

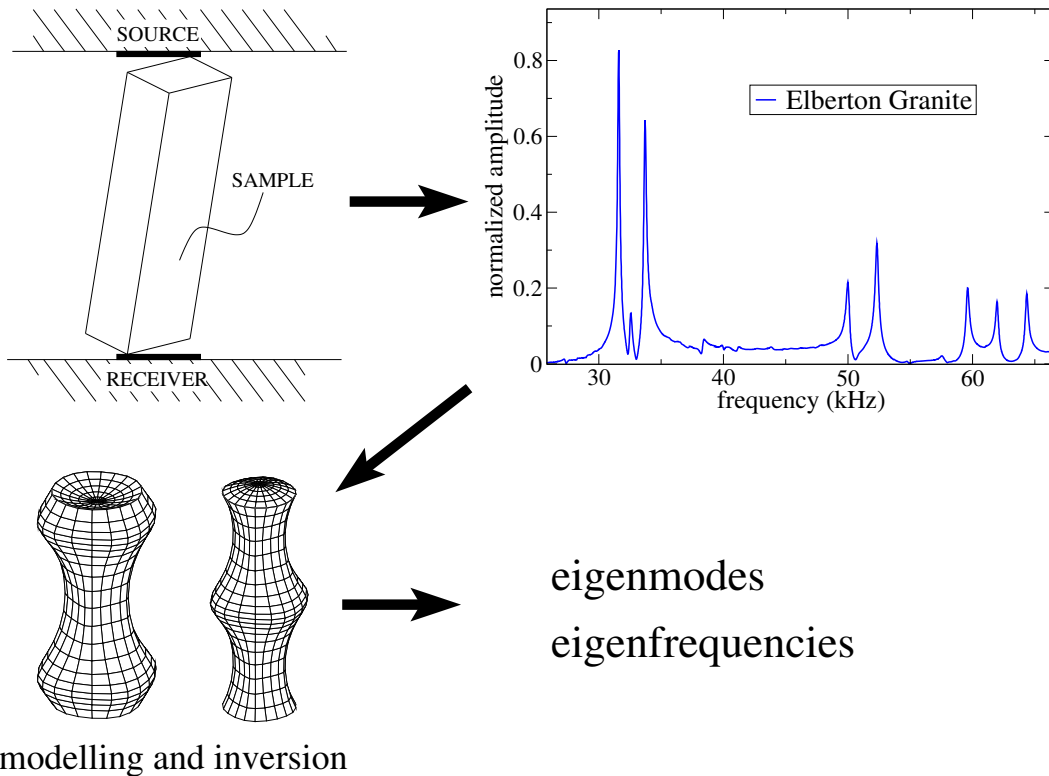
Installation and Help Guide to

Fitspectra¹ and RUS-inverse²

Resonance peak fitting, forward modeling and inversion

by ¹Brian Zadler and ²Jérôme H.L. Le Rousseau

Physical Acoustics Lab Center for Wave Phenomena
Department of Geophysics
Colorado School of Mines
Golden, CO 80401



Contents

1	Introduction	1
2	Downloading the necessary files	2
2.1	Fitspectra for Matlab: spectrum fitting code	3
2.2	Fitspectra for Scilab: spectrum fitting code	4
2.2.1	If you DO NOT have Scilab installed	4
2.2.2	If you DO have Scilab installed	4
2.2.3	Scilab and Plotlib are installed	4
2.3	C code: dependencies	6
2.3.1	SU libraries: libsu, libpar, libcwp and Makefile.config	6
2.3.2	libm	7
2.3.3	libblas, liblapack, libg2c	7
2.4	Forward model: formod_aniso.c	8
2.4.1	Testing the formod_aniso install	8
2.5	Inversion: RUS-inverse.c	10
2.5.1	Testing the RUS-inverse install	11
3	Example: aluminum cylinder	12
3.1	Spectrum fitting of aluminum	13
3.2	Setting up for inversion of c_{ij}	16
3.2.1	Constructing the parameter input file: data	18
3.3	Inverting for c_{ij}	19
3.3.1	Running RUS-inverse	20
3.3.2	The final results	21

3.4	EXPLANATIONS OF AREAS ON THE GUI AND THEIR BUTTONS . .	23
3.4.1	Windows	23
3.4.2	Plot options	23
3.4.3	Data options	23
3.4.4	Save options	24
3.4.5	Plot range	25
3.4.6	Get peaks	25
3.4.7	Directories	26
3.4.8	Files	26
3.4.9	Miscellaneous buttons	26
3.5	Some important files and directories	27
3.6	Useful formulas and relations	27
3.7	Anisotropic materials	28
3.8	Details	28

Chapter 1

Introduction

This manual is to be used as an installation guide and a help manual for the associated resonance modeling codes:

- fitspectra - Scilab and Matlab codes for resonance peak fitting
- formod-aniso - C code for normal mode forward modeling
- RUS-inverse - C code for stiffness tensor inversion

These tools are used in the Physical Acoustics Laboratory at the Colorado School of Mines for performing resonant ultrasound spectroscopy on rocks and other materials.

This *HELP* manual is divided into (hopefully) easy-to-use chapters. These cover what files to download, how to install them, how to use them, and some extra information that may be of use.

Chapter 2

Downloading the necessary files

Separate from this manual, you should first download the following files:

- **rusGuiScilab.tar.gz** if you like Matlab
- **rusGuiMatlab.tar.gz** if you like Scilab
- **forinv.tar.gz**

Second, find a permanent directory where you would like these programs to be. One suggestion is to make the following directory structure in your home directory.

```
mkdir resonance
```

Then move the downloaded files inside `\home\[your username]\resonance` and untar them.

```
mv rusGuiScilab.tar.gz rusGuiMatlab.tar.gz forinv.tar.gz resonance\  
cd ~/resonance  
gunzip *  
tar -xvf forinv.tar  
tar -xvf rusGuiScilab.tar  
tar -xvf rusGuiMatlab.tar
```

If you do an `ls`, you'll see there are 3 new directories containing the untarred files: **rusGuiScilab/**, **rusGuiMatlab/** and **modeling/** which contains the subdirectories **for/**, **inv/** and **example/**. We'll cover the spectrum fitting codes first, then the modeling and inversion.

2.1 Fitspectra for Matlab: spectrum fitting code

A few small differences aside, the Matlab GUI works very similar to the Scilab GUI which is described in extensive detail in the following sections.

Descend in the `rusGuiMatlab` directory. Start Matlab from here first. Type `'fitspectra'` into the Matlab console to open the GUI. The most important thing to note is that unlike the Scilab GUI, the Matlab version only reads data files from the current Matlab directory... there is no directory switching option. You can change directories inside the Matlab console if necessary. This will update the GUI.

A note on file extensions. There is a pulldown box of different file extensions under the file list box. Be sure to select the appropriate one. If necessary, more extensions can easily be added or removed inside the `fitspectra.m` file.

To start, you will see 3 files in the file list box. Left click on `sample1.dat`. You will see two resonance peaks. The only button available to you is the 'Get Peaks' button, so click on it. Now left click near the apex of the two peaks (that means two separate clicks) and hit the ENTER button to finish. A rough starting model is shown in red which may or may not be close to your data. Don't worry, Matlab is smart. Now click the 'Fit Model' button. After a few seconds, you should have two fit peaks. The parameters are shown below as B0, B1, C, D, Gam and Freq.

The model being fit is a Breit Wigner model, which consists of a constant background parameter B0, a linear background parameter B1, and 4 other parameters per peak. That means 6 parameters for 1 peak, 10 for 2 peaks, etc. The parameter C is the peak amplitude, D is the skewness, Gam is the peak width and Freq is the resonance frequency. Below the blue save button is a pull down menu labeled 'Peak #'. We've just fit 2 peaks, but only 6 parameters are showing in the GUI, the 2 background parameters and the 4 parameters specific to peak #1. Use the pull down menu to select peak #2. You'll see that B0 and B1 stay the same, but the C, D, Gam and Freq now show the fit values for the second peak.

Just click the save button to save the peaks. Three files are created. The `*.params` file keeps the full list of parameters, which may not be useful to you. The `*.fq` file stores in a 2 column format each fit resonance frequency and its associated Q value (resonant frequency divided by peak width). The last file, `*.f`, contains two columns. The first column holds the resonance frequencies, the second column holds the default weighting values that will be used later with the inversion code RUS-inverse. These two columns are preceded by a single number which simply states how many resonance frequencies are in the file. The usefulness of this will be explained in Section 3.3. Values are appended to the files as necessary.

The Scilab version of `fitspectra` has a more detailed explanation. Please refer to it for help with spectrum fitting and general resonant ultrasound spectroscopy techniques.

2.2 Fitspectra for Scilab: spectrum fitting code

Go ahead and descend in the `rusGuiScilab` directory. Since Fitspectra is written in Scilab, a Matlab clone, you must have Scilab installed on your Linux system. The current version can be downloaded from <http://www-rocq.inria.fr/scilab>.

Also needed is the Matlab-like plotting library written by Stephane Mottelet, available at <http://www.dma.utc.fr/mottelet/plotlib/> This gives Scilab a Matlab-like plotting feel.

2.2.1 If you DO NOT have Scilab installed

Download the latest version from the site mentioned above. It is likely you will need to have root access in order to complete the installation. Follow the instructions given with Scilab, as they are quite thorough. Once Scilab is running cleanly, continue.

2.2.2 If you DO have Scilab installed

Again, as mentioned above you need to install the Matlab-like plotting library *plotlib*. Please try to use either of the two links provided above. Follow the instructions provided for you for a Linux or Unix installation. Again, you may need to have root access on your system to do this.

2.2.3 Scilab and Plotlib are installed

Almost done. In your **resonance/** directory, you should see a directory called **rusGuiScilab** which contains all of the .sci files for Fitspectra. In your *home directory* either make(if you don't have one) or edit your .scilab file and include the lines

```
getd [path of fitspectra.sci file]
mprintf(' loading FITSPECTRA\n').
```

Be ABSOLUTELY sure to put a carriage return at the end of the second line. Scilab is picky about that. For me, this is:

```
getd /home/bzadler/computing/scilab/gui/
mprintf(' loading FITSPECTRA\n')
```

This command executes a `getf()` on every .sci file in the specified directory, loading all of the functions defined in the .sci files. The `mprintf(...)` is just a print statement so that when you

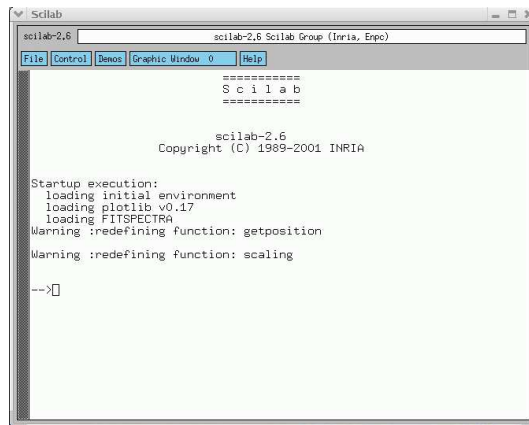


Figure 2.1: The scilab environment with Plotlib v0.17 and FITSPECTA loaded correctly.

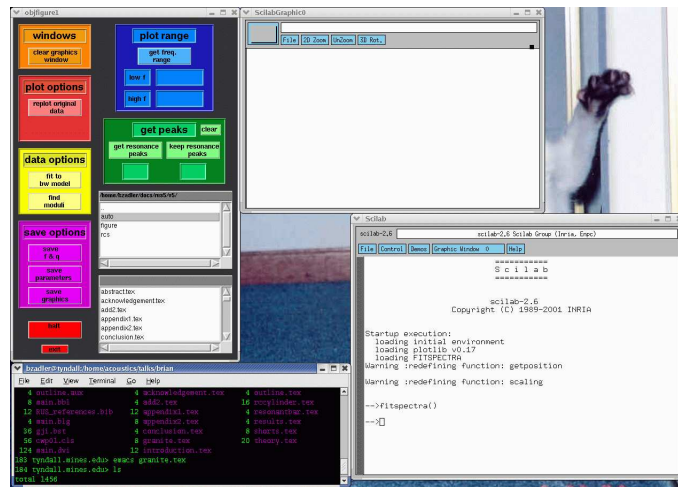


Figure 2.2: Fitspectra up and running.

run SCILAB, you know that the .scilab file executed and that Fitspectra has been loaded correctly. It is possible that newer versions of Scilab put the .scilab file into the directory .Scilab/scilab-x.x.x/, so watch for this.

That's it! Now open Scilab in a new terminal window. You should see something that looks like Figure 2.1. In the Scilab window type

```
-- > fitspectra
```

to run the program. Fitspectra should load and look like Figure 2.2. That's it. For a detailed example on using Fitspectra, see Chapter 3 Good luck, and enjoy!!


```
# Makefile for ...su/main
include ${CWPROOT}/src/Makefile.config
B = ${HOME}/bin
D = $L/libcwp.a $L/libpar.a $L/libsu.a
OPTC = -O0 -Wall -ansi -pedantic -DHAVE_UNISTD_H
LFLAGS= -lm -L$L -lsu -lpar -lcwp -lblas -llapack -lg2c
PROGS = $B/formod_aniso

INSTALL: $(PROGS)
touch $@

${PROGS}:
$(CC) $(CFLAGS) $(@F).c $(LFLAGS) -o $@
@chmod 755 $@
@echo $@ installed in $B

remake:
-touch *.c
-rm -f ${PROGS}
$(MAKE)

clean:
rm -f a.out junk* JUNK* core

-- Makefile (Makefile CVS:1.1.1.1)--L11--All-----
Wrote /home/bzadler/RUS/RUS_src/Makefile
```

Figure 2.3: A snapshot of the formod-aniso Makefile.

2.3 C code: dependencies

The C codes used for generating the eigenfunction and eigenfrequencies have quite a few dependencies on outside libraries and packages. I'll try to take you through this step-by-step. All of these packages should be installed during the the SU installation, but in case something is amiss, I give links to other sites where these packages can be found.

You shouldn't need *root* privileges on you system to install most of these, but if you need help, ask your System Administrator. And remember, Linux only (as far as I know).

Both C codes, formod-aniso.c and RUS-inverse.c are compiled via a *Makefile* which looks like Figure 2.3. The list of packages that are needed are given by the line

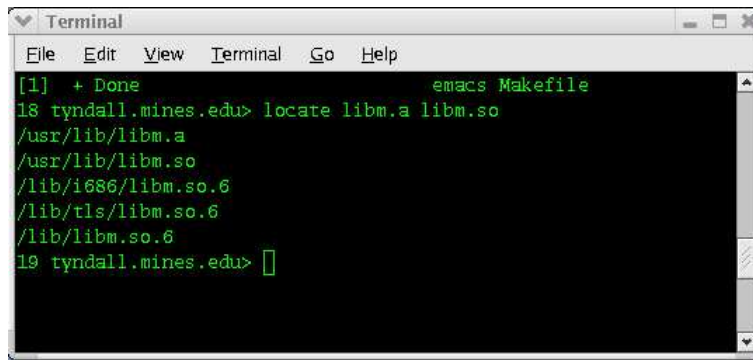
LFLAGS= -L\$L -lsu -lpar -lcwp -lm -lblas -llapack -lg2c.

As noted, all of these installed during the Seismic Un*x package installation.

2.3.1 SU libraries: libsu, libpar, libcwp and Makefile.config

These packages are contained in the Seismic Un*x (SU) package available from the Center for Wave Phenomena (CWP) at the Colorado School of Mines. The download site is

<http://www.cwp.mines.edu/cwpcodes/index.html>.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Terminal, Go, Help). The prompt is "18 tyndall.mines.edu>". The command "locate libm.a libm.so" has been entered. The output is displayed in green text: "/usr/lib/libm.a", "/usr/lib/libm.so", "/lib/i686/libm.so.6", "/lib/tls/libm.so.6", and "/lib/libm.so.6". The prompt "19 tyndall.mines.edu>" is visible at the bottom.

```
Terminal
File Edit View Terminal Go Help
[1] + Done emacs Makefile
18 tyndall.mines.edu> locate libm.a libm.so
/usr/lib/libm.a
/usr/lib/libm.so
/lib/i686/libm.so.6
/lib/tls/libm.so.6
/lib/libm.so.6
19 tyndall.mines.edu> 
```

Figure 2.4: What you'll see if libm is installed.

Please refer to the download and installation instructions given there, as they are straightforward and complete.

Makefile.config should also come with the SU tools. This will be needed when you finally run the *make INSTALL* command once all of the dependencies are ready. It may require a little editing (for GNU make users). Just read the README files if you have install problems.

2.3.2 libm

Libm is a standard math library for C. It is part of just about every glib development library. Try a *locate libm.a libm.so* to see if you have these files. If you do, you'll see something like Figure 2.4. If not, download them from

<http://rpmfind.net>.

Just do a search for *libm.a*, download one of the packages, and install it via rpm. These should be installed during the SU installation process, however.

2.3.3 libblas, liblapack, libg2c

The last three libraries you should also check to see if they are already installed. Just use *locate libblas liblapack libg2c*. If they aren't installed, go to <http://rpmfind.net> and download them from there. Again, if you've installed the SU package, they should already be there.

That's it for now. Let's take a look at our C source code!

2.4 Forward model: `formod_aniso.c`

OK, let's get ready to compile the first C file. First, make sure the line

```
include $(CWPROOT)/src/Makefile.config
```

is pointing to the correct location. For me, this is

```
/usr/local/cwp/src/Makefile.config
```

If you don't already have a **bin/** directory in your home directory, then create one. That is where we'll be putting our compiled executables. OK, here we go. Descend into the **for/** directory and type the following command.

```
>> make INSTALL
```

If you get warnings about explicitly defined functions, just ignore them. Otherwise, that should be it. If you are getting errors during the compile, make sure you have the necessary files downloaded and installed from Sections [2.3.1](#), [2.3.2](#), and [2.3.3](#). If all else fails, ask your System Administrator or a friend.

2.4.1 Testing the `formod_aniso` install

Also inside the **for/** directory is a little shell script by the name **mod-shell**. It is a quick test to check whether `formod_aniso` is functioning correctly. Just type

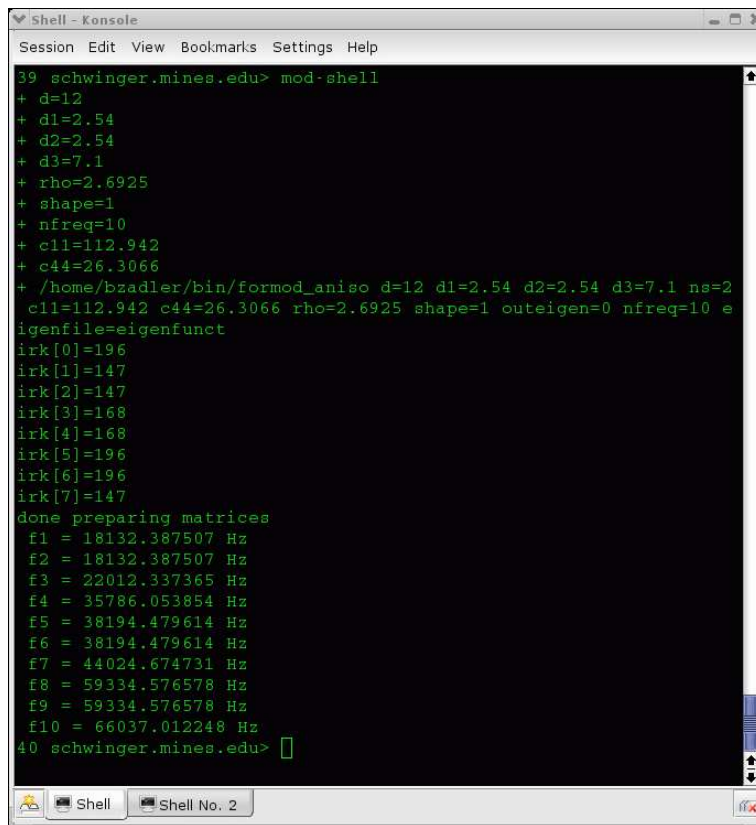
```
>> mod-shell
```

at the command line in the terminal. If `./` isn't in your Linux path, then you need to type

```
>> ./mod-shell
```

instead. You should see an output like Figure [2.5](#). Congratulations. Let's compile the inversion code.

Note that you can use this little shell script to forward model your materials. Without delving into details, you need only change the material properties and parameters in this file forward model your parallelepipeds, cylinders or spheres.



```
39 schwinger.mines.edu> mod-shell
+ d=12
+ d1=2.54
+ d2=2.54
+ d3=7.1
+ rho=2.6925
+ shape=1
+ nfreq=10
+ c11=112.942
+ c44=26.3066
+ /home/bzadler/bin/formod_aniso d=12 d1=2.54 d2=2.54 d3=7.1 ns=2
  c11=112.942 c44=26.3066 rho=2.6925 shape=1 outeigen=0 nfreq=10 e
  igenfile=eigenfunct
irk[0]=196
irk[1]=147
irk[2]=147
irk[3]=168
irk[4]=168
irk[5]=196
irk[6]=196
irk[7]=147
done preparing matrices
f1 = 18132.387507 Hz
f2 = 18132.387507 Hz
f3 = 22012.337365 Hz
f4 = 35786.053854 Hz
f5 = 38194.479614 Hz
f6 = 38194.479614 Hz
f7 = 44024.674731 Hz
f8 = 59334.576578 Hz
f9 = 59334.576578 Hz
f10 = 66037.012248 Hz
40 schwinger.mines.edu> 
```

Figure 2.5: An example of output from running **mod-shell**.

```

Terminal
File Edit View Terminal Go Help
irk[6]=126
irk[7]=90
ndata=6
y0=0.018242
y1=0.018289
y2=0.022200
y3=0.035668
y4=0.038340
y5=0.038397
starting eigenvalues calculation
eigenvalues calculation done
f1=0.018210
f2=0.018210
f3=0.022050
f4=0.035845
f5=0.038312
f6=0.038312
chisq=0.000004

starting eigenvalues calculation
eigenvalues calculation done
f1=0.018209
f2=0.018209
f3=0.022050
f4=0.035845
f5=0.038312
f6=0.038312
chisq=0.000004

```

Figure 2.6: An example of output from running **RUS-inverse**.

2.5 Inversion: **RUS-inverse.c**

Descend into the **inv/** directory. We need to make a couple changes to the C code. Open **RUS-inverse.c** in your favorite editor, and move down to lines 197, 198, and 343. Change them from

```
(197) char measurement[]="/home/bzadler/RUS/RUS_LM/freq_data";
```

```
(198) char parameters[]="/home/bzadler/RUS/RUS_LM/data";
```

```
(343) char freqs[]="/home/bzadler/RUS/RUS_LM/predictedf";
```

to

```
(197) char measurement[]="/home/[your username]/resonance/example/freq_data";
```

```
(198) char parameters[]="/home/[your username]/resonance/example/data";
```

```
(343) char freqs[]="/home/[your username]/resonance/example/predictedf";
```

There are two filenames stated explicitly inside the **RUS-inverse.c** file, **freq_data** and **data**. These two files contain the observed resonance frequencies and input parameters,

respectively. The details of these files are explained in Section 3.2. The file **predictedf** will contain the predicted frequencies from the output of the inversion. Once the example has been finished, feel free to use any directory you'd like for these files. Just remember to re-make the executable if you change the paths in the **RUS-inverse.c** file. Now, type the following command.

```
>> make INSTALL
```

If you get warnings about explicitly defined functions, just ignore them. Otherwise, that should be it. If you are getting errors during the compile, make sure you have the necessary files downloaded and installed from Section 2.3. If all else fails, ask your System Administrator or a friend.

2.5.1 Testing the RUS-inverse install

Inside the **example/** directory are two files, **data** and **freq_data**. We'll use these two files to do a quick check of the install. A detailed example for an aluminum cylinder is given in Chapter 3. Go ahead and descend into the **example/** directory. Then type

```
>> RUS-inverse.
```

If things are working correctly, you should see an output on the terminal screen like Figure 2.6. This output will repeat, as it is performing iterations in a least-squares minimization. The chisquared value should be about $\text{chisq}=0.000010$ or less and it won't drop much, since the starting values for the fit are close to the true values. If this is the case, it is working fine and you should continue to the detailed example in the next section.

Chapter 3

Example: aluminum cylinder

This chapter will take you through a detailed example of using all of these resonance tools together. The example I included is a 2.5 cm diameter, 7 cm long cylinder of standard aluminum. Aluminum generally has a Young's modulus of about 70-72 GPa and a shear modulus of 24-26 GPa.

The example data set recorded at PAL is shown in Figure 3.1. The data was acquired using an aluminum cylinder mounted between two piezo-electric transducers. A swept sine wave of 10 V_{PP} was used to drive the sample over a frequency range of 10KHz to 100KHz.

We're going to step through using Fit-spectra to load the data file, selected resonance peaks, and fit them with a Breit and Wigner model. This will give us three files when complete: **j.dat.f**, **j.dat.fq**, and **freq_data**. The first file, **j.dat.f**, contains a list of the peaks that were fit in chronological order. The second file, **j.dat.fq**, contains the same frequencies along with the associated Q value of the resonance peak. The third file, **freq_data**, contains the frequencies once again and an associated weight for each one. This file will be used in the following steps.

With the data (observed frequencies) collected into the file **freq_data**, we'll then construct the file **param_data** which will contain the measured parameters of the aluminum cylinder and starting values for the inversion. The inversion will attempt to use the starting stiffness coefficients for our aluminum model, which will be assumed to be isotropic, to pro-

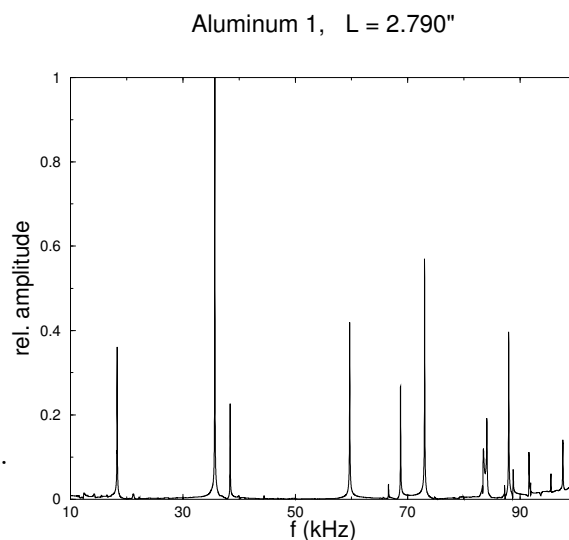


Figure 3.1: An example of the resonance spectrum of an aluminum cylinder recorded at the PAL.

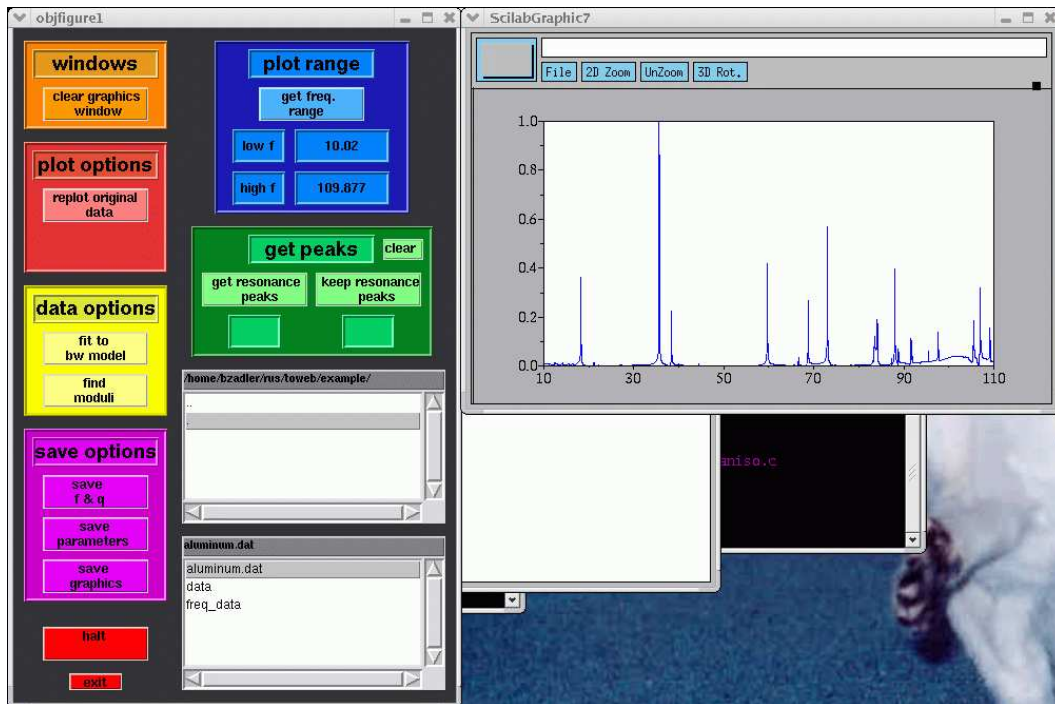


Figure 3.2: Fitspectra up and running with the aluminum test data set loaded.

duces a frequency spectrum exactly like our observed spectrum. The final result is a set of predicted frequencies, hopefully close to the observed ones, and a final *fit* value of the stiffness coefficients that produced that spectrum. Those coefficients can be easily transformed into elastic moduli and velocities.

3.1 Spectrum fitting of aluminum

Open a new terminal window. Descend into `home/[your username]/resonance/modeling`. Open Scilab, then open Fitspectra. In the directory list box you should see **for/**, **inv/**, and **example/**. Single-click **example/** to highlight it (if it isn't already), then single-click on the button above the list box. This should move you into the **example/** directory and in the file list box at the bottom right you should see three files: **param_data**, **freq_data**, and **aluminum.dat**. Select **aluminum.dat** and single-click the file button above it. This will load the data set and you should see it plotted as in Figure 3.2.

Looking at the spectrum, it looks like the first peak is around 15-20KHz. So, we want to select a range of frequencies around the resonance peak. To do this, click the *get freq. range* button in the blue section and move the mouse cursor onto the plot. Click the left button ONCE near 10 KHz (any height is OK) and ONCE near 30 KHz. The figure should replot the data over the frequency range you selected. We can chop the data set down a little more.

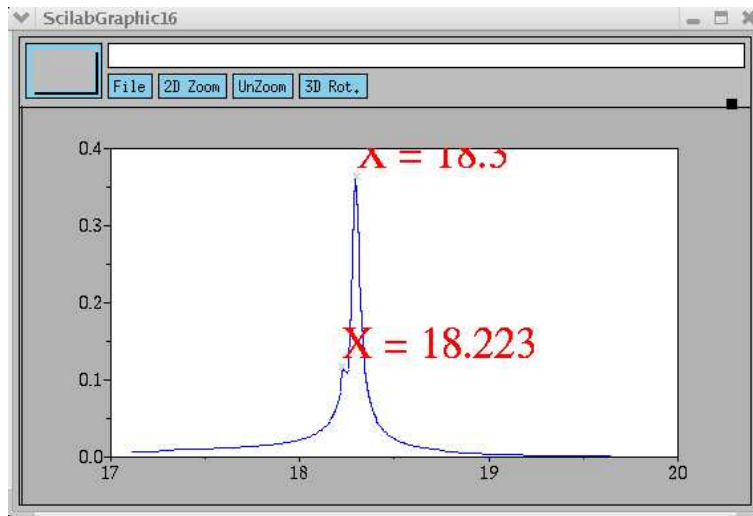


Figure 3.3: Fitspectrum up and running with the aluminum test data set loaded.

Click the *get freq. range* button again and select from about 15KHz to 21KHz. If you get in trouble, click the *replot original data* button. If all else fails, try to quit and reload Scilab, but you shouldn't have to do this.

You should now have the data from 15KHz to 21KHz showing. It looks like only one resonance peak, but if you click the *2D Zoom* button on the graphics window and select about a 2KHz window of just the peak itself, you can see it looks as if this is a degenerate mode since there is a small hump on the left side of the peak. So, let's try to fit it with 2 peaks.

Unzoom using the *UnZoom* button in the graphics window. Use the *get freq. range* button again and select just the range 17KHz to 20KHz. Now we can see the two peaks clearly. Click the *get resonance peaks* button. This enables you to select as many peaks as you want using the RIGHT button. When you are done, click the left button to stop. Move the mouse to the tip of the small peak and click the right button once. This should put a very small grey 'x' on that spot. Now click the right button again near the top of the large peak. Click the left button to finish. Try to always select the peaks in increasing order. This saves times sorting them later before using them in the inversion scheme. Your figure should now look like Figure 3.3.

Now we fit the two selected peaks by clicking the *fit to bw model* button. The first time you do this, rough starting values of the parameters are chosen automatically. In the terminal window you can watch the iterations of the fit. It stops when *simuls* hits 100. When finished, the f value on my decreased from $f = 0.3042950E+03$ to $f = 0.6117257E-02$. We can do better. On the scilab window, and output of the fit parameters is shown. I'll explain those later. Right now, we want to increase our 'goodness of fit'.

Click the *keep resonance peaks* button. This means you want to use the output of the

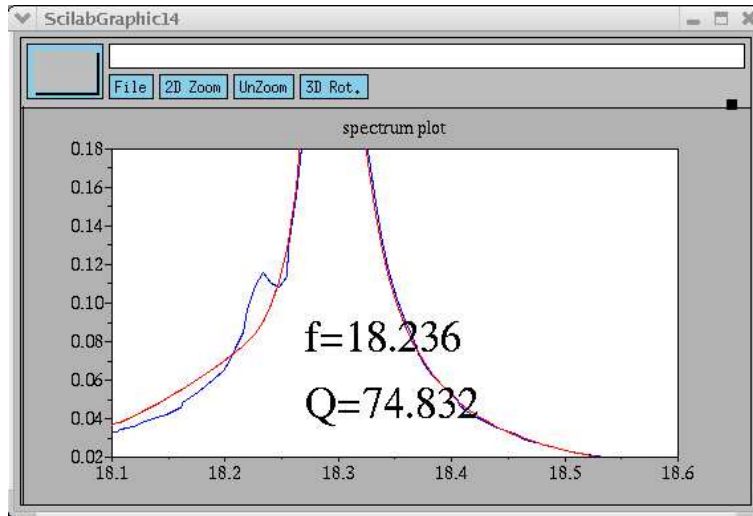


Figure 3.4: A zoomed view of the poor fit of what is guessed to be a low-response resonance peaks on the side of a prominent peak.

last fit to as the starting parameters of a new fit. So we're going to fit the peaks again, but this time we have much better 'starting values' if you'd like to think of it that way. Hit the *fit to bw model* button again. You can repeat this process until you're certain the fit will not converge further by observing the f = parameter in the terminal window. If you're unhappy with the fit, click the *clear* button in the *Get Peaks* section to erase your old choices and select new peaks. My best fit gave me

$$f = 18.236 \text{ and } Q = 76$$

$$f = 18.290 \text{ and } Q = 368$$

for my two peaks. It is difficult to say how accurate the fit to the smaller peak is since it is dominated by the large one (Figure 3.4). Therefore, I say I have good confidence in the $f = 18.290\text{KHz}$ peak and low confidence in the $f = 18.236\text{KHz}$ peak. Let's save this fit and move to the next peak.

Click the *save f and Q button* to open a new dialog box showing the two peaks we're saving, their confidences, and the path and filename to save them as. Change the confidence of 1 for the first peak to a zero for now since we're not completely sure the peak exists, anyway. Using the default path and filename, choosing the *select and save* button will create three files in a directory called **aluminum.dat_results/**. In the Scilab window you'll see output that looks like

New directory made: /home/bzadler/rus/example/aluminum.dat_results/

File created: /home/bzadler/rus/example/aluminum.dat_results/aluminum.dat.fq

File created: /home/bzadler/rus/example/aluminum.dat_results/aluminum.dat.f

File created: freq_data in directory /home/bzadler/rus/example/aluminum.dat_results/

For now we only care about the file **freq_data**. The other two files are explained in Section 3.4.4. In **freq_data**, the frequencies and confidences were saved along with the number 2.0 in the first line of the file. The value at the top of the file will change as we add more peaks and will come into play in the inversion.

Time to fit the next peak. Click *replot original data* to replot the full data set. Looks like the next peaks are at about 36KHz and 38KHz. Let's fit them separate for now. Zoom in the frequency range from about 34.5KHz to 37KHz and fit the first peak as we did the last two. After a few iterations, I get values of

$$f = 35.668 \text{ and } Q = 582$$

for the peak. Save that data as we did before, and be sure to use the SAME filenames and directories. Using the default names is safest for now. Later you can hack the Scilab code and change the default names if you want. I get

$$f = 38.397 \text{ and } Q = 579$$

for that peak. Looking very closely, there is something around 44KHz also. Basically, for the first 18 peaks I get the results in Table 3.1. I've no doubt that some peaks were missed. Since the data is from a cylinder, there are bound to be degenerate modes. the only one that stood out was at 18KHz. The ones at 83KHz, 84KHz and 97KHz looked suspicious, but we'll find out for sure when we run the forward modeling code.

3.2 Setting up for inversion of c_{ij}

Once you've fit all of the peaks (which is overkill, but fun), go ahead and close down Fit-spectra and Scilab. If you look at the **freq_data** file in an editor, it should look contain these lines:

```
5
0.018236 0.000000
0.018290 1.000000
0.035668 1.000000
0.038397 1.000000
0.044420 0.000000
0.059671 1.000000
0.066572 1.000000
0.068718 1.000000
0.073007 1.000000
```

$f^{(obs)}$ (Hz)	Q	confidence	freq. #
18236	76	0	1
18290	368	1	2
35668	582	1	3
38397	579	1	4
44420	1022	0	5
59671	669	1	6
66572	974	1	7
68718	890	1	8
73007	883	1	9
83467	333	1	10
84097	540	1	11
87256	1782	1	12
87973	1010	1	13
88748	1410	1	14
91565	1495	1	15
91902	93	0	16
95490	2452	1	17
97610	1075	1	18

Table 3.1: The first 18 peaks as far as I can tell from the example data set of an aluminum cylinder. Three of the modes are given zero weight because they are poorly defined and we're not sure if they are really there yet.

```
0.083467 1.000000
0.084097 1.000000
0.087256 1.000000
0.087973 1.000000
0.088748 1.000000
0.091565 1.000000
0.091902 1.000000
0.095490 1.000000
0.097610 1.000000
```

We're going to use the **freq_data** file in a moment, but first we need build ourselves a file containing all of the known properties of our aluminum cylinder.

3.2.1 Constructing the parameter input file: data

The file **param_data** needs the following information in order as they are listed.

```
order of polynomial to fit (6-14)
shape (0=rect piped, 1=cyl, 2=sphere)
# of cij's (2,3,5,6,9)
dim1 (cm)
dim2 (cm)
dim3 (cm)
density (grams/cm3)
lower f bound (Mhz)
upper f bound (MHz)
c11 guess (GPa, 109 Pa)
c44 guess (GPa, 109 Pa)
```

Lets start with a small order polynomial, like 6, to do a rough fit. The shape is a cylinder, which corresponds to the value 1. We're assuming the cylinder is isotropic for now, so there are 2 stiffness coefficients. The 3 dimensions for the cylinder are diameter, diameter, and length, all expressed in cm. These are 2.54508, 2.54508, and 7.0866.

The next line is the density, 2.6925 grams/cm³. The next two lines are the upper and lower bounds on the fit. Let's start with trying to fit just the first five frequencies, so that looks like 0.016MHz for a lower bound and 0.050MHz for an upper bound, just to be safe. Last of all, we need an inital guess for our two stiffness coefficients, c_{11} and c_{44} . Looking at literature, we'll start with values of 112 GPa and 28 GPa, respectively. Open a file in an editor, input the following lines:

```
6
1
```

```
2
2.54508
2.54508
7.6088
2.6925
0.016
0.050
112.
26.
```

and save the file as **param_data**.

Now we're ready to start our inversion to find the REAL moduli of this PARTICULAR piece of aluminum.

We now have the two essential files ready. If you remember back to Section 2.5, we edited the **RUS-inverse.c** file to look for **param_data** and **freq_data** in the **example/** directory. The two files currently residing in that directory aren't needed, but they are essentially the same as the two files we just created. Go ahead and rename the old files to **parms_data.old** and **freq_data.old** just in case you need to look at them later. Now, move **freq_data.old** from what should be

```
home/[your username]/resonance/modeling/example/aluminum.dat_results/
```

back one level to

```
home/[your username]/resonance/modeling/example/.
```

Now, go ahead and move **param_data** from wherever you created it to the same directory. That's it. We're ready to find the moduli of our sample.

3.3 Inverting for c_{ij}

Whenever I'm attempting to invert for the c_{ij} of a sample, I always have both **param_data** and **freq_data** open in emacs.

This is important for a few reasons. First, as the inversion converges toward the optimal results, you'll be changing the order of the polynomial from six to twelve or fourteen to get the best fit possible. Also during the inversion, you start with about 5 frequencies and just fit those first. Then you slowly add one or two more of your observed frequencies by changing the number at the top of the **freq_data** file from five to seven, then eight, etc. This seems to be the best way to make sure you don't miss any of the degenerate modes. Finally, it

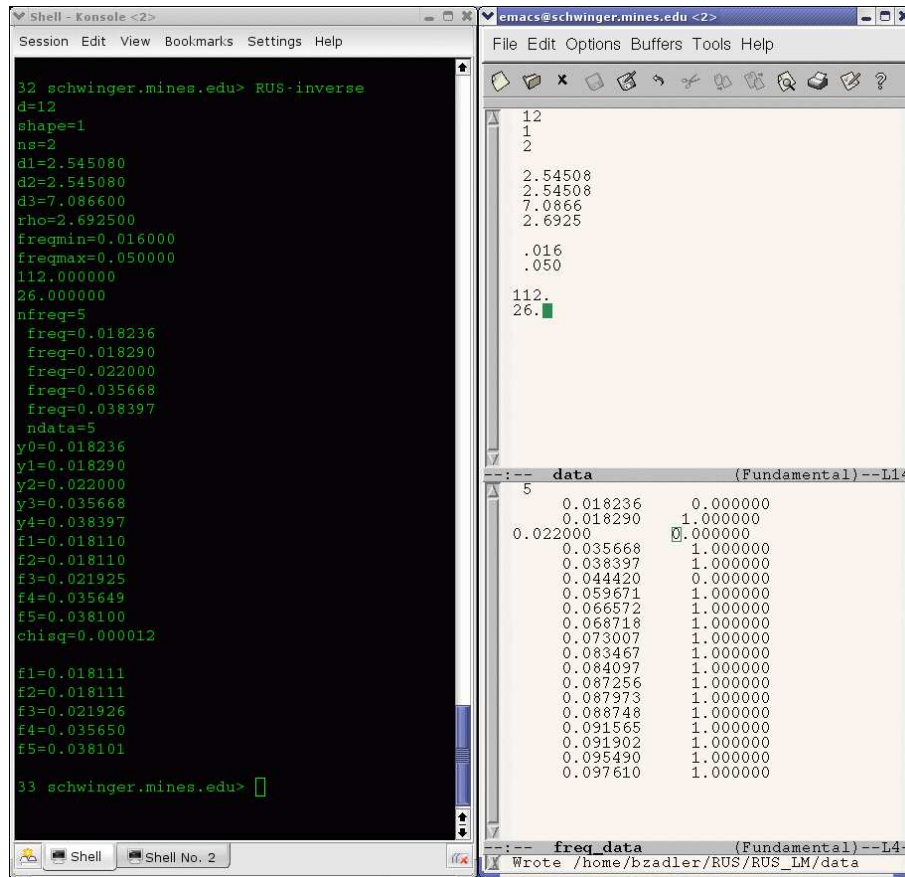


Figure 3.5: A screenshot of my screen during the inversion process. The terminal window running **RUS-inverse** is expanded vertically so I can watch the output. The input files are ready to be constantly updated on the right.

is likely you'll want to stop the inversion (via Ctl-c) when the convergence slows or to add another frequency to the fit. With the file **param_data** already open in an editor, you can copy and paste in the last fit values for the c_{ij} , save the file, and you're ready to go from where you left off.

3.3.1 Running RUS-inverse

With everything ready to go, my screen always looks like Figure 3.5. Type 'RUS-inverse' at the prompt. If you watch carefully, you see the program is fitting the first 5 frequencies by changing c_{11} and c_{44} and printing out a chisq value for each fit. If I wait until the fit is done, the last output looks like

113.0310

```

25.5869
starting eigenvalues calculation
eigenvalues calculation done
f1=0.017996
f2=0.017996
f3=0.021751
f4=0.035419
f5=0.038648
chisq=0.000021.

```

Comparing the first 5 frequencies to our observed frequencies, we can see we're right on so far. Add two more frequencies to the inversion by changing the '5' at the top of **freq_data** to a '7'. In the **param_data** file, change the maximum frequency line from '0.050' to '0.070'. Run it again.

You can see right away the chisq value is MUCH higher, about 0.022788 for me, and by looking at the predicted frequencies (output), the first five frequencies match with our observed frequencies, but the sixth frequency is a degenerate of the fifth. So our sixth observed frequency should actually be the seventh. Fix this by adding the missing frequency. Then **freq_data** looks like:

```

0.038397  1.000000
0.038400  0.000000
0.044420  0.000000

```

Run the inversion again. The chisq is now back down again. Let's add two more frequencies. Change the '7' to a '9' and run it again. It might also be a good time to change the order of the polynomial fit from '6' to '7' or '8'. This will allow a better fit to the higher frequencies.

We can see that we missed the doublet at 60 KHz, also. Add that line and run again. Keep repeating this until you are content with the number of frequencies you are using or until you feel you've filled in all of the missing gaps.

3.3.2 The final results

When finished, my **freq_data** file looks like the following:

28	
0.018236	0.000000
0.018290	1.000000
0.022000	0.000000
0.035668	1.000000
0.038397	1.000000
0.038400	0.000000
0.044420	0.000000
0.059671	1.000000
0.060000	0.000000
0.066572	1.000000
0.068718	1.000000
0.073007	1.000000
0.073000	0.000000
0.083467	1.000000
0.083000	0.000000
0.084097	1.000000
0.084000	0.000000
0.087256	1.000000
0.088300	0.000000
0.087973	1.000000
0.088000	0.000000
0.088748	1.000000
0.091565	1.000000
0.091600	0.000000
0.091902	1.000000
0.091900	0.000000
0.095490	1.000000
0.097610	1.000000

That list of 28 frequencies yields a chisq value of about 0.000205 for me. The final fit values of c_{11} and c_{44} are $c_{11} = 113.4809$ and $c_{44} = 26.3690$. That corresponds to an elastic modulus of $E = 71.13 GPa$ and a shear modulus of $\mu = 26.37 GPa$. These correspond to velocities of $V_P = 6492 \frac{m}{s}$ and $V_S = 3130 \frac{m}{s}$ which are dead on with literature values [SD90].

Congratulation! That's how it's done.

3.4 EXPLANATIONS OF AREAS ON THE GUI AND THEIR BUTTONS

3.4.1 Windows

create graphics window

This button will spawn a new graphics window offset from an already existing graphics window. This process will repeat indefinitely. If no graphics window is open, a new window will be spawned next to the gui. The size and position of the window is set inside the fitspectra.sci file using the variables wpos_xy and wdim_xy.

clear graphics window

If the current window is blank, no action occurs. Otherwise, pressing this button executes the xclear() command. This button is seldom used since the actions of either the Clear button or the Plot Original Freq. Range button are usually desired.

3.4.2 Plot options

replot original frequency range

At any time this button can be used to replot the data set as it was originally loaded from the filelist. This action removes any peaks that were selected and clears those variables.

add plot label

Use this to open a simple figure containing two edittable text boxes for the title, x-axis and y-axis labels.

3.4.3 Data options

fit to Breit-Wigner model

This attempts to fit a Breit and Wigner [BW36] model using the selected peaks to the data specified by the frequency range contained within the lowf and highf boxes in the Plot Range section. This model is a superposition of Lorentzians. There are two background parameters, B0, a constant term, and B1, a linear term. For each peak there are four parameters: frequency, width, amplitude and skew. Initial values these are calculated in

the *findinitialparams()* function. Once the desired peaks for fitting are selected, pressing the fit button calls the scilab function *datafit()* which performs a non- linear least squares regression of the data. The resulting model is plotted below the model created by the initial parameters and shows the frequency of the resonance peak and the Q value of the mode, calculated as $\frac{f}{\delta f}$, where δf is the full width at half max.

3.4.4 Save options

When saving files, FITSPECTRA is fairly nice.

GRAPHICS: At any time after loading a data file, graphics can be saved with the Save Graphics button. This can be either the original data set, a truncated data set with the range given by the lowf and highf values in the Plot Range section, or a plot of the original data with the B-W model fit over it.

PARAMETERS: Once a fit has been performed, it is also possible to save the new model parameters. The option to save just frequencies and Q values or to save all parameters is given. For specifics on these options, see the detailed button descriptions below.

DEFAULT DIRECTORY: This name is found by tagging '_results/' on to the original filename.

Example: Data filename is **ppcoated**

New Directory: **ppcoated_results**

DEFAULT FILENAME: This depends on what is being saved. Using the example above:

frequencies	ppcoated.f
frequencies and Q	ppcoated.fq
parameters	ppcoated.params
graphics	ppcoated.graphics

save f and Q

Once FITSPECTRA has fit the data to a Breit-Wigner model, the parameters frequency and Q (constructed from frequency and width parameters) can be saved via this button. The default directory is as described above.

NOTE: This option creates 2 save files, ***.fq** and ***.f**

- ***.fq**: Contains the frequency and Q of each mode fit. The default filename is [**data set name**].**fq** The format is 2 columns by n rows, n is the number of resonance peaks fit.

- ***.f**: Contains the frequency of each mode fit. The default filename is **[data set name].f** The format is 1 column by n rows, n is the number of resonance peaks fit.

NOTE: If the ***.f** and ***.fq** files already exist, all other fitting results on that data set will be put at the end of these files. This means that all results from fitting on the same data set are saved in column format in these two files.

save parameters

Same as Save f and Q button, but all parameters for the fit are save in a single column. The default filename is **[dataset name]_lowf_highf.params**. Lowf and highf are the frequency range the fit was performed on. Since these parameters will be optimized to this exact data range, the range is given if one wishes to use them to reconstruct the model.

save graphics

Only the optimized model is saved.

3.4.5 Plot range

get frequency range

Clicking on this button halts Scilab until 2 left mouse button clicks are made somewhere on the current graphics window, after which Scilab resumes. The two clicks define a frequency range over which the current data set is replotted. This is the range that will be fit by the Breit and Wigner model. The *2D Zoom* button on the Scilab graphics window CANNOT be used to do this (THIS being truncate the data set as you zoom in).

3.4.6 Get peaks

get resonance peaks

Once a frequency range is specified in the *Plot Range* section that contains the peak(s) that will be fit, use this button to select the desired peaks. Left-clicking on the button will again halt Scilab, but this time the right button MUST be used to select the peaks. A small grey X will appear where a click was made. When peak selection is complete, click the left mouse button to end. Red X's should now appear with the frequency values of the position of the X's. The data is now ready to be fit (see Section [3.4.2](#)).

keep resonance peaks

By watching the output of the fit on the terminal window, you may notice that for multiple peaks or for large data sets that it seems that the fit was still converging when the optimization ended. Click this button. If you now press the Fit button again, the fit will use the optimized parameters of the first fit as the starting parameters of the new fit. You can select all of the fit peaks this way. If you wish to de-select any peaks, use the *clear* button and start over.

clear

If you are unhappy with the peaks chosen or a mistake was made, click this button to replot the frequency range specified in the *Plot Range* section of the gui.

3.4.7 Directories

The bar above the directory listbox is a button used to change directories. Use the mouse to select either '..' to go back one directory, '.' to reload the current directory or name given within the listbox. Once chosen, click the dark gray button and the contents of the new directory will be updated within the directory listbox and the file listbox, as well as the button itself showing the path to the current directory.

3.4.8 Files

The bar above the file listbox is a button used to either load a data set of frequencies and amplitudes in a 2 column format or to load a graphics file. Note that a loaded graphics file CANNOT have analysis performed on it, as it is loaded as graphics, not a data set. Select the desired file with the mouse and click the button above the listbox. This will load the selected file and write the name of the file to the button.

3.4.9 Miscellaneous buttons

halt

Use this to halt the fitting process.

exit

Closes all figures and windows and exits Scilab.

3.5 Some important files and directories

- **scilab.star**: startup file for Scilab instructions in this file are executed when Scilab is executed. Note that you can also have your own startup file **.scilab** or **scilab.ini** in your current directory.
- **fitspectra.sci**: this is the main function to which all callbacks are made
- **specgui.sci**: this creates the graphical user interface and sets the gui properties and callbacks
- **bw.sci**: contains the Breit-Wigner formula

3.6 Useful formulas and relations

The general stress-strain relations for an elastic material are

$$\sigma_i = C_{ij} \epsilon_j \quad (3.1)$$

where σ_i is stress, ϵ_j is strain, the C_{ij} are elastic constants and $i, j = 1, 2, 3, 4, 5, 6$.

For Isotropic materials,

$$C_{11} = C_{22} = C_{33} = \lambda + 2\mu \quad (3.2)$$

$$C_{12} = C_{13} = C_{23} = \lambda \quad (3.3)$$

$$C_{44} = C_{55} = C_{66} = \mu \quad (3.4)$$

where λ and μ are Lamé constants. To relate these to the moduli of the material, we use

$$E = \frac{\mu (3\lambda + 2\mu)}{\lambda + \mu} \quad (3.5)$$

$$K = \frac{3\lambda + 2\mu}{3} \quad (3.6)$$

$$\nu = \frac{\lambda}{2(\lambda + \mu)} \quad (3.7)$$

$$(3.8)$$

where E is Young's modulus, K is the bulk modulus, and ν is Poisson's ratio.

Finally, to convert these to velocities:

$$\lambda = \rho (V_P^2 - 2V_S^2) \quad (3.9)$$

$$\mu = \rho V_S^2 \quad (3.10)$$

$$(3.11)$$

3.7 Anisotropic materials

The analysis for anisotropic materials is the same as isotropic. However, there are more moduli (up to 9 for our code) that define the elasticity of the body. For equations on converting the C_{ij} output from an anisotropic fit to velocities, see the material by Tsvankin [Tsv97].

What that means is, instead of having only two moduli, C_{11} and C_{44} for example, as the last two lines in the file **data**, you should put in the other C_{ij} as well. Also, don't forget to then change line #3 in **data** to reflect the correct number of moduli. For example, if you want to fit a cubic symmetry, line #3 will have the value 3 and you should put C_{11} , C_{12} , and C_{44} in the last 3 lines (in that order). Be sure you put the C_{ij} in the correct order. For all of the available symmetries, these are:

- Isotropic: C_{11} , C_{44}
- Cubic: C_{11} , C_{12} , C_{44}
- Hexagonal: C_{33} , C_{23} , C_{12} , C_{44} , C_{66}
- Tetragonal: C_{11} , C_{33} , C_{23} , C_{12} , C_{44} , C_{66}
- Orthorhombic: C_{11} , C_{22} , C_{33} , C_{23} , C_{13} , C_{12} , C_{44} , C_{55} , C_{66}

3.8 Details

For those of you who want exacting and specific details about the codes, you can email me questions at bzadler@acoustics.mines.edu. The scilab code is pretty simple, and likely very sloppy. It is very similar to Matlab, so the learning curve is steep. However, it's trial and error since (in my opinion) the help documentation is tough to follow. Regardless, feel free to hack the .sci files as you see fit, just try to give me a little credit. All of it is my own coding except for the Breit and Wigner model equation found in the bibliography.

The C code has great documentaion if you hack into it. However, unless you understand C programming, it'll look overwhelming to you. Most of the code follows the book *Resonant Ultrasound Spectroscopy* by Migliori and Sarrao [MS97]. The rest is Jerome's genius.

Bibliography

- [BW36] G. Breit and E.P. Wigner. Capture of slow neutrons. *Physical Review*, 49:519, 1936.
- [MS97] A. Migliori and J. L. Sarrao. *Resonant ultrasound spectroscopy: applications to physics, materials measurements, and non-destructive evaluation*. John Wiley & Sons, INC., New York, 1997.
- [SD90] C. B. Scruby and L. E. Drain. *Laser Ultrasonics: Techniques and Applications*, chapter 5-8-4, page 273. Adam Hilgor by IOP Publishing Ltd., 1 edition, 1990.
- [Tsv97] I. Tsvankin. Anisotropic parameters and p-wave velocity for orthorhombic media. *Geophysics*, 62(4):1292–1309, July-August 1997.