Stanford Exploration Project, Report 102, October 25, 1999, pages 187–??

Additions to SEPlib

Robert G. Clapp[1]

keywords: *not available*

ABSTRACT

The release of this report coincides with the release of new version of the SEP software "SEPlib". The new version of SEPlib contains a significant number of new features that allows better handling of 3-D data. The release also includes support for running Seismic Unix (SU) programs with SEP data, a new installation mechanism, and machine independent makefile rules that should allow easy building of SEP reports.

INTRODUCTION

SEPlib in its current form began in the early 1990s as a collection of utility and graphic programs (Claerbout, 1991; Dellinger and Tálas, 1992). In 1994 Nichols et al. (1994) introduced a new I/O interface that made working with files, pipes, and sockets a much simpler matter. In 1995 SEP began adding the ability to work with 3-D and irregular data, creating SEP3D (Biondi et al., 1996). These changes allowed SEP students to begin working with 3-D data, but it was notoriously difficult. The initial software was buggy (especially when dealing with large files), the process to read the data could be convoluted, and there was a lack of utility programs that could deal with 3-D data. In the process of adding the ability to deal with 3-D datasets and new platforms, the amount of time spent on general code maintenance decreased. As a result SEPlib became difficult to install and difficult to use.

The new version of SEPlib fixes many of the problems seen in earlier versions. A number of new functions and programs have been added to deal with irregular data and large data sizes. In addition two new libraries have been added. The first simplifies the accessing of data, and the second, complimentary, library allows SU programs to be used with SEPlib datasets. To make it easier to use SEPlib a new installation mechanism has been added. The new install mechanism not only makes it easier to install SEPlib, but also to use the software written for this and other SEP reports.

[1]**email:** bob@sep.stanford.edu

3-D SUPPORT

The design of SEPlib3D seemed ideal for 3-D data. Unfortunately, as is often the case, the path from design to implementation was littered with seen and unforeseen hurdles. At this time we have overcome many of these stumbling blocks.

Large files

One problem when handling 3-D data is there large size. Much of SEPlib used an `int` to specify the location within a file. Unfortunately, the dynamic range of an `int` is limited to 2GB. As a result, many of the SEPlib library had to be rewritten to handle the additional file size in a rather opaque manner[2] that could still efficiently access files. In addition to dealing with the 2GB limit in our own software, we had to overcome problems with Unix systems and standards that did not account for more than 2GB file sizes. For example, only recently has Linux begun to support file sizes over 2GB and portable `tar`'s are limited to 2GB files. To get around these limitations Dave Nichols wrote some preliminary support for multiple-file datasets. This support has been expanded upon to allow the user to create a dataset composed of multiple files, each not exceeding a user provided file size in Megabytes `filesize`.

The support for multiple file dataset provided two other benefits. First, the dreaded `File system full` error is avoided. By specifying multiple directories for the datapath `datapath=/scrka1/bob/;/scrka2/bob/` SEPlib will switch the directory it's writing its binary data to when the file system is full or when a user-specified size limit (`dirsize`) is reached. Second, `Cat3d` can create a virtual SEPlib dataset by concatenating and updating grid and header pointers, but leaving the large binary data files untouched.

The initial implementation of SEPlib3D did not allow piping between programs. As a result, many large, intermediate results were required. The new version of SEPlib allows piping by opening up additional sockets for the header and the grid. However piping is only allowed between SEPlib3D programs when certain conditions are met:

- `init_3d()` is called at the beginning of the program
- `sep_3d_close()` is called before the first writing of data
- no parameters (such as the number of traces) are changed after `sep_3d_close`
- data is written sequentially by the first program and read sequentially by the second
- the first and second programs read and write the same type of information (data, header, and/or grid)

[2]There is no machine independent way to specify an integer with more dynamic range.

Superset

SEP3D is good at dealing with 3-D data, but requires a significant coding overhead. As a result Clapp and Crawley (1996) wrote SEPF90, a Fortran90 library that simplified dealing with 3-D data. Unfortunately the design, like all early prototypes, head serious limitations. Among them,

- it forced referencing through a structure to access the data (something that was incredibly slow with early Fortran90 versions)
- it required programs to be written in Fortran90 (even though many programs are more suited for C)
- it did not easily allow for headers and data to be read separately (a very powerful option in SEP3D)

The new version of SEPlib comes with a replacement for SEPF90, `superset`. The purpose of `superset` is the same, but the implementation is significantly different. The basic idea of `superset` is to maintain an invisible *sep3d* structure copy of each SEPlib3D dataset. The structure contains

- the type of data (float, complex, byte, integer)
- the type of SEPlib file (regular, header, grid)
- the axes of the data (`n,o,d,label,unit`)
- the header keys associated with the data (key name, key type, key format)
- current section of the grid being processed
- current section of the headers being processed
- mapping from the headers to the traces

This internal structure can be initialized through a SEPlib tag, from another structure, or created manually by the programmer. Information is passed to and from the structure through a `sep3dtag`.

Reading of any SEPlib data then can be done in two simple steps: First the programmer makes a call to read in the headers (either all or a portion) and is returned the number of headers read. The library will automatically read in the grid, find the valid headers, check for a `data_record_number` and create a list of pointers to the traces. Once the headers have been read the user can ask for all of the data associated with the header block to read in, or read in sections of the data.

Writing is also simple. The programmer first initializes the output format files. He then makes a call(s) to write data (data, headers, and/or grid), and finally asks for the number of traces in the dataset to be updated in the format files if it wasn't known until the end of the program. The library does all the work figuring out what files to write, what trace number it is currently writing out, etc.

For added convenience I also wrote a F90 module which provides wrappers around the C function calls. The module allows the programmer to access a Fortran90 type which contains all properties of the dataset (except the header and grid values). The programmer than can then access and modify these values. When done they can synchronize the C and Fortran90 version. This added flexibility further simplifies dealing with SEPlib3D data.

SU SUPPORT

The creation of the `superset` library made possible another new element included in this release of SEPlib: the ability to use SU programs with SEPlib data. A SEPlib3D dataset with a header is similar but more free form than the SU format. SEPlib3D allows the user to have any number of keys in any order, named arbitrary, and doesn't require them to be in the same order as the data. SU data is a single file containing a series of traces. Each trace is made up to 82 keys and data. Access is done almost exclusively sequentially, through the `puttr` and `gettr` routines. The routines `gettr` and `puttr` are in turn aliased to **f**`gettr` and **f**`puttr`, where the **f** refers to a file. The library `sepsu` contains two new routines: **t**`gettr` and **t**`puttr`, which instead use the `sep3dtag` to access the data. These two routines are calls to the `superset` library read and write routines with a conversion to and from the SU `segy` structure. To add a little more flexibility the library provides some additional command line arguments:

nmem the library buffers it reads and writes. **nmem** is the number of traces that buffers.

sukey=sepkey tells the library that **sepkey** should be treated as this **sukey**

sukey.fract=val tells the library to scale the key value it reads by **val** and writes by **1./val**

suinput tells the program that the input file is in SU format

suoutput tells the program that the output file should be in SU format

The following program converts from SEPlib3D to SU, and shows just how easy it is to write code that can take advantages of both software packages.

Example

```
segy tr;
int main(int argc, char **argv)
{
  int i;
```

```
  int verb;
  /* hook up getpar */
  initpar(argc,argv);  getch_add_string("suoutput=1");
  initargs(argc, argv);
  verb=1000000;
  getch("verb","d",&verb);
  requestdoc(1);  i=0;
  if (!gettr(&tr)) err("can't get first trace");
  do {    i++;
    fputtr(stdout,(&tr));
    if(i%verb==1) fprintf(stderr,"converted %d  traces \n",i);
  } while (gettr(&tr));
  return EXIT_SUCCESS;
}
```

In the above, the `gettr` reads SEPlib3D and the `fputtr` writes out SU data. One thing to note is the `EXIT_SUCCESS`. In order to make SEPlib3D data work the total number of traces must be known. The `EXIST_SUCCESS` call is aliased to a call to `finish_susep` which updates the number of traces if it has changed within the program. To compile and run SU programs you need to:

- have the SEPlib include directory specified before the SU include directory
- link with `libsepsu.a` and `libsuperset.a` before linking with any of the SU libraries

INSTALLATION

For the last few years SEP has required anyone who wanted to install SEPlib to wade through and edit a series of complicated configuration Makefiles. This system was convenient for us, but difficult to near impossible for anyone outside SEP. Part of the difficulty was that SEPlib relied heavily on machine dependent `ifdefs` for:

- compiler options
- locations of programs and libraries
- different locations of functionality of the basic C libraries

With each new version of on operating system we were left with the choice of creating more `ifdef`'s or not completely supporting the old OS. Looking at past experience I was left with three choices: continue the `ifdef` cycle, simplify SEPlib so it wasn't reliant on so many non-standard operations (required less `ifdef`), or find a different way to do the installation. I chose to follow the third option, and found some tools that were well suited for the purpose.

Automake, Autoconf, Libtool

Gnu's Not Unix (GNU) provides a series of utilities that install easily and run successfully on an amazing number of platforms. Their installation mechanism involves running a shell script `configure` that examines features of an operating system and then creates machine dependent makerules, header files, and code. From the distributor point of view this involves learning to integrate two programs, `Autoconf` and `Automake` into your software.

Autoconf Creates a configuration script that determines machine dependent features

For the distributer `Autoconf` can remove the need for almost all machine dependent `ifdef`'s in your code. Unfortunately, the removal of these `ifdef`'s does not come without a cost. Generally machine dependent `ifdef`'s are replaced with feature dependent `ifdef`'s, and a resulting test is written to determine whether the installing machine has the feature. For example replacing a machine dependent `ifdef` which was dependent on whether numbers represent in big or little Endian fashion with corresponding `ifdef` and test.

Automake Creates makefile rules for creating libraries and binaries from a list of source files and required linked libraries

Generally `Automake` replaced all of SEP's makerules by replacing our standard shorthand conventions with a more generic shorthand convention. The advantage of the `Automake` philosophy is it is faster, provides automatic distribution, install, and clean mechanisms that are not too easy to implement.

Both programs do an excellent job in supporting variations in C compilers but are significantly lacking in their Fortran support. As a result the writing of many Fortran tests and the changing of the `Automake` source code were necessary. For instructions on how to install SEPlib see the attached Appendix.

REPRODUCIBLE RESEARCH

The idea that research is reproducible is one of the basic tenets of good research. SEP has a long history in attempting to define and produce reproducible research. Unfortunately what we have discovered is that our definition of "reproducible" has always been very subjective, and on the most part limited to reproducible at SEP.

SEP's began its quest for reproducibility with the introduction of interactive documents (Claerbout, 1990). We then started producing SEP reports on CDROM and adopting the GNU, makefile system as our framework for reproducibility (Schwab et al., 1996). This method effectively created a snapshot of the current SEP environment and guaranteed that if you had a certain type of a computer, a certain OS, and the CDROM you could reproduce SEP research. Unfortunately, OS's change quickly and CDROMS aren't an ideal distribution mechanism. Recently we have abandoned CDROMs and instead use a web based method to distribute our research, but we still make an attempt to test every report for reproducibility (Prucha, 1999). The web based solution is a better distribution method but requires you to have

- the author's source code
- the version of SEPlib they used to produce the results
- the a version of the SEP makefile rules that they used to produce the results

Overall, a system that requires even more work than the CDROM solution.

This version of SEPlib attempts to correct some of these issues. The SEPlib configure script will produce a set of SEPlib makerules that are tuned to the platform where it is run, but should compile all of the SEPlib codes. This system allows SEP research to reproduce on any platform where the configure script can be successfully run.

ACKNOWLEDGMENTS

I would like to thank Matt Schwab who pointed a number of years ago that GNU configure was the way to go.

REFERENCES

Biondi, B., Clapp, R., and Crawley, S., 1996, Seplib90: Seplib for 3-D prestack data: SEP–**92**, 343–364.

Claerbout, J. F., 1990, Active documents and reproducible results: SEP–**67**, 139–144.

Claerbout, J. F., 1991, Introduction to Seplib and SEP utility software: SEP–**70**, 413–436.

Clapp, R. G., and Crawley, S., 1996, SEPF90: SEP–**93**, 293–304.

Dellinger, J., and Tálas, S., 1992, A tour of SEPlib for new users: SEP–**73**, 461–502.

Nichols, D., Karrenbach, M., and Urdaneta, H., 1994, What's new in SEPLIB?: SEP–**82**, 257–264.

Prucha, M., 1999, Revisiting the half-derivative filter: SEP–**102**, 137–142.

Schwab, M., Karrenbach, M., and Claerbout, J., 1996, Making scientific computations reproducible: SEP–**92**, 327–342.

APPENDIX A

Programs in **bold** are new to SEPlib, programs in *italics* have been significantly modified (often converted from Fortran 77 to C or Fortran 90).

UTILITIES

Programs that perform utility (non-seismic) functions.

Unix

This group of programs perform functions similar to their corresponding UNIX commands, but on SEPlib datasets.

Program	
Cp3d	Copy a history and binary file
Ls	List all SEPlib history and associated binary files
Mv3d	Move a SEP77 history and binary file
Rm3d	Remove a history and binary file
Zero	Create a null SEPlib file

Vector

Programs that perform vector operations, where the vector is defined as an SEP dataset.

Program	**Purpose**
Add	Add two SEPlib real datasets
Cabs	Take the Complex absolute value of a complex dataset
Cabs2	Take the Complex absolute values squared of a complex dataset
Cadd	Add two complex datasets
Conj	Take the conjugate of a complex dataset
Math	Generic routine to do vector operations on files
Log	Take the log of a file
Scale	Scale a dataset
Cmult	Perform complex multiplication of two datasets
arithpar	Perform mathematical operations on SEPlib history file parameters
Cmplx	Create a complex dataset from two real datasets
Rtoc	Convert a real dataset to complex by setting the imaginary part to 0
Imag	Take the imaginary part of a complex dataset
atoF	convert typed numbers to a SEPlib dataset
Real	Take the real part of a complex dataset
Pow	Raise a SEPlib dataset to a power

Cube

Do operations that rely on the SEPlib hypercube format.

Program	**Purpose**
Cat	concatenate SEPlib-2D datasets
Merge	Merge SEPlib-2D datasets
Pad	Pad with zeros a SEPlib Dataset
Window3d	Window a SEPlib dataset
Reverse	Reverse an axis of a 2-D dataset
Transp	Transpose an axis of a 2-D dataset
Interleave	Interleave the traces of two 2-D datasets
Window	Window a 2-D dataset
Cat3d	Concatenate SEPlib-3D datasets
Fold3d	Calculate the fold of a SEPlib dataset
Grid_fold	Calculate the fold of a given set of gridding parameters
Sort3d	Sort and grid a SEPlib-3D dataset
Stack3d	Stack a SEPlib-3D dataset
Stack	Stack a SEPlib-2d dataset

Headers

Programs that operate on header values.

Program	**Purpose**
Headermath	Perform mathematical operation on header keys
Create3d	Create a SEPlib-3D dataset from SEPlib-2D dataset(s)
Synch3d	Synch the headers and data of a SEPlib-3D dataset
Window_key	Window a SEPlib dataset according to a key

Info

Programs that give information about SEPlib datasets.

Program	**Purpose**
Attr	Calculate the attributes of a SEPlib data file
Dd	Convert back and forth from an ascii-binary file
Disfil	Display the values of a SEPlib data set
Get	Get values from a SEPlib history file
In3d	Print a description of a SEPlib dataset
Dis3dhead	Display header keys
Attr3dhead	Display the attributes (min,max,etc) of header keys

SEISMIC

Programs that do seismic operations.

Filter

Programs that do some type of filtering.

Program	Purpose
Again	Arctangent gain
Agc	Automatic gain control
Balance	Trace balancing
Gpow	Apply a gainpow to a dataset
Lpfilt	Low-pass (Butterworth) filtering
Clip	Clip a dataset
Interp	Interpolate a SEPlib dataset
Trcamp	Calculate total amp/energey
Ft3d	Fourier transform a complex dataset
Energy	Calculate energy in a running window
Edit	Patching driver
Fx2d	2-D FX decon
Txdec	2-D TX decon
Spectra	Obtain averaged amplitude spectra
Bandpass	Bandpass a dataset
Envelope	Analytic signal amplitude
Halfint	Take a half-derivative of a dataset
Smooth	Smooth a dataset with triangular filters
Filter	Filter a dataset
Mute	Mute a SEPlib-2D dataset
Noise	Add noise or create noise
Median	Median filter a dataset
Tpow	Raise a dataset by a power of the time axis

Model

Programs that do modeling.

Program	Purpose
Aniso2d	Anisotropic heterogeneous elastic modeling
Iso2d	Isotropic heterogeneous elastic model
Gauss	Create a model with Gaussian blobs
Spike	Create a model with spikes
Kirmod3d	Perform v(z) Kirchoff modeling
Vel	Create a velocity model
Marine_geom3d	Create a typical marine geometry
Gfgradz	Create v(z) traveltimes and amplitudes
Scat3d	Create scaterers (for input to Kirmod3d)

Velocity

Programs that work with velocity.

Program	Purpose
Unmo	Inverse nmo
Vconvert	Convert velocity (z-t) (interval-depth)
Velan	Perform 2-D velocity analysis
Velan3d	Perform 3-D velocity analysis
Hypint	Velocity space transform (integration)
Hypmovie	Velocity space transform movie
Hypsum	Velocity space transform
MCvfit	Monte Carlo automatic velocity picks
Stretch	Generic stretching routine
NMO	Perform NMO on SEPlib-2D dataset
Nmo3d	Perform NMO on a SEPlib-3D dataset
Radial	Transform a 2-D dataset to radial trace domain
Radnmo	Do radial NMO

Travel times

Routines that calculate travel times.

Program	Purpose
FMeikonal	Calculate travel times using fast marching eikonal
MTTmaps	Band limited maximum energy Green's function maps
Hwt3d	Rays or travel times using Huygens wavefront tracing

Imaging

Programs that to do imaging.

Program	Purpose
Phase	Do phase shift migration
Stolt	Do stolt migration

Anisotropy

Software dealing with anisotropic data.

Program	Purpose
Combine/combine	Combine two sets of elastic layer coefficients
Uncombine/uncombine	Subtract layer coefficients
Uncrack/uncrack	Compare fractured and unfractured rock sample
Crack/crack	Crack a set of elastic layer coefficients
Create/create	Create a stiffness tensor
Laymod/laymod	Create a layered of model
Laymod21/laymod21	Create a layered model with all 21 coefs.
Laymodthin/laymodthin	Create elastic parameter model with thin interbedding
Laythin1d/laythin1d	Create elastic parameters with two inter-bedded substances
Display/display	Display the parameters of a anisotropic dataset
Rotate/rotate	Rotate the coordinate frame

GRAPHICS

Programs that display graphics.

SEP VPLOT INTERFACE

SEP programs that interface with vplot to display functions.

Program	**Purpose**
Box	Draw a vplot balloon
Dots	Create a dot plot
Graph	Graph a function
Grey	Display a 2-D plane variable density plane
Histogram	Create a histogram
Operplot	Shaded box plot
Thplot	Hidden line plotting program
Ftplot	Plot Fourier amplitude or phase spectrum
Wiggle	Create a wiggle plot
Contour	Contour a dataset
Cubeplot	Create a cubeplot of a 3-D dataset

VPLOT

Vplot shell programs and pens for different devices.

Program	**Purpose**
Vppen/vppen	Generic vplot plotting routine
Xtpen/xtpen	XT vplot pen
vp_Movie	Combine vplot frames into a movie
vp_OverUnderAniso	Plot vplot frames vertically
vp_OverUnderIso	Plot vplot frames vertically, preserve ratio
vp_Overlay	Overlay vplot frames
vp_SideBySideAniso	Plot vplot frames horizontally
vp_SideBySideIso	Plot vplot frames horizontally, preserve ratio
vp_Unrotate	Unrotate vplot frames
vp_annotate	Annotate a vplot frame
plas	Convert vplot file to ascii
pldb	Convert vplot file to binary
Raspen/raspen	Create raster files
Pspen/pspen	Create postcript files
Tube/tube	Display a vplot file
pstexpen	Convert a vplot file to postscript

Movie

Programs that diplay 3-D volumes.

Program	**Purpose**
Rickmovie	A great movie program

TOOLS

Program	**Purpose**
sat	Handle f77 dynamic and temporary allocation, SEPlib shorthand
saw	Handle f77 memory, SEPlib shorthand
sawf90	Handle SEPlib shorthand in fortran90
cmsawf	Handle SEPlib shorthand for the CM5
ExtractPOD	Extract Pod documentation from f90 source code
Makedepend	Create f90 depenicies according module usage
key-word	Search directories for keywords
shortfort	Fold f77 lines
spp	Generic pre-processor
lop2f90	Convert linear-operator shorthand to ratfor
ratfor77	Convert from ratfor to fortran77
ratfor90	Convert from ratfor90 to fortran90
Snoop	Search directories for keywords

CONVERTERS

Programs that convert from or to SEP internal formats.

Program	**Purpose**
Segy2sep	Convert from SEG-Y to SEPlib
Sep2su	Convert from SEPlib to SU
Su2sep	Convert from SU to SEPlib
Byte2mpeg	Convert a SEPlib byte file to mpeg format
pstogif	Convert a postscript to gif
vplot2gif	Convert a vplot file to gif
vplot2mpeg	Convert a vplot file to mpeg
vplot2ras	Convert a vplot fike to a raster plot

APPENDIX B

INSTALLING SEPLIB

SEPlib now uses a GNU-style configure mechanism for installation. So far this installation mechanism has been tested on:

- Linux (Redhat 5.0,5.2,6.0)
- IRIX6.5
- DecAlpha
- HPUX (10.1)
- Solaris (2.6)

It is the first release with this new installation mechanism so please report bugs to `bob@sep.stanford.edu`. Follow the following steps to install SEPlib

- Download the software, ftp://sepftp.stanford.edu/pub/sep-distr/seplib-5.0.tar.gz
- `gunzip seplib-5.0 -c |tar xf -`
- `cd seplib-5.0`
- Make sure lex or flex is in your path
- `./configure`
- `gmake install`

Following the above procedure should install the core seplib libraries and programs into the directory `/usr/local/SEP`.

–prefix=`/other/directory` specify another directory to install SEPlib in

–with-local install the less tested, newer portions of SEPlib

–with-su=`/su/directory` Compile SU support

–with-motif=`/motif/directory` Specify motif directory or **–without-motif** the absence of motif

–with-ppm=`/my/pbmplus/dir/` The directory containing the pbmplus package.

If you run into problems (for example you need to add an additional library path when compiling programs) you can often solve your problem by setting environmental variables that the configure script will then use. For example:

F90 The F90 compiler

F77 The F90 compiler

CFLAGS F90 compiler flags

F77FLAGS F77 compiler flags

F90FLAGS F90 compiler flags

LDFLAGS Directories and libraries to link when compiling C programs

F77LDFLAGS Directories and libraries to link when compiling F77 programs

F90LDFLAGS Directories and libraries to link when compiling F77 programs

CPFLAGS C Processor flags

etc Look configure.in the main directory to find other variables that can be set in the environment

APPENDIX C

SETTING UP THE SEP ENVIRONMENT

Before running SEPlib do the following:

create /.datapath SEPlib files are composed of ascii/binary pairs. The ascii portion describes the data (the size, the type, and the location of the binary). The binary portion is the actual data. The two are separated to allow processing to be done in a centralized location (a home directory for example) while the data is written where ever there is space. The datapath file tells SEPlib where to put binary data and should look something like this:

```
datapath=/scrka3/bob/;/scrka2/bob/
spur     datapath=/scrka2/bob/
oas      datapath=/scrsa1/bob/
vesuvio  datapath=/SDA/bob/
santorin datapath=/scrsa4/bob/
```

By default SEPlib first checks the command line for `datapath=`, then the directory where the program is run for a `.datapath` file, and finaly the home directory. The above `.datapath` files tells SEPlib to put binary data by default in `/scrka3/bob` and if it runs out of space in `/scrka2/bob`, but when on the computer "santorin" to put the data in `/scrsa4/bob`.

setenv VPLOTSPOOLDIR `/tmp/` The next step is to tell SEPlib where to put temporary vplot files. It is best to put these in a location such as `/tmp` which is periodically cleaned.

setenv VPLOTFONTDIR `SEPINCDIR`

This tells vplot where to find fonts.

setenv SEPINC `SEPINCDIR`

The final step is only necessary if you want to compile and run programs from SEP reports, theses, or books. This environmental variable is needed by our `Makefile`'s to find out its compile and install rules.