seg rft$az270.seg +1.0
mv bsumrft$az90.seg trsm.seg

bsum rfr$az90.seg rfr$az270.seg +1.0
mv bsumrfr$az90.seg rrsm.seg

bsum rfv$az90.seg rfv$az270.seg +1.0
mv bsumrfv$az90.seg vrsm.seg

#-------------------------------------------
#make links to alias names like Merge
ln -s tddf.seg twav.seg
ln -s trdf.seg tref.seg
ln -s vdsm.seg pwav.seg
ln -s vrsm.seg vref.seg
```

```
#SH-WAVE SHAPING
#...take last trace as a target
bkil trdf.seg 1 1 1 $eklmax    1>/dev/null
bstk bkiltrdf.seg              1>/dev/null
bscl bstkbkil.seg 1 1 0 $oklmax.0  1>/dev/null
mv bsclbstk.seg targ.seg

#...determine shaping filters and apply to T-Reference rft*.seg
bshp trdf.seg targ.seg 1 1 $tmin $tmax $npf $stab
#...apply shaping filters to downhole T-data          swt*.seg
bshp trdf.seg targ.seg 1 0 $tmin $tmax $npf $stab tddf.seg
mv bshptddf.seg twave.seg
#-------------------------------------------

#P-WAVE SHAPING
#...take last trace as a target
bkil vrsm.seg 1 1 1 $eklmax    1>/dev/null
bstk bkilvrsm.seg              1>/dev/null
bscl bstkbkil.seg 1 1 0 $oklmax.0 1>/dev/null
mv bsclbstk.seg varg.seg

#...determine shaping filters and apply to V-Reference rfv*.seg
bshp vrsm.seg varg.seg 1 1 $tmin $tmax $npf $stab
#...apply shaping filters to downhole T-data          swv*.seg
bshp vrsm.seg varg.seg 1 0 $tmin $tmax $npf $stab vdsm.seg
mv bshpvdsm.seg pwave.seg
#-------------------------------------------

#...quick view of results
bequ twave.seg 0 .25
bplt bequtwav.seg 3 0 0 1 500 0  .5  1  2 200
mv bplt.jpg twave.jpg

#...quick view of results
bequ pwave.seg 0 .025
bplt bequpwav.seg 3 0 0 1 500 0  .25  1  2 200
mv bplt.jpg pwave.jpg

echo "  "
echo "----------------------------------------"
echo "|                                      |"
echo "|To display QC plots:                  |"
echo "|                                      |"
echo "| SH-Wave (horizontal rotated data):   |"
echo "|                                      |"
echo "|       display twave.jpg              |"
echo "|                                      |"
echo "| P-Wave (vertical component data):    |"
echo "|                                      |"
echo "|       display pwave.jpg              |"
echo "|                                      |"
echo "----------------------------------------"
```

C Appendix (*Merge2*)

The following bash script is used in down-hole surveys to form the various resorted data sets. This script is designed for a vertical impact source with a load-cell signal on channel 7.

```
#!/bin/sh
#  $Id: Merge2,v 1.1.1.1 2017/04/13 21:40:01 pm Exp $
#set -x
# USE THIS FOR 7 TRACE DATA SETS:
# Merge2:  basic script to merge traces into P-wave gather and parse off load_cell trace
# assumes channels in following order 1,2,3 down-hole, 4,5,6 reference, 7=load_cell
# !!!Does NOT do enhanced processing. Simple trace extraction with bmrg program.!!!
#
# This combines all the shot gathers into receiver gathers for the different
# geophone components in a down-hole survey.
#
#    NOTE:!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
# This script was designed for a vertical impact source. There is no S-wave enhancement
# since the source is excited with only one polarization.
#
#     Copyright (c) 2017 Paul Michaels
#     <pm@cgiss.boisestate.edu>
#     This program is free software; you can
#     redistribute it and/or modify it under the terms
#     of the GNU General Public License as published
#     by the Free Software Foundation; either version
#     2 of the License, or (at your option) any later
#     version.  This program 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 General Public
#     License for more details.
#     You should have received a copy of the GNU
#     General Public License along with this program;
#     if not, write to the Free Software Foundation,
#     Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

#set -x

#...define parameters  YOU MUST DEFINE THESE FOR EACH NEW SURVEY
#                      SINCE IT IS LIKELY THAT THE NUMBER OF RECORDS
#                      WILL VARY WITH EACH SURVEY !!!!

#...define parameters
odmin=01
odmax=83
PRFX=brot10

#pull off load_cell channels
bmrg $PRFX $odmin $odmax 1 7 7 1>/dev/null
mv bmrg.seg ldcl.seg
echo "created ldcl.seg (load cell traces)"

#pull off vertical reference phone
bmrg $PRFX $odmin $odmax 1 4 4 1>/dev/null
mv bmrg.seg refv.seg
echo "created refv.seg (vertical reference traces) "
```

```
#pull off radial   reference phone
bmrg $PRFX $odmin $odmax 1 5 5 1>/dev/null
mv bmrg.seg refr.seg
echo "created refr.seg (radial   reference traces) "

#pull off transverse reference phone
bmrg $PRFX $odmin $odmax 1 6 6 1>/dev/null
mv bmrg.seg reft.seg
echo "created reft.seg (transverse reference traces)"

#pull off vertical down-hole phone
bmrg $PRFX $odmin $odmax 1 1 1 1>/dev/null
mv bmrg.seg swcv.seg
echo "created swcv.seg (vertical swc phone traces) "

#pull off radial   down-hole phone
bmrg $PRFX $odmin $odmax 1 2 2 1>/dev/null
mv bmrg.seg swcr.seg
echo "created swcr.seg (radial   swc phone traces) "

#pull off transverse down-hole phone
bmrg $PRFX $odmin $odmax 1 3 3 1>/dev/null
mv bmrg.seg swct.seg
echo "created swct.seg (transverse swc phone traces)"

echo done
```

D GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each

party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program 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 General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author> This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'. This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library.

If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

E IBM LICENSE

The following license was included with the library downloaded in the archive, **libascii.tar.Z**. The only function from this library included in BUS is file, xdrfloa.c, which is used to make BUS library libibm.a.
<http://www-03.ibm.com/systems/z/os/zos/features/unix/libascii.html

```
*********************************************************************
* libascii - ascii-ebcidic interface layer - README file            *
* Version 1.1.9                                                     *
*                                                                   *
* To report problems or ask questions send e-mail to:               *
*                                                                   *
*             libascii@nvet.ibm.com                                 *
*                                                                   *
* Copyright:   Licensed Materials - Property of IBM.                *
*              (C) Copyright IBM Corp. 1997, 1998.                  *
*              All rights reserved.                                 *
*                                                                   *
* License information:                                              *
*   The libascii source code is provided free of charge and         *
*   may be distributed freely.  No fee may be charged if you        *
*   distribute the libascii source code (except for such things     *
*   as the price of a disk or tape, postage ).  The libascii        *
*   makefile will compile and produce a libascii.a archive file.    *
*   The libascii.a archive may be link edited  with any software    *
*   vendor  product.  Any software vendor product that is link      *
*   edit with libascii.a archive is free to distribute and charge   *
*   for that product.                                               *
*                                                                   *
*   THIS PROGRAM IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY       *
*   KIND, EXPRESS  OR IMPLIED, INCLUDING THE IMPLIED  WARRANTIES    *
*   OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.        *
*   IBM does not warrant uninterrupted or error free operation of   *
*   the  Program, or that the Program is free from claims by a      *
*   third party of copyright, patent, trademark, trade secret,      *
*   or any other intellectual property infringement. IBM  has       *
*   no obligation to provide service, defect correction, or any     *
*   maintenance for the Program. IBM  has  no  obligation to        *
*   supply any Program updates or enhancements to you even if       *
*   such are or later become available.                             *
*                                                                   *
*    Under no circumstances is IBM liable for any of the            *
*    following:                                                     *
*                                                                   *
*      1.  third-party claims against you for losses or damages;    *
*      2.  loss of, or damage to, your records or data; or          *
*      3.  direct  damages,  lost  profits, lost savings,           *
*          incidental, special, or indirect damages or other        *
*          consequential damages,  even if IBM or its authorized    *
*          supplier, has been advised of the possibility of         *
*          such damages.                                            *
*                                                                   *
*    Some jurisdictions do not allow these limitations or           *
*    exclusions, so they may not apply to you.                      *
*********************************************************************
```

Index