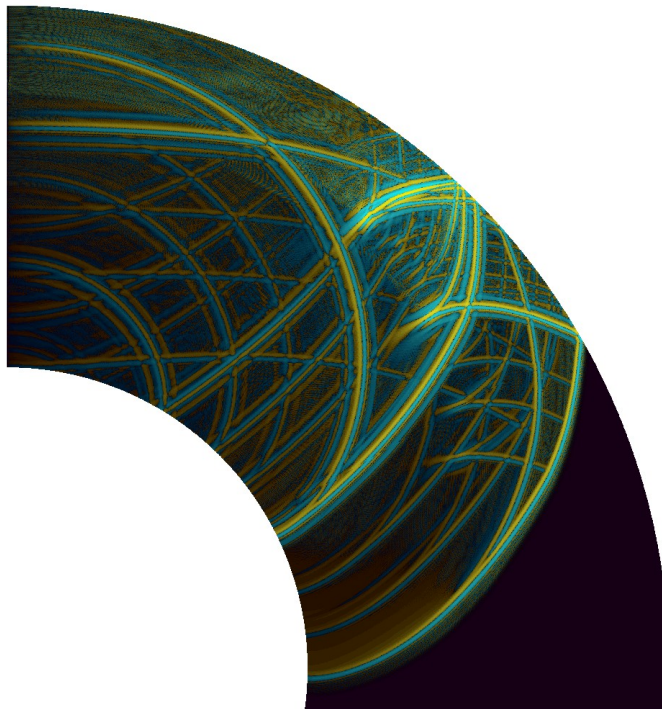


# SHaxi User's Guide

**version 1.0**

Gunnar Jahnke<sup>1</sup>, Michael S. Thorne<sup>2</sup>, Heiner Igel<sup>3</sup>

Created on 01.07.2008



<sup>1</sup>BGR

<sup>2</sup>Dept. of Geology and Geophysics, University of Utah

<sup>3</sup>LMU

# Contents

## Table of Contents

1. Introduction.....	4
2. Getting Started.....	4
2.1 Where to download SHaxi.....	4
2.2 Supported systems.....	5
2.3 Software requirements.....	5
2.4 Unpacking the source.....	6
2.5 Compiling.....	6
2.5.1 Adjusting the Makefile.....	6
2.5.2 Compiling the code.....	7
2.5.3 Interactive testing.....	7
2.5.4 Batch jobs / NQS.....	8
3 Input Files.....	8
3.1 The Parameter file (Par-file).....	8
3.2 Receiver files.....	13
3.3 Example queue files.....	13
3.3.1 Load Leveller Example:.....	13
3.3.2 PBS Example:.....	14
4 The Grid.....	14
4.1 Geometry of the SHaxi grid.....	14
4.2 Changing the grid size.....	16
4.3 Tables.....	19
5. Output Files.....	20
5.1 Info file.....	20
5.2 Par-out file.....	20
5.2 Stdout files.....	21
5.3 Seismogram files.....	21
5.3.1 ASCII seismograms.....	21

5.3.2	SAC seismograms.....	22
5.5	Snapshot Files.....	23
5.6	netCDF Model Files.....	24
6	Benchmarks.....	26
7	Models.....	28
7.1	Models included in SHaxi release.....	28
7.1.1	Homogeneous Earth Model.....	28
7.1.2	PREM 1-D reference model.....	29
7.1.3	Crustal thickening.....	29
7.1.4	Basic Slab Structure.....	30
7.1.5	Basic Rift Structure.....	31
7.1.6	Cross section through tomography model TXBW.....	32
7.2	Constructing User Defined Models.....	34
7.2.1	Model Parameters.....	35
7.2.2	Model Driver.....	37
7.2.3	Specifying the velocity anomaly.....	39
7.2.4	Incorporating the model into SHaxi.....	41
7.2.5	Adding the user module to the makefile.....	42
7.2.6	Visualizing SHaxi models.....	42
8	Additional Tools.....	45
8.1	Processing Seismograms.....	45
8.2	Producing snapshots of wave propagation.....	45
9	File Structure.....	47
10	References.....	48
10.1	SHaxi References.....	48
10.2	General References.....	48

# 1. Introduction

The SHaxi program generates global high resolution *SH* wave seismograms and wavefield snapshots using high performance parallel computers or PC networks.

Possible models are restricted to axi-symmetric geometries which allows the computation of synthetic seismograms at dominant periods down to 2.5 seconds for global mantle models.

The code is verified against an analytical solution and the implemented ring-source is well understood. Detailed information about the method can be found in Jahnke et al. (2008) as well as in Jahnke's PhD thesis.

The code has already been used for several applications, appropriate routines for convenient definition of user-defined models are implemented and explained.

The SHaxi method has already proven to be a valuable method for computing global synthetic seismograms at high frequencies and for studying the seismic waveform effects from models where rotational symmetry may be assumed.

## 2. Getting Started

### 2.1 Where to download SHaxi

- HTML download:

The latest version of the SHaxi code is publicly available at the home page of the “*Seismic wave Propagation and Imaging in Complex media: a European network*” Research Training Network (SPICE, [www.spice-rtn.org](http://www.spice-rtn.org), Igel, 2004). The SPICE software library provides open-source codes in computational seismology, training material on the numerical methods employed, benchmark solutions for forward and inverse problems, and publications of the SPICE team (Gallovič, 2007). After the 4-year SPICE project officially has ended in January 2008 the software library is now maintained by the Geophysics section of the LMU Munich (<http://www.geophysik.uni-muenchen.de>).

- Download from the Trac/Subversion Server

The development of the SHaxi program uses the Trac/Subversion server of the Geophysics section, LMU Munich which may in future provide download capability via its Trac interface: <http://svn.geophysik.uni-muenchen.de/trac/projects>.

If you have an individual account on the Subversion server, you can check out the newest version with the command:

svn checkout [https://\[LOGIN\]@svn.geophysik.uni-muenchen.de/svn/shaxi](https://[LOGIN]@svn.geophysik.uni-muenchen.de/svn/shaxi)

## **2.2 Supported systems**

The following combinations of systems and compilers is supported and can be selected in the SHaxi Makefile:

- **x86 Linux System & NAG Fortran Compiler**
- **x86 Linux System & Intel Fortran Compiler**
- **ICEBERG System & mpixlf95 Compiler**
- **Hitachi SR8000-F1 System & Hitachi Fortran Compiler**

In general, compilation with a different system and compiler should work after with minor adjustments of the compiler options in the *Makefile* (see 2.5 “Compiling”).

## **2.3 Software requirements**

Besides a Fortran 90 compliant compiler an implementation of the MPI Library (Message Passing Interface) is needed. On Linux systems the free MPICH distribution is recommended (<http://www.mcs.anl.gov/research/projects/mpich2>) although other implementations should also work. In general, the installation of MPI and its side effects on the compiling process may be a little tricky when done for the first time, and it is recommended to first assure that MPI works on the used system before starting with SHaxi.

## 2.4 Unpacking the source

The source is supplied as a gnu-zipped tar-file (.tgz) and can be unpacked with the commands:

```
> gunzip SHaxi.tgz.  
> tar xf Shaxi.tar
```

After unpacking, the following directory structure appears:

```
Benchmarks/ (setup files and output for some test runs)  
Docs/       (various documentation, e.g. this one here)  
OUTPUT/     (default output dir for simulations)  
Par_Files/  (default dir for simulation setups)  
RECEIVERS/ (for specifying synthetic receivers)  
Source/     (SHaxi source code)  
Tools/      (tools for processing the generated output)
```

## 2.5 Compiling

Before compiling it should be assured that a Fortran90 Compiler and an implementation of the MPI library is installed. Basic knowledge about the MPI library can be achieved from the *User's Guide* and the *Installer's Guide* available at the MPICH homepage (<http://www.mcs.anl.gov/research/projects/mpich2/>). Moreover it is recommended at the very beginning to compile a sample Fortran90 MPI Program which e.g. just spans some processes in order to test whether the MPI environment is correctly configured. Such sample programs are part of most MPI tutorials which can be found on the web.

### 2.5.1 Adjusting the Makefile

“cd” to the *Source/* directory and load the *Makefile* in your editor.

If your system is one of those listed in section 2.2 you may enable the appropriate lines in the *Makefile*. And disable the previously enabled system.

If your system is not listed, you can create your own set of rules:

- 1) find out how to invoke the Fortran 90 compiler on your system for the compilation of MPI-programs and add an appropriate *F90=[compiler's name]* statement.
- 2) Check your compiler's optimization options in the compiler's documentation. This can increase the performance of *SHaxi* significantly. Add the appropriated options in the *OPTIONS=[your compiler options]* line.
- 3) NetCDF Support: If you want to use NetCDF (network Common Data Form, <http://www.unidata.ucar.edu/software/netcdf>) and the library is installed, adjust the *NETCDF=[path\_to\_netcdf\_installation]* and set the variable *NCOBJECTS* to the object file *mod\_ncoutput.o* (*NCOBJECTS=mod\_ncoutput.o*)  
If you don't want to use the library insert *NETCDF=.* and set *NCOBJECTS* to *NCOBJECTS= mod\_ncoutput\_stub.o* .

## 2.5.2 Compiling the code

If the *Makefile* is properly adjusted, invoke *make* (on some systems such as the Hitachi SR8000 use *gmake* instead). If you use a compiler other than those listed in the *Makefile*, it may occur that the compilation process cancels and reports syntax errors. This is then caused by differences in Fortran syntax which different compilers support. If you are a little used to Fortran90 it should be no big deal to modify the appropriate lines in the *SHaxi* code, even without deeply understanding the underlying numerics. If the source could be compiled successfully, the executable *SHaxi.x* and a bunch of object files (\*.o files) reside in the *Source/* directory.

## 2.5.3 Interactive testing

If your system allows running (small) MPI-runs interactively, you should test that *SHaxi* works by manually starting a small job. To do so you have to find out how MPI jobs are started on your system which will in any case be a call similar to:

```
> mpirun -np 2 SHaxi.x < Parfile_test
```

This starts a parallel *SHaxi* run using 2 processors (np=2) with the simulation parameters from file *Parfile\_test*. On your system, the *mpirun* (or equivalent, e.g. *mpiexec*) command may require additional options, such as a machine-file or partition name etc.

## 2.5.4 Batch jobs / NQS

On large parallel systems, MPI jobs can be usually started only via batch jobs which are administrated by a queuing software like NQS (Network Queueing System). The usage of NQS differs on different platforms so it is recommended to have a look in the individual documentation of your system. On large parallel systems, sample NQS setups are usually given in the system documentation or can be requested from the administrator.

# 3 Input Files

## 3.1 The Parameter file (Par-file)

The central parameter file for defining the properties of a simulation is the *Par-file* which is an ASCII-file containing the parameters that have to be adjusted. Sample Par-files can be found in the *Par\_files/* directory and an example is given and explained here:

```
=====
seisfile      =OUTPUT/debug
modelfile     =MODELS/none
recfile       =RECEIVERS/recs
time          =50.          ! simulation time (sec)
nop           =4            ! operator length (keep at 4)
izfree        =4            ! free surface layer (izfree >= nop/2+1)
igraves       =0            ! method 1=graves 0=symmetry
attenuate     =0            ! Attenuation (0=off, 1=on)
Nk            =8            ! Number of Memory Variables (should =8)
Model -----
model_type    =103          ! 1=hom,2=prem,...=user models
vs0           =5000.        ! vs for homogeneous model
rho0          =2000.        ! rho for homogeneous model
Q0            =1000.        ! Q for homogeneous model
Tmax          =95.0         ! Angular size of model (180=full Earth)
Source -----
source_type   =1            ! should always be =1
sdepth        =200.0        ! source depth [km]
aa            =-1.          ! period of source signal (-1 for delta)
it0           =10           ! time step where signal starts
Receivers-----
izrec         =4            ! depth of receivers (= 'izfree')
OUTPUT-----
gfx_exp       =.33          !
=====
```



```

icheck          =1000          ! Output estimations each 'icheck'
timesteps
tsnap           =10000.        ! output snaps each 'tsnap' seconds
snapstart       =1             ! First snapshot to write
iout            =1             ! Snapshot size
iflush          =10000        ! flush seismograms every 'iflush' time steps
sacout          =1             ! (0/1)-output seismograms in SAC format
kevn            =debug        ! character descriptor (len=16) 'sacout'=1
nabs            =10            ! absorbing boundary at tmax (# grid points)
=====

```

The parameters in the Par-file are now explained in more detail:

- **seisfile:** Path of the output directory for the generated data together with the *root-file* name of the generated files. Each name of the individual files (e.g. seismogram files or snapshot files) consist of the root-file name together with an appropriate ending. The path should be given as a relative path (not starting with a “/”) which has the consequence that the SHaxi program has to be started with a current-directory so that the relative path is valid. Although it is possible to use the same output directory for multiple simulations by using different root-file names it is recommended to keep the results separately by creating an individual output directory for each run.
- **modelfile:** In case the selected *model\_type* (see description below) requires additional data which specifies the seismic properties individually at each grid point, these data is stored in the *modelfile*. Modelfiles are usually stored in the *MODELS/* folder. For built in model types such as the homogeneous model or PREM the no model file is needed and a stub name to a non-existing file can be given (e.g. *MODELS/none*). A description on how to create custom models is given in Section 7.2 “Constructing User Defined Models”.
- **recfile:** Specifies the path to the receiver file which should be placed in the *RECEIVERS/* folder. The format of the receiver file is described in a separate section below.
- **time:** Specifies the length of the seismograms to generate (simulation time). The actual time which needed for the calculation is approximately linear to the *time* value. Note that by increasing time also the number of time steps of the simulation increases linearly and that the the maximum number of time steps is hard-wired limit in the

code. If its value is too small, one can easily edit the *maxnt* variable in the source file *mod\_global.f90* and recompile the SHaxi code.

- **nop:** Length of the used finite-difference operator. This should be usually set to 4, since for the implemented time extrapolation scheme both longer and smaller operator lengths cause a decrease in overall accuracy of the generated seismograms. However, for convergence tests an operator length of 2 might be of interest, since this converges theoretically to the exact solution with no numerical dispersion at all, which is not exactly the case for  $nop \geq 4$ .
- **izfree:** Depth index of the free surface. Due to the finite difference implementation of the free surface boundary operator the topmost few grid points are used for the formulation of the free surface symmetry condition. The actual position of the free surface is therefore shifted a few grid points downwards. The value for *izfree* must be larger than half the operator length, so a value of 3 or 4 is sufficient for  $nop=4$ .
- **igraves:** Method used for the free surface implementation. *igraves=0* specifies the ordinary stress symmetry condition, whereas *igraves=1* activates the Graves boundary condition which is expected to be more precise. However, when investigating body waves this option should have no significant effect.
- **attenuate:** Switch for attenuation. 0: elastic wave propagation, no attenuation. 1: anelastic wave propagation.
- **Nk:** Number of memory variables used for the attenuation. The value influences both the accuracy and the needed memory during simulation.

The following settings define the model geometry:

- **model\_type:** Integer specifying the model type. 1: homogeneous model, 2: PREM model, >2: user defined models. See Section 7.2 “Constructing User Defined Models” for more information.
- **vs0:** S-Velocity (m/sec) for model type 1.
- **rho0:** Density (kg/m<sup>3</sup>) for model type 1.
- **Q0:** Q (attenuation) for model type 1.

- **Tmax:** *Theta-max* is the maximum lateral size of the model in degrees. If  $Tmax=180$  the boundary condition at the “right end” of the model is just the symmetry condition, similar to the condition at  $Theta=0$ . If  $Tmax<180$  an absorbing boundary is activated at the right end of the model to avoid artificial reflections.

The next variables specify the source parameters and the receiver depth:

- **source\_type:** Integer specifying the radiation pattern of the source. Currently, only the SH ring source ( $source\_type=1$ ) is implemented.
- **sdepth:** Source depth (km).
- **aa:** Dominant period (sec) of the source signal. For generating seismograms set  $aa$  to -1 which activates the discrete representation of the delta-function as source-time function. For generating snapshots set  $aa>0$  to suppress the numerical oscillations at high frequencies. The optimal value for  $aa$  depends of the grid size since the number of grid points which sample one period of the source signal. For snapshot runs  $aa$  should correspond to about 20 grid points per wave length. Fine tuning of  $aa$  can be done by calibration with a small model (large grid spacing) and then for larger setups scaling  $aa$  proportional to the decreased grid spacing.
- **it0:** For the delta-function source-time function  $it0$  defines the time step where the non-zero sample is set in the source-time function vector. It must be  $it0>1$  (e.g. 10) since otherwise the source time function would act as a step function rather than a delta function.
- **izrec:** Depth index where the receivers are defined. For receivers placed at the surface this must be equal to  $izfree$  (e.g.  $izrec=4$ ).

The last section controls the output of the generated data:

- **gfx\_exp:** Optional dynamic reduction of generated snapshots. Each amplitude of the wave field is raised to the power  $gfx\_exp$ . Thus, for  $gfx\_exp=1$  the amplitudes remain unchanged, whereas values between 0 and 1 cause the dynamic reduction which is stronger for smaller values of  $gfx\_exp$ . This can be used to suppress surface waves or to empathize small converted phases.

- **icheck:** Integer specifying the output interval when information about the maximum and minimum amplitude of the fields are written to the console (or to the redirection file). *icheck* should be chosen not too small since the amount of print statements may decrease the performance of the run drastically.
- **tsnap:** Output interval (in seconds) of successive snapshots. Small values may make animations based on the generated snapshot files smoother but decrease the performance due to the larger amount of 2D data which has to be written. Also the available storage on the output partition should be taken into consideration.
- **snapstart:** Index of the first snapshot which is actually written to the output folder. *snapstart=1* saves all snapshots from the beginning, *values > 1* are reasonable if one is not interested in the wave field of the first part of the simulation increase performance and decrease the needed disk space for the generated snapshot files.
- **iout:** Integer controlling the snapshot size. If *iout=1* each grid point is stored in the snapshot file. For *iout* values *> 1*, say *iout=2*, every 2<sup>nd</sup> grid point is written to file. Small values *> 1* can significantly reduce the snapshot size while the relevant information in relatively smooth wave fronts is preserved.
- **iflush:** Writes synthetic seismograms to file every *iflush* time steps. For interactive testing set *iflush* to a fraction of the total number of time steps of the simulation. This allows to check the output while the simulation is still running. Too small values cause unnecessary disk i/o which may unnecessarily reduce the performance. For large batch jobs which are usually inspected after completion *iflush* can be set to a very large number (larger than or equal to the last time step of the simulation) which causes the seismograms to be written not before the simulation is done.
- **sacout:** Flag specifying if seismograms are to be written in SAC format. Set to 1 for alphanumeric SAC format and to 0 for plain ASCII format.
- **kevnrm:** String defining a label in the SAC seismogram header. For SAC output *sacout* must be active (see above). The maximum length of *kevnrm* is 16 characters.
- **nabs:** In case of non-full earth models ( $T_{max} < 180$ ) *nabs* defines the taper thickness in grid points which is applied at the far end of the model.

All lines in the parameter file which follow the last variable are ignored and can be used for comments.

### **3.2 Receiver files**

The receiver file is a plain ASCII file located in the RECEIVERS/ folder. The format of the file is quite simple: The first line contains the total number of receivers which are defined in the file. The following lines contain the source-receiver distance in degrees for each individual receiver. An example receiver file with five receivers and 20 deg receiver spacing is given here:

---

```
5
 20.000000
 40.000000
 60.000000
 80.000000
100.000000
```

---

### **3.3 Example queue files**

To run SHaxi in a queued system a queue file should be constructed. Two examples are given below using LoadLeveller and PBS.

#### **3.3.1 Load Leveller Example:**

```
#!/bin/csh
#
# @ wall_clock_limit=28800
# @ environment = MP_SHARED_MEMORY=yes; COPY_ALL
# @ error = debug.err
# @ output = debug.out
# @ notification = never
# @ job_type = parallel
# @ node = 4
# @ tasks_per_node = 1
# @ network.MPI = sn_single,shared,us
# @ node_usage = shared
# @ class = debug
```

```
# @ queue

limit datasize 1048576
limit stacksize 1048576

date
./Source/SHaxi.x < ./Par_Debug
date
```

### 3.3.2 PBS Example:

```
#!/bin/bash

#PBS -N benchmark
#PBS -q workq
#PBS -l walltime=500:00:00
#PBS -e bench_stderr.txt
#PBS -o bench_stdout.txt
#PBS -l nodes=2:ppn=2

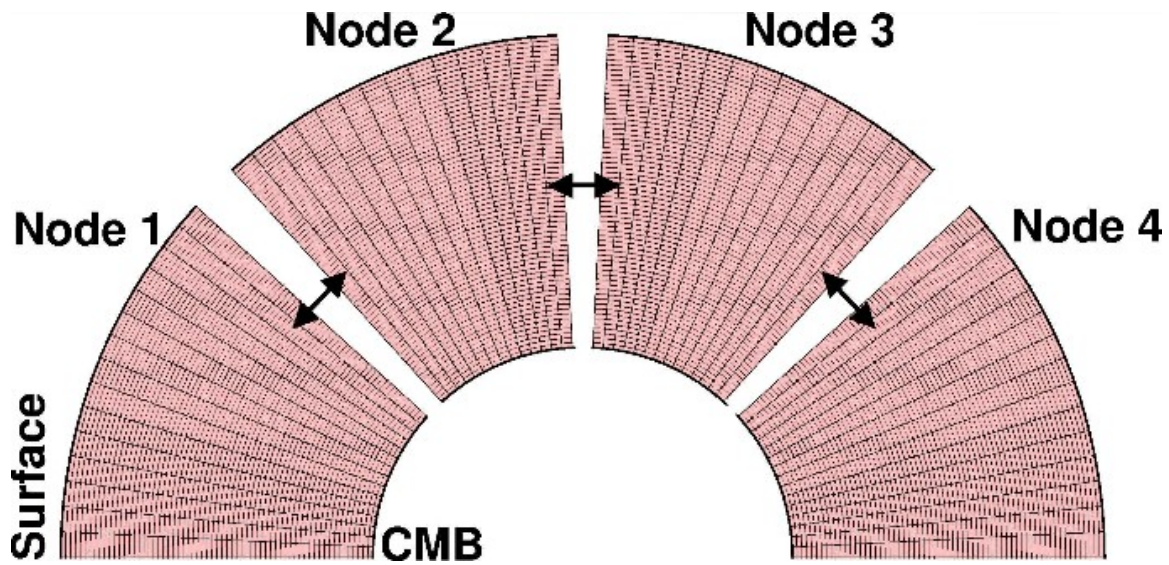
cd $PBS_O_WORKDIR
echo $PBS_NODEFILE

/opt/mpich/bin/mpirun -np 4 -machinefile ./machine_list ./Source/SHaxi.x <
Par_Benchmark01
```

## 4 The Grid

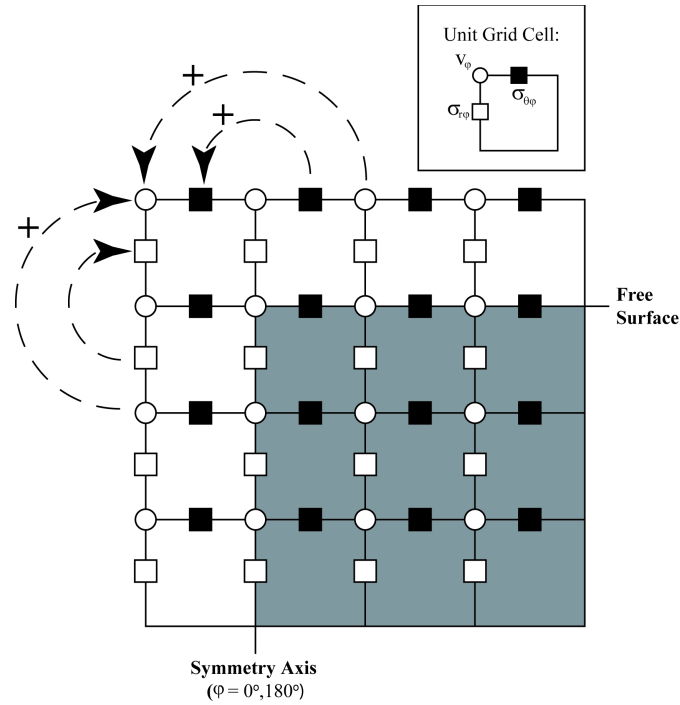
### 4.1 Geometry of the SHaxi grid

In the SHaxi program, the whole mantle is discretized by one large matrix (grid) with the left and right border corresponding to the symmetry axis at  $\theta=0^\circ$  and  $\theta=180^\circ$ , respectively. The upper border of the matrix corresponds to the Earth's surface and the lower border is the Core-Mantle boundary (Fig. 4.1). For parallel runs this large grid is subdivided into segments of equal size



**Figure 4.1** Sketch of the domain decomposition of the model grid, here shown for four nodes. All grid segments have the same size and cover an individual distance interval and the whole depth range of the mantle. For transportation of the wave field parameters between different segments, at each time step during the simulation the boundary elements of the grids have to be exchanged (indicated by double-arrows).

Each individual grid cell (Fig. 4.2, top) contains the three seismic parameters which are needed for axi-symmetric SH-wave calculation, namely S-velocity and two off-diagonal components of the stress tensor. Due to the used staggering the parameters are not defined exactly at the same location but shifted by  $\frac{1}{2}$  grid spacing in horizontal or vertical direction (Fig 4.2, bottom).



**Figure 4.2** Sketch of a small section of the staggered grids. The grey area denotes the leftmost and topmost part of the model space. Additional grid points above and left of the model space are used for the boundary conditions at the symmetry axis and the free surface. On the right top the unit grid-cell geometry is shown where the needed parameters for SH-wave generation

## 4.2 Changing the grid size

The model grid size is hard-wired in the *SHaxi* executable and can be only changed in the source code which efforts to recompile the program. This may be a little inconvenient, especially if dealing with several executables in case different model sizes are of interest. However, an implementation of *dynamic* memory allocation which would allow to define the grid size at run time, e.g. from a parameter file, is possible although it is not clear (to me) whether this causes an increase of performance on parallel systems compared to the used static arrays.

The variables defining the grid size are defined in `mod_global.f90` and the corresponding section is shown below:



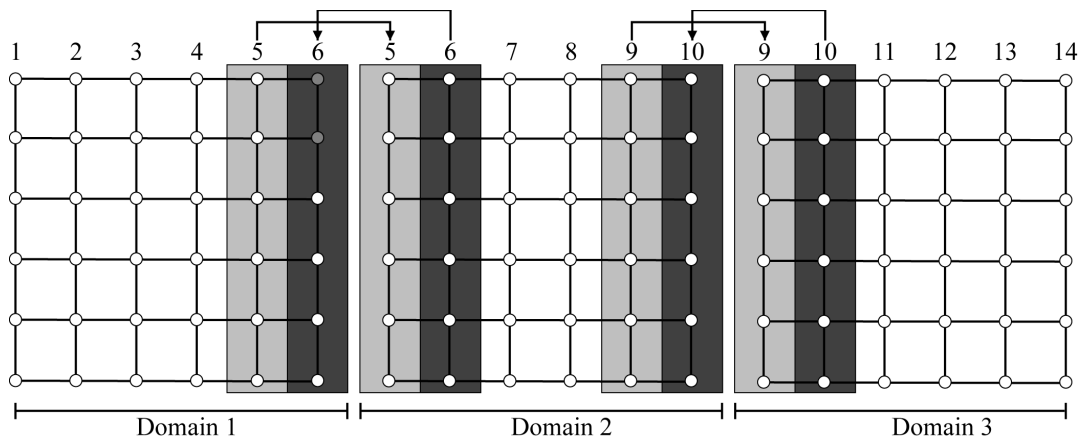
```

...
! nx,ny: grid size in horiz/vert direction
! ~~~~~
! SOME SUGGESTIONS                                !S(80 deg)  (2700 sec sim)

integer, parameter :: XSIZE=5000/4,  ZSIZE=1000    !Tdom=18s
!integer, parameter :: XSIZE=10000/4,  ZSIZE=1800    !Tdom=12s
!integer, parameter :: XSIZE=10000/8,  ZSIZE=1800    !Tdom=12s
!integer, parameter :: XSIZE=15000/8,  ZSIZE=2900    !Tdom=10s
!integer, parameter :: XSIZE=15000/12, ZSIZE=2900    !Tdom=10s
!integer, parameter :: XSIZE=20000/12, ZSIZE=3800    !Tdom= 8s
!integer, parameter :: XSIZE=20000/16, ZSIZE=3800    !Tdom= 8s
!integer, parameter :: XSIZE=30000/16, ZSIZE=5200    !Tdom= 6s
...

```

The variables XSIZE and ZSIZE define the size of the grid which is allocated by *each* of the individual nodes (more precisely: the size of each of the staggered grids defining the seismic parameters) in horizontal (XIZE) and vertical (ZSIZE) direction. Since the model space is decomposited in horizontal direction (Fig. 4.3), the total number of grid points of the whole simulation corresponds to  $XSIZE \cdot NP \cdot ZSIZE$  with NP the number of nodes (processors).



**Figure 4.3** Sketch of the model grid, here split in three domains which are processed by three individual nodes. The grey columns illustrate the exchange of wavefield information between the nodes which is not of further interest in this section.

In the examples above, the value for XSIZE is specified as fraction rather than an integer value. This is convenient if the total number of nodes (processors) to be used for the

simulation is already known, since this directly reflects the total grid size in horizontal direction but of course one has to assure that in the expression the nominator is a multiple of the denominator. On an algorithmic point of view there is no difference between **XSIZE=1250** instead of **XSIZE=5000/4**.

Although in principle any combination of XSIZE and ZSIZE can be chosen one should be aware that the grid spacing in vertical direction and the overall grid spacing in lateral direction should be approximately the same. This assures to get the optimum accuracy of the results by not wasting computational resources. The vertical grid spacing corresponds to approximately  $D_{max}/ZSIZE$  where  $D_{max}$  is the extension of the model in vertical direction. Usually,  $D_{max}$  corresponds to the thickness of the mantle which is 2891 km. It is possible to change  $D_{max}$  which is defined also in `mod_global.f90`, for example to perform modeling of waves which turn well above the CMB. However, for such simulations an additional tapering of the lower model boundary has to be applied which is not yet coded in SHaxi. However, based on the implemented taper which is initialized in `fd2_init.f90` and applied at each time step in `fd2_evolution.f90` it should be possible for an experienced Fortran 90 programmer to add such a feature. After modifying the file `mod_global.f90` the code has to be recompiled as explained in section 2.5.2 “Compiling the code”.

### 4.3 Tables

The two tables below show the computational resources needed for different grid sizes on two different systems: The first table shows the results obtained using 24 nodes of a x86-based Linux cluster whereas the second table shows the performance of up to 16 nodes (each consisting of 8 processors) of a high performance parallel system based on IBM Power4 CPUs. In addition to the run time and needed memory as a function of the grid size and grid spacing, the highest achievable dominant period of the direct S wave for four selected different distances is shown.

**Table 1. Example SHaxi parameters and performance – Linux Cluster.**

Grid Size				Number of Time Steps	Memory Usage <sup>c</sup> (Mb)	Dominant Period <sup>d</sup> (s)				Run Time <sup>e</sup>
npts <sup>a</sup> ( $\theta$ )	d $\theta$ <sup>b</sup> (km)	Npts (r)	dr (km)			S (40°)	S (80°)	SS (120°)	SS (160°)	
5000/24	4.0/2.2	1000	2.9	16894	17	16	18	25	30	19 m
10000/24	2.0/1.1	1800	1.6	33785	52	10	12	17	19	2 h 9 m
15000/24	1.3/0.7	2900	1.0	50758	122	8	10	12	15	7 h 39 m
20000/24	1.0/0.5	3800	0.76	67649	210	6	8	10	11	17 h 33 m
30000/24	0.7/0.4	5200	0.55	101512	428	5	6	8	9	2 d 6 h 21 m

<sup>a</sup>Values are: Total number of grid points / Number of processors used.

<sup>b</sup>Values are: d $\theta$  (at Earth surface) / d $\theta$  (at CMB)

<sup>c</sup>Memory is reported as total memory (code size + data size + stack size) for one processor. Code size is ~800 kb.

<sup>d</sup>Dominant Period based on phase and epicentral distance listed for a source depth of 500 km.

<sup>e</sup>Total run time is based on 2700.0s of simulation time.

**Table 2. Example SHaxi parameters and performance – “Iceberg” Supercomputer.**

Grid Size <sup>a</sup>		Number of Nodes <sup>b</sup>	Simulation Time = 1800 s		Simulation Time = 2700 s	
npts ( $\theta$ )	npts (r)		Number of Time Steps	Run Time	Number of Time Steps	Run Time
5000	1000	4	11281	13 m	16921	21 m
10000	1800	4	22560	1 h 31 m	33840	2 h 16 m
10000	1800	8	22560	57 m	33840	1 h 24 m
15000	2900	8	33838	3 h 2 m	50758	4 h 24 m
15000	2900	12	33838	2 h 20 m	50758	3 h 32 m
20000	3800	12	45099	5 h 46 m	67649	8 h 14 m
20000	3800	16	45117	4 h 30 m	67676	6 h 23 m
30000	5200	16	67675	12 h 26 m	101512	17 h 38 m

<sup>a</sup>Corresponding grid spacing is listed in Table 1.

<sup>b</sup>Each node consists of an IBM p655+ node, with 8 processors per node, and 16 Gb shared memory. Processor speed is 1.5 GHz.

## 5. Output Files

The following types of output files are written to the output folder. For the root-name (see description of the Par-file for the root-name) the string `mysim` is assumed which has to be replaced by the individual chosen string:

<code>mysim_inf</code>	- info file: general information about the run
<code>mysim_rank###_input</code>	- par-out file: settings read from Parfile by rank no. ###
<code>mysim_rank###_stdout</code>	- stdout file: redirected console output of rank no. ###
<code>mysim_rx##rz00snap###</code>	- Snap no. ### of rank no. ##
<code>mysim_seis###</code>	- Seismogram file of seismometer no. ###

### 5.1 Info file

The info file gives a compact overview about the simulation settings, e.g. on the model size and number of time stamps. The content of an info-file is self-explanatory therefore we give just an example file here:

```
Simulation_Info
xsize:    nx    =      842
ysize:    nz    =      508
timesteps: nt   =     2159
isnap:    isnap =      215
num. of snaps =      10
ranks in x-dir =       3
ranks in z-dir =       1
mpi buffer size =       4
dr (km)    =     5.770459
```

### 5.2 Par-out file

The Par-out file (`mysim_rank000_input`, `mysim_rank001_input`, etc.) contains all values read from the Par-file by the corresponding rank, whose rank-number is part of the file name (starting from 0). The input file enables to check whether mal formatted entries in the Par-file are the reason for a problem, e.g. if float numbers without a colon are given.

## 5.2 Stdout files

The stdout files (mysim\_rank000\_stdout, mysim\_rank001\_stdout, ...) contain status information about the actual progress of the simulation. Each MPI-process writes its output to a separate file whose rank-number is part of the file name and the rank number which is between 0 and N-1 (N: total number of ranks). While a simulation is running, it is possible with “tail -f filename” to continuously print the cumulative file to a console window.

## 5.3 Seismogram files

### 5.3.1 ASCII seismograms

The format of the ASCII seismogram files consist of a header of 11 entries followed by a single row of float values of the recorded amplitudes. The top of an example file is given below (the comments on the right are not part of the original file):

---

2159	number of samples
0.4632811	sampling rate
600.1280	exact source depth
30.00000	source receiver distance
10	it0
001	receiver number
4	operator length
842	grid size in x-direction
508	grid size in y-direction
3	total number of mpi-processes
1	model type (see Par-file description)
0.000000	first sample
0.000014	second sample
0.000042	third sample
...	

---

- **total number of time steps:** The number of samples (or amplitude values) which follow the header of this file.
- **sampling rate:** Time interval between two samples. The sampling rate does usually not correspond to an integer sampling frequency and for further processing a resampling may be a good idea.

- **exact source depth:** If the source depth which is specified in the Par-file does not match exactly the grid, the source is mapped to the closest grid point and the corresponding value is stored here.
- **source receiver distance:** The source receiver distance of the current trace in degrees (111.1949 km)
- **it0:** In case the source-time function is the delta-function *it0* describes the position of the non-zero sample of the source time vector. See the description of *it0* in the Par-file section for more information.
- **receiver number:** Position of the receiver as given in the receiver file (see Par-file description).
- **operator length:** Length of the finite-difference operator (see Par-file description).
- **grid size in x-direction:** Horizontal (lateral) grid size.
- **grid size in y-direction:** Vertical (radial) grid size.
- **total number of mpi-processes:** Number of mpi-processes used for the simulation.
- **model type:** Integer describing the type of the model ( see Par-file description).

After the header, all amplitudes of the recorded trace are given in a single column as ASCII float values.

### 5.3.2 SAC seismograms

If the *sacout* parameter is set equal to 1, SHaxi will output seismograms in Seismic Analysis Code (SAC) alphanumeric format. These seismograms can be read into SAC using the command:

```
SAC> read alpha filename
```

The SAC header is filled in with the most basic header variables. It is important to note how a few of the header variables have been filled in. Table 5.1 shows how certain header variables are filled.

**Table 5.1 Selected SAC header variables**

Variable	Description	How variable is set
evla	Event Latitude	Always set to 0°
evlo	Event Longitude	Always set to 0°
stla	Station Latitude	Always set to 0°
stlo	Station Longitude	Always set to the great circle arc distance.
kstnm	Station Name	Set as a the receiver number (e.g., 000, 001, etc.) in the order the receiver is specified in the receiver file.
kevn	Event Name	16 character description of the model as given by the <i>kevn</i> parameter in the parameter file.
kzdate	Event Date	Set as the date the seismogram was computed.
kztime	Event Time	Set as the time the seismogram was computed.

## 5.5 Snapshot Files

The format of the ASCII snapshot files consist of a header of 7 entries followed by a single row of float values of the snapshot amplitude data. The top of an example snapshot file is listed here (the comments on the right are not part of the snapshot file):

---

```

      1      iout
0.4632811  dt (time increment)
      2150  it (output time)
      508   nz
3462689.   minimum radius (km)
6388312.   maximum radius (km)
      842   nx
2.094814   theta_min (rad)
3.150379   theta_max (rad)
0.002456   first amplitude
0.005323   second amplitude
...

```

---

- **iout:** Store every *iout*-th sample in the snapshot file. For *iout*=1 every grid point is stored, for e.g. *iout*=2 only every 2<sup>nd</sup> grid point in x- and y-direction is stored, resulting in a 75% reduction of the size of the snapshot files.
- **dt:** Time increment of the simulation. In combination with the next variable the actual point in simulation time can be calculated.

- **it (time index)** : Time step of the generation of the current snapshot. The simulation time (in sec) of the snapshot generation is  $dt*it$ .
- **nz** : Vertical (radial) grid size of the full grid, not taking into account *iout*.
- **minimum radius**: Distance of the deepest row of grid points from the Earth's center in km.
- **maximum radius (km)** : Distance of the shallowest row of grid points from the Earth's center in km.
- **nx** : x Horizontal (lateral) grid size of the full grid, not taking into account *iout*.
- **theta\_min**: Theta angle (rad) of the column of grid points which is closest to the symmetry axis.
- **theta\_max**: Largest theta angle (rad) which occurs in the grid.

After this line, a single column with amplitude values of the snapshot follows. The amplitude values are written from inside of two nested loops, where the outer loop increments the horizontal (x) index of the grid and the inner loop increments the vertical (y) index, respectively being  $a_{xn,ym}$  the value of the grid at index (xn,ym) this results in:

---

```

ax1,y1
ax1,y2
ax1,y3
. . .
ax2,y1
ax2,y2
ax2,y3
. . .

```

---

## 5.6 netCDF Model Files

If netCDF is installed model files are written out in COARDS compliant netCDF (.nc) files. Two options exist for the method in which the files are written out. No flag is supplied to switch the option, but can be changed by editing the program *fd2\_model.f90*. The subroutine call is as follows:

```
CALL write_nc_model(r1,theta1,mu1,mu2,rho,Q1,mpi_rank,1)
```



The final argument can be set either to either 1 or 2. By default the argument is set equal to 1. This rotates the model by 90° making checking with *ncview* easier (see Section 7.2.6). Setting the argument equal to 2 makes visualizing the models with GMT easier. GMT expects that arrays be flipped vertically compared to what one would normally expect. Option 2 performs that flip. Viewing the format of the files can be done using the netCDF program *ncdump*. For example, use `ncdump -h` to view the netCDF header information. Model files contain the SHaxi grid and the fields *Vs1* (Velocity field on grid 1), *Vs2* (Velocity field on grid 2), *Rho* (Density field), and *Q* (Quality factor field). To conserve space only grid 1 is stored in the netCDF files, hence the field *Vs2* is slightly shifted compared to where the values actually lie, but still allows checking to make sure the basic structure of the *Vs2* field is correct.

## 6 Benchmarks

The SHaxi release contains four benchmark set of seismograms the user can use to compare their output with. These files are included in the directory *Benchmarks*. Each benchmark is contained in its own subdirectory. The contents of each subdirectory are described in Table 6.1.

**Table 6.1 Contents of benchmark subdirectories**

File/Directory	Description
Par_Benchmark	Parameter File used for benchmark
Run_Benchmark.ll	Load Leveller script used to run benchmark
recs_benchmark	Receiver file used for benchmark
README.txt	Readme file showing steps to run each benchmark
seismos_unfilt/	Raw unfiltered SAC formatted seismograms for the benchmark
seismos_filt/	Filtered SAC formatted seismograms for the benchmark
gemini_soln/	GEMINI solutions for same model setup.

For a first time use of SHaxi one can quickly test if the code is working properly by running Benchmark01. This is a homogenous model on a small grid without attenuation turned on. On most systems this benchmark can be run in just a few minutes. To run this benchmark (the other benchmarks can be run using similar steps) perform the following steps:

- 1) cd into the shaxi/trunk/ directory
- 2) cp Par\_Benchmark01 ./
- 3) cp Run\_Benchmark01.ll ./
- 4) cp recs\_b01 ./RECEIVERS/
- 5) cd Source
- 6) open up mod\_global.f90 for editing (e.g., using the vi editor)
- 7) change the parameter XSIZE=5000/4
- 8) change the parameter ZSIZE=1000
- 9) recompile SHaxi
- 10) cd back into the shaxi/trunk directory
- 11) Submit Run\_Benchmark01.ll (e.g., llsbmit Run\_Benchmark01.ll)

The basic steps for running each benchmark are outlined in the readme file supplied with each benchmark. The benchmarks supplied in the SHaxi release are outlined in Table 6.2.

**Table 6.2 Benchmark descriptions**

<b>Benchmark</b>	<b>Model Type</b>	<b>XGRID Size</b>	<b>ZGRID Size</b>
Benchmark01	Homogeneous*	5000/4	1000
Benchmark02	PREM*	20000/16	3800
Benchmark03	Homogeneous	5000/4	1000
Benchmark04	PREM	20000/16	3800

\*These benchmarks do not have attenuation turned on.

Each benchmark is supplied with SHaxi seismograms as either filtered or unfiltered. Additionally, seismograms produced with the GEMINI technique (Friederich and Dalkolmo, 1995) for a similar model setup are also included and are filtered in the same manner as the SHaxi synthetics. The filtered synthetics are filtered by convolution with a Gaussian pulse with a dominant period of 10 sec.

## 7 Models

A variety of simple models are included in the SHaxi release. These models are summarized in section 7.1. Section 7.2 gives examples of how the user can construct their own model types.

### 7.1 Models included in SHaxi release

Table 7.1 shows a summary of the models included in the current SHaxi release and the *model\_type* used to call them in the parameter file. Subsequent sections of this chapter display the various parameters used in defining each model.

**Table 7.1 SHaxi models**

<b>model_type</b>	<b>Program Name</b>	<b>Description</b>
1	fd2_model.f90	Homogeneous Earth
2	fd2_model.f90	PREM 1-D reference model
100	mod_circle.f90	Circular velocity anomaly (see Section 7.2)
101	mod_simpleslab.f90	Basic slab structure
102	mod_simplerift.f90	Basic rift structure
103	mod_modifycrust.f90	Thickened crust
200	mod_tomogrand.f90	Cross section through tomography model TXBW

#### 7.1.1 Homogeneous Earth Model

The parameters for a homogeneous earth are specified solely in the parameter file. These parameters are:

**Table 7.2 Parameters used for Homogeneous Earth Model**

<b>Parameter</b>	<b>Description</b>	<b>Units</b>
vs0	Shear-wave velocity	meters
rho0	Density	kg/m <sup>3</sup>
Q0	Quality Factor	NA

## 7.1.2 PREM 1-D reference model

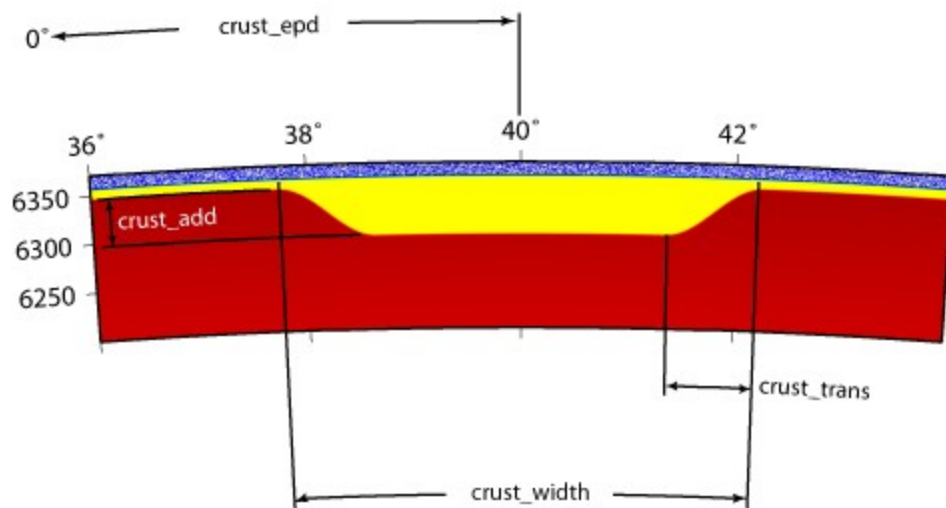
To use the PREM reference model (Dziewonski & Anderson, 1981) nothing needs to be specified other than *model\_type* = 2. The subroutine *sh\_prem* in the program *fd2\_model.f90* specifies the PREM model.

## 7.1.3 Crustal thickening

This model thickens the PREM crust. Table 7.3 shows the parameters used which are shown graphically in Figure 7.1. The parameters are hardwired in the module *mod\_modifycrust.f90*. Changing the parameters requires recompiling SHaxi. The shape of the crustal transition from thin to thick crust is a cosine in the range 0 to  $\pi$ .

**Table 7.3 Parameters used in crustal thickening model**

Parameter	Description	Units
<i>crust_add</i>	Add this many km of thickness to base of crust.	km
<i>crust_epd</i>	Epicentral Distance to center of thickened crustal region.	deg
<i>crust_width</i>	Total width of thickened crustal region.	km
<i>crust_trans</i>	Width of smoothed side regions.	km



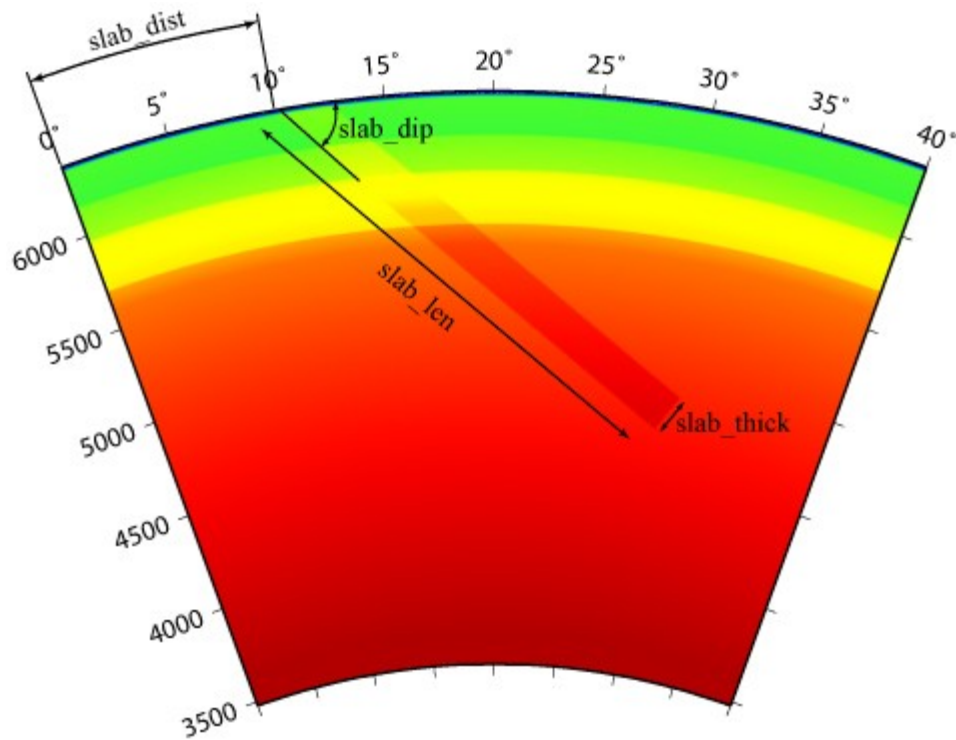
**Figure 7.1** Parameters used in crustal thickening model.

### 7.1.4 Basic Slab Structure

This model constructs a linear slab. Table 7.4 shows the parameters used which are shown graphically in Figure 7.2. The parameters are hardwired in the module `mod_simpleslab.f90`. Changing the parameters requires recompiling SHaxi. If gradient is turned off, then the slab takes on a constant velocity and/or density jump within the slab region. If gradient is turned on the slab velocity and/or density has a Gaussian profile, with the velocity and/or density reaching its maximum value in the center of the slab.

**Table 7.4 Parameters used in slab model**

Parameter	Description	Units
slab_dip	Dip of slab from horizontal (deg)	deg
slab_dist	Epicentral Distance to slab (deg)	deg
slab_len	Length of slab (end to end) (km)	km
slab_thick	Thickness of slab (km)	km
slab_v	Slab velocity (dVs - % difference from mantle)	%
slab_rho	Slab density (dp - % difference from mantle)	%
gradient	0=no gradient; 1=gradient	(0,1)



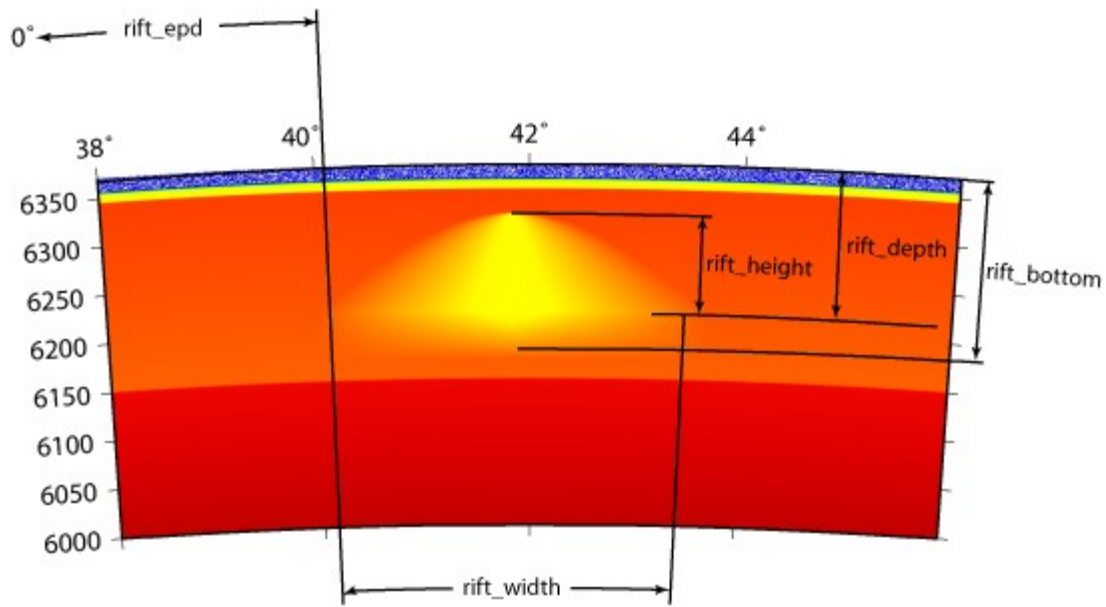
**Figure 7.2** Parameters used in slab model.

### 7.1.5 Basic Rift Structure

This model constructs a rift-like structure. Table 7.5 shows the parameters used which are shown graphically in Figure 7.3. The parameters are hardwired in the module `mod_simplerift.f90`. Changing the parameters requires recompiling SHaxi. The rift structure is constructed as one half of a cosine curve.

**Table 7.5** Parameters used in rift model

Parameter	Description	Units
<code>rift_width</code>	Total width of rift	km
<code>rift_depth</code>	Depth at which cos curve starts	km
<code>rift_height</code>	Total amplitude of cos curve	km
<code>rift_bottom</code>	Very lowermost edge of rift	km
<code>rift_epd</code>	Epicentral distance where rift starts.	deg
<code>rift_v</code>	Rift velocity (dVs - % difference from mantle)	%
<code>rift_rho</code>	Rift density (dp - % difference from mantle)	%



**Figure 7.3** Parameters used in rift model.

### 7.1.6 Cross section through tomography model TXBW

Cross sections through the S-wave velocity tomography model TXBW (Grand 2002) can be constructed and imported into SHaxi.

The first step is to create a base cross-section file using the scripts in the directory *Tools/Tomo\_Grand*. The steps to do this are outlined below:

(1) *Decide on great circle arc to cut through the tomography model.*

The great circle arc in the scripts is defined by an event, receiver pair. The scripts will draw a great circle through the pair and determine the values from Grand's model for this great circle path.

The parameters are:

evlat: Event latitude (deg)

evlon: Event longitude (deg)

stlat: Receiver latitude (deg)

stlon: Receiver longitude (deg)



The great circle arc can be checked using the provided GMT script *Plot\_globe.csh*.

Using the parameters:

```
evlat = -13.74  
evlon = 291.21  
stlat = 33.49  
stlon = 243.33
```

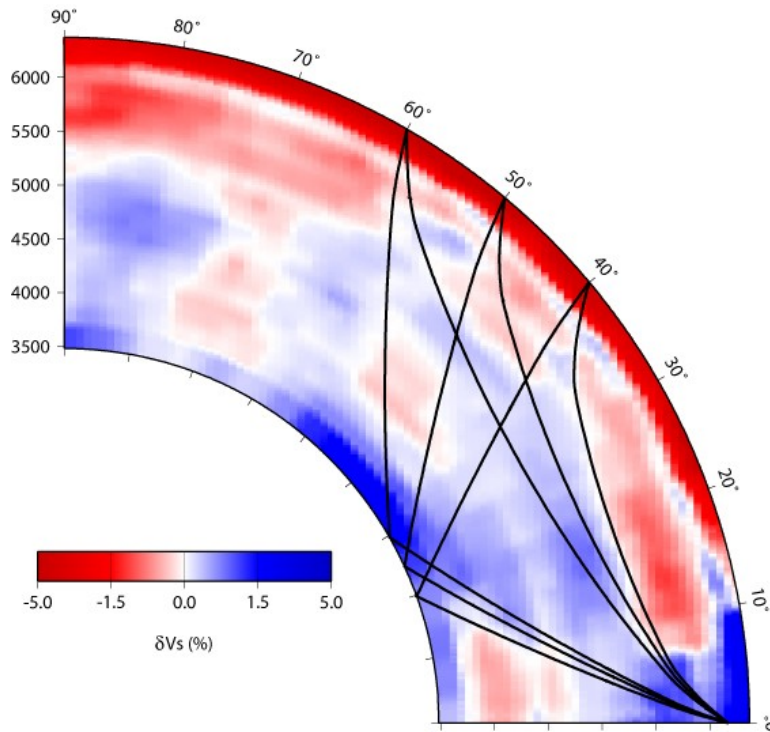
produces the map shown in Figure 7.4.



**Figure 7.4** Defining a cross section through tomography model TXBW.

## *(2) Create the base cross section*

The parameters (*evlat*, *evlon*, etc.) must be changed in the code *Grand\_Xsect.f90*. Then the code can be compiled (*make*), and run (*./xsect.x*). The output of the code is a file called *Base\_xsctn\_grand2002*. This file contains the basic information that will be needed by SHaxi to produce the model. This is the only file SHaxi will require. To check the output of the base cross-section file, the script *Plot\_xsect.csh* is an example GMT script that can be used. The output plot for the above example should look something like as shown in Figure 7.5.



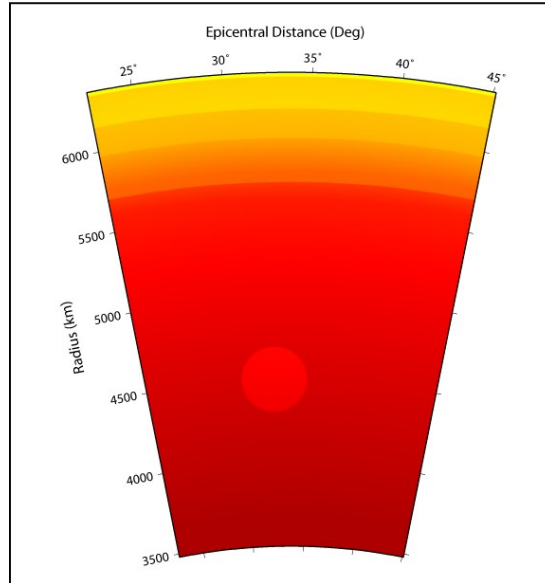
**Figure 7.5** Example cross section through model TXBW.

(3) Change filename in *mod\_tomogrand.f90*

The only parameter in *mod\_tomogrand.f90* that needs to be set is the variable *xfile*, which contains the name of the base cross-section file to use. This can be changed from the default filename *Base\_xsctn\_grand2002*.

## 7.2 Constructing User Defined Models

To incorporate a user defined model in SHaxi we recommend adding your own modules to the code. In this section we show a simple example of how to do this. The example we show will produce a circular shaped velocity anomaly. This may not be the most exciting example, but it is an easy to understand example and demonstrates the process by which more complicated models may be built. Figure 7.6 displays what the model we will construct looks like.



**Figure 7.6** Circular velocity anomaly.

My individual user modules typically consist of three main subroutines:

- i) A subroutine setting model parameters.
- ii) A subroutine that drives the whole process, to make sure every grid point is handled and that conversion to the correct units for SHaxi are applied.
- iii) A subroutine that specifies  $S$ -wave velocity and density (and possibly  $Q$ ) at each grid point in SHaxi.

As an example, I will show the above three subroutines to create the circular velocity anomaly. These subroutines are provided in the SHaxi source code in the module named, *mod\_circle.f90*.

### 7.2.1 Model Parameters

The first subroutine specifies the various parameters about the anomaly I may want to vary. Alternatively, a subroutine could be written to read these parameters in from a file, but this example is the easiest solution. All this subroutine does is set aside the parameters that I may want to change.

```

!
=====!!
! SETCIRCLE
!
! Set parameters for circular anomaly
! !
=====!

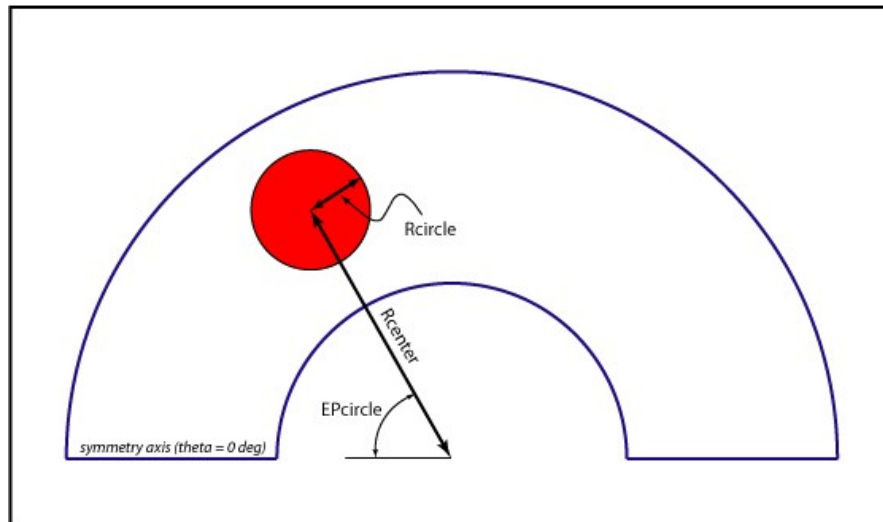
SUBROUTINE setcircle(Rcenter,EPcircle,Rcircle,circle_v,circle_rho)
IMPLICIT NONE
REAL, INTENT(OUT) :: Rcenter      !Radius to center of circular anomaly (km)
REAL, INTENT(OUT) :: EPcircle     !Epicentral Distance to center of anomaly (deg)
REAL, INTENT(OUT) :: Rcircle      !Radius of circular anomaly (km)
REAL, INTENT(OUT) :: circle_v     !Vs inside circle (% difference from surrounding mantle)
REAL, INTENT(OUT) :: circle_rho   !Density inside circle (% difference from mantle)

!Set parameters for circle - to be used by makecircle
Rcenter   = 4500.0
EPcircle  = 32.5
Rcircle   = 200.0
circle_v  = -5.0
circle_rho = -2.0

END SUBROUTINE setcircle
!
=====!

```

The parameters are shown graphically in Figure 7.7.



**Figure 7.7** Parameters used in defining circular velocity anomaly.

## 7.2.2 Model Driver

The next subroutine drives the process of building the model. This is the subroutine that the main part of the SHaxi code would make a subroutine call to (called in the program *fd2\_model.f90*). The driving subroutine is shown below.

```

=====
!
! CIRCLEDRIVER
!
! Driver routine to produce circular anomaly
!!
=====
SUBROUTINE circledriver(rho,mu1,mu2)
USE global, only: nx, nz, r1, theta1, r2, theta2, pi
IMPLICIT NONE
REAL, DIMENSION(nx,nz), INTENT(INOUT) :: rho, mu1, mu2
REAL :: trad, ttheta, Vsin, Rhoin, Vsout, Rhoout
INTEGER :: z, x

CALL sh_prem

DO z=1,nz
  DO x=1,nx

    trad = r1(x,z)/1000.      !trad = radius in km
    ttheta = theta1(x,z)*(180.0/pi) !ttheta = theta in deg
    Vsin = mu1(x,z)          !Vs (km/sec)
    Rhoin = rho(x,z)         !Rho
    CALL makecircle(trad,ttheta,Vsin,Vsout,Rhoin,Rhoout)
    mu1(x,z) = Vsout
    rho(x,z) = Rhoout

    trad = r2(x,z)/1000.0
    ttheta = theta2(x,z)*(180.0/pi)
    Vsin = mu2(x,z)
    CALL makecircle(trad,ttheta,Vsin,Vsout,Rhoin,Rhoout)
    mu2(x,z) = Vsout

  ENDDO
ENDDO
rho=rho*1000.      ! adjust units
mu1=mu1*1000.
mu2=mu2*1000.
mu1=rho*mu1**2    ! convert vs-> mu
mu2=rho*mu2**2    ! ( vs=rho*mu^2 )

```

This subroutine can almost be directly copied for each individual use. Before we describe what this subroutine does we should define the global variables that are used and

are important for building models. Table 7.6 shows the global variables in SHaxi that are important for building models.

**Table 7.6 Important SHaxi Global Variables**

Variable	Description
<i>nx</i>	The integer number of grid points (for the current rank) in the theta-direction.
<i>nz</i>	The integer number of grid points (for the current rank) in the radial-direction.
<i>r1</i> & <i>r2</i>	An array of size ( <i>nx,nz</i> ) containing the radius (units of meters) for each grid point. <i>r1</i> and <i>r2</i> contain the radii for <i>mu1</i> and <i>mu2</i> respectively.
<i>theta1</i> & <i>theta2</i>	An array of size ( <i>nx,nz</i> ) containing the epicentral distance (from the source – or symmetry axis) (units of radians) to each grid point. <i>theta1</i> and <i>theta2</i> contain the distance to <i>mu1</i> and <i>mu2</i> respectively.
<i>rho</i>	An array of size ( <i>nx,nz</i> ) containing the density (units kg/m <sup>3</sup> ) of each grid point.
<i>mu1</i>	An array of size ( <i>nx,nz</i> ) containing the shear modulus (kg/m·sec) of each grid point (on grid defined by <i>r1</i> , <i>theta1</i> ).
<i>mu2</i>	an array of size ( <i>nx,nz</i> ) containing the shear modulus (kg/m·sec) of each grid point (on grid defined by <i>r2</i> , <i>theta2</i> ).

The first thing this subroutine does is make a call to *sh\_prem*. This initializes each grid point in the model to values given in the PREM model. A note of possible confusion is that this initial step does not actually store *rho*, *mu1*, and *mu2* in the units described above, but instead temporarily holds *S*-wave velocity (units of km/sec) in *mu1* and *mu2* and holds density (units of g/cm<sup>3</sup>) in *rho*. This step does not have to be done if your subroutine will fill every grid point on its own.

Next we loop through all grid points in the model making a call to the subroutine *makecircle*. The *makecircle* subroutine takes as input the radius (in km) and theta (deg) position, and the currently assigned *S*-wave velocity and density value of the current grid point. And it spits out a modified *S*-wave velocity and Density to the variables *Vsout* and *Rhoout*. Because of the staggered grid this must be done twice once for the SHaxi variable *mu1* and again for *mu2*. Inside the DO loops *mu1* and *mu2* take on the values of *S*-wave velocity (units of km/sec) as given by the *sh\_prem* subroutine. However, *mu1*

and *mu2* are assigned for the shear moduli, and are thus converted to such at the end of the subroutine.

As noted, this driver subroutine can be directly copied into other user specified routines, with just a replacement of the *makecircle* subroutine.

### **7.2.3 Specifying the velocity anomaly**

Creating the velocity anomaly is done simply. In the above driving routine we looped through all grid points. Here all we do is test each grid point in the model and see if they are within the specified radius (*Rcircle*). If they are we change the velocity and density, if not we leave the velocity and density alone. The following subroutine accomplishes this:

```

=====
! MAKECIRCLE
! Subroutine that creates the circular anomaly
!   r      = radius of grid point (km) EXPECTS UNITS == KM!
!   theta  = theta of grid point (deg) EXPECTS UNITS == DEG!
!   Vsin   = Current Vs of grid point (km/sec)
!   Vsout  = Output velocity of grid point (km/sec)
!   Rhoin  = Current density of grid point (g/cm^3)
!   Rhoout = Output density of grid point
=====
SUBROUTINE makecircle(r,theta,Vsin,Vsout,Rhoin,Rhoout)
IMPLICIT NONE
REAL, INTENT(IN) :: r, theta, Vsin, Rhoin
REAL, INTENT(OUT) :: Vsout, Rhoout
REAL, PARAMETER :: dtr=.01745329252222
REAL :: x, y, xc, yc, d
REAL :: Rcenter, EPcircle, Rcircle, circle_v, circle_rho

CALL setcircle(Rcenter,EPcircle,Rcircle,circle_v,circle_rho)

!Convert r,theta position to x,y
x = r*sin(theta*dtr)
y = r*cos(theta*dtr)

!Find center of circle
xc = Rcenter*sin(EPcircle*dtr)
yc = Rcenter*cos(EPcircle*dtr)

!Calculate distance between current position and center of circle
d = sqrt(((xc-x)**2 + (yc-y)**2))
IF (d <= Rcircle) THEN
  Vsout = (Vsin)*(circle_v/100.0) + Vsin
  Rhoout = (Rhoin)*(circle_rho/100.0) + Rhoin
ELSE
  Vsout = Vsin
  Rhoout = Rhoin
ENDIF

```



## 7.2.4 Incorporating the model into SHaxi

To incorporate an individual user module into SHaxi, it must be added to the program that specifies the models. This program is *fd2\_model.f90*. In the above example the subroutines were added in a module named *circle*. So, a *USE circle* statement must be added to *fd2\_model.f90* as shown here:

```
!=====!  
! FD_MODEL  
!=====!  
subroutine fd_model  
  USE global  
  USE mod_cart2D, only: mpi_rank,mpi_xrank  
  USE attenuation  
  USE ncoutput  
  USE circle  
  implicit none  
  ---- the rest of the code ----
```

Next we need to add a call to the driver subroutine and define a model number (SHaxi variable *model\_type*) to the module. This can be done inside *fd2\_model.f90* as shown:

```
IF (model_type==1) THEN           ! Homogeneous Model  
  --- several lines of code ---  
ELSEIF (model_type == 48) THEN    ! circular anomaly  
  IF (attenuate == 1) THEN        ! initialize Quality factors to PREM values  
    CALL Qprem(Q1,1)  
    CALL Qprem(Q2,2)  
  ENDIF  
  CALL circledriver(rho,mu1,mu2)  
ELSEIF (model_type == 49) THEN    ! yet another model  
  --- more code ----
```

## 7.2.5 Adding the user module to the makefile

In addition, the user module should be added to the makefile. Here two things must be done. First, it needs to be added to the OBJECTS variable:

```
OBJECTS= mod_sacoutput.o mod_global.o mod_circle.o mod_cart2D.o ...
```

Second, we add the lines to compile the module file itself:

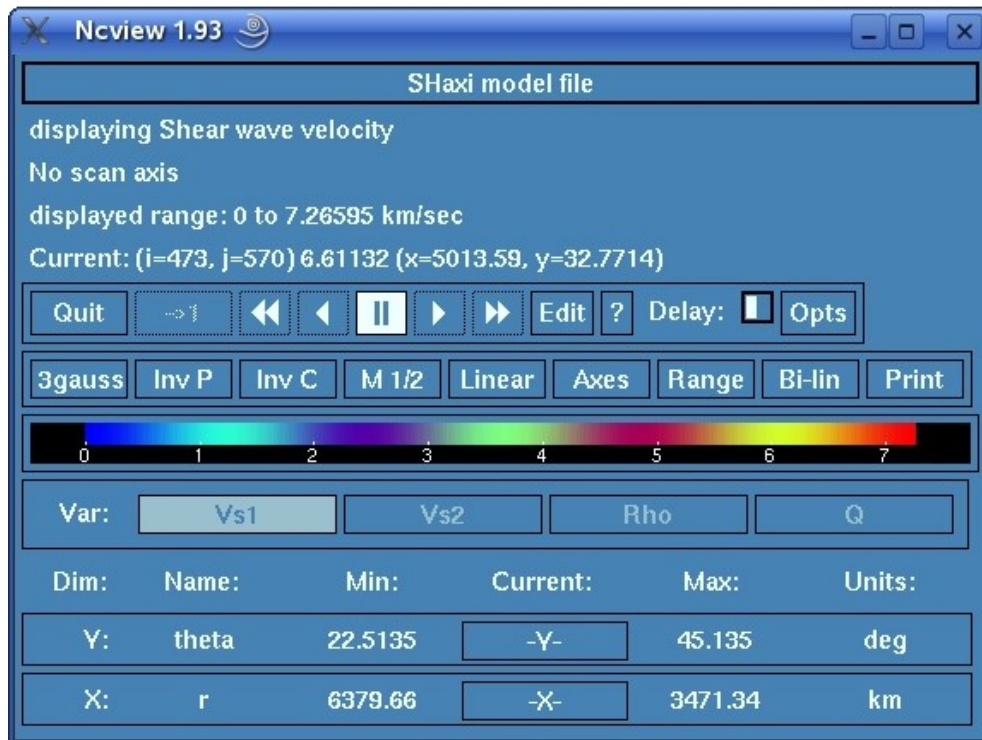
```
mod_circle.o : mod_circle.f90 mod_global.f90  
    $(F90) $(OPTIONS) -c mod_circle.f90
```

In this case, we add the *mod\_global.f90* dependency since we used several of the global variables in our module.

## 7.2.6 Visualizing SHaxi models

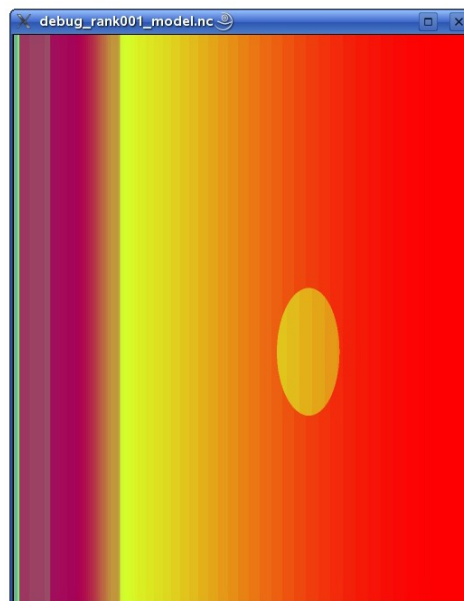
How do we know our subroutine's actually produced the model we expected in SHaxi? For the purpose of checking models we supply a subroutine for writing out the models in netCDF format. A model file is written for each process rank. These files have a naming convention \*\_model.nc.

Each netCDF file contains the parameters,  $\rho$ ,  $Vs1$ ,  $Vs2$ , and  $Q$ . Where  $Vs1$  and  $Vs2$  are the shear wave velocities calculated from the shear moduli and density. The quickest way to check the model file is to use a freeware program such as ncview ([http://meteora.ucsd.edu/~pierce/ncview\\_home\\_page.html](http://meteora.ucsd.edu/~pierce/ncview_home_page.html)). A Ncview prompter used with a SHaxi model file is shown in Figure 7.8. Each of the above variables can be selected in the Ncview prompter and one can quickly check if the model looks correct (although Ncview only displays the files on a Cartesian grid).



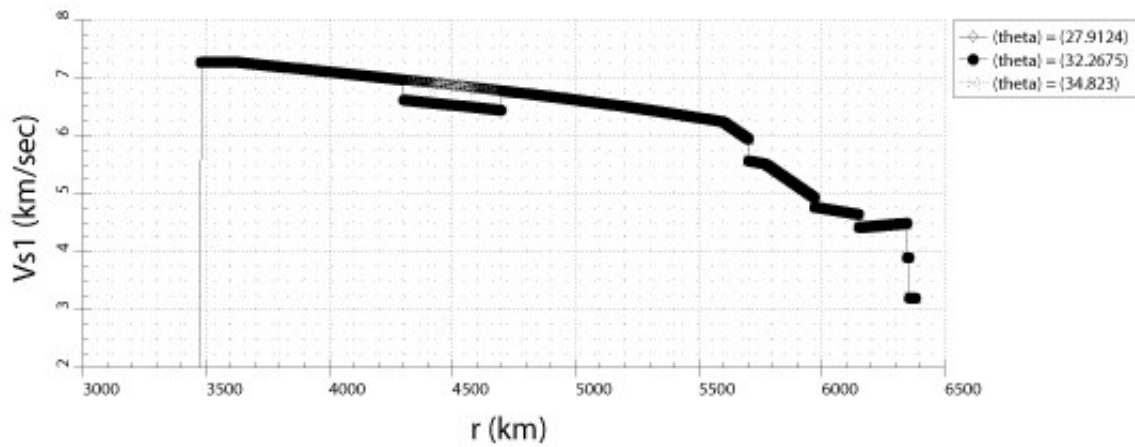
**Figure 7.8** Ncview prompter for SHaxi model file.

Clicking on Vs1 in the Ncview prompt will display the velocities for the model as shown in Figure 7.9.



**Figure 7.9** Ncview representation of S-wave velocities in SHaxi model.

One of the advantages of using Ncview is that clicking inside the model space produces a profile. So, for example clicking in a few different places in the image displayed in Figure 7.9 produces an *S*-wave velocity profile that can be inspected, zoomed into, printed, etc. An example is shown in Figure 7.10.



## Shear wave velocity from SHaxi model file

**Figure 7.10** Velocity profile of SHaxi model displayed in Ncview.

## 8 Additional Tools

Additional processing tools are located in the directory *shaxi/trunk/Tools/*

### 8.1 Processing Seismograms

The subdirectory *process\_seismos/* contains an example script of how to process SHaxi seismograms. With SHaxi the best practice is to produce seismograms with the variable *aa* in the parameter file set equal to -1. This uses a step as the source-time function. After the SHaxi job is complete, one needs to filter the output seismograms and supply a source-time function. This is most easily accomplished by convolving the unfiltered seismograms with the source-time function. The example given in the *process\_seismos/* directory shows how to convolve the SHaxi output seismograms with a Gaussian pulse. In general, the user will want to design their own scripts for processing seismograms.

### 8.2 Producing snapshots of wave propagation

The subdirectory *shnap/* contains some scripts for visualizing snapshots of the wave propagation. When generating seismograms it is best practice to set *aa* in the parameter file equal to -1. However, if snapshots is the desired output it is desirable to set *aa* to a positive number. The parameter *aa* should be set to the dominant period of the wavefield to be displayed in the snapshots.

Snapshot data is written in SHaxi as one file per process rank. In order to visualize the full wavefield these data for all ranks must be combined. This can be accomplished using the program *shnap.f90*. Program useage is as follows:

```
shell > ./shnap snap_number
```

where *snap\_number* is 000, 001, 010, etc. The program *shnap.f90* uses the Generic Mapping Tools (GMT) to visualize the wavefield. A GMT script (*gmtscript.csh*) is used to perform the GMT operations. This script is not standalone, but is called by the program *shnap.f90*. However, the GMT script can be altered to achieve the desired plot. To use the routines in *shnap.f90* one must first perform the following two steps:

- 1) The GMT script uses a special type of media to perform the plotting. Open the file `$GMTHOME/share/gmtmedia.d` for editing and add the line:  
`snap            672            504`
- 2) Edit the file *shnapdefaults*. This file contains the radial ( $minr \leq r \leq maxr$ ) and angular ( $mina \leq \theta \leq maxa$ ) range of the final plot. And a non-linear scaling factor *scale*. Set *scale* equal to 1 if no scaling of wavefield amplitudes is desired.

An example script *shnapall* shows how to run *shnap.f90* on all snapshot files in the current working directory.

The supplied GMT script also contains a line using ImageMagick's *convert* program to turn the GMT postscript files into JPEG files. Once the series of JPEG files is created, it is simple to generate an animation of the wave propagation. Adobe's ImageReady is a good choice as it has a function: `File > Import > Folder as Frames...` which will load all the JPEG images into one animation file.

## 9 File Structure

A complete list of SHaxi source files is shown in Table 9.1. The list is sorted in the order that the routine is called. The tree structure indicates within which program each subsequent program is called.

**Table 9.1 SHaxi File Structure**

Program Name	Description
<i>Modules used by various programs</i>	
mod_global.f90	Stores global variables
mod_cart2D.f90	MPI routines for Cartesian grid
plot_mod.f90	PGPlot X11 output routines
mod_attenuation.f90	Attenuation routines
<i>Main Calling sequence</i>	
fd2_main.f90	Main SHaxi driving routine
fd2_allocate.f90	Dynamic grid allocation
fd2_input.f90	Reads parameter file input
fd2_model.f90	Fills grid with model parameters
mod_simpleslab.f90	Slab model
mod_simplerift.f90	Rift model
mod_tomogrand.f90	Tomography cross sections
mod_modifycrust.f90	Thickened crust
mod_circle.f90	Circular velocity anomaly
mod_ncoutput.f90	netCDF output routines
mod_ncoutput_stub.f90	used if netCDF is unavailable
fd2_init.f90	Initialize grids, taper, source, ...
fd2_check.f90	Consistency checks of input parameters
fd2_evolution.f90	FD time evolution
fd2_deriv.f90	FD operators
fd2_output.f90	Seismogram and snapshot output
mod_sacoutput.f90	Writes SAC seismograms
plot_mod.f90	polar2cart transform and PGPlot output
mkpert.f90	standalone program for model perturbation

## 10 References

### 10.1 SHaxi References

Jahnke, G., Thorne, M.S., Cochard, A., Igel, H., Global SH-wave propagation using a parallel axi-symmetric spherical finite-difference scheme: application to whole mantle scattering, *Geophysical Journal International*, doi: 10.1111/j.1365-246X.2008.03744.x, 2008.

Thorne, M.S., Lay, T., Garnero, E.J., Jahnke, G., Igel, H., Seismic imaging of the laterally varying D" region beneath the Cocos Plate, *Geophysical Journal International* (170), pp. 635-648, doi: 10.1111/j.1365-246X.2006.03279.x, 2007.

Igel, H., and Weber, M. SH-wave propagation in the whole mantle using high-order finite differences, *Geophysical Research Letters*, 22 (6), pp. 731-734, 1995.

### 10.2 General References

Dziewonski, A.M., and Anderson, D.L. Preliminary reference Earth model, *Physics of the Earth and Planetary Interiors*, 25 (4), pp. 297-356, 1981.

Friederich, W., and Dalkolmo, J., Complete synthetic seismograms for a spherically symmetrical Earth by a numerical computation of the Greens-function in the frequency domain, *Geophys. J. Int.*, 122, pp. 537-550, 1995.

Gallovič, F., Barsch, R., Alvarez, J. P. & Igel, H., 2007. Digital Library for Computational Seismology, *Eos Trans. AGU*, **88**(50), 559.

Grand, S.P. (2002), Mantle shear-wave tomography and the fate of subducted slabs, *Philos. Trans. R. Soc. London, Ser. A*, 360 (1800), pp. 2475-2491, 2002.

Igel, H., 2004. New European training network to improve young scientists' capabilities in computational wave propagation, *Eos Trans. AGU*, **5**(28), 267.



## 11 Authors/Contact

The Authors of the SHaxi code and contact information are listed below.

**Gunnar Jahnke** ([gunnar@geophysik.uni-muenchen.de](mailto:gunnar@geophysik.uni-muenchen.de)) MPI parallelization, PGPlot graphics output, verification against analytical solution.

**Michael S. Thorne** ([michael.thorne@utah.edu](mailto:michael.thorne@utah.edu)) SAC output, user-interface routines, netCDF output, attenuation, basic models.

**Heiner Igel** ([heiner.igel@geophysik.uni-muenchen.de](mailto:heiner.igel@geophysik.uni-muenchen.de)) Author of the original serial version of the code.

## Development/Download

The reference URL of the SHaxi code is:

<http://www.spice-rtn.org/library/software/SHaxi>.

This page contains a basic description of the code together with a download link.

Currently, the software itself is located at the *Trac* server of the Geophysics section of the Ludwig-Maximilians-University in Munich:

<http://svn.geophysik.uni-muenchen.de/trac/shaxi> .