Stanford Exploration Project, Report 73, November 18, 1997, pages ??–??

A tour of SEPlib for new users

Joe Dellinger and Sándor Tálas[1]

keywords: *SEPlib, algorithm, graphics*

ABSTRACT

SEPlib is a data-processing software package developed at the Stanford Exploration Project. Many researchers at universities and oil companies would find SEPlib extremely useful if they could only get over the initial hump of learning how SEPlib programs are used. By working through several illustrative examples this document attempts to painlessly introduce novice users to the fundamental SEPlib concepts they will need to understand the available SEPlib self-documentation and manual pages. Concepts covered include self-documenting programs, history files and data files, SEPlib-style pipes, command-line and history-file "getpar" parameters, auxiliary input and output, and how to use some of the more useful SEPlib utilities.

INTRODUCTION

Researchers spend too much time hacking together personal special-purpose programs to do their work, usually reinventing "square wheels" again and again in the process. By sharing programs in a common framework researchers can spend their programming time writing only those special-purpose programs they need for their particular work.

The SEPlib software package evolved at the Stanford Exploration Project over the last decade as a partial solution to this problem. SEPlib is organized like UNIX; most programs perform relatively basic operations. Complex manipulations are performed by stringing these simple programs together via pipes and files. If you like this philosophy of programming you will probably find SEPlib easy and natural to use. Advanced users fluent in FORTRAN or C can quickly write their own SEPlib utilities; if well written, these can be added to the pool of common programs available for all to use.

While documentation does exist for SEPlib, it is minimal and mostly intended for users already familiar with UNIX, FORTRAN, and C. Traditionally, beginning SEPlib users have learned the basics directly from more experienced users, and only later

[1]**email:** not available

with experience are they able to use the software-locating utilities and documentation that exists to discover and use new programs on their own. You may not have the opportunity to learn SEPlib this way, so we have prepared this document to help get you started.

Accessing SEPlib depends on your site, so we have relegated this information to an Appendix. Assuming you can type the name of a SEPlib program and have it run and have some sort of graphical output device available, this document will take you on a "guided tour" through several of the most important SEPlib programs and utilities. On the way, if you pay close attention, you should be able to pick up the basic knowledge you will need to proceed further on your own. There will be a brief summary of the important points at the end.

After you have finished reading and trying out the examples in this document, try reading Claerbout (1990), and at a more advanced level Claerbout (1986) to learn more.

GETTING GOING

On-line version

If you are already reading this document on a computer you will probably have no problems running SEPlib programs. Let's see whether it works or not.

Click on this box with the middle button of the mouse: make **test** pushd ./../joe1;cake welcome;popd

Did it work? Great!

If you want to try the examples as you read along (and we strongly encourage you to try it) open a window for yourself or simply click on this box: make **window** pushd ./../joe1;cake window;popd

Paper version

If you are reading the paper version of this document and you think that your computer is set up to run SEPlib, test it by carefully typing:

```
Spike n1=1000 k1=300 title="\s200 Welcome to \c2 SEP\c5 lib" \
 | Bandpass fhi=2. phase=1 | Wiggle | Tube
```

This might look pretty frightening at a first glance, but if it works you are evidently able to run SEPlib. If it did not work, go look at the Appendix on "Basic troubleshooting" before continuing.

General SEPlib features

Although SEPlib currently consists of over 100 different programs, they share many common features. First of all, by convention SEPlib programs start with a Capital Letter. More importantly, most SEPlib programs are "filters"; they read from standard input and write to standard output:

```
Prog < input   > output
```

Complex functions are created by joining filters with pipes:

```
Prog1 < in | Prog2 | Prog3 > out
```

We will use the SEPlib program "`Wiggle`" to demonstrate some basic SEPlib-program properties. "`Wiggle`" is just a simple program that converts raw data into wiggle traces, but it is used a lot because people are usually curious to see what they have done to their data.

Try typing:

```
Wiggle
```

You should get a couple of screens' worth of documentation. This is called "self documentation": run the program without arguments, input, or output, and it will display a brief documentation summary. Almost all SEPlib programs will self document, which is good because very few of them have real manual pages.

If you get an error message `doc(): No such file or directory`, complain to the person who installed SEPlib at your site! If you don't feel like complaining (perhaps because you are that person) and you know where the SEPlib source is installed, you can tell SEPlib programs an alternate place to look for their source by setting the environment variable `SEP_DOC_PATH`. See the "Basic troubleshooting" Appendix.

Getting the test data

Hopefully you have now worked out any software-environment problems you might have had before, and you are ready for your "test drive". Make a directory somewhere convenient (like under your home directory) and `cd` or `pushd` (as you prefer) into it. (If you are reading the electronic document and opened the window by pushing the button on the previous page, you will find yourself in the directory where the data is. We still recommend that you make a private directory for test purposes.)

You need some data in this directory to experiment on. To get some, do:[2]

```
cp /sepcd2/sep73/joe1+ \verb/Tx?.H .+
mkdir Data
cp /sepcd2/sep73/joe1+ \verb/Data/Tx?.H@ ./Data+
```

(If the files were not there, tell the person who installed SEPlib on your computer that they need to customize this document for your particular site by modifying the list of site-dependent paths kept in the file "sitepaths.tex".)

You should now be the proud owner of six files: `Data/Txx.H@`, `Data/Txy.H@`, and `Data/Txz.H@`, the actual data files, and `Txx.H`, `Txy.H`, and `Txz.H`, the so-called "history" files. History files contain all the information about the data that is needed by SEPlib programs, such as the path to the corresponding data file. This way you can keep large data files in places where they are more welcome. We will introduce history files in detail in the next section.

Plot one of the data files by doing

```
Wiggle < Txx.H | Tube
```

When you run the command above `Wiggle` creates a plot which `Tube` then tries to display on your screen. Did it work? Hopefully your screen will look something like the one in Figure **??**. (If when you try it "`Tube`" does the plot using `Xtpen`, like in the figure, exit the program by clicking on the `QUIT` button at the top of the window. If you are using some other sort of graphics device you may have to hit return or space to exit, or the program may simply exit when the plot is done without any prompting from you.) Try plotting `Txy.H` and `Txz.H` too.

Hardcopy

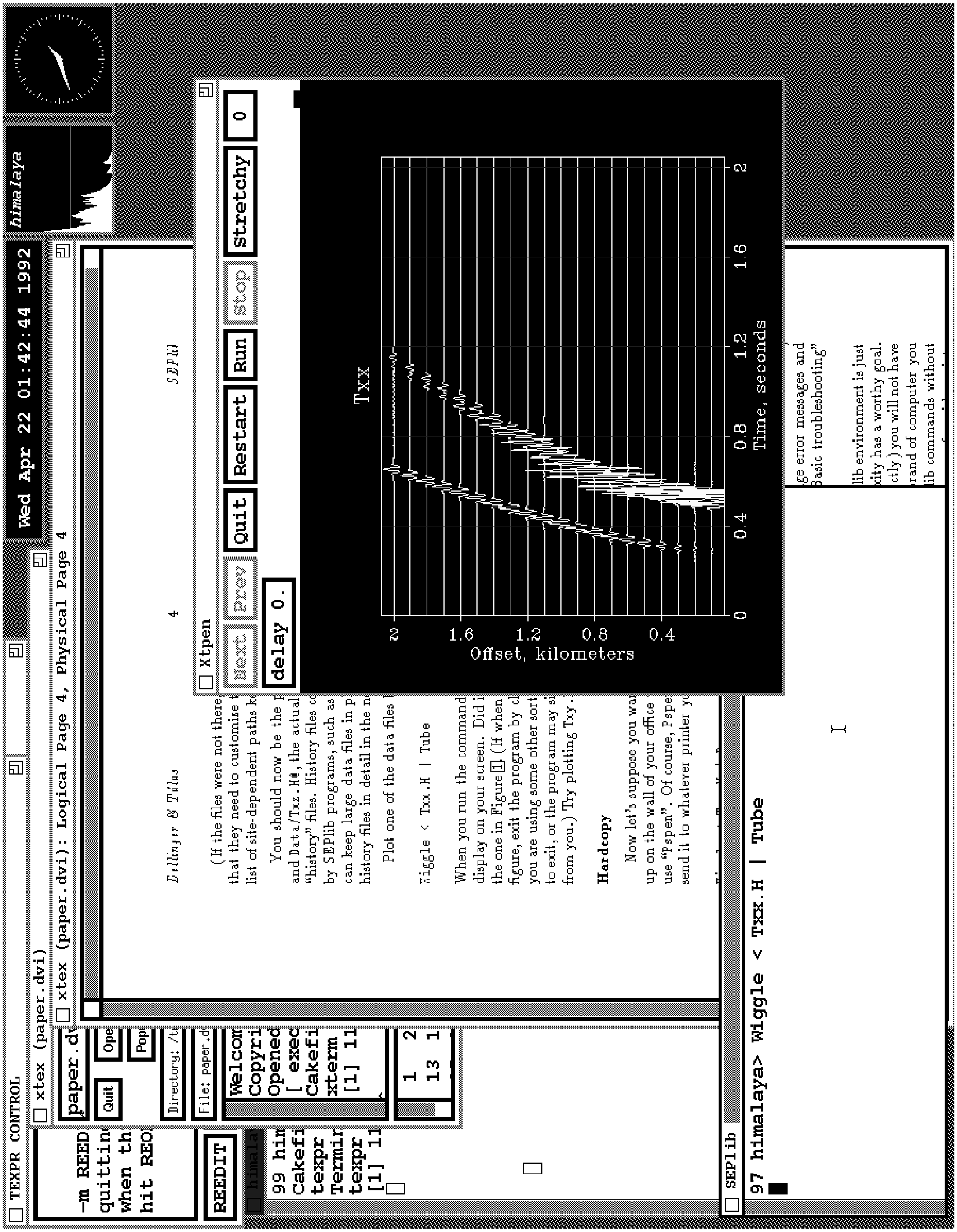
Now let's suppose you want a hardcopy plot instead, something that you can put up on the wall of your office to demonstrate your efforts. Simple: instead of "`Tube`" use "`Pspen`". Of course, `Pspen` has to know where to send the plot. By default it will send it to whatever printer your local machine thinks is called "postscript". Try:

```
Wiggle < Txx.H | Pspen
```

Hopefully this will work, producing something like Figure **??**.

If the printer your workstation calls "postscript" turns out to be old and slow, on another floor or in another building, or you often get strange error messages and

[2] The final ".+'' on the first line is *not* a period to end the sentence. We assume that you are familiar with basic UNIX commands. If not, we strongly advise you to read the first couple of chapters of any introductory book on UNIX before continuing.


Wed Apr 22 01:42:44 1992
himalaya
TEXPR CONTROL
REEDIT
xtex (paper.dvi)
xtex (paper.dvi): Logical Page 4, Physical Page 4
Xtpen
Next
Prev
Quit
Restart
Run
Stop
stretchy
0
delay 0.
Txx
Time, seconds
Offset, kilometers
SEPlib
97 himalaya> Wiggle < Txx.H | Tube