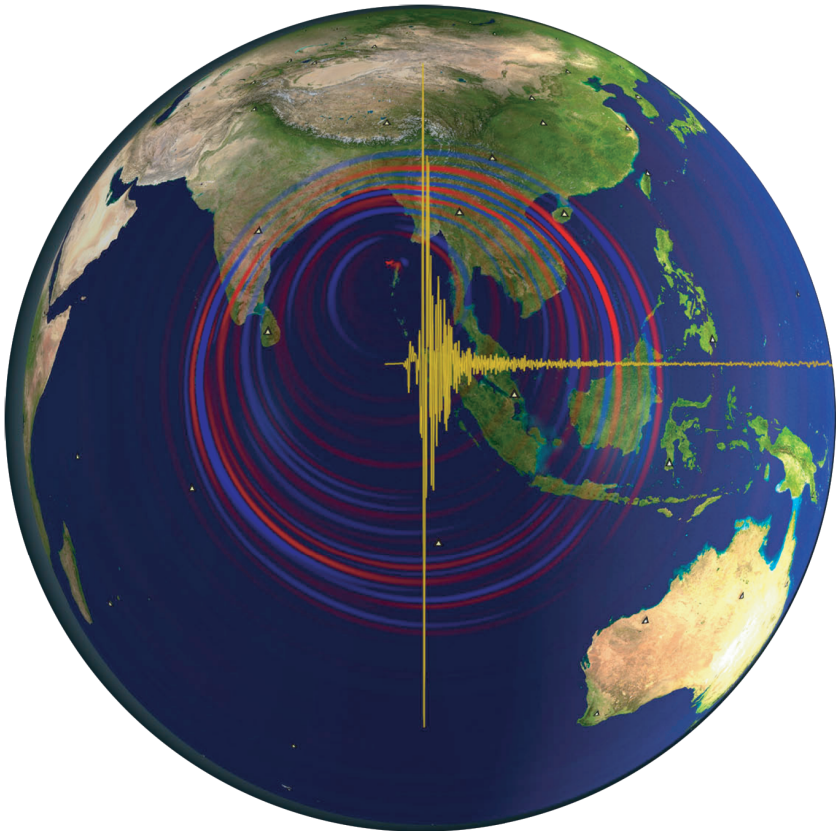
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)
PRINCETON UNIVERSITY (USA)
CNRS and UNIVERSITY OF MARSEILLE (FRANCE)
ETH ZÜRICH (SWITZERLAND)

SPECFEM 3D Globe

User Manual

Version 6.0



SPECFEM3D_GLOBE
User Manual

Version 6.0

February 26, 2021

Authors

The SPECFEM3D_GLOBE package was first developed by Dimitri Komatitsch and Jean-Pierre Vilotte at Institut de Physique du Globe (IPGP) in Paris, France from 1995 to 1997 and then by Dimitri Komatitsch and Jeroen Tromp at Harvard University and Caltech, USA, starting in 1998. The story started on March 28, 1995, when Prof. Yvon Maday from CNRS and University of Paris, France, gave a lecture to Dimitri Komatitsch and Jean-Pierre Vilotte at IPG about the nice properties of the spectral-element method that he had used for other equations. We are deeply indebted and thankful to him for that.

Since then it has been developed and maintained by a development team: in alphabetical order, Ebru Bozdağ, Joseph Charles, Min Chen, Dominik Göddeke, Vala Hjörleifsdóttir, Sue Kientz, Dimitri Komatitsch, Jesús Labarta, Nicolas Le Goff, Pieyre Le Loher, Qinya Liu, Yang Luo, Alessia Maggi, Roland Martin, Dennis McRitchie, Matthias Meschede, Peter Messmer, David Michéa, Tarje Nissen-Meyer, Daniel Peter, Max Rietmann, Elliott Sales de Andrade, Brian Savage, Bernhard Schuberth, Anne Sieminski, Leif Strand, Carl Tape, Jeroen Tromp, Jean-Pierre Vilotte, Zhinan Xie, Hejun Zhu.

The cover graphic of the manual was created by Santiago Lombeyda from Caltech's Center for Advanced Computing Research (CACR) (`http://www.cacr.caltech.edu`).

Current and past main participants or main sponsors

Contents

Chapter 1

Introduction

The software package SPECFEM3D_GLOBE simulates three-dimensional global and regional seismic wave propagation based upon the spectral-element method (SEM). The SEM is a continuous Galerkin technique [**??**], which can easily be made discontinous [**????????**]; it is then close to a particular case of the discontinuous Galerkin technique [**??????????????????????**], with optimized efficiency because of its tensorized basis functions [**??**]. In particular, it can accurately handle very distorted mesh elements [**?**].

It has very good accuracy and convergence properties [**??????????**]. The spectral element approach admits spectral rates of convergence and allows exploiting hp-convergence schemes. It is also very well suited to parallel implementation on very large supercomputers [**?????**] as well as on clusters of GPU accelerating graphics cards [**????**]. Tensor products inside each element can be optimized to reach very high efficiency [**?**], and mesh point and element numbering can be optimized to reduce processor cache misses and improve cache reuse [**?**]. The SEM can also handle triangular (in 2D) or tetrahedral (in 3D) elements [**?????**] as well as mixed meshes, although with increased cost and reduced accuracy in these elements, as in the discontinuous Galerkin method.

Note that in many geological models in the context of seismic wave propagation studies (except for instance for fault dynamic rupture studies, in which very high frequencies or supershear rupture need to be modeled near the fault, see e.g. **????**) a continuous formulation is sufficient because material property contrasts are not drastic and thus conforming mesh doubling bricks can efficiently handle mesh size variations [**?????**]. This is particularly true at the scale of the full Earth.

For a detailed introduction to the SEM as applied to global and regional seismic wave propagation, please consult **???????????**. A detailed theoretical analysis of the dispersion and stability properties of the SEM is available in **?**, **?**, **?**, **?** and **?**.

Effects due to lateral variations in compressional-wave speed, shear-wave speed, density, a 3D crustal model, ellipticity, topography and bathymetry, the oceans, rotation, and self-gravitation are included. The package can accommodate full 21-parameter anisotropy [**?**] as well as lateral variations in attenuation [**?**]. Adjoint capabilities and finite-frequency kernel simulations are also included [**??????**].

The SEM was originally developed in computational fluid dynamics [**??**] and has been successfully adapted to address problems in seismic wave propagation. Early seismic wave propagation applications of the SEM, utilizing Legendre basis functions and a perfectly diagonal mass matrix, include **?**, **?**, **?**, **?**, **?** and **?**, whereas applications involving Chebyshev basis functions and a nondiagonal mass matrix include **?**, **?** and **?**.

All SPECFEM3D_GLOBE software is written in Fortran2003 with full portability in mind, and conforms strictly to the Fortran2003 standard. It uses no obsolete or obsolescent features of Fortran. The package uses parallel programming based upon the Message Passing Interface (MPI) [**??**].

SPECFEM3D_GLOBE won the Gordon Bell award for best performance at the SuperComputing 2003 conference

in Phoenix, Arizona (USA) (see **?** and `www.sc-conference.org/sc2003/nr_finalaward.html`). It was a finalist again in 2008 for a run at 0.16 petaflops (sustained) on 149,784 processors of the 'Jaguar' Cray XT5 system at Oak Ridge National Laboratories (USA) [**?**]. It also won the BULL Joseph Fourier supercomputing award in 2010.

It reached the sustained one petaflop performance level for the first time in February 2013 on the Blue Waters Cray supercomputer at the National Center for Supercomputing Applications (NCSA), located at the University of Illinois at Urbana-Champaign (USA).

The package includes support for GPU graphics card acceleration [**????**] and also supports OpenCL.

The next release of the code will include support for Convolutional or Auxiliary Differential Equation Perfectly Matched absorbing Layers (C-PML or ADE-PML) [**?????**] for the case of single-chunk simulations in regional models.

1.1 Citation

If you use SPECFEM3D_GLOBE for your own research, please cite at least one of the following articles: **??????????????????** or **?**.

If you use this new version 6.0, which has non blocking MPI for much better performance for medium or large runs, please cite at least one of these five articles, in which results of 3D non blocking MPI runs are presented: **?????**.

If you use GPU graphics card acceleration please cite e.g. **?**, **?**, **?**, and/or **?**.

If you work on geophysical applications, you may be interested in citing some of these application articles as well, among others: **?????????????**. If you use 3D mantle model S20RTS, please cite **?**.

Domain decomposition is explained in detail in **?**, and excellent scaling up to 150,000 processor cores in shown for instance in **?????**.

The corresponding BibTeX entries may be found in file `doc/USER_MANUAL/bibliography.bib`.

1.2 Support

This material is based upon work supported by the U.S. National Science Foundation under Grants No. EAR-0406751 and EAR-0711177, by the French CNRS, French INRIA Sud-Ouest MAGIQUE-3D, French ANR NUMASIS under Grant No. ANR-05-CIGC-002, and European FP6 Marie Curie International Reintegration Grant No. MIRG-CT-2005-017461. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. National Science Foundation, CNRS, INRIA, ANR or the European Marie Curie program.

Chapter 2

Getting Started

2.1 Configuring and compiling the source code

To get the SPECFEM3D_GLOBE software package, type this:

```
git clone -recursive -branch devel
https://github.com/geodynamics/specfem3d_globe.git
```

We recommend that you add `ulimit -S -s unlimited` to your `.bash_profile` file and/or `limit stacksize unlimited` to your `.cshrc` file to suppress any potential limit to the size of the Unix stack.

Then, to configure the software for your system, run the `configure` shell script. This script will attempt to guess the appropriate configuration values for your system. However, at a minimum, it is recommended that you explicitly specify the appropriate command names for your Fortran compiler and MPI package (another option is to define FC, CC and MPIF90 in your .bash_profile or your .cshrc file):

```
./configure FC=ifort MPIFC=mpif90
```

Before running the `configure` script, you should probably edit file `flags.guess` to make sure that it contains the best compiler options for your system. Known issues or things to check are:

`Intel ifort compiler` See if you need to add `-assume byterecl` for your machine.

`IBM compiler` See if you need to add `-qsave` or `-qnosave` for your machine.

`Mac OS` You will probably need to install `XCODE`.

When compiling on an IBM machine with the `xlf` and `xlc` compilers, we suggest running the `configure` script with the following options:

```
./configure FC=xlf90_r MPIFC=mpif90 CC=xlc_r CFLAGS="-O3 -q64" FCFLAGS="-O3 -q64".
```

On SGI systems, `flags.guess` automatically informs `configure` to insert "`TRAP_FPE=OFF`" into the generated `Makefile` in order to turn underflow trapping off.

If you run very large meshes on a relatively small number of processors, the static memory size needed on each processor might become greater than 2 gigabytes, which is the upper limit for 32-bit addressing (dynamic memory allocation is always OK, even beyond the 2 GB limit; only static memory has a problem). In this case, on some compilers you may need to add "`-mcmodel=medium`" (if you do not use the Intel ifort / icc compiler) or "`-mcmodel=medium -shared-intel`" (if you use the Intel ifort / icc compiler) to the configure options of CFLAGS, FCFLAGS and LDFLAGS otherwise the compiler will display an error message (for instance "`relocation truncated to fit: R_X86_64_PC32 against .bss`" or something similar); on an IBM machine with the `xlf` and `xlc`

compilers, using `-q64` is usually sufficient.

A summary of the most important configuration variables follows.

FC Fortran compiler command name. By default, `configure` will execute the command names of various well-known Fortran compilers in succession, picking the first one it finds that works.

MPIFC MPI Fortran command name. The default is `mpif90`. This must correspond to the same underlying compiler specified by `FC`; otherwise, you will encounter compilation or link errors when you attempt to build the code. If you are unsure about this, it is usually safe to set both `FC` and `MPIFC` to the MPI compiler command for your system:

```
./configure FC=mpif90 MPIFC=mpif90
```

FLAGS_CHECK Compiler flags.

LOCAL_PATH_IS_ALSO_GLOBAL If you want the parallel mesher to write a parallel (i.e., split) database for the solver on the local disks of each of the compute nodes, set this flag to `.false.`. Some systems have no local disks (e.g., BlueGene) and other systems have a fast parallel file system (LUSTRE, GPFS) that is easy and reliable to use, in which case this variable should be set to `.true.`. Note that this flag is not used by the mesher nor the solver; it is only used for some of the (optional) post-processing. If you do not know what is best on your system, setting it to `.true.` is usually fine; or else, ask your system administrator.

In addition to reading configuration variables, `configure` accepts the following options:

--enable-double-precision The package can run either in single or in double precision mode. The default is single precision because for almost all calculations performed using the spectral-element method using single precision is sufficient and gives the same results (i.e. the same seismograms); and the single precision code is faster and requires exactly half as much memory. To specify double precision mode, simply provide `--enable-double-precision` as a command-line argument to `configure`. On many current processors (e.g., Intel, AMD, IBM Power), single precision calculations are significantly faster; the difference can typically be 10% to 25%. It is therefore better to use single precision. What you can do once for the physical problem you want to study is run the same calculation in single precision and in double precision on your system and compare the seismograms. If they are identical (and in most cases they will), you can select single precision for your future runs.

--help Directs `configure` to print a usage screen which provides a short description of all configuration variables and options. Note that the options relating to installation directories (e.g., `--prefix`) do not apply to SPECFEM3D_GLOBE.

The `configure` script runs a brief series of checks. Upon successful completion, it generates the files `Makefile`, `constants.h`, and `precision.h` in the working directory.

Note: If the `configure` script fails, and you don't know what went wrong, examine the log file `config.log`. This file contains a detailed transcript of all the checks `configure` performed. Most importantly, it includes the error output (if any) from your compiler.

The `configure` script automatically runs the script `flags.guess`. This helper script contains a number of suggested flags for various compilers; e.g., Portland, Intel, Absoft, NAG, Lahey, NEC, IBM and SGI. The software has run on a wide variety of compute platforms, e.g., various PC clusters and machines from Sun, SGI, IBM, Compaq, and NEC. The `flags.guess` script attempts to guess which compiler you are using (based upon the compiler command name) and choose the related optimization flags. The `configure` script then automatically inserts the suggested flags into `Makefile`. Note that `flags.guess` may fail to identify your compiler; and in any event, the default flags chosen by `flags.guess` are undoubtedly not optimal for your system. So, we encourage you to experiment with these flags (by editing the generated `Makefile` by hand) and to solicit advice from your system administrator. Selecting the right compiler and compiler flags can make a tremendous difference in terms of performance. We welcome feedback on your experience with various compilers and flags.

When using a slow or not too powerful shared disk system or when running extremely large simulations (on tens of thousands of processor cores), one can add `-DUSE_SERIAL_CASCADE_FOR_IOs` to the compiler flags in file `flags.guess` before running `configure` to make the mesher output mesh data to the disk for one MPI slice after the other, and to make the solver do the same thing when reading the files back from disk. Do not use this option if you do not need it because it will slow down the mesher and the beginning of the solver if your shared file system is fast and reliable.

If you run scaling benchmarks of the code, for instance to measure its performance on a new machine, and are not interested in the physical results (the seismograms) for these runs, you can set `DO_BENCHMARK_RUN_ONLY` to `.true.` in file `setup/constants.h.in` before running the `configure` script.

If your compiler has problems with the `use mpi` statements that are used in the code, use the script called `replace_use_mpi_with_include_mpif_dot_h.pl` in the root directory to replace all of them with `include 'mpif.h'` automatically.

We recommend that you ask for exclusive use of the compute nodes when running on a cluster or a supercomputer, i.e., make sure that no other users are running on the same nodes at the same time. Otherwise your run could run out of memory if the memory of some nodes is used by other users, in particular when undoing attenuation using the UNDO_ATTENUATION option in DATA/Par_file. To do so, ask your system administrator for the option to add to your batch submission script; it is for instance `#BSUB -x` with SLURM and `#$ -l exclusive=TRUE` with Sun Grid Engine (SGE).

2.2 Compiling on an IBM BlueGene

More recent installation instruction for IBM BlueGene, from October 2012:

Edit file `flags.guess` and put this for `FLAGS_CHECK`:

```
-g -qfullpath -O2 -qsave -qstrict -qtune=qp -qarch=qp -qcache=auto -qhalt=w
-qfree=f90 -qsuffix=f=f90 -qlanglvl=95pure -Q -Q+rank,swap_all -Wl,-relax
```

The most relevant are the -qarch and -qtune flags, otherwise if these flags are set to “auto” then they are wrongly assigned to the architecture of the frond-end node, which is different from that on the compute nodes. You will need to set these flags to the right architecture for your BlueGene compute nodes, which is not necessarily “qp”; ask your system administrator. On some machines if is necessary to use -O2 in these flags instead of -O3 due to a compiler bug of the XLF version installed. We thus suggest to first try -O3, and then if the code does not compile or does not run fine then switch back to -O2. The debug flags (-g, -qfullpath) do not influence performance but are useful to get at least some insights in case of problems.

Before running `configure`, select the XL Fortran compiler by typing `module load bgq-xl/1.0` or `module load bgq-xl` (another, less efficient option is to load the GNU compilers using `module load bgq-gnu/4.4.6` or similar).

Then, to configure the code, type this:

```
./configure FC=bgxlf90_r MPIFC=mpixlf90_r CC=bgxlc_r LOCAL_PATH_IS_ALSO_GLOBAL=true
```

Older installation instruction for IBM BlueGene, from 2011:

To compile the code on an IBM BlueGene, Laurent Léger from IDRIS, France, suggests the following: compile the code with

```
 FLAGS_CHECK="-O3 -qsave -qstrict -qtune=auto -qarch=450d -qcache=auto
-qfree=f90 -qsuffix=f=f90 -g -qlanglvl=95pure -qhalt=w -Q -Q+rank,swap_all -Wl,-relax"
```

Option "-Wl,-relax" must be added on many (but not all) BlueGene systems to be able to link the binaries `xmeshfem3D` and `xspecfem3D` because the final link step is done by the GNU `ld` linker even if one uses `FC=bgxlf90_r`,

MPIFC=mpixlf90_r and CC=bgxlc_r to create all the object files. On the contrary, on some BlueGene systems that use the native AIX linker option "-Wl,-relax" can lead to problems and must be suppressed from flags.guess. One then just needs to pass the right commands to the configure script:

```
 ./configure -prefix=/path/to/SPECFEM3DG_SP -host=Babel -build=BGP
FC=bgxlf90_r MPIFC=mpixlf90_r CC=bgxlc_r
LOCAL_PATH_IS_ALSO_GLOBAL=false
```

This trick can be useful for all hosts on which one needs to cross-compile.
On BlueGene, one also needs to run the xcreate_header_file binary file manually rather than in the Makefile:

```
bgrun -np 1 -mode VN -exe ./bin/xcreate_header_file
```

2.3 Using a cross compiler

The "configure" script assumes that you will compile the code on the same kind of hardware as the machine on which you will run it. On some systems (for instance IBM BlueGene, see also the previous section) this might not be the case and you may compile the code using a cross compiler on a frontend computer that does not have the same architecture. In such a case, typing "make all" on the frontend will fail, but you can use one of these two solutions:

1/ create a script that runs "make all" on a node instead of on the frontend, if the compiler is also installed on the nodes

2/ after running the "configure" script, create two copies of the Makefiles:
TODO: this has not been tested out yet, any feedback is welcome

In src/create_header_file/Makefile put this instead of the current values:

```
FLAGS_CHECK = -O0
```

and replace

```
create_header_file: $O/create_header_file.o $(XCREATE_HEADER_OBJECTS)
 ${FCCOMPILE_CHECK} -o ${E}/xcreate_header_file $O/create_header_file.o $(XCREATE_HEADER_O
```

with

```
xcreate_header_file: $O/create_header_file.o $(XCREATE_HEADER_OBJECTS)
  ${MPIFCCOMPILE_CHECK} -o ${E}/xcreate_header_file $O/create_header_file.o $(XCREATE_HEADE
```

In src/specfem3D/Makefile comment out these last two lines:

```
#${OUTPUT}/values_from_mesher.h: reqheader
#       (mkdir -p ${OUTPUT}; cd ${S_TOP}/; ./bin/xcreate_header_file)
```

Then:

```
 make clean
 make create_header_file
 ./bin/xcreate_header_file
 make clean
 make meshfem3D
 make specfem3D
```

should work.

2.4 Adding OpenMP support in addition to MPI

OpenMP support can be enabled in addition to MPI. However, in many cases performance will not improve because our pure MPI implementation is already heavily optimized and thus the resulting code will in fact be slightly slower. A possible exception could be IBM BlueGene-type architectures.

To enable OpenMP, uncomment the OpenMP compiler option in two lines in file `src/specfem3D/Makefile.in` (before running `configure`) and also uncomment the `#define USE_OPENMP` statement in file `src/specfem3D/specfem3D.F90`.

The DO-loop using OpenMP threads has a SCHEDULE property. The OMP_SCHEDULE environment variable can set the scheduling policy of that DO-loop. Tests performed by Marcin Zielinski at SARA (The Netherlands) showed that often the best scheduling policy is DYNAMIC with the size of the chunk equal to the number of OpenMP threads, but most preferably being twice as the number of OpenMP threads (thus chunk size = 8 for 4 OpenMP threads etc). If OMP_SCHEDULE is not set or is empty, the DO-loop will assume generic scheduling policy, which will slow down the job quite a bit.

2.5 Visualizing the subroutine calling tree of the source code

Packages such as `doxywizard` can be used to visualize the subroutine calling tree of the source code. `Doxywizard` is a GUI front-end for configuring and running `doxygen`.

Chapter 3

Running the Mesher `xmeshfem3D`

You are now ready to compile the mesher. In the directory with the source code, type '`make meshfem3D`'. If all paths and flags have been set correctly, the mesher should now compile and produce the executable `xmeshfem3D`. Note that all compiled executables are placed into the directory `bin/`. To run the executables, you must call them from the root directory, for example type '`mpirun -np 64 ./bin/meshfem3D`' to run the mesher in parallel on 64 CPUs. This will allow the executables to find the parameter file `Par_file` in the relative directory location `./DATA`.

Input for the mesher (and the solver) is provided through the parameter file `Par_file`, which resides in the subdirectory `DATA`. Before running the mesher, a number of parameters need to be set in the `Par_file`. This requires a basic understanding of how the SEM is implemented, and we encourage you to read **????????** and **?**. A detailed theoretical analysis of the dispersion and stability properties of the SEM is available in **?**, **?** and **?**.

In this chapter we will focus on simulations at the scale of the entire globe. Regional simulations will be addressed in Chapter 5. The spectral-element mesh for a SPECFEM3D_GLOBE simulation is based upon a mapping from the cube to the sphere called the *cubed sphere* [**??**]. This cubed-sphere mapping breaks the globe into 6 chunks, each of which is further subdivided in terms of n^2 mesh slices, where $n \geq 1$ is a positive integer, for a total of $6 \times n^2$ slices (Figure 3.1). Thus the minimum number of processors required for a global simulation is 6 (although it is theoretically possible to run more than one slice per processor).

To run the mesher for a global simulation, the following parameters need to be set in the `Par_file`:

`SIMULATION_TYPE` is set to 1 for forward simulations, 2 for adjoint simulations for sources (see Section 6.1) and 3 for kernel simulations (see Section 8.3).

`SAVE_FORWARD` is only set to `.true.` for a forward simulation with the last frame of the simulation saved, as part of the finite-frequency kernel calculations (see Section 8.3). For a regular forward simulation, leave `SIMULATION_TYPE` and `SAVE_FORWARD` at their default values.

`NCHUNKS` must be set to 6 for global simulations.

`ANGULAR_WIDTH_XI_IN_DEGREES` Not needed for a global simulation. (See Chapter 5 for regional simulations.)

`ANGULAR_WIDTH_ETA_IN_DEGREES` Not needed for a global simulation. (See Chapter 5 for regional simulations.)

`CENTER_LATITUDE_IN_DEGREES` Not needed for a global simulation. (See Chapter 5 for regional simulations.)

`CENTER_LONGITUDE_IN_DEGREES` Not needed for a global simulation. (See Chapter 5 for regional simulations.)

`GAMMA_ROTATION_AZIMUTH` Not needed for a global simulation. (See Chapter 5 for regional simulations.)

`NEX_XI` The number of spectral elements along one side of a chunk in the cubed sphere (see Figure 3.1); this number *must* be a multiple of 16 and 8 $\times$ a multiple of `NPROC_XI` defined below. We do not recommend using `NEX_XI` less than 64 because the curvature of the Earth cannot be honored if one uses too few elements, and distorted elements can lead to inaccurate and unstable simulations, i.e., smaller values of `NEX_XI` are likely to

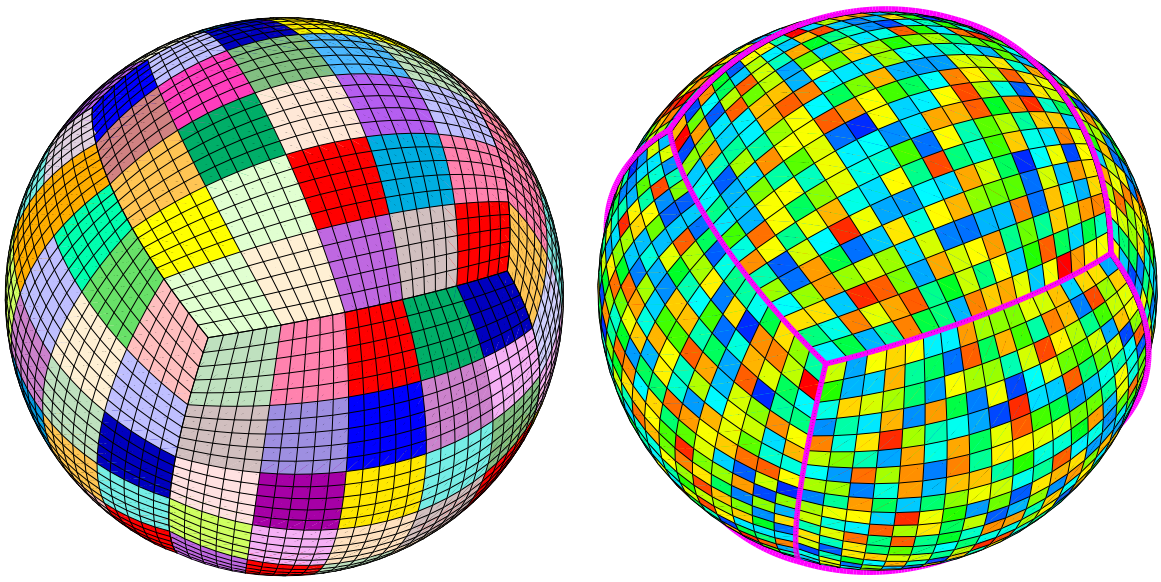


Figure 3.1: Each of the 6 chunks that constitutes the cubed sphere is subdivided in terms of n^2 slices of elements, where $n \geq 1$ is a positive integer, for a total of $6 \times n^2$ slices (and therefore processors). The figure on the left shows a mesh that is divided in terms of $6 \times 5^2 = 150$ slices as indicated by the various colors. In this cartoon, each slice contains $5 \times 5 = 25$ spectral elements at the Earth's surface. The figure on the right shows a mesh that is divided over $6 \times 18^2 = 1944$ processors as indicated by the various colors. Regional simulations can be accommodated by using only 1, 2 or 3 chunks of the cubed sphere. One-chunk simulations may involve a mesh with lateral dimensions smaller than 90°, thereby accommodating smaller-scale simulations.

result in spectral elements with a negative Jacobian, in which case the mesher will exit with an error message. Table 3 summarizes various suitable choices for NEX_XI and the related values of NPROC_XI. Based upon benchmarks against semi-analytical normal-mode synthetic seismograms, **??** determined that a NEX_XI $= 256$ run is accurate to a shortest period of roughly 17 s. Therefore, since accuracy is determined by the number of grid points per shortest wavelength, for any particular value of NEX_XI the simulation will be accurate to a shortest period determined approximately by

$$\text{shortest period (s)} \simeq (256/\texttt{NEX\_XI}) \times 17. \tag{3.1}$$

The number of grid points in each orthogonal direction of the reference element, i.e., the number of Gauss-Lobatto-Legendre points, is determined by NGLLX in the `constants.h` file. In the globe we use NGLLX $= 5$, for a total of $5^3 = 125$ points per elements. We suggest not to change this value.

NEX_ETA For global simulations NEX_ETA must be set to the same value as NEX_XI.

NPROC_XI The number of processors or slices along one chunk of the cubed sphere (see Figure 3.1); we must have NEX_XI $= 8 \times c \times$ NPROC_XI, where $c \geq 1$ is a positive integer. See Table 3 for various suitable choices.

NPROC_ETA For global simulations NPROC_ETA must be set to the same value as NPROC_XI.

MODEL Must be set to one of the following:

1D models with real structure:

1D_isotropic_prem Isotropic version of the spherically symmetric Preliminary Reference Earth Model (PREM) [**?**].

1D_transversely_isotropic_prem Transversely isotropic version of PREM.

1D_iasp91 Spherically symmetric isotropic IASP91 model [**?**].

`1D_1066a` Spherically symmetric earth model 1066A [**?**]. When `ATTENTUATION` is on, it uses an unpublished 1D attenuation model from Scripps.

`1D_ak135f_no_mud` Spherically symmetric isotropic AK135 model [**?**] modified to use the density and Q attenuation models of **?**. That modified model is traditionally called AK135-F, see `http://rses.anu.edu.au/seismology/ak135/ak135f.html` for more details. As we do not want to use the 300 m-thick mud layer from that model nor the ocean layer, above the d120 discontinuity we switch back to the classical AK135 model of **?**, i.e., we use AK135-F below and AK135 above.

`1D_ref` A recent 1D Earth model developed by **?**. This model is the 1D background model for the 3D models s362ani, s362wmani, s362ani_prem, and s29ea.

For historical reasons and to provide benchmarks against normal-mode synthetics, the mesher accommodates versions of various 1D models with a single crustal layer with the properties of the original upper crust. These 'one-crust' models are:

`1D_isotropic_prem_onecrust`
`1D_transversely_isotropic_prem_onecrust`
`1D_iasp91_onecrust`, `1D_1066a_onecrust`
`1D_ak135f_no_mud_onecrust`

Fully 3D models:

`transversely_isotropic_prem_plus_3D_crust_2.0` This model has CRUST2.0 [**?**] on top of a transversely isotropic PREM. We first extrapolate PREM mantle velocity up to the surface, then overwrite the model with CRUST2.0

`s20rts` By default, the code uses 3D mantle model S20RTS [**?**] and 3D crustal model Crust2.0 [**?**]. Note that S20RTS uses transversely isotropic PREM as a background model, and that we use the PREM radial attenuation model when `ATTENUATION` is incorporated. See Chapter 10 for a discussion on how to change 3D models.

`s40rts` A global 3D mantle model [**?**] succeeding S20RTS with a higher resolution. S40RTS uses transversely isotropic PREM as a background model and the 3D crustal model Crust2.0 [**?**]. We use the PREM radial attenuation model when `ATTENUATION` is incorporated.

`s362ani` A global shear-wave speed model developed by **?**. In this model, radial anisotropy is confined to the uppermost mantle. The model (and the corresponding mesh) incorporate tomography on the 650~km and 410~km discontinuities in the 1D reference model REF.

`s362wmani` A version of S362ANI with anisotropy allowed throughout the mantle.

`s362ani_prem` A version of S362ANI calculated using PREM as the 1D reference model.

`s29ea` A global model with higher resolution in the upper mantle beneath Eurasia calculated using REF as the 1D reference model.

`3D_anisotropic` See Chapter 10 for a discussion on how to specify your own 3D anisotropic model.

`3D_attenuation` See Chapter 10 for a discussion on how to specify your own 3D attenuation model.

`PPM` For a user-specified 3D model (Point-Profile-Model) given as ASCII-table, specifying Vs-perturbations with respect to PREM. See Chapter 10 for a discussion on how to specify your own 3D model.

NOTE:

When a 3D mantle model is chosen in `Par_file`, the simulations are performed together with the 3D crustal model Crust2.0. Alternatively, Crust2.0 can be combined with a higher resolution European crustal model EUCrust07 [**?**]. This can be done by setting the crustal type to `ICRUST_CRUSTMAPS` in the `constant.h` file. It is also possible to run simulations using a 3D mantle model with a 1D crustal model on top. This can be done by setting the model in `Par_file` to `<3D mantle>_1Dcrust`, e.g., `s20rts_1Dcrust`, `s362ani_1Dcrust`, etc. In this case, the 1D crustal model will be the one that is used in the 3D mantle model as a reference model (e.g., transversely isotropic PREM for s20rts, REF for s362ani, etc.).

OCEANS Set to `.true.` if the effect of the oceans on seismic wave propagation should be incorporated based upon the approximate treatment discussed in **?**. This feature is inexpensive from a numerical perspective, both in terms of memory requirements and CPU time. This approximation is accurate at periods of roughly 20 s and longer. At shorter periods the effect of water phases/reverberations is not taken into account, even when the flag is on.

ELLIPTICITY Set to `.true.` if the mesh should make the Earth model elliptical in shape according to Clairaut's equation [**?**]. This feature adds no cost to the simulation. From **?**: "Spherically-symmetric Earth models all have the same hydrostatic surface ellipticity 1/299.8. This is 0.5 percent smaller than observed flattening of best-fitting ellipsoid 1/298.3. The discrepancy is referred to as the "excess equatorial bulge of the Earth", an early discovery of artificial satellite geodesy." From Paul Melchior, IUGG General Assembly, Vienna, Austria, August 1991 Union lecture, available at http://www.agu.org/books: "It turns out that the spheroidal models constructed on the basis of the spherically-symmetric models (PREM, 1066A) by using the Clairaut differential equation to calculate the flattening in function of the radius vector imply hydrostaticity. These have surface ellipticity 1/299.8 and a corresponding dynamical flattening of .0033 (PREM). The actual ellipticty of the Earth for a best-fitting ellipsoid is 1/298.3 with a corresponding dynamical flattening of .0034." Thus, flattening f = 1/299.8 is what is used in SPECFEM3D_GLOBE, as it should. And thus eccentricity squared $e^2 = 1 - (1-f)^2 = 1 - (1-1/299.8)^2 = 0.00665998813529$, and the correction factor used in the code to convert geographic latitudes to geocentric is $1-e^2 = (1-f)^2 = (1-1/299.8)^2 = 0.9933400118647$. As a comparison, the classical World Geodetic System reference ellipsoid WGS 84 (see e.g. http://en.wikipedia.org/wiki/World_Geodetic_System) has $f = 1/298.2572236$.

TOPOGRAPHY Set to `.true.` if topography and bathymetry should be incorporated based upon model ETOPO5 [**?**]. This feature adds no cost to the simulation.

GRAVITY Set to `.true.` if self-gravitation should be incorporated in the Cowling approximation [**??**]. Turning this feature on is relatively inexpensive, both from the perspective of memory requirements as well as in terms of computational speed.

ROTATION Set to `.true.` if the Coriolis effect should be incorporated. Turning this feature on is relatively cheap numerically.

ATTENUATION Set to `.true.` if attenuation should be incorporated. Turning this feature on increases the memory requirements significantly (roughly by a factor of 2), and is numerically fairly expensive. Of course for realistic simulations this flag should be turned on. See **??** for a discussion on the implementation of attenuation based upon standard linear solids.

ABSORBING_CONDITIONS Set to `.false.` for global simulations. See Chapter 5 for regional simulations.

RECORD_LENGTH_IN_MINUTES Choose the desired record length of the synthetic seismograms (in minutes). This controls the length of the numerical simulation, i.e., twice the record length requires twice as much CPU time. This feature is not used at the time of meshing but is required for the solver, i.e., you may change this parameter after running the mesher.

MOVIE_SURFACE Set to `.false.`, unless you want to create a movie of seismic wave propagation on the Earth's surface. Turning this option on generates large output files. See Section 8.2 for a discussion on the generation of movies. This feature is not used at the time of meshing but is relevant for the solver.

MOVIE_VOLUME Set to `.false.`, unless you want to create a movie of seismic wave propagation in the Earth's interior. Turning this option on generates huge output files. See Section 8.2 for a discussion on the generation of movies. This feature is not used at the time of meshing but is relevant for the solver.

NTSTEP_BETWEEN_FRAMES Determines the number of timesteps between movie frames. Typically you want to save a snapshot every 100 timesteps. The smaller you make this number the more output will be generated! See Section 8.2 for a discussion on the generation of movies. This feature is not used at the time of meshing but is relevant for the solver.

HDUR_MOVIE determines the half duration of the source time function for the movie simulations. When this parameter is set to be 0, a default half duration that corresponds to the accuracy of the simulation is provided.

SAVE_MESH_FILES Set this flag to .true. to save AVS (www.avs.com), OpenDX (www.opendx.org), or ParaView (www.paraview.org) mesh files for subsequent viewing. Turning the flag on generates large (distributed) files in the LOCAL_PATH directory. See Section 8.1 for a discussion of mesh viewing features.

NUMBER_OF_RUNS On machines with a run-time limit, for instance for a batch/queue system, a simulation may need to be completed in stages. This option allows you to select the number of stages in which the simulation will be completed (1, 2 or 3). Choose 1 for a run without restart files. This feature is not used at the time of meshing but is required for the solver. At the end of the first or second stage of a multi-stage simulation, large files are written to the file system to save the current state of the simulation. This state is read back from the file system at the beginning of the next stage of the multi-stage run. Reading and writing the states can be very time consuming depending on the nature of the network and the file system (in this case writing to the local file system, i.e., the disk on a node, is preferable).

NUMBER_OF_THIS_RUN If you choose to perform the run in stages, you need to tell the solver what stage run to perform. This feature is not used at the time of meshing but is required for the solver.

LOCAL_PATH Directory in which the databases generated by the mesher will be written. Generally one uses a directory on the local disk of the compute nodes, although on some machines these databases are written on a parallel (global) file system (see also the earlier discussion of the LOCAL_PATH_IS_ALSO_GLOBAL flag in Chapter 2). The mesher generates the necessary databases in parallel, one set for each of the $6 \times \text{NPROC\_XI}^2$ slices that constitutes the mesh (see Figure 3.1). After the mesher finishes, you can log in to one of the compute nodes and view the contents of the LOCAL_PATH directory to see the (many) files generated by the mesher.

NTSTEP_BETWEEN_OUTPUT_INFO This parameter specifies the interval at which basic information about a run is written to the file system (timestamp* files in the OUTPUT_FILES directory). If you have access to a fast machine, set NTSTEP_BETWEEN_OUTPUT_INFO to a relatively high value (e.g., at least 100, or even 1000 or more) to avoid writing output text files too often. This feature is not used at the time of meshing. One can set this parameter to a larger value than the number of time steps to avoid writing output during the run.

NTSTEP_BETWEEN_OUTPUT_SEISMOS This parameter specifies the interval at which synthetic seismograms are written in the LOCAL_PATH directory. The seismograms can be created in three different formats by setting the parameters OUTPUT_SEISMOS_ASCII_TEXT, OUTPUT_SEISMOS_SAC_ALPHANUM and OUTPUT_SEISMOS_SAC_BINARY. One can choose any combination of these parameters (details on the formats follow in the description of each parameter). SAC (www.iris.edu/software/sac/) is a signal-processing software package. If a run crashes, you may still find usable (but shorter than requested) seismograms in this directory. On a fast machine set NTSTEP_BETWEEN_OUTPUT_SEISMOS to a relatively high value to avoid writing to the seismograms too often. This feature is not used at the time of meshing.

NTSTEP_BETWEEN_READ_ADJSRC The number of adjoint sources read in each time for an adjoint simulation.

OUTPUT_SEISMOS_ASCII_TEXT Set this flag to .true. if you want to have the synthetic seismograms written in two-column ASCII format (the first column contains time in seconds and the second column the displacement in meters of the recorded signal, no header information). Files will be named with extension .ascii, e.g., STA.NT.?X?.sem.ascii where STA and NT are the station and network codes given in STATIONS file, and ?X? is the channel code (see Appendix E).

OUTPUT_SEISMOS_SAC_ALPHANUM Set this flag to .true. if you want to have the synthetic seismograms written in alphanumeric (human readable) SAC format, which includes header information on the source and receiver parameters (e.g., source/receiver coordinates, station name, etc., see Appendix D). For details on the format, please check the SAC webpage (www.iris.edu/software/sac/). Files will be named with extension .sacan.

OUTPUT_SEISMOS_SAC_BINARY Set this flag to .true. if you want to have the synthetic seismograms written in binary SAC format. The header information included is the same as for the alphanumeric SAC format (see Appendix D). Using this format requires the least disk space, which may be particulary important if you have

a large number of stations. For details on the binary format please also check the SAC webpage (`www.iris.edu/software/sac/`) and Appendix D. Files will be named with extension `.sac`.

ROTATE_SEISMOGRAMS_RT Set this flag to `.true.` if you want to have radial (R) and transverse (T) horizontal components of the synthetic seismograms (default is `.false.` → East (E) and North (N) components).

WRITE_SEISMOGRAMS_BY_MASTER Set this flag to `.true.` if you want to have all the seismograms written by the master (no need to collect them on the nodes after the run).

SAVE_ALL_SEISMOS_IN_ONE_FILE Set this flag to `.true.` if you want to have all the seismograms saved in one large combined file instead of one file per seismogram to avoid overloading shared non-local file systems such as GPFS for instance.

USE_BINARY_FOR_LARGE_FILE Set this flag to `.true.` if you want to use binary instead of ASCII for that large file (not used if SAVE_ALL_SEISMOS_IN _ONE_FILE = .false.)

RECEIVERS_CAN_BE_BURIED This flag accommodates stations with instruments that are buried, i.e., the solver will calculate seismograms at the burial depth specified in the `STATIONS` file. This feature is not used at the time of meshing.

PRINT_SOURCE_TIME_FUNCTION Turn this flag on to print information about the source time function in the file `OUTPUT_FILES/plot_source_time_function.txt`. This feature is not used at the time of meshing.

NPROC_XI	processors	NEX_XI									
1	6	64	80	96	112	128	144	160	176	192	208
2	24	64	80	96	112	128	144	160	176	192	208
3	54	96	144	192	240	288	336	384	432	480	528
4	96	64	96	128	160	192	224	256	288	320	352
5	150	80	160	240	320	400	480	560	640	720	800
6	216	96	144	192	240	288	336	384	432	480	528
7	294	112	224	336	448	560	672	784	896	1008	1120
8	384	64	128	192	256	320	384	448	512	576	640
9	486	144	288	432	576	720	864	1008	1152	1296	1440
10	600	80	160	240	320	400	480	560	640	720	800
11	726	176	352	528	704	880	1056	1232	1408	1584	1760
12	864	96	192	288	384	480	576	672	768	864	960
13	1014	208	416	624	832	1040	1248	1456	1664	1872	2080
14	1176	112	224	336	448	560	672	784	896	1008	1120
15	1350	240	480	720	960	1200	1440	1680	1920	2160	2400
16	1536	128	256	384	512	640	768	896	1024	1152	1280
17	1734	272	544	816	1088	1360	1632	1904	2176	2448	2720
18	1944	144	288	432	576	720	864	1008	1152	1296	1440
19	2166	304	608	912	1216	1520	1824	2128	2432	2736	3040
20	2400	160	320	480	640	800	960	1120	1280	1440	1600
21	2646	336	672	1008	1344	1680	2016	2352	2688	3024	3360
22	2904	176	352	528	704	880	1056	1232	1408	1584	1760
23	3174	368	736	1104	1472	1840	2208	2576	2944	3312	3680
24	3456	192	384	576	768	960	1152	1344	1536	1728	1920
25	3750	400	800	1200	1600	2000	2400	2800	3200	3600	4000
26	4056	208	416	624	832	1040	1248	1456	1664	1872	2080
27	4374	432	864	1296	1728	2160	2592	3024	3456	3888	4320
28	4704	224	448	672	896	1120	1344	1568	1792	2016	2240
29	5046	464	928	1392	1856	2320	2784	3248	3712	4176	4640
30	5400	240	480	720	960	1200	1440	1680	1920	2160	2400
31	5766	496	992	1488	1984	2480	2976	3472	3968	4464	4960
32	6144	256	512	768	1024	1280	1536	1792	2048	2304	2560
33	6534	528	1056	1584	2112	2640	3168	3696	4224	4752	5280
34	6936	272	544	816	1088	1360	1632	1904	2176	2448	2720
35	7350	560	1120	1680	2240	2800	3360	3920	4480	5040	5600
36	7776	288	576	864	1152	1440	1728	2016	2304	2592	2880
37	8214	592	1184	1776	2368	2960	3552	4144	4736	5328	5920
38	8664	304	608	912	1216	1520	1824	2128	2432	2736	3040
39	9126	624	1248	1872	2496	3120	3744	4368	4992	5616	6240
40	9600	320	640	960	1280	1600	1920	2240	2560	2880	3200
41	10086	656	1312	1968	2624	3280	3936	4592	5248	5904	6560
42	10584	336	672	1008	1344	1680	2016	2352	2688	3024	3360
43	11094	688	1376	2064	2752	3440	4128	4816	5504	6192	6880
44	11616	352	704	1056	1408	1760	2112	2464	2816	3168	3520
45	12150	720	1440	2160	2880	3600	4320	5040	5760	6480	7200
46	12696	368	736	1104	1472	1840	2208	2576	2944	3312	3680
47	13254	752	1504	2256	3008	3760	4512	5264	6016	6768	7520
48	13824	384	768	1152	1536	1920	2304	2688	3072	3456	3840
49	14406	784	1568	2352	3136	3920	4704	5488	6272	7056	7840
50	15000	400	800	1200	1600	2000	2400	2800	3200	3600	4000
51	15606	816	1632	2448	3264	4080	4896	5712	6528	7344	8160

NPROC_XI	processors	NEX_XI									
52	16224	416	832	1248	1664	2080	2496	2912	3328	3744	4160
53	16854	848	1696	2544	3392	4240	5088	5936	6784	7632	8480
54	17496	432	864	1296	1728	2160	2592	3024	3456	3888	4320
55	18150	880	1760	2640	3520	4400	5280	6160	7040	7920	8800
56	18816	448	896	1344	1792	2240	2688	3136	3584	4032	4480
57	19494	912	1824	2736	3648	4560	5472	6384	7296	8208	9120
58	20184	464	928	1392	1856	2320	2784	3248	3712	4176	4640
59	20886	944	1888	2832	3776	4720	5664	6608	7552	8496	9440
60	21600	480	960	1440	1920	2400	2880	3360	3840	4320	4800
61	22326	976	1952	2928	3904	4880	5856	6832	7808	8784	9760
62	23064	496	992	1488	1984	2480	2976	3472	3968	4464	4960
63	23814	1008	2016	3024	4032	5040	6048	7056	8064	9072	10080
64	24576	512	1024	1536	2048	2560	3072	3584	4096	4608	5120
65	25350	1040	2080	3120	4160	5200	6240	7280	8320	9360	10400
66	26136	528	1056	1584	2112	2640	3168	3696	4224	4752	5280
67	26934	1072	2144	3216	4288	5360	6432	7504	8576	9648	10720
68	27744	544	1088	1632	2176	2720	3264	3808	4352	4896	5440
69	28566	1104	2208	3312	4416	5520	6624	7728	8832	9936	11040
70	29400	560	1120	1680	2240	2800	3360	3920	4480	5040	5600
71	30246	1136	2272	3408	4544	5680	6816	7952	9088	10224	11360
72	31104	576	1152	1728	2304	2880	3456	4032	4608	5184	5760
73	31974	1168	2336	3504	4672	5840	7008	8176	9344	10512	11680
74	32856	592	1184	1776	2368	2960	3552	4144	4736	5328	5920
75	33750	1200	2400	3600	4800	6000	7200	8400	9600	10800	12000
76	34656	608	1216	1824	2432	3040	3648	4256	4864	5472	6080
77	35574	1232	2464	3696	4928	6160	7392	8624	9856	11088	12320
78	36504	624	1248	1872	2496	3120	3744	4368	4992	5616	6240
79	37446	1264	2528	3792	5056	6320	7584	8848	10112	11376	12640
80	38400	640	1280	1920	2560	3200	3840	4480	5120	5760	6400
81	39366	1296	2592	3888	5184	6480	7776	9072	10368	11664	12960
82	40344	656	1312	1968	2624	3280	3936	4592	5248	5904	6560
83	41334	1328	2656	3984	5312	6640	7968	9296	10624	11952	13280
84	42336	672	1344	2016	2688	3360	4032	4704	5376	6048	6720
85	43350	1360	2720	4080	5440	6800	8160	9520	10880	12240	13600
86	44376	688	1376	2064	2752	3440	4128	4816	5504	6192	6880
87	45414	1392	2784	4176	5568	6960	8352	9744	11136	12528	13920
88	46464	704	1408	2112	2816	3520	4224	4928	5632	6336	7040
89	47526	1424	2848	4272	5696	7120	8544	9968	11392	12816	14240
90	48600	720	1440	2160	2880	3600	4320	5040	5760	6480	7200
91	49686	1456	2912	4368	5824	7280	8736	10192	11648	13104	14560
92	50784	736	1472	2208	2944	3680	4416	5152	5888	6624	7360
93	51894	1488	2976	4464	5952	7440	8928	10416	11904	13392	14880
94	53016	752	1504	2256	3008	3760	4512	5264	6016	6768	7520
95	54150	1520	3040	4560	6080	7600	9120	10640	12160	13680	15200
96	55296	768	1536	2304	3072	3840	4608	5376	6144	6912	7680
97	56454	1552	3104	4656	6208	7760	9312	10864	12416	13968	15520
98	57624	784	1568	2352	3136	3920	4704	5488	6272	7056	7840
99	58806	1584	3168	4752	6336	7920	9504	11088	12672	14256	15840
100	60000	800	1600	2400	3200	4000	4800	5600	6400	7200	8000
101	61206	1616	3232	4848	6464	8080	9696	11312	12928	14544	16160
102	62424	816	1632	2448	3264	4080	4896	5712	6528	7344	8160

NPROC_XI	processors	NEX_XI									
103	63654	1648	3296	4944	6592	8240	9888	11536	13184	14832	16480
104	64896	832	1664	2496	3328	4160	4992	5824	6656	7488	8320
105	66150	1680	3360	5040	6720	8400	10080	11760	13440	15120	16800
106	67416	848	1696	2544	3392	4240	5088	5936	6784	7632	8480
107	68694	1712	3424	5136	6848	8560	10272	11984	13696	15408	17120
108	69984	864	1728	2592	3456	4320	5184	6048	6912	7776	8640
109	71286	1744	3488	5232	6976	8720	10464	12208	13952	15696	17440
110	72600	880	1760	2640	3520	4400	5280	6160	7040	7920	8800
111	73926	1776	3552	5328	7104	8880	10656	12432	14208	15984	17760
112	75264	896	1792	2688	3584	4480	5376	6272	7168	8064	8960
113	76614	1808	3616	5424	7232	9040	10848	12656	14464	16272	18080
114	77976	912	1824	2736	3648	4560	5472	6384	7296	8208	9120
115	79350	1840	3680	5520	7360	9200	11040	12880	14720	16560	18400
116	80736	928	1856	2784	3712	4640	5568	6496	7424	8352	9280
117	82134	1872	3744	5616	7488	9360	11232	13104	14976	16848	18720
118	83544	944	1888	2832	3776	4720	5664	6608	7552	8496	9440
119	84966	1904	3808	5712	7616	9520	11424	13328	15232	17136	19040
120	86400	960	1920	2880	3840	4800	5760	6720	7680	8640	9600
121	87846	1936	3872	5808	7744	9680	11616	13552	15488	17424	19360
122	89304	976	1952	2928	3904	4880	5856	6832	7808	8784	9760
123	90774	1968	3936	5904	7872	9840	11808	13776	15744	17712	19680
124	92256	992	1984	2976	3968	4960	5952	6944	7936	8928	9920
125	93750	2000	4000	6000	8000	10000	12000	14000	16000	18000	20000
126	95256	1008	2016	3024	4032	5040	6048	7056	8064	9072	10080
127	96774	2032	4064	6096	8128	10160	12192	14224	16256	18288	20320
128	98304	1024	2048	3072	4096	5120	6144	7168	8192	9216	10240
129	99846	2064	4128	6192	8256	10320	12384	14448	16512	18576	20640

Table 3.2: Sample choices for NEX_XI given NPROC_XI based upon the relationship NEX_XI $= 8 \times c \times$ NPROC_XI, where the integer $c \geq 1$. The number of MPI slices, i.e., the total number of required processors, is $6 \times$ NPROC_XI^2, as illustrated in Figure 3.1. The approximate shortest period at which the global simulation is accurate for a given value of NEX_XI can be estimated by running the small serial program `xcreate_header_file`.

Finally, you need to provide a file that tells MPI what compute nodes to use for the simulations. The file must have a number of entries (one entry per line) at least equal to the number of processors needed for the run. A sample file is provided in the file `mymachines`. This file is not used by the mesher or solver, but is required by the `go_mesher` and `go_solver` default job submission scripts. See Chapter 9 for information about running the code on a system with a scheduler, e.g., LSF.

Now that you have set the appropriate parameters in the `Par_file` and have compiled the mesher, you are ready to launch it! This is most easily accomplished based upon the `go_mesher` script. When you run on a PC cluster, the script assumes that the nodes are named n001, n002, etc. If this is not the case, change the `tr -d 'n'` line in the script. You may also need to edit the last command at the end of the script that invokes the `mpirun` command.

Mesher output is provided in the `OUTPUT_FILES` directory in `output_mesher.txt`; this file provides lots of details about the mesh that was generated. Alternatively, output can be directed to the screen instead by uncommenting a line in `constants.h`:

```
! uncomment this to write messages to the screen
! integer, parameter :: IMAIN = ISTANDARD_OUTPUT
```

Note that on very fast machines, writing to the screen may slow down the code.

Another file generated by the mesher is the header file OUTPUT_FILES/values_from_mesher.h. This file specifies a number of constants and flags needed by the solver. These values are passed statically to the solver for reasons of speed. Some useful statistics about the mesh are also provided in this file.

For a given model, set of nodes, and set of parameters in Par_file, one only needs to run the mesher once and for all, even if one wants to run several simulations with different sources and/or receivers (the source and receiver information is used in the solver only).

Please note that it is difficult to correctly sample S waves in the inner core of the Earth because S-wave velocity is very small there. Therefore, correctly sampling S waves in the inner core would require a very dense mesh, which in turn would drastically reduce the time step of the explicit time scheme because the P wave velocity is very high in the inner core (Poisson's ratio is roughly equal to 0.44). Because shear wave attenuation is very high in the inner core (Q_μ is approximately equal to 85), we have therefore decided to design the inner core mesh such that P waves are very well sampled but S waves are right at the sampling limit or even slightly below. This works fine because spurious numerical oscillations due to S-wave subsampling are almost completely suppressed by attenuation. However, this implies that one should not use SPECFEM3D_GLOBE with the regular mesh and period estimates of Table 3 to study the PKJKP phase very precisely. If one is interested in that phase, one should use typically 1.5 times to twice the number of elements NEX indicated in the table.

Regarding fluid/solid coupling at the CMB and ICB, in SPECFEM3D_GLOBE we do not use the fluid-solid formulation of **?** and **?** anymore, we now use a displacement potential in the fluid (rather than a velocity potential as in **?** and **?**). This leads to the simpler fluid-solid matching condition introduced by **?** with no numerical iterations at the CMB and ICB.

For accuracy reasons, in the mesher the coordinates of the mesh points (arrays xstore, ystore and zstore) are always created in double precision. If the solver is compiled in single precision mode, the mesh coordinates are converted to single precision before being saved in the local mesh files.

3.1 Memory requirements

The SPECFEM3D_GLOBE memory requirements can be estimated before or after running the mesher using the small serial program ./bin/xcreate_header_file, which reads the input file DATA/Par_file and displays the total amount of memory that will be needed by the mesher and the solver to run it. This way, users can easily modify the parameters and check that their simulation will fit in memory on their machine. The file created by xcreate_header_file is called OUTPUT_FILES/values_from_mesher.h and contains even more details about the future simulation.

Please note that running these codes is optional because no information needed by the solver is generated.

Chapter 4

Running the Solver `xspecfem3D`

Now that you have successfully run the mesher, you are ready to compile the solver. For reasons of speed, the solver uses static memory allocation. Therefore it needs to be recompiled (type '`make clean`' and '`make specfem3D`') every time one reruns the mesher with different parameters. To compile the solver one needs a file generated by the mesher in the directory `OUTPUT_FILES` called `values_from_mesher.h`, which contains parameters describing the static size of the arrays as well as the setting of certain flags.

The solver needs three input files in the `DATA` directory to run: the `Par_file` that was discussed in detail in Chapter 3, the earthquake source parameter file `CMTSOLUTION`, and the stations file `STATIONS`. Most parameters in the `Par_file` should be set prior to running the mesher. Only the following parameters may be changed after running the mesher:

- the simulation type control parameters: `SIMULATION_TYPE` and `SAVE_FORWARD`
- the record length `RECORD_LENGTH_IN_MINUTES`
- the movie control parameters `MOVIE_SURFACE`, `MOVIE_VOLUME`, and `NTSTEPS_BETWEEN_FRAMES`
- the multi-stage simulation parameters `NUMBER_OF_RUNS` and `NUMBER_OF_THIS_RUN`
- the output information parameters `NTSTEP_BETWEEN_OUTPUT_INFO`, `NTSTEP_BETWEEN_OUTPUT_SEISMOS`, `OUTPUT_SEISMOS_ASCII_TEXT`, `OUTPUT_SEISMOS_SAC_ALPHANUM`, `OUTPUT_SEISMOS_SAC_BINARY` and `ROTATE_SEISMOGRAMS_RT`
- the `RECEIVERS_CAN_BE_BURIED` and `PRINT_SOURCE_TIME_FUNCTION` flags

Any other change to the `Par_file` implies rerunning both the mesher and the solver.

For any particular earthquake, the `CMTSOLUTION` file that represents the point source may be obtained directly from the Harvard Centroid-Moment Tensor (CMT) web page (`www.globalcmt.org`). It looks like this:

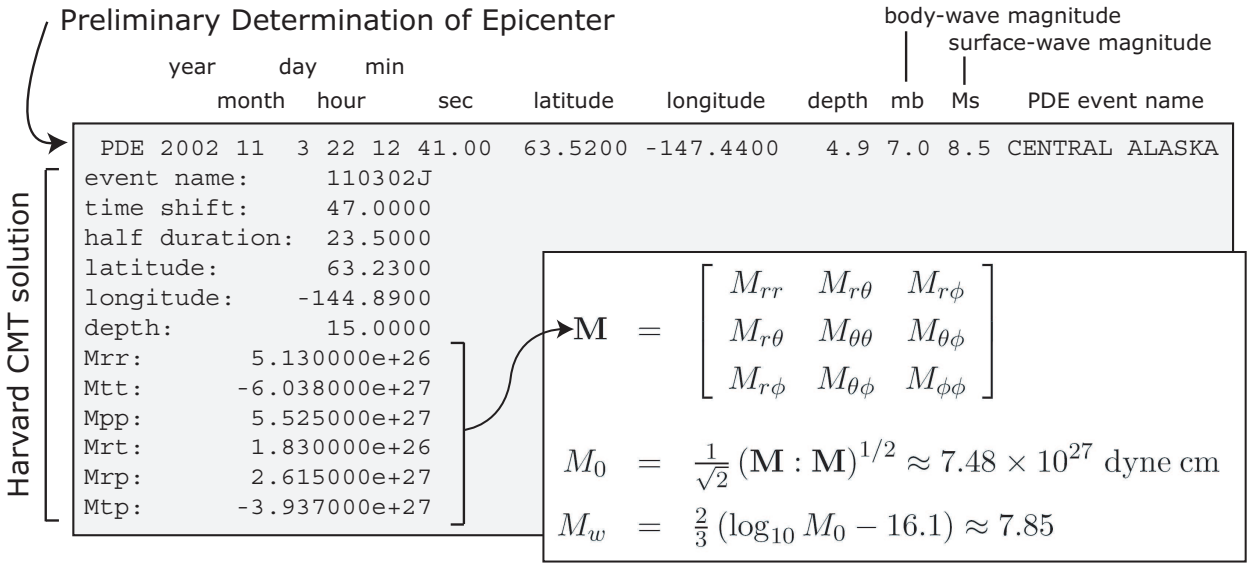


Figure 4.1: CMTSOLUTION file obtained from the Harvard CMT catalog. The top line is the initial estimate of the source, which is used as a starting point for the CMT solution. **M** is the moment tensor, M_0 is the seismic moment, and M_w is the moment magnitude.

The CMTSOLUTION should be edited in the following way:

- For point-source simulations (see finite sources, page 22) we recommend setting the source half-duration parameter half duration equal to zero, which corresponds to simulating a step source-time function, i.e., a moment-rate function that is a delta function. If half duration is not set to zero, the code will use a Gaussian (i.e., a signal with a shape similar to a 'smoothed triangle', as explained in **?** and shown in Fig 4.2) source-time function with half-width half duration. We prefer to run the solver with half duration set to zero and convolve the resulting synthetic seismograms in post-processing after the run, because this way it is easy to use a variety of source-time functions (see Section 11.2). **?** determined that the noise generated in the simulation by using a step source time function may be safely filtered out afterward based upon a convolution with the desired source time function and/or low-pass filtering. Use the postprocessing script process_syn.pl (see Section 11.2.2) with the -h flag, or the serial code convolve_source_timefunction.f90 and the script UTILS/convolve_source_timefunction.csh for this purpose, or alternatively use signal-processing software packages such as SAC (www.iris.edu/software/sac/). Type

  ```
  make convolve_source_timefunction
  ```

  to compile the code and then set the parameter hdur in UTILS/convolve_source_timefunction.csh to the desired half-duration.

- The zero time of the simulation corresponds to the center of the triangle/Gaussian, or the centroid time of the earthquake. The start time of the simulation is $t = -1.5 * \texttt{half duration}$ (the 1.5 is to make sure the moment rate function is very close to zero when starting the simulation). To convert to absolute time $t_{\rm abs}$, set

  $$t_{\rm abs} = t_{\rm pde} + \texttt{time shift} + t_{\rm synthetic}$$

  where $t_{\rm pde}$ is the time given in the first line of the CMTSOLUTION, time shift is the corresponding value from the original CMTSOLUTION file and $t_{\rm synthetic}$ is the time in the first column of the output seismogram.

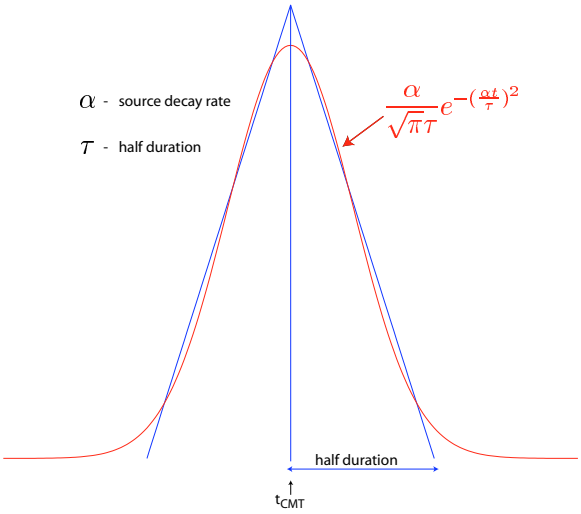


Figure 4.2: Comparison of the shape of a triangle and the Gaussian function actually used.

If you know the earthquake source in strike/dip/rake format rather than in `CMTSOLUTION` format, use the C code `utils/strike_dip_rake_to_CMTSOLUTION.c` to convert it. The conversion formulas are given for instance in **?**. Note that the **?** convention is slightly different from the Harvard `CMTSOLUTION` convention (the sign of some components is different). The C code outputs both.

Centroid latitude and longitude should be provided in geographical coordinates. The code converts these coordinates to geocentric coordinates [**?**]. Of course you may provide your own source representations by designing your own `CMTSOLUTION` file. Just make sure that the resulting file adheres to the Harvard CMT conventions (see Appendix A). Note that the first line in the `CMTSOLUTION` file is the Preliminary Determination of Earthquakes (PDE) solution performed by the USGS NEIC, which is used as a seed for the Harvard CMT inversion. The PDE solution is based upon P waves and often gives the hypocenter of the earthquake, i.e., the rupture initiation point, whereas the CMT solution gives the 'centroid location', which is the location with dominant moment release. The PDE solution is not used by our software package but must be present anyway in the first line of the file.

In the current version of the code, the solver can run with a non-zero `time shift` in the `CMTSOLUTION` file. Thus one does not need to set `time shift` to zero as it was the case for previous versions of the code. `time shift` is only used for writing centroid time in the SAC headers (see Appendix D). CMT time is obtained by adding `time shift` to the PDE time given in the first line in the `CMTSOLUTION` file. Therefore it is recommended not to modify `time shift` to have the correct timing information in the SAC headers without any post-processing of seismograms.

To simulate a kinematic rupture, i.e., a finite-source event, represented in terms of N_{sources} point sources, provide a `CMTSOLUTION` file that has N_{sources} entries, one for each subevent (i.e., concatenate N_{sources} `CMTSOLUTION` files to a single `CMTSOLUTION` file). At least one entry (not necessarily the first) must have a zero `time shift`, and all the other entries must have non-negative `time shift`. If none of the entries has a zero `time shift` in the `CMTSOLUTION` file, the smallest `time shift` is subtracted from all sources to initiate the simulation. Each subevent can have its own half duration, latitude, longitude, depth, and moment tensor (effectively, the local moment-density tensor).

Note that the zero in the synthetics does NOT represent the hypocentral time or centroid time in general, but the timing of the *center* of the source triangle with zero `time shift` (Fig 4.3).

Although it is convenient to think of each source as a triangle, in the simulation they are actually Gaussians (as they have better frequency characteristics). The relationship between the triangle and the Gaussian used is shown in Fig 4.2. For finite fault simulations it is usually not advisable to use a zero half duration and convolve afterwards, since the half duration is generally fixed by the finite fault model.

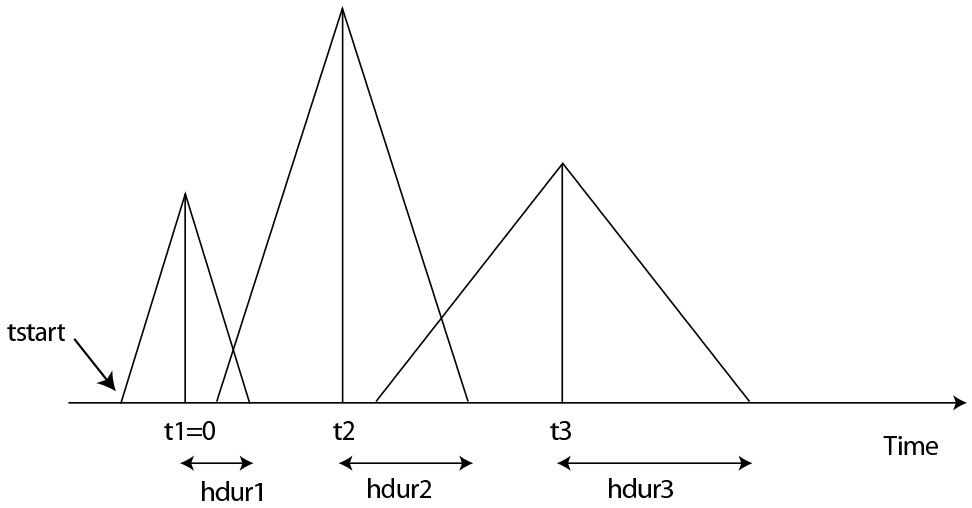


Figure 4.3: Example of timing for three sources. The center of the first source triangle is defined to be time zero. Note that this is NOT in general the hypocentral time, or the start time of the source (marked as tstart). The parameter `time shift` in the `CMTSOLUTION` file would be t1(=0), t2, t3 in this case, and the parameter `half duration` would be hdur1, hdur2, hdur3 for the sources 1, 2, 3 respectively.

The solver can calculate seismograms at any number of stations for basically the same numerical cost, so the user is encouraged to include as many stations as conceivably useful in the `STATIONS` file, which looks like this:

Station	Network	Latitude (deg)	Longitude (deg)	Elevation (m)	Burial (m)
SFJD	IU	66.9960	-50.6215	328.0	0.0
SAML	IU	-8.9488	-63.1832	130.0	0.0
BBSR	IU	32.3712	-64.6962	6.0	0.0
RAO	IU	-29.2517	-177.9183	110.0	0.0
LAST	GE	35.1610	25.4790	870.0	0.0
MCK	AK	63.7323	-148.9349	618.0	0.0
CTAO	AS	-20.0882	146.2545	357.0	0.0
KONO	AS	59.6491	9.5982	216.0	0.0
MAJO	AS	36.5409	138.2083	431.0	0.0
⋮	⋮	⋮	⋮	⋮	⋮

Figure 4.4: Sample `STATIONS` file. Station latitude and longitude should be provided in geographical coordinates. The width of the station label should be no more than 32 characters (see `MAX_LENGTH_STATION_NAME` in the `constants.h` file), and the network label should be no more than 8 characters (see `MAX_LENGTH_NETWORK_NAME` in the `constants.h` file).

Each line represents one station in the following format:

```
Station Network Latitude (degrees) Longitude (degrees) Elevation (m) burial (m)
```

If you want to put a station on the ocean floor, just set elevation and burial depth in the STATIONS file to 0. Equivalently you can also set elevation to a negative value equal to the ocean depth, and burial depth to 0.

Solver output is provided in the `OUTPUT_FILES` directory in the `output_solver.txt` file. Output can be directed to the screen instead by uncommenting a line in `constants.h`:

```
! uncomment this to write messages to the screen
! integer, parameter :: IMAIN = ISTANDARD_OUTPUT
```

Note that on very fast machines, writing to the screen may slow down the code.

While the solver is running, its progress may be tracked by monitoring the '`timestamp*`' files in the `OUTPUT_FILES` directory. These tiny files look something like this:

```
Time step #           200
Time:    0.6956667      minutes
Elapsed time in seconds =     252.6748970000000
Elapsed time in hh:mm:ss =    0 h 04 m 12 s
Mean elapsed time per time step in seconds =     1.263374485000000
Max norm displacement vector U in solid in all slices (m) =    1.9325
Max non-dimensional potential Ufluid in fluid in all slices = 1.1058885E-22
```

The `timestamp*` files provide the `Mean elapsed time per time step in seconds`, which may be used to assess performance on various machines (assuming you are the only user on a node), as well as the `Max norm displacement vector U in solid in all slices (m)` and `Max non-dimensional potential Ufluid in fluid in all slices`. If something is wrong with the model, the mesh, or the source, you will see the code become unstable through exponentionally growing values of the displacement and/or fluid potential with time, and ultimately the run will be terminated by the program when either of these values becomes greater than `STABILITY_THRESHOLD` defined in `constants.h`. You can control the rate at which the timestamp files are written based upon the parameter `NTSTEP_BETWEEN_OUTPUT_INFO` in the `Par_file`.

Having set the `Par_file` parameters, and having provided the `CMTSOLUTION` and `STATIONS` files, you are now ready to launch the solver! This is most easily accomplished based upon the `go_solver` script (see Chapter 9 for information about running the code through a scheduler, e.g., LSF). You may need to edit the last command at the end of the script that invokes the `mpirun` command. Another option is to use the `runall` script, which compiles and runs both mesher and solver in sequence. This is a safe approach that ensures using the correct combination of mesher output and solver input.

It is important to realize that the CPU and memory requirements of the solver are closely tied to choices about attenuation (`ATTENUATION`) and the nature of the model (i.e., isotropic models are cheaper than anisotropic models). We encourage you to run a variety of simulations with various flags turned on or off to develop a sense for what is involved.

For the same model, one can rerun the solver for different events by simply changing the `CMTSOLUTION` file, and/or for different stations by changing the `STATIONS` file. There is no need to rerun the mesher. Of course it is best to include as many stations as possible, since this does not add significantly to the cost of the simulation.

Chapter 5

Regional Simulations

The code has the option of running in 1-, 2-, 3- or 6-chunk mode. The 1- or 2-chunk options may be used for higher resolution regional simulations. A one-chunk mesh may have lateral dimensions other than the customary 90° per chunk, which can further increase the resolution of the mesh, and thus reduce the shortest period in the synthetic seismograms (but of course then also reducing the time step in order for the simulation to remain stable). A disadvantage of regional simulations is that one needs to use approximate absorbing boundary conditions on the side and bottom edges of the model (e.g., see **?** for a description of the paraxial boundary conditions used). Figure 5.1 on the following page and Figure 5.2 on page 27 show an example of a one-chunk mesh centered on the Japan subduction zone, applied in the Japan regional waveform simulation [**?**].

5.1 One-Chunk Simulations

For a one-chunk regional simulation the following parameters need to be set in the `Par_file`:

`NCHUNKS` Must be set to 1.

`ANGULAR_WIDTH_XI_IN_DEGREES` Denotes the width of one side of the chunk (90°or less).

`ANGULAR_WIDTH_ETA_IN_DEGREES` Denotes the width of the second side of the chunk (90°or less). Note that this value may be different from `ANGULAR_WIDTH_XI_IN_DEGREES`.

`CENTER_LATITUDE_IN_DEGREES` Defines the latitude of the center of the chunk (degrees).

`CENTER_LONGITUDE_IN_DEGREES` Defines the longitude of the center of the chunk (degrees).

`GAMMA_ROTATION_AZIMUTH` Defines the rotation angle of the chunk about its center measured counter clockwise from due North (degrees). The corners of the mesh are output in `OUTPUT_FILES/values_from_mesher.h`. The output corner progression in `OUTPUT_FILES/values_from_mesher.h` is bottom left, bottom right, top left, top right. The rotation azimuth can be changed in the `Par_file` and the corners output (`OUTPUT_FILES/values_from_mesher.h`) by using `xcreate_header_file`. It is important to note that the mesher or the solver does not need to be run to determine the limits of a 1-chunk simulation.

`NEX_XI` The number of spectral elements along the ξ side of the chunk. This number *must* be 8 $\times$ a multiple of `NPROC_XI` defined below. For a 90° chunk, we do not recommend using `NEX_XI` less than 64 because the curvature of the Earth cannot be honored if one uses too few elements, which results in inaccurate and unstable simulations.

`NEX_ETA` The number of spectral elements along the η side of the chunk. This number *must* be 8 $\times$ a multiple of `NPROC_ETA` defined below. Note that in order to get elements that are close to square on the Earth's surface, the following ratios should be similar:

```
ANGULAR_WIDTH_XI_IN_DEGREES/NEX_XI
ANGULAR_WIDTH_ETA_IN_DEGREES/NEX_ETA
```

Because of the geometry of the cubed sphere, the option of having different values for NEX_XI and NEX_ETA is available only for regional simulations when NCHUNKS $= 1$ (1/6th of the sphere).

NPROC_XI The number of processors or mesh slices along the ξ side of the chunk. To accommodate the mesh doubling layers, we must have NEX_XI $= 8 \times c \times$ NPROC_XI, where $c \geq 1$ is a positive integer. See Table 3 for various suitable choices.

NPROC_ETA The number of processors or slices along the η side of the chunk; we must have NEX_ETA $= 8 \times c \times$ NPROC_ETA, where $c \geq 1$ is a positive integer. NPROC_XI and NPROC_ETA must be equal when NCHUNKS $=$ 6.

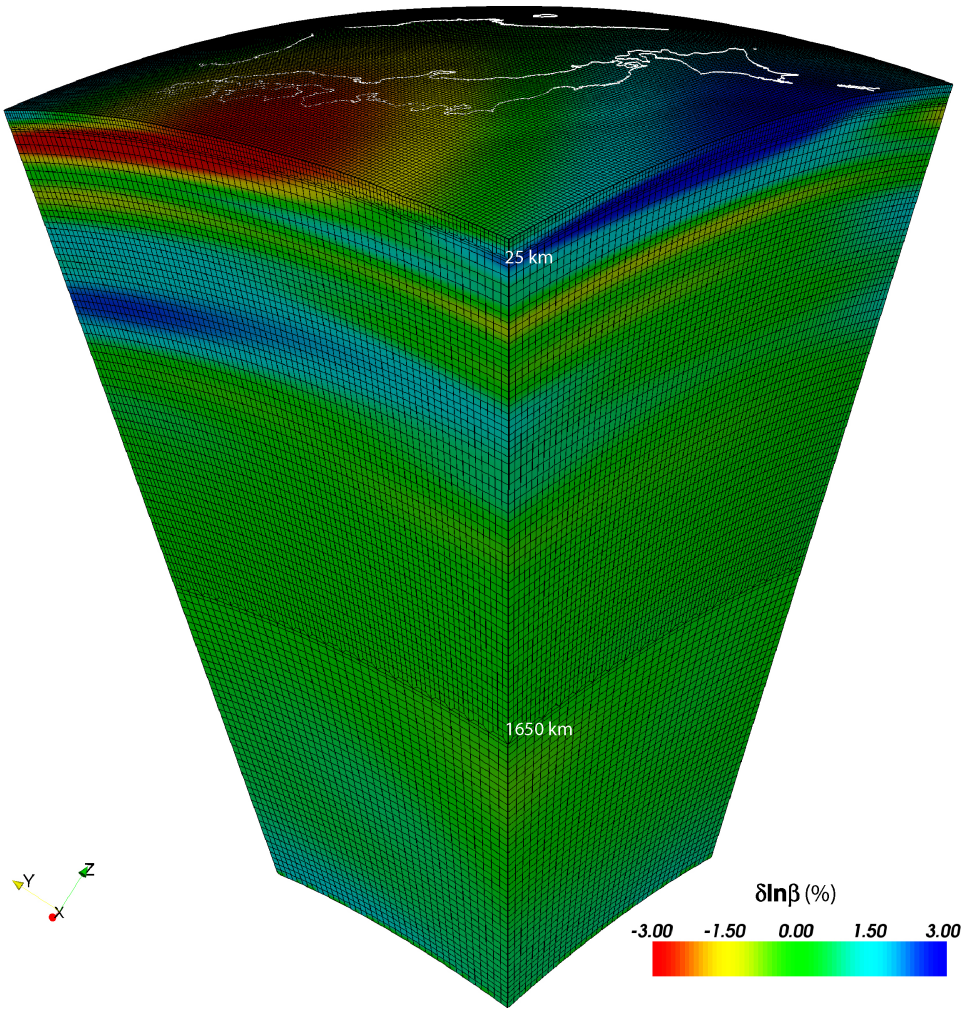


Figure 5.1: S-wave velocity anomalies from the global tomographic model s20rts [**?**] are superimposed on the mesh. For parallel computing purposes, the one-chunk SEM simulation is subdivided in terms of 64 slices. The center of the chunk is at (38.5° N, 137.5° E), and the lateral dimensions are 30° × 30°. Two doubling layers are indicated at a depth of 25 km (PREM Moho depth) and a depth of about 1650 km. Shows full view of 25 neighboring slices; see Figure 5.2 for close-up of upper mantle mesh.

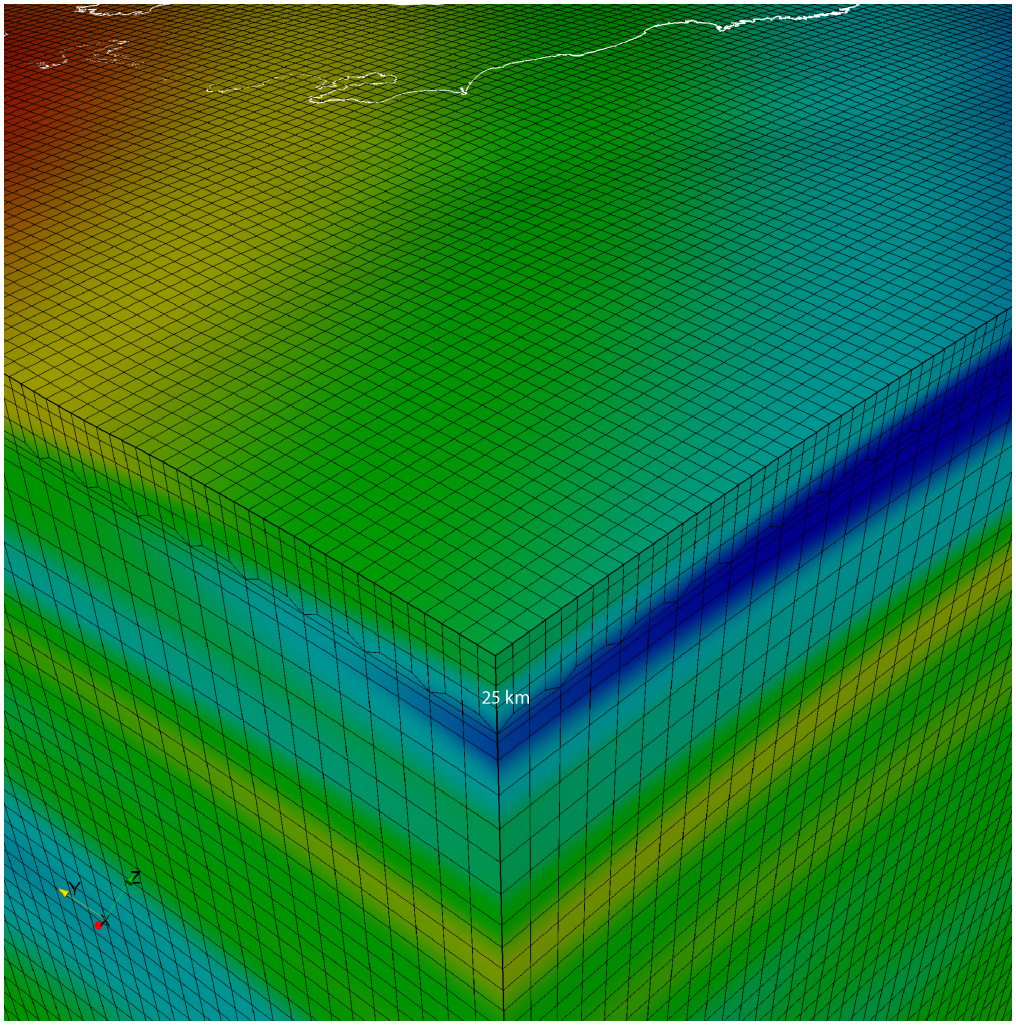


Figure 5.2: Close-up view of the upper mantle mesh shown in Figure 5.1. Note that the element size in the crust (top layer) is 13 km $\times$ 13 km, and that the size of the spectral elements is doubled in the upper mantle. The velocity variation is captured by NGLL = 5 grid points in each direction of the elements [**??**].

ABSORBING_CONDITIONS Set to `.true.` for regional simulations. For instance, see **?** for a description of the paraxial boundary conditions used. Note that these conditions are never perfect, and in particular surface waves may partially reflect off the artificial boundaries. Note also that certain arrivals, e.g., PKIKPPKIKP, will be missing from the synthetics.

When the width of the chunk is different from 90° (or the number of elements is greater than 1248), the radial distribution of elements needs to be adjusted as well to maintain spectral elements that are as cube-like as possible. The code attempts to do this, but be sure to view the mesh with your favorite graphics package to make sure that the element are well behaved. Remember: a high-quality mesh is paramount for accurate simulations. In addition to a reorganization of the radial distribution of elements, the time stepping and period range in which the attenuation is applied is automatically determined. The minimum and maximum periods for attenuation are:

$$\omega_{max} = \omega_{min} \times 10^{W_3}$$

where W_3 is the optimal width in frequency for 3 Standard Linear Solids, about 1.75. See `read_compute_parameters.f90` for more details.

The time stepping is determined in a similar fashion as Equation (48) in **?**:

`dt =` S_c `Element Width in km (`r`=ICB) / Velocity (`r`=ICB)`

where S_c is the stability condition (about 0.4). We use the radius at the inner core boundary because this is where the maximum velocity/element width occurs. Again, see `read_compute_parameters.f90` for all the details.

The approximate shortest period at which a regional simulation is accurate may be determined based upon the relation

$$\text{shortest period (s)} \simeq (256/\texttt{NEX\_XI}) \times (\texttt{ANGULAR\_WIDTH\_XI\_IN\_DEGREES}/90) \times 17. \tag{5.1}$$

5.2 Two-Chunk Simulations

For a two-chunk regional simulation the following parameters need to be set in the `Par_file`:

`NCHUNKS` Must be set to 2

`ANGULAR_WIDTH_XI_IN_DEGREES` Denotes the width of one side of the chunk, and it has to be 90 degrees.

`ANGULAR_WIDTH_ETA_IN_DEGREES` Denotes the width of the second side of the chunk, and it also has to be 90 degrees.

`NEX_XI` and `NEX_ETA` follow the same description in Section 5.1, however, they need to be the same in this case. All other parameters are similar to the one-chunk simulations, refer to Section 5.1 for details.

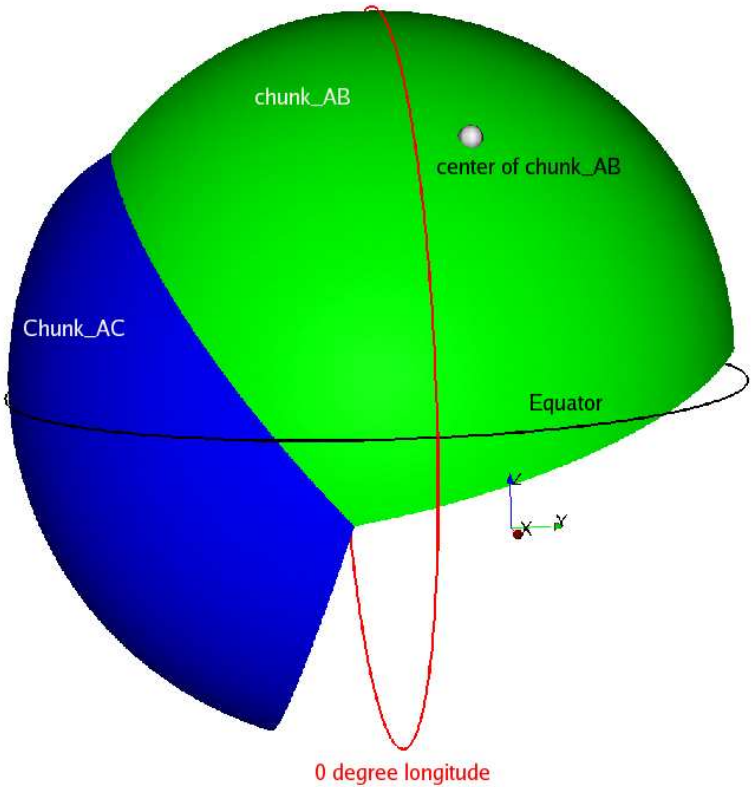


Figure 5.3: Geometry of a 2-chunk simulation, where the first chunk (`CHUNK_AB`) centers at 40° latitude, 10° longitude, and has been rotated by 20° counter clockwise, and the second chunk (`CHUNK_AC`) connects to the first chunk through one face.

Chapter 6

Adjoint Simulations

Adjoint simulations are generally performed for two distinct applications. First, they can be used for earthquake source inversions, especially earthquakes with large ruptures such as the Sumatra-Andaman event [**???**]. Second, they can be used to generate finite-frequency sensitivity kernels that are a critical part of tomographic inversions based upon 3D reference models [**????**]. In either case, source parameter or velocity structure updates are sought to minimize a specific misfit function (e.g., waveform or traveltime differences), and the adjoint simulation provides a means of computing the gradient of the misfit function and further reducing it in successive iterations. Applications and procedures pertaining to source studies and finite-frequency kernels are discussed in Sections 6.1 and 6.2, respectively. The two related parameters in the `Par_file` are `SIMULATION_TYPE` (1, 2 or 3) and `SAVE_FORWARD` (boolean).

6.1 Adjoint Simulations for Sources Only (not for the Model)

In the case where a specific misfit function is minimized to invert for the earthquake source parameters, the gradient of the misfit function with respect to these source parameters can be computed by placing time-reversed seismograms at the receivers and using them as sources in an adjoint simulation, and then the value of the gradient is obtained from the adjoint seismograms recorded at the original earthquake location.

1. **Prepare the adjoint sources**

   (a) First, run a regular forward simlation (`SIMULATION_TYPE = 1` and `SAVE_FORWARD = .false.`). You can automatically set these two variables using the `UTILS/change_simulation_type.pl` script:

   ```
   UTILS/change_simulation_type.pl -f
   ```

   and then collect the recorded seismograms at all the stations given in `DATA/STATIONS`.

   (b) Then select the stations for which you want to compute the time-reversed adjoint sources and run the adjoint simulation, and compile them into the `DATA/STATIONS_ADJOINT` file, which has the same format as the regular `DATA/STATIONS` file.

   - Depending on what type of misfit function is used for the source inversion, adjoint sources need to be computed from the original recorded seismograms for the selected stations and saved in the `SEM/` directory with the format `STA.NT.?X?.adj`, where `STA`, `NT` are the station name and network code given in the `DATA/STATIONS_ADJOINT` file, and `?X?` represents the channel name of a particular adjoint seismogram where the first letter corresponds to the band code governed by the resolution of simulations, for example, generally `MX?` for the current resolution of global simulations (see Appendix E for details). The last letter of channel names is the component name `E/N/Z`.
   - The adjoint seismograms are in the same format as the original seismogram (`STA.NT.?X?.sem?`), with the same start time, time interval and record length.

   (c) Notice that even if you choose to time reverse only one component from one specific station, you still need to supply all three components because the code is expecting them (you can set the other two components to be zero).

(d) Also note that since time-reversal is done in the code itself, no explicit time-reversing is needed for the preparation of the adjoint sources, i.e., the adjoint sources are in the same forward time sense as the original recorded seismograms.

2. **Set the related parameters and run the adjoint simulation**
In the `DATA/Par_file`, set the two related parameters to be `SIMULATION_TYPE = 2` and `SAVE_FORWARD = .false.`. More conveniently, use the scripts `UTILS/change_simulation_type.pl` to modify the `Par_file` automatically (`change_simulation_type.pl -a`). Then run the solver to launch the adjoint simulation.

3. **Collect the seismograms at the original source location**

After the adjoint simulation has completed successfully, get the seismograms from directory `OUTPUT_FILES`.

- These adjoint seismograms are recorded at the locations of the original earthquake sources given by the `DATA/CMTSOLUTION` file, and have names of the form `S?????.NT.S??.sem` for the six-component strain tensor (`SNN, SEE, SZZ, SNE, SNZ, SEZ`) at these locations, and `S?????.NT.?X?.sem` for the three-component displacements (i.e., `MXN, MXE, MXZ`) recorded at these locations.
- `S?????` denotes the source number; for example, if the original `CMTSOLUTION` provides only a point source, then the seismograms collected will start with `S00001`.
- These adjoint seismograms provide critical information for the computation of the gradient of the misfit function.

6.2 Adjoint Simulations for Finite-Frequency Kernels (Kernel Simulation)

Finite-frequency sensitivity kernels are computed in two successive simulations (please refer to **?** and **?** for details).

1. **Run a forward simulation with the state variables saved at the end of the simulation**

Prepare the `CMTSOLUTION` and `STATIONS` files, set the parameters `SIMULATION_TYPE = 1` and `SAVE_FORWARD = .true.` in the `Par_file` (`change_simulation_type -F`), and run the solver.

- Notice that attenuation is not fully implemented yet for the computation of finite-frequency kernels; if `ATTENUATION = .true.` is set in the `Par_file`, only effects on phase shift are accounted for but not on amplitude of the signals. However, we suggest you use the same setting for `ATTENUATION` as for your forward simulations.
- We also suggest you modify the half duration of the `CMTSOLUTION` to be similar to the accuracy of the simulation (see Equation 3.1 or 5.1) to avoid too much high-frequency noise in the forward wavefield, although theoretically the high-frequency noise should be eliminated when convolved with an adjoint wavefield with the proper frequency content.
- This forward simulation differs from the regular simulations (`SIMULATION_TYPE = 1` and `SAVE_FORWARD = .false.`) described in the previous chapters in that the state variables for the last time step of the simulation, including wavefields of the displacement, velocity, acceleration, etc., are saved to the `LOCAL_PATH` to be used for the subsequent simulation.
- For regional simulations, the files recording the absorbing boundary contribution are also written to the `LOCAL_PATH` when `SAVE_FORWARD = .true.`.

2. **Prepare the adjoint sources**

The adjoint sources need to be prepared the same way as described in Section 6.1, item 1.

- In the case of traveltime finite-frequency kernel for one source-receiver pair, i.e., point source from the `CMTSOLUTION`, and one station in the `STATIONS_ADJOINT` list, we supply a sample program in `UTILS/adjoint_sources/traveltime` to cut a certain portion of the original displacement seismograms and convert them into the proper adjoint source to compute the finite-frequency kernel.

```
xcreate_adjsrc_traveltime t1 t2 ifile[0-5] E/N/Z-ascii-files [baz]
```

where t1 and t2 are the start and end time of the portion you are interested in, ifile denotes the component of the seismograms to be used (0 for all three components, 1 for east, 2 for north, and 3 for vertical, 4 for transverse, and 5 for radial component), E/N/Z-ascii-files indicate the three-component displacement seismograms in the right order, and baz is the back-azimuth of the station from the event location. Note that baz is only supplied when ifile = 4 or 5.

- Similarly, a sample program to compute adjoint sources for amplitude finite-frequency kernels may be found in UTILS/adjoint_sources/amplitude and used in the same way as described for traveltime measurements

```
xcreate_adjsrc_amplitude t1 t2 ifile[0-5] E/N/Z-ascii-files [baz].
```

3. **Run the kernel simulation**

   With the successful forward simulation and the adjoint source ready in SEM/, set SIMULATION_TYPE = 3 and SAVE_FORWARD = .false. in the Par_file(change_simulation_type.pl -b), and rerun the solver.

   - The adjoint simulation is launched together with the back reconstruction of the original forward wavefield from the state variables saved from the previous forward simulation, and the finite-frequency kernels are computed by the interaction of the reconstructed forward wavefield and the adjoint wavefield.
   - The back-reconstructed seismograms at the original station locations are saved to the OUTPUT_FILES directory at the end of the kernel simulations.
   - These back-constructed seismograms can be compared with the time-reversed original seismograms to assess the accuracy of the backward reconstruction, and they should match very well (in the time-reversed sense).
   - The files containing the density, P-wave speed and S-wave speed kernels are saved in the LOCAL_PATH with the names of proc??????_reg_?_rho(alpha,beta)_kernel.bin, where proc?????? represents the processor number, and reg_? denotes the region these kernels are for, including mantle (reg_1), outer core (reg_2), and inner core (reg_3). The output kernels are in the unit of s/km^3.
   - Note that if you set the flag APPROXIMATE_HESS_KL = .true. in the constants.h file and re-compile the solver, the adjoint simulation also saves files proc??????_reg_1_hess_kernel.bin which can be used as preconditioners in the crust-mantle region for iterative inverse optimization schemes.

4. **Run the boundary kernel simulation**

   If you set the SAVE_BOUNDARY_MESH = .true. in the constants.h file before the simulations, i.e., at the beginning of step 1, you will get not only the volumetric kernels as described in step 3, but also boundary kernels for the Earth's internal discontinuities, such as Moho, 410-km discontinuity, 670-km discontinuity, CMB and ICB. These kernel files are also saved in the local scratch directory defined by LOCAL_PATH and have names such as proc??????_reg_1(2)_Moho(d400,d670,CMB,ICB)_kernel.bin. For a theoretical derivation of the boundary kernels, refer to **?**, and for the visualization of the boundary kernels, refer to Section 8.3.

5. **Run the anisotropic kernel simulation**

   Instead of the kernels for the isotropic wave speeds, you can also compute the kernels for the 21 independent components C_{IJ}, $I, J = 1, ..., 6$ (using Voigt's notation) of the elastic tensor in the (spherical) geographical coordinate system. This is done by setting ANISOTROPIC_KL = .true. in constants.h before step 3. The definition of the parameters C_{IJ} in terms of the corresponding components $c_{ijkl}, ijkl, i, j, k, l = 1, 2, 3$ of the elastic tensor in spherical coordinates follows **?**. The computation of the anisotropic kernels is only implemented in the crust and mantle regions. The 21 anisotropic kernels are saved in the LOCAL_PATH in one file with the name of proc??????_reg1_cijkl_kernel.bin (with proc?????? the processor number). The output kernels correspond to perturbation δC_{IJ} of the elastic parameters and their unit is in

$s/GPa/km^3$. For consistency, the output density kernels with this option turned on are for a perturbation $\delta\rho$ (and not $\frac{\delta\rho}{\rho}$) and their unit is in s / (kg/m^3) / km^3. These 'primary' anisotropic kernels can then be combined to obtain the kernels related to other descriptions of anisotropy. This can be done, for example, when combining the kernel files from slices into one mesh file (see Section 8.3).

In general, the first three steps need to be run sequentially to ensure proper access to the necessary files at different stages. If the simulations are run through some cluster scheduling system (e.g., LSF), and the forward simulation and the subsequent kernel simulations cannot be assigned to the same set of computer nodes, the kernel simulation will not be able to access the database files saved by the forward simulation. Solutions for this problem are provided in Chapter 9. Visualization of the finite-frequency kernels is discussed in Section 8.3.

Chapter 7

Noise Cross-correlation Simulations

The new version of SPECFEM3D_GLOBE includes functionality for seismic noise tomography. Users are recommended to familiarize themselves first with the procedures for running regular earthquake simulations (Chapters 3–5) and adjoint simulations (Chapter 6). Also, make sure you read the paper 'Noise cross-correlation sensitivity kernels' [**?**] in order to understand noise simulations from a theoretical perspective.

7.1 New Requirements on 'Old' Input Parameter Files

As usual, the three main input files are crucial: `DATA/Par_file`, `DATA/CMTSOLUTION` and `DATA/STATIONS`.

`DATA/CMTSOLUTION` is required for all simulations. However, it may seem unexpected to have it listed here, since the noise simulations should have nothing to do with the earthquake – hence the `DATA/CMTSOLUTION`. For noise simulations, it is critical to have no earthquakes. In other words, the moment tensor specified in `DATA/CMTSOLUTION` must be set to ZERO!

`DATA/STATIONS` remains the same as in previous earthquake simulations, except that the order of stations listed in `DATA/STATIONS` is now important. The order will be used later to identify the 'master' receiver, i.e., the one that simultaneously cross correlates with the others. Please be noted that the actual station file used in the simulation is `OUTPUT_FILES/STATIONS_FILTERED`, which is generated when you run your simulations. (e.g., in regional simulations we may have included stations out of the region of our interests in `DATA/STATIONS`, so we have to get rid of them.)

`DATA/Par_file` also requires careful attention. New to this version of SPECFEM3D_GLOBE, a parameter called `NOISE_TOMOGRAPHY` has been added that specifies the type of simulation to be run. `NOISE_TOMOGRAPHY` is an integer with possible values 0, 1, 2 and 3. For example, when `NOISE_TOMOGRAPHY` equals 0, a regular earthquake simulation will be run. When `NOISE_TOMOGRAPHY` is equal to 1/2/3, you are about to run step 1/2/3 of the noise simulations respectively. Should you be confused by the three steps, refer to **?** for details.

Another change to `DATA/Par_file` involves the parameter `RECORD_LENGTH_IN_MINUTES`. While for regular earthquake simulations this parameter specifies the length of synthetic seismograms generated, for noise simulations it specifies the length of the seismograms used to compute cross correlations. The actual cross correlations are thus twice this length. The code automatically makes modification accordingly, if `NOISE_TOMOGRAPHY` is not zero.

There are other parameters in `DATA/Par_file` which should be given specific values. For instance, `NUMBER_OF_RUNS` and `NUMBER_OF_THIS_RUN` must be 1; `ROTATE_SEISMOGRAMS_RT`, `SAVE_ALL_SEISMOGRAMS_IN_ONE_FILES`, `USE_BINARY_FOR_LARGE_FILE` and `MOVIE_COARSE` should be `.false.`. Moreover, since the first two steps for calculating noise cross-correlation kernels correspond to forward simulations, `SIMULATION_TYPE` must be 1 when `NOISE_TOMOGRAPHY` equals 1 or 2. Also, we have to reconstruct the ensemble forward wavefields in adjoint simulations, therefore we need to set `SAVE_FORWARD` to `.true.` for the second step, i.e., when `NOISE_TOMOGRAPHY`

equals 2. The third step is for kernel constructions. Hence SIMULATION_TYPE should be 3, whereas SAVE_FORWARD must be .false..

7.2 Noise Simulations: Step by Step

Proper parameters in those 'old' input files are not enough for noise simulations to run. We have a lot more 'new' input parameter files to specify: for example, the ensemble-averaged noise spectrum, the noise distribution etc. However, since there are a few 'new' files, it is better to introduce them sequentially. Read through this section even if you don't understand some parts temporarily, since some examples you can go through are provided in this package.

7.2.1 Pre-simulation

- As usual, we first configure the software package using:

  ```
  ./configure FC=ifort MPIFC=mpif90
  ```

- Next, we need to compile the source code using:

  ```
  make xmeshfem3D
  make xspecfem3D
  ```

  Before compilation, the DATA/Par_file must be specified correctly, e.g., NOISE_TOMOGRAPHY shouldn't be zero; RECORD_LENGTH_IN_MINUTES, NEX_XI and NEX_ETA must be what you want in your real simulations. Otherwise you may get wrong informations which will cause problems later. (it is good to always re-complie the code before you run simulations)

- After compiling, you will find two important numbers besides the needed executables:

  ```
  number of time steps = 31599
  time_stepping of the solver will be:  0.19000
  ```

  The first number will be denoted as NSTEP from now on, and the second one as dt. The two numbers are essential to calculate the ensemble-averaged noise spectrum from either Peterson's noise model or just a simple flat power spectrum (corresponding to 1-bit preprocessing). Should you miss the two numbers, you can run ./xcreate_header_file to bring them up again (with correct DATA/Par_file!). FYI, NSTEP is determined by RECORD_LENGTH_IN_MINUTES in DATA/Par_file, which is automatically doubled in noise simulations; whereas dt is derived from NEX_XI and NEX_ETA, or in other words your element sizes.

- A Matlab script is provided to generate the ensemble-averaged noise spectrum.

  EXAMPLES/noise_examples/NOISE_TOMOGRAPHY.m (main program)
  EXAMPLES/noise_examples/PetersonNoiseModel.m

  In Matlab, simply run:

  ```
  NOISE_TOMOGRAPHY(NSTEP, dt, Tmin, Tmax, NOISE_MODEL)
  ```

  NSTEP and dt have been given when compiling the specfem3D source code; Tmin and Tmax correspond to the period range you are interested in; NOISE_MODEL denotes the noise model you will be using. Details can be found in the Matlab script.

  After running the Matlab script, you will be given the following information (plus a figure in Matlab):

  ```
  *******************************************************
  the source time function has been saved in:
  ```

```
/data2/yangl/3D_NOISE/S_squared (note this path must be different)
S_squared should be put into directory:
./NOISE_TOMOGRAPHY/ in the SPECFEM3D_GLOBE package
```

In other words, the Matlab script creates a file called `S_squared`, which is the first 'new' input file we encounter for noise simulations.

One may choose a flat noise spectrum rather than Peterson's noise model. This can be done easily by modifying the Matlab script a little bit.

- Create a new directory in the SPECFEM3D_GLOBE package, name it as `NOISE_TOMOGRAPHY`. In fact, this new directory should have been created already when checking out the package. We will add/replace some information needed in this folder.
- Put the Matlab-generated-file `S_squared` in `NOISE_TOMOGRAPHY`.

  That's to say, you will have a file `NOISE_TOMOGRAPHY/S_squared` in the SPECFEM3D_GLOBE package.
- Create a file called `NOISE_TOMOGRAPHY/irec_master_noise`. Note that this file should be put in directory `NOISE_TOMOGRAPHY` as well. This file contains only one integer, which is the ID of the 'master' receiver. For example, if in this file shows 5, it means that the fifth receiver listed in `DATA/STATIONS` becomes the 'master'. That's why we mentioned previously that the order of receivers in `DATA/STATIONS` is important.

  Note that in regional (1- or 2-chunk) simulations, the `DATA/STATIONS` may contain receivers not within the selected chunk(s). Therefore, the integer in `NOISE_TOMOGRAPHY/irec_master_noise` is actually the ID in `DATA/STATIONS_FILTERED` (which is generated by `xspecfem3D`).
- Create a file called `NOISE_TOMOGRAPHY/nu_master`. This file holds three numbers, forming a (unit) vector. It describes which component we are cross-correlating at the 'master' receiver, i.e., $\hat{\nu}^{\alpha}$ in **?**. The three numbers correspond to N/E/Z components respectively.
- Describe the noise direction and distributions in `src/specfem3d/noise_tomography.f90`. Search for a subroutine called `noise_distribution_direction` in `noise_tomography.f90`. It is actually located at the very beginning of `noise_tomography.f90`. The default assumes vertical noises and a uniform distribution across the whole physical domain. It should be quite self-explanatory for modifications. Should you modify this part, you have to re-compile the source code. (again, that's why we recommend that you always re-compile the code before you run simulations)

7.2.2 Simulations

As discussed in **?**, it takes three simulations to obtain one contribution of the ensemble sensitivity kernels:

- Step 1: simulation for generating wavefield
  `SIMULATION_TYPE=1`
  `NOISE_TOMOGRAPHY=1`
  `SAVE_FORWARD` not used, can be either .true. or .false.
- Step 2: simulation for ensemble forward wavefield
  `SIMULATION_TYPE=1`
  `NOISE_TOMOGRAPHY=2`
  `SAVE_FORWARD=.true.`
- Step 3: simulation for ensemble adjoint wavefield and sensitivity kernels
  `SIMULATION_TYPE=3`
  `NOISE_TOMOGRAPHY=3`
  `SAVE_FORWARD=.false.`

  Note Step 3 is an adjoint simulation, please refer to previous chapters on how to prepare adjoint sources and other necessary files, as well as how adjoint simulations work.

It's better to run the three steps continuously within the same job, otherwise you have to collect the saved surface movies from the old nodes to the new nodes.

7.2.3 Post-simulation

After those simulations, you have all stuff you need, either in the `OUTPUT_FILES` or in the directory specified by `LOCAL_PATH` in `DATA/Par_file` (most probably on local nodes). Collect whatever you want from the local nodes to your workstation, and then visualize them. This process is the same as what you may have done for regular earthquake simulations. Refer Chapter 8 if you have problems.

Simply speaking, two outputs are the most interesting: the simulated ensemble cross correlations and one contribution of the ensemble sensitivity kernels.

The simulated ensemble cross correlations are obtained after the second simulation (Step 2). Seismograms in `OUTPUT_FILES` are actually the simulated ensemble cross correlations. Collect them immediately after Step 2, or the Step 3 will overwrite them. Note that we have a 'master' receiver specified by `NOISE_TOMOGRAPHY/irec_master_noise`, the seismogram at one station corresponds to the cross correlation between that station and the 'master'. Since the seismograms have three components, we may obtain cross correlations for different components as well, not necessarily the cross correlations between vertical components.

One contribution of the ensemble cross-correlation sensitivity kernels are obtained after Step 3, stored in the `LOCAL_PATH` on local nodes. The ensemble kernel files are named the same as regular earthquake kernels.

You need to run another three simulations to get the other contribution of the ensemble kernels, using different forward and adjoint sources given in **?**.

7.3 Examples

In order to illustrate noise simulations in an easy way, three examples are provided in `EXAMPLES/noise_examples/`. Note however that they are created for a specific cluster (SESAME@PRINCETON). You have to modify them to fit your own cluster.

The three examples can be executed using (in directory `EXAMPLES/noise_examples/`):

```
./pre-processing regional
./pre-processing global_short
./pre-processing global_long
```

Each corresponds to one example, but they are pretty similar.

Although the job submission only works on SESAME@PRINCETON, the procedure itself is universal. You may review the whole process described in the last section by following commands in those examples.

Finally, note again that those examples show only one contribution of the ensemble kernels!

Chapter 8

Graphics

8.1 Meshes

Use the serial code `combine_AVS_DX.f90` (type '`make combine_AVS_DX`' and then '`xcombine_AVS_DX`') to generate AVS (`www.avs.com`) output files (in AVS UCD format) or OpenDX (`www.opendx.org`) output files showing the mesh, the MPI partition (slices), the `NCHUNKS` chunks, the source and receiver location, etc. Use the AVS UCD files `AVS_continent_boundaries.inp` and `AVS_plate_boundaries.inp` or the OpenDX files `DX_continent_boundaries.dx` and `DX_plate_boundaries.dx` (that can be created using Perl scripts located in `UTILS/Visualization/opendx_AVS`) for reference.

8.2 Movies

To make a surface or volume movie of the simulation, set parameters `MOVIE_SURFACE`, `MOVIE_VOLUME`, and `NTSTEP_BETWEEN_FRAMES` in the `Par_file`. Turning on the movie flags, in particular `MOVIE_VOLUME`, produces large output files. `MOVIE_VOLUME` files are saved in the `LOCAL_PATH` directory, whereas `MOVIE_SURFACE` output files are saved in the `OUTPUT_FILES` directory. We save the velocity field. The look of a movie is determined by the half-duration of the source. The half-duration should be large enough so that the movie does not contain frequencies that are not resolved by the mesh, i.e., it should not contain numerical noise. This can be accomplished by selecting a CMT `HALF_DURATION` > 1.1 $\times$ smallest period (see figure 4.1). When `MOVIE_SURFACE` = `.true.` or `MOVIE_VOLUME` = `.true.`, the half duration of each source in the `CMTSOLUTION` file is replaced by

$$\sqrt{(\texttt{HALF\_DURATION}^2 + \texttt{HDUR\_MOVIE}^2)}$$

NOTE: If `HDUR_MOVIE` is set to 0.0, the code will select the appropriate value of 1.1 $\times$ smallest period. As usual, for a point source one can set `HALF_DURATION` in the `Par_file` to be 0.0 and `HDUR_MOVIE` = 0.0 to get the highest frequencies resolved by the simulation, but for a finite source one would keep all the `HALF_DURATION`s as prescribed by the finite source model and set `HDUR_MOVIE` = 0.0.

8.2.1 Movie Surface

When running `xspecfem3D` with the `MOVIE_SURFACE` flag turned on the code outputs `moviedata??????` files in the `OUTPUT_FILES` directory. The files are in a fairly complicated binary format, but there are two programs provided to convert the output into more user friendly formats. The first one, `create_movie_AVS_DX.f90` outputs data in ASCII, OpenDX, AVS, or ParaView format. Run the code from the source directory (type '`make create_movie_AVS_DX`' first) to create an input file in your format of choice. The code will prompt the user for input parameters. The second program `create_movie_GMT_global.f90` outputs ASCII xyz files, convenient for use with GMT. This codes uses significantly less memory than `create_movie_AVS_DX.f90` and is therefore useful for high resolution runs. A README file and sample Perl scripts to create movies using GMT are provided in directory `UTILS/Visualization/GMT`.

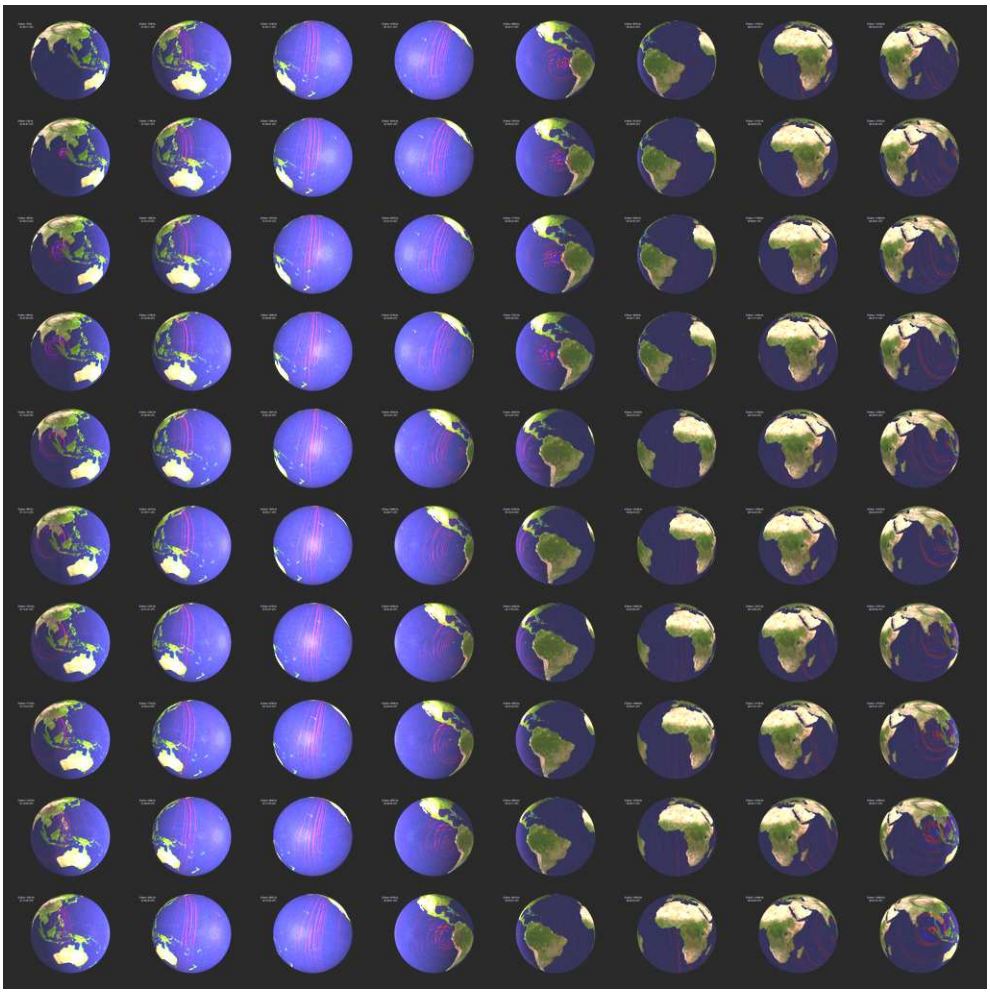


Figure 8.1: Snapshots from a global movie for the December 26, 2004, M=9.2 Sumatra-Andaman earthquake. Time runs down successive columns.

8.2.2 Movie Volume

When running xspecfem3D with the `MOVIE_VOLUME` flag turned on, the code outputs several files in `LOCAL_DIR`. As the files can be very large, there are several flags in the `Par_file` that control the region in space and time that is saved. These are: `MOVIE_TOP_KM`, `MOVIE_BOTTOM_KM`, `MOVIE_WEST_DEG`, `MOVIE_EAST_DEG`, `MOVIE_NORTH_DEG`, `MOVIE_SOUTH_DEG`, `MOVIE_START` and `MOVIE_STOP`. The code will save a given element if the center of the element is in the prescribed volume.

The Top/Bottom: Depth below the surface in kilometers, use `MOVIE_TOP = -100.0` to make sure the surface is stored.

West/East: Longitude, degrees East [-180.0/180.0]

North/South: Latitute, degrees North [-90.0/90.0]

Start/Stop: Frames will be stored at `MOVIE_START + i*NSTEP_BETWEEN_FRAMES`, where `i=(0,1,2..)` while `i*NSTEP_BETWEEN_FRAMES <= MOVIE_STOP`

The code saves several files, and the output is saved by each processor. The first is `proc??????_movie3D_info.txt` which contains two numbers, first the number of points within the prescribed volume within this particular slice, and

second the number of elements. The next files are proc??????_movie3D_x.bin, proc??????_movie3D_y.bin, proc??????_movie3D_z.bin which store the locations of the points in the 3D mesh.

Finally the code stores the "value" at each of the points. Which value is determined by MOVIE_VOLUME_TYPE in the Par_file. Choose 1 to save the strain, 2 to save the time integral of strain, and 3 to save μ*time integral of strain in the subvolume. Choosing 4 causes the code to save the trace of the stress and the deviatoric stress in the whole volume (not the subvolume in space), at the time steps specified. The name of the output file will depend on the MOVIE_VOLUME_TYPE chosen.

Setting MOVIE_VOLUME_COARSE = .true. will make the code save only the corners of the elements, not all the points within each element for MOVIE_VOLUME_TYPE = 1,2,3.

To make the code output your favorite "value" simply add a new MOVIE_VOLUME_TYPE, a new subroutine to write_movie_volume.f90 and a subroutine call to specfem3D.F90.

A utility program to combine the files produced by MOVIE_VOLUME_TYPE = 1,2,3 is provided in combine_paraview_strain_data.f90. Type xcombine_paraview_strain_data to get the usage statement. The program combine_vol_data.f90 can be used for MOVIE_VOLUME_TYPE = 4.

8.3 Finite-Frequency Kernels

The finite-frequency kernels computed as explained in Section 6.2 are saved in the LOCAL_PATH at the end of the simulation. Therefore, we first need to collect these files on the front end, combine them into one mesh file, and visualize them with some auxilliary programs. Examples of kernel simulations may be found in the EXAMPLES directory.

1. **Create slice files**

   We will only discuss the case of one source-receiver pair, i.e., the so-called banana-doughnut kernels. Although it is possible to collect the kernel files from all slices onto the front end, it usually takes up too much storage space (at least tens of gigabytes). Since the sensitivity kernels are the strongest along the source-receiver great circle path, it is sufficient to collect only the slices that are along or close to the great circle path.

   A Perl script UTILS/Visualization/Paraview/global_slice_number.pl can help to figure out the slice numbers that lie along the great circle path (both the minor and major arcs), as well as the slice numbers required to produce a full picture of the inner core if your kernel also illuminates the inner core.

   (a) You need to first compile the utility programs provided in the UTILS/Visualization/Paraview/global_slice_util directory. Then copy the CMTSOLUTION file, STATIONS_ADJOINT, and Par_file, and run:

   ```
   global_slice_number.pl CMTSOLUTION STATIONS_ADJOINT Par_file
   ```

   In the case of visualization boundary kernels or spherical cross-sections of the volumetric kernels, it is necessary to obtain the slice numbers that cover a belt along the source and receiver great circle path, and you can use the hybrid version:

   ```
   globe_slice_number2.pl CMTSOLUTION STATIONS_ADJOINT
        Par_file belt_width_in_degrees
   ```

   A typical value for belt_width_in_degrees can be 20.

   (b) For a full 6-chunk simulation, this script will generate the slice_minor, slice_major, slice_ic files, but for a one- or two-chunk simulation, this script only generates the slice_minor file.

   (c) For cases with multiple sources and multiple receivers, you need to provide a slice file before proceeding to the next step.

2. **Collect the kernel files**

   After obtaining the slice files, you can collect the corresponding kernel files from the given slices.

   (a) To accomplish this, you can use or modify the scripts in UTILS/collect_database directory:

```
copy_m(oc,ic)_globe_database.pl slice_file lsf_machine_file filename [jobid]
```

for volumetric kernels, where `lsf_machine_file` is the machine file generated by the LSF scheduler, `filename` is the kernel name (e.g., `rho_kernel`, `alpha_kernel` and `beta_kernel`), and the optional `jobid` is the name of the subdirectory under `LOCAL_PATH` where all the kernel files are stored. For boundary kernels, you need to use

```
copy_surf_globe_database.pl slice_file lsf_machine_file filename [jobid]
```

where the filename can be `Moho_kernel`, `d400_kernel`, `d670_kernel`, `CMB_kernel` and `ICB_kernel`.

(b) After executing this script, all the necessary mesh topology files as well as the kernel array files are collected to the local directory on the front end.

3. **Combine kernel files into one mesh file**

We use an auxiliary program `combine_vol_data.f90` to combine the volumetric kernel files from all slices into one mesh file, and `combine_surf_data.f90` to combine the surface kernel files.

(a) Compile it in the global code directory:

```
make combine_vol_data
./bin/xcombine_vol_data slice_list kernel_filename  input_topo_dir  input_file_dir output_dir
low/high-resolution-flag-0-or-1 [region]
```

where `input_dir` is the directory where all the individual kernel files are stored, and `output_dir` is where the mesh file will be written. Give 0 for low resolution and 1 for high resolution. If region is not specified, all three regions (crust and mantle, outer core, inner core) will be collected, otherwise, only the specified region will be. Here is an example: `./xcombine_vol_data slices_major alpha_kernel input_topo_dir input_file_dir output_dir 1`

```
./bin/xcombine_surf_data slice_list filename surfname input_dir output_dir
low/high-resolution 2D/3D
```

where `surfname` should correspond to the specific kernel file name, and can be chosen from `Moho`, `400`, `670`, `CMB` and `ICB`.

(b) Use 1 for a high-resolution mesh, outputting all the GLL points to the mesh file, or use 0 for low resolution, outputting only the corner points of the elements to the mesh file. Use 0 for 2D surface kernel files and 1 for 3D volumetric kernel files.

(c) Use region = 1 for the mantle, region =2 for the outer core, region = 3 for the inner core, and region = 0 for all regions.

(d) The output mesh file will have the name `reg_?_rho(alpha,beta)_kernel.mesh`, or `reg_?_Moho(d400,d670,CMB,ICB)_kernel.surf`.

4. **Convert mesh files into .vtu files**

(a) We next convert the `.mesh` file into the VTU (Unstructured grid file) format which can be viewed in ParaView, for example:

```
mesh2vtu -i file.mesh -o file.vtu
```

(b) Notice that this program `mesh2vtu`, in the `UTILS/Visualization/Paraview/mesh2vtu` directory, uses the VTK (`www.vtk.org`) run-time library for its execution. Therefore, make sure you have it properly installed.

5. **Copy over the source and receiver .vtk file**

In the case of a single source and a single receiver, the simulation also generates the `OUTPUT_FILES/sr.vtk` file to describe the source and receiver locations, which can be viewed in Paraview in the next step.

6. **View the mesh in ParaView**

   Finally, we can view the mesh in ParaView (`www.paraview.org`).

   (a) Open ParaView.

   (b) From the top menu, File → Open data, select `file.vtu`, and click the Accept button.

   - If the mesh file is of moderate size, it shows up on the screen; otherwise, only the outline is shown.

   (c) Click Display Tab → Display Style → Representation and select wireframe of surface to display it.

   (d) To create a cross-section of the volumetric mesh, choose Filter → cut, and under Parameters Tab, choose Cut Function → plane.

   (e) Fill in center and normal information given by the standard output from `global_slice_number.pl` script.

   (f) To change the color scale, go to Display Tab → Color → Edit Color Map and reselect lower and upper limits, or change the color scheme.

   (g) Now load in the source and receiver location file by File → Open data, select `sr.vtk`, and click the Accept button. Choose Filter → Glyph, and represent the points by 'spheres'.

   (h) For more information about ParaView, see the ParaView Users Guide (`www.paraview.org/files/v1.6/ParaViewUsersGuide.PDF`).

For illustration purposes, Figure 8.2 shows P-wave speed finite-frequency kernels from cross-correlation traveltime and amplitude measurements for a P arrival recorded at an epicentral distance of 60° for a deep event.

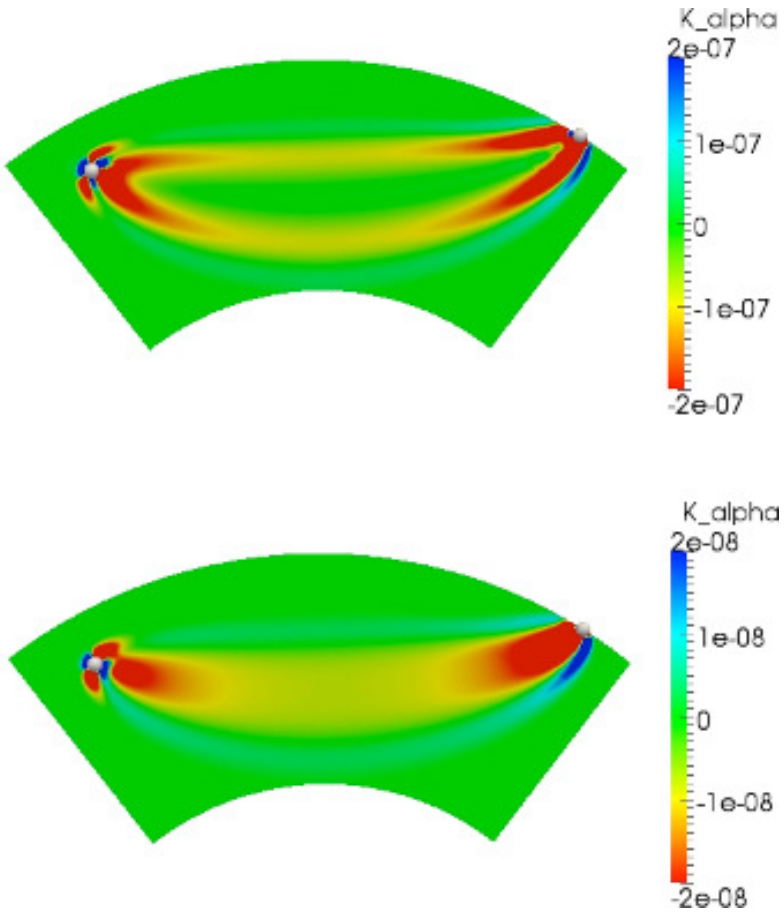


Figure 8.2: P-wave speed finite-frequency kernels from cross-correlation traveltime (top) and amplitude (bottom) measurements for a P arrival recorded at an epicentral distance of 60°. The kernels together with the associated files and routines to reproduce them may be found in `EXAMPLES/global_PREM_kernels/`.

Chapter 9

Running through a Scheduler

The code is usually run on large parallel machines, often PC clusters, most of which use schedulers, i.e., queuing or batch management systems to manage the running of jobs from a large number of users. The following considerations need to be taken into account when running on a system that uses a scheduler:

- The processors/nodes to be used for each run are assigned dynamically by the scheduler, based on availability. Therefore, in order for the mesher and the solver (or between successive runs of the solver) to have access to the same database files (if they are stored on hard drives local to the nodes on which the code is run), they must be launched in sequence as a single job.
- On some systems, the nodes to which running jobs are assigned are not configured for compilation. It may therefore be necessary to pre-compile both the mesher and the solver. A small program provided in the distribution called `create_header_file.f90` can be used to directly create `OUTPUT_FILES/values_from_mesher.h` using the information in the `DATA/Par_file` without having to run the mesher (type '`make create_header_file`' to compile it and '`./bin/xcreate_header_file`' to run it; refer to the sample scripts below). The solver can now be compiled as explained above.
- One feature of schedulers/queuing systems is that they allow submission of multiple jobs in a "launch and forget" mode. In order to take advantage of this property, care needs to be taken that output and intermediate files from separate jobs do not overwrite each other, or otherwise interfere with other running jobs.

We describe here in some detail a job submission procedure for the Caltech 1024-node cluster, CITerra, under the LSF scheduling system. We consider the submission of a regular forward simulation. The two main scripts are `run_lsf.bash`, which compiles the Fortran code and submits the job to the scheduler, and `go_mesher_solver_lsf.bash`, which contains the instructions that make up the job itself. These scripts can be found in `UTILS/Cluster/lsf` directory and can straightforwardly be modified and adapted to meet more specific running needs.

9.1 `run_lsf.bash`

This script first sets the job queue to be 'normal'. It then compiles the mesher and solver together, figures out the number of processors required for this simulation from `DATA/Par_file`, and submits the LSF job.

```
#!/bin/bash
# use the normal queue unless otherwise directed queue="-q normal"
if [ $# -eq 1 ]; then
        echo "Setting the queue to $1"
        queue="-q $1"
fi

# compile the mesher and the solver
d=`date` echo "Starting compilation $d"
```

```
make clean
make meshfem3D
make create_header_file
./bin/xcreate_header_file
make specfem3D
d=‘date’
echo "Finished compilation $d"

# compute total number of nodes needed
NPROC_XI=‘grep NPROC_XI DATA/Par_file | cut -c 34- ’
NPROC_ETA=‘grep NPROC_ETA DATA/Par_file | cut -c 34- ’
NCHUNKS=‘grep NCHUNKS DATA/Par_file | cut -c 34- ’

# total number of nodes is the product of the values read
numnodes=$(( $NCHUNKS * $NPROC_XI * $NPROC_ETA ))

echo "Submitting job"
bsub $queue -n $numnodes -W 60 -K <go_mesher_solver_lsf_globe.bash
```

9.2 go_mesher_solver_lsf_globe.bash

This script describes the job itself, including setup steps that can only be done once the scheduler has assigned a job-ID and a set of compute nodes to the job, the `run_lsf.bash` commands used to run the mesher and the solver, and calls to scripts that collect the output seismograms from the compute nodes and perform clean-up operations.

1. First the script directs the scheduler to save its own output and output from `stdout` into `OUTPUT_FILES/%J.o`, where `%J` is short-hand for the job-ID; it also tells the scheduler what version of `mpich` to use (`mpich_gm`) and how to name this job (`go_mesher_solver_lsf`).

2. The script then creates a list of the nodes allocated to this job by echoing the value of a dynamically set environment variable `LSB_MCPU_HOSTS` and parsing the output into a one-column list using the Perl script `UTILS/Cluster/lsf/remap_lsf_machines.pl`. It then creates a set of scratch directories on these nodes (`/scratch/$USER/DATABASES_MPI`) to be used as the `LOCAL_PATH` for temporary storage of the database files. The scratch directories are created using `shmux`, a shell multiplexor that can execute the same commands on many hosts in parallel. `shmux` is available from Shmux (`web.taranis.org/shmux/`). Make sure that the `LOCAL_PATH` parameter in `DATA/Par_file` is also set properly.

3. The next portion of the script launches the mesher and then the solver using `run_lsf.bash`.

4. The final portion of the script performs clean up on the nodes using the Perl script `cleanmulti.pl`

```
#!/bin/bash -v
#BSUB -o OUTPUT_FILES/%J.o
#BSUB -a mpich_gm
#BSUB -J go_mesher_solver_lsf
BASEMPIDIR=/scratch/$USER/DATABASES_MPI
echo "$LSB_MCPU_HOSTS" > OUTPUT_FILES/lsf_machines
echo "$LSB_JOBID" > OUTPUT_FILES/jobid
./remap_lsf_machines.pl OUTPUT_FILES/lsf_machines >OUTPUT_FILES/machines
# Modif : create a directory for this job
shmux -M50 -Sall -c "mkdir -p /scratch/$USER;
mkdir -p $BASEMPIDIR.$LSB_JOBID" - < OUTPUT_FILES/machines >/dev/null
# Set the local path in Par_file
sed -e "s:^LOCAL_PATH .*:LOCAL_PATH = $BASEMPIDIR.$LSB_JOBID:"
```

```
< DATA/Par_file > DATA/Par_file.tmp
mv DATA/Par_file.tmp DATA/Par_file
current_pwd=$PWD
mpirun.lsf  --gm-no-shmem --gm-copy-env $current_pwd/bin/xmeshfem3D
mpirun.lsf --gm-no-shmem --gm-copy-env $current_pwd/bin/xspecfem3D
# clean up
cleanbase_jobid.pl OUTPUT_FILES/machines DATA/Par_file
```

9.3 `run_lsf.kernel` and `go_mesher_solver_globe.kernel`

For kernel simulations, you can use the sample run scripts `run_lsf.kernel` and `go_mesher_solver_globe.kernel` provided in `UTILS/Cluster` directory, and modify the command-line arguments of `xcreate_adjsrc_traveltime` in `go_mesher_solver_globe.kernel` according to the start and end time of the specific portion of the forward seismograms you are interested in.

Chapter 10

Changing the Model

In this section we explain how to change the crustal, mantle, or inner core models. These changes involve contributing specific subroutines that replace existing subroutines in the SPECFEM3D_GLOBE package.

10.1 Changing the Crustal Model

The 3D crustal model Crust2.0 [**?**] is superimposed onto the mesh by the subroutine model_crust.f90. To accomplish this, the flag CRUSTAL, set in the subroutine get_model_parameters.f90, is used to indicate a 3D crustal model. When this flag is set to .true., the crust on top of the 1D reference model (PREM, IASP91, or AK135F_NO_MUD) is removed and replaced by extending the mantle. The 3D crustal model is subsequently overprinted onto the crust-less 1D reference model. The call to the 3D crustal routine is of the form

```
call model_crust(lat,lon,r,vp,vs,rho,moho,foundcrust,CM_V,elem_in_crust)
```

Input to this routine consists of:

lat Latitude in degrees.

lon Longitude in degrees.

r Non-dimensionalized radius ($0 < r < 1$).

Output from the routine consists of:

vp Non-dimensionalized compressional wave speed at location (lat,lon,r).

vs Non-dimensionalized shear wave speed.

rho Non-dimensionalized density.

moho Non-dimensionalized Moho depth.

found_crust Logical that is set to .true. only if crust exists at location (lat,lon,r), i.e., .false. for radii r in the mantle. This flags determines whether or not a particular location is in the crust and, if so, what parameters to assign to the mesh at this location.

CM_V Fortran structure that contains the parameters, variables and arrays that describe the model.

elem_in_crust Logical that is used to force the routine to return crustal values, even if the location would be below the crust.

All output needs to be non-dimensionalized according to the convention summarized in Appendix B. You can replace this subroutine by your own routine *provided you do not change the call structure of the routine*, i.e., the new routine should take exactly the same input and produce the required, properly non-dimensionalized output.

Part of the file `model_crust.f90` is the subroutine `model_crust_broadcast`. The call to this routine takes argument `CM_V` and is used to once-and-for-all read in the databases related to Crust2.0 and broadcast the model to all parallel processes. If you replace the file `model_crust.f90` with your own implementation, you *must* provide a `model_crust_broadcast` routine, even if it does nothing. Model constants and variables read by the routine `model_crust_broadcast` are passed to the subroutine `read_crust_model` through the structure `CM_V`. An alternative crustal model could use the same construct. Please feel free to contribute subroutines for new models and send them to us so that they can be included in future releases of the software.

> **NOTE:** If you decide to create your own version of file `model_crust.f90`, you must add calls to `MPI_BCAST` in the subroutine `model_crust_broadcast` after the call to the `read_crust_model` subroutine that reads the isotropic mantle model once and for all in the mesher. This is done in order to read the (potentially large) model data files on the master node (which is the processor of rank 0 in our code) only and then send a copy to all the other nodes using an MPI broadcast, rather than using an implementation in which all the nodes would read the same model data files from a remotely-mounted home file system, which could create a bottleneck on the network in the case of a large number of nodes. For example, in the current call to that routine from `model_crust.f90`, we write:

```
! the variables read are declared and stored in structure CM_V
  if(myrank == 0) call read_crust_model(CM_V)
! broadcast the information read on the master to the nodes
  call MPI_BCAST(CM_V%thlr,NKEYS_CRUST*NLAYERS_CRUST,MPI_DOUBLE_PRECISION,
                 0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(CM_V%velocp,NKEYS_CRUST*NLAYERS_CRUST,MPI_DOUBLE_PRECISION,
                 0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(CM_V%velocs,NKEYS_CRUST*NLAYERS_CRUST,MPI_DOUBLE_PRECISION,
                 0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(CM_V%dens,NKEYS_CRUST*NLAYERS_CRUST,MPI_DOUBLE_PRECISION,
                 0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(CM_V%abbreviation,NCAP_CRUST*NCAP_CRUST,MPI_CHARACTER,
                 0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(CM_V%code,2*NKEYS_CRUST,MPI_CHARACTER,0,MPI_COMM_WORLD,ier)
```

10.2 Changing the Mantle Model

This section discusses how to change isotropic and anisotropic 3D mantle models. Usually such changes go hand-in-hand with changing the 3D crustal model.

10.2.1 Isotropic Models

The 3D mantle model S20RTS [**?**] is superimposed onto the mantle mesh by the subroutine `model_s20rts.f90`. The call to this subroutine is of the form

```
call model_s20rts(radius,theta,phi,dvs,dvp,drho,D3MM_V)
```

Input to this routine consists of:

`radius` Non-dimensionalized radius (`RCMB/R_ EARTH` $<$ `r` $<$ `RMOHO/R_ EARTH`; for a given 1D reference model, the constants `RCMB` and `RMOHO` are set in the `get_model_parameters.f90` file). The code expects the isotropic mantle model to be defined between the Moho (with radius `RMOHO` in m) and the core-mantle boundary (CMB; radius `RCMB` in m) of a 1D reference model. When a 3D crustal model is superimposed, as will usually be the case, the 3D mantle model is stretched to fill any potential gap between the radius of the Moho in the 1D reference model and the Moho in the 3D crustal model. Thus, when the Moho in the 3D crustal model is shallower than the Moho in the reference model, e.g., typically below the oceans, the mantle model is extended to fill this gap.

theta Colatitude in radians.

phi Longitude in radians.

Output from the routine are the following non-dimensional perturbations:

dvs Relative shear-wave speed perturbations $\delta\beta/\beta$ at location (radius,theta,phi).

dvp Relative compressional-wave speed perturbations $\delta\alpha/\alpha$.

drho Relative density perturbations $\delta\rho/\rho$.

D3MM_V Fortran structure that contains the parameters, variables and arrays that describe the model.

You can replace the model_s20rts.f90 file with your own version *provided you do not change the call structure of the routine*, i.e., the new routine should take exactly the same input and produce the required relative output.

Part of the file model_s20rts.f90 is the subroutine model_s20rts_broadcast. The call to this routine takes argument D3MM_V and is used to once-and-for-all read in the databases related to S20RTS. If you replace the file model_s20rts.f90 with your own implementation, you *must* provide a model_s20rts_broadcast routine, even if it does nothing. Model constants and variables read by the routine model_s20rts_broadcast are passed to the subroutine read_model_s20rts through the structure D3MM_V. An alternative mantle model should use the same construct.

> **NOTE:** If you decide to create your own version of file model_s20rts.f90, you must add calls to MPI_BCAST in the subroutine model_s20rts_broadcast after the call to the read_model_s20rts subroutine that reads the isotropic mantle model once and for all in the mesher. This is done in order to read the (potentially large) model data files on the master node (which is the processor of rank 0 in our code) only and then send a copy to all the other nodes using an MPI broadcast, rather than using an implementation in which all the nodes would read the same model data files from a remotely-mounted home file system, which could create a bottleneck on the network in the case of a large number of nodes. For example, in the current call to that routine from model_s20rts.f90, we write:

```
! the variables read are declared and stored in structure D3MM_V
  if(myrank == 0) call read_model_s20rts(D3MM_V)
! broadcast the information read on the master to the nodes
  call MPI_BCAST(D3MM_V%dvs_a,(NK+1)*(NS+1)*(NS+1),MPI_DOUBLE_PRECISION,
          0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%dvs_b,(NK+1)*(NS+1)*(NS+1),MPI_DOUBLE_PRECISION,
          0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%dvp_a,(NK+1)*(NS+1)*(NS+1),MPI_DOUBLE_PRECISION,
          0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%dvp_b,(NK+1)*(NS+1)*(NS+1),MPI_DOUBLE_PRECISION,
          0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%spknt,NK+1,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%qq0,(NK+1)*(NK+1),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(D3MM_V%qq,3*(NK+1)*(NK+1),MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ier)
```

10.2.2 Anisotropic Models

Three-dimensional anisotropic mantle models may be superimposed on the mesh based upon the subroutine

```
model_aniso_mantle.f90
```

The call to this subroutine is of the form

```
call model_aniso_mantle(r,theta,phi,rho, &
    c11,c12,c13,c14,c15,c16,c22,c23,c24,c25,c26, &
    c33,c34,c35,c36,c44,c45,c46,c55,c56,c66,AMM_V)
```

Input to this routine consists of:

r Non-dimensionalized radius (RCMB/R_ EARTH $<$ r $<$ RMOHO/R_ EARTH; for a given 1D reference model, the constants RCMB and RMOHO are set in the get_model_parameters.f90 file). The code expects the anisotropic mantle model to be defined between the Moho and the core-mantle boundary (CMB). When a 3D crustal model is superimposed, as will usually be the case, the 3D mantle model is stretched to fill any potential gap between the radius of the Moho in the 1D reference model and the Moho in the 3D crustal model. Thus, when the Moho in the 3D crustal model is shallower than the Moho in the reference model, e.g., typically below the oceans, the mantle model is extended to fill this gap.

theta Colatitude in radians.

phi Longitude in radians.

Output from the routine consists of the following non-dimensional model parameters:

rho Non-dimensionalized density ρ.

c11, $\cdots$**, c66** 21 non-dimensionalized anisotropic elastic parameters.

AMM_V Fortran structure that contains the parameters, variables and arrays that describe the model.

You can replace the model_aniso_mantle.f90 file by your own version *provided you do not change the call structure of the routine*, i.e., the new routine should take exactly the same input and produce the required relative output. Part of the file model_aniso_mantle.f90 is the subroutine model_aniso_mantle_broadcast. The call to this routine takes argument AMM_V and is used to once-and-for-all read in the static databases related to the anisotropic model. When you choose to replace the file model_aniso_mantle.f90 with your own implementation you *must* provide a model_aniso_mantle_broadcast routine, even if it does nothing. Model constants and variables read by the routine model_aniso_mantle_broadcast are passed through the structure AMM_V. An alternative anisotropic mantle model should use the same construct.

> **NOTE:** If you decide to create your own version of file model_aniso_mantle.f90, you must add calls to MPI_BCAST in file model_aniso_mantle.f90 after the call to the read_aniso_mantle_model subroutine that reads the anisotropic mantle model once and for all in the mesher. This is done in order to read the (potentially large) model data files on the master node (which is the processor of rank 0 in our code) only and then send a copy to all the other nodes using an MPI broadcast, rather than using an implementation in which all the nodes would read the same model data files from a remotely-mounted home file system, which could create a bottleneck on the network in the case of a large number of nodes. For example, in the current call to that routine from model_aniso_mantle.f90, we write:

```
! the variables read are declared and stored in structure AMM_V
  if(myrank == 0) call read_aniso_mantle_model(AMM_V)
! broadcast the information read on the master to the nodes
  call MPI_BCAST(AMM_V%npar1,1,MPI_INTEGER,0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(AMM_V%beta,14*34*37*73,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ier)
  call MPI_BCAST(AMM_V%pro,47,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ier)
```

Rotation of the anisotropic tensor elements from one coordinate system to another coordinate system may be accomplished based upon the subroutine rotate_aniso_tensor. Use of this routine requires understanding the coordinate system used in SPECFEM3D_GLOBE, as discussed in Appendix A.

10.2.3 Point-Profile Models

In order to facilitate the use of your own specific mantle model, you can choose PPM as model in the DATA/Par_file file and supply your own model as an ASCII-table file. These generic models are given as depth profiles at a specified lon/lat location and a perturbation (in percentage) with respect to the shear-wave speed values from PREM. The ASCII-file should have a format like:

```
#lon(deg), lat(deg), depth(km), Vs-perturbation wrt PREM(%), Vs-PREM (km/s)
-10.00000 31.00000 40.00000 -1.775005 4.400000
-10.00000 32.00000 40.00000 -1.056823 4.400000
...
```

where the first line is a comment line and all following ones are specifying the Vs-perturbation at a lon/lat location and a given depth. The last entry on each line is specifying the absolute value of Vs (however this value is only given as a supplementary information and not used any further). The background model is PREM with a transverse isotropic layer between Moho and 220 km depth. The specified Vs-perturbations are added as isotropic perturbations. Please see the file `DATA/PPM/README` for more informations how to setup the directory `DATA/PPM` to use your own ASCII-file.

To change the code behavior of these PPM-routines, please have a look at the implementation in the source code file `model_ppm.f90` and set the flags and scaling factors as needed for your purposes. Perturbations in density and Vp may be scaled to the given Vs-perturbations with constant scaling factors by setting the appropriate values in this source code file. In case you want to change the format of the input ASCII-file, see more details in the Appendix F.

10.3 Anelastic Models

Three-dimensional anelastic (attenuation) models may be superimposed onto the mesh based upon your subroutine . `model_atten3D.f90`. The call to this routine would be as follows

```
call model_atten3D(radius, colatitude, longitude, Qmu, QRFSI12_Q, idoubling)
```

Input to this routine consists of:

`radius` scaled radius of the earth: $0\,(\text{center}) <= r\ <= 1$(surface)

`latitude` Colatitude in degrees: $0^\circ <= \theta <= 180^\circ$

`longitude` Longitude in degrees: $-180^\circ <= \phi <= 180^\circ$

`QRFSI12_Q` Fortran structure that contains the parameters, variables and arrays that describe the model

`idoubling` value of the doubling index flag in each radial region of the mesh

Output to this routine consists of:

`Qmu` Shear wave quality factor: $0 < Q_\mu < 5000$

A 3-D attenuation model QRFSI12 [**?**] is provided, as well as 1-D models with a PREM and a 1DREF attenuation structure. By default the PREM attenuation model is taken, using the routine `model_attenuation_1D_PREM`, found in `model_attenuation.f90`.

To create your own three-dimensional attenuation model, you add your model using a routine like the `model_atten3D_QRFSI12` subroutine and the example routine above as a guide and replace the call in file `meshfem3D_models.f90` to your own subroutine.

Note that the resolution and maximum value of anelastic models are truncated. This speeds the construction of the standard linear solids during the meshing stage. To change the resolution, currently at one significant figure following the decimal, or the maximum value (5000), consult `constants.h`. In order to prevent unexpected results, quality factors Q_μ should never be equal to 0 outside of the inner core.

Chapter 11

Post-Processing Scripts

Several post-processing scripts/programs are provided in the `UTILS/seis_process` directory, most of which need to be adjusted for different systems, for example, the path of the executable programs. Here we only list the available scripts and provide a brief description, and you can either refer to the related sections for detailed usage or, in many cases, type the script/program name without arguments to see its usage.

11.1 Clean Local Database

After all the simulations are done, you may need to clean the local scratch disks for the next simulation. This is especially important in the case of 1- or 2-chunk kernel simulations, where very large files are generated for the absorbing boundaries to help with the reconstruction of the regular forward wavefield. A sample script is provided in `UTILS/Cluster/lsf`:

```
cleanbase.pl machines
```

11.2 Process Data and Synthetics

In many cases, the SEM synthetics are calculated and compared to observed seismograms recorded at seismic stations. Since the SEM synthetics are accurate for a certain frequency range, both the original data and the synthetics need to be processed before a comparison can be made. For such comparisons, the following steps are recommended:

1. Make sure that both synthetic and observed seismograms have the correct station/event and timing information.
2. Convolve synthetic seismograms with a source time function with the half duration specified in the `CMTSOLUTION` file, provided, as recommended, you used a zero half duration in the SEM simulations.
3. Resample both observed and synthetic seismograms to a common sampling rate.
4. Cut the records using the same window.
5. Remove the trend and mean from the records and taper them.
6. Remove the instrument response from the observed seismograms (recommended) or convolve the synthetic seismograms with the instrument response.
7. Make sure that you apply the same filters to both observed and synthetic seismograms. Preferably, avoid filtering your records more than once.
8. Now, you are ready to compare your synthetic and observed seismograms.

We generally use the following scripts for processing:

11.2.1 process_data.pl

This script cuts a given portion of the original data, filters it, transfers the data into a displacement record, and picks the first P and S arrivals. For more functionality, type 'process_data.pl' without any argument. An example of the usage of the script:

```
process_data.pl -m CMTSOLUTION -s 1.0 -l 0/4000 -i -f -t 40/500 -p -x bp DATA/1999.330*.LH?.SAC
```

which has resampled the SAC files to a sampling rate of 1 seconds, cut them between 0 and 4000 seconds, transfered them into displacement records and filtered them between 40 and 500 seconds, picked the first P and S arrivals, and added suffix 'bp' to the file names.

Note that all of the scripts in this section actually use SAC, saclst and/or IASP91 to do the core operations; therefore make sure that the SAC, saclst and IASP91 packages are installed on your system, and that all the environment variables are set properly before running these scripts.

11.2.2 process_syn.pl

This script converts the synthetic output from the SEM code from ASCII to SAC format, and performs similar operations as 'process_data.pl'. An example of the usage of the script:

```
process_syn.pl -m CMTSOLUTION -h -a STATIONS -s 1.0 -l 0/4000 -f -t 40/500 -p -x bp SEM/*.MX?.sem
```

which will convolve the synthetics with a triangular source-time function from the CMTSOLUTION file, convert the synthetics into SAC format, add event and station information into the SAC headers, resample the SAC files with a sampling rate of 1 seconds, cut them between 0 and 4000 seconds, filter them between 40 and 500 seconds with the same filter used for the observed data, pick the first P and S arrivals, and add the suffix 'bp' to the file names.

More options are available for this script, such as adding a time shift to the origin time of the synthetics, convolving the synthetics with a triangular source time function with a given half duration, etc. Type process_syn.pl without any argument for detailed usage.

11.2.3 rotate.pl

To rotate the horizontal components of both the data and the synthetics (i.e., MXN and MXE) to the transverse and radial directions (i.e., MXT and MXR), use rotate.pl:

```
rotate.pl -l 0 -L 4000 -d DATA/*.LHE.SAC.bp
rotate.pl -l 0 -L 4000 SEM/*.MXE.sem.sac.bp
```

where the first command performs rotation on the SAC data obtained through IRIS (which may have timing information written in the filename), while the second command rotates the processed synthetics.

For synthetics, another (simpler) option is to set flag ROTATE_SEISMOGRAMS_RT to .true. in the parameter file DATA/Par_file.

11.2.4 clean_sac_headers_after_crash.sh

Note: You need to have the sismoutil-0.9b package installed on your computer if you want to run this script on binary SAC files. The software is available via the ORFEUS web site (www.orfeus-eu.org).

In case the simulation crashes during run-time without computing and writing all time steps, the SAC files (if flags OUTPUT_SEISMOS_SAC_ALPHANUM or OUTPUT_SEISMOS_SAC_BINARY have been set to .true.) are corrupted and cannot be used in SAC. If the simulation ran long enough so that the synthetic data may still be of use, you can run the script called clean_sac_headers_after_crash.sh (located in the UTILS directory) on the SAC files to correct the header variable NPTS to the actually written number of time steps. The script must be called from the SPECFEM3D main directory, and the input argument to this script is simply a list of SAC seismogram files.

11.3 Map Local Database

A sample program `remap_database` is provided to map the local database from a set of machines to another set of machines. This is especially useful when you want to run mesher and solver, or different types of solvers separately through a scheduler (refer to Chapter 9).

```
run_lsf.bash --gm-no-shmem --gm-copy-env remap_database
old_machines 150 [old_jobid new_jobid]
```

where `old_machines` is the LSF machine file used in the previous simulation, and `150` is the number of processors in total. Note that you need to supply `old_jobid` and `new_jobid(%J)` which are the LSF job-IDs for the old and new run if your databases are stored in a sub-directory named after the jobid on the scratch disk.

Bug Reports and Suggestions for Improvements

To report bugs or suggest improvements to the code, please send an e-mail to the CIG Computational Seismology Mailing List (`cig-seismo@geodynamics.org`).

Notes and Acknowledgements

In order to keep the software package thread-safe in case a multithreaded implementation of MPI is used, developers should not add modules or common blocks to the source code but rather use regular subroutine arguments (which can be grouped in "derived types" if needed for clarity).

The Gauss-Lobatto-Legendre subroutines in `gll_library.f90` are based in part on software libraries from the Massachusetts Institute of Technology, Department of Mechanical Engineering (Cambridge, Massachusetts, USA). The non-structured global numbering software was provided by Paul F. Fischer (Brown University, Providence, Rhode Island, USA, now at Argonne National Laboratory, USA).

OpenDX (`www.opendx.org`) is open-source based on IBM Data Explorer, AVS (`www.avs.com`) is a trademark of Advanced Visualization Systems, and ParaView (`www.paraview.org`) is an open-source visualization platform.

Please e-mail your feedback, questions, comments, and suggestions to the CIG Computational Seismology Mailing List (`cig-seismo@geodynamics.org`).

Copyright

Evolution of the code:

v. 6.0, Daniel Peter (ETH Zürich, Switzerland), Dimitri Komatitsch and Zhinan Xie (CNRS / University of Marseille, France), Elliott Sales de Andrade (University of Toronto, Canada), and many others, in particular from Princeton University, USA, April 2014: more flexible MPI implementation, GPU support, exact undoing of attenuation, LDDRK4-6 higher-order time scheme, etc...

v. 5.1, Dimitri Komatitsch, University of Toulouse, France and Ebru Bozdag, Princeton University, USA, February 2011: non blocking MPI for much better scaling on large clusters; new convention for the name of seismograms, to conform to the IRIS standard; new directory structure.

v. 5.0, many developers, February 2010: new Moho mesh stretching honoring crust2.0 Moho depths, new attenuation assignment, new SAC headers, new general crustal models, faster performance due to Deville routines and enhanced loop unrolling, slight changes in code structure (see also trivia at program start).

v. 4.0 David Michéa and Dimitri Komatitsch, University of Pau, France, February 2008: new doubling brick in the mesh, new perfectly load-balanced mesh, more flexible routines for mesh design, new inflated central cube with optimized shape, far fewer mesh files saved by the mesher, global arrays sorted to speed up the simulation, seismograms can be written by the master, one more doubling level at the bottom of the outer core if needed (off by default).

v. 3.6 Many people, many affiliations, September 2006: adjoint and kernel calculations, fixed IASP91 model, added AK135F_NO_MUD and 1066a, fixed topography/bathymetry routine, new attenuation routines, faster and better I/Os on very large systems, many small improvements and bug fixes, new 'configure' script, new Pyre version, new user's manual etc..

v. 3.5 Dimitri Komatitsch, Brian Savage and Jeroen Tromp, Caltech, July 2004: any size of chunk, 3D attenuation, case of two chunks, more precise topography/bathymetry model, new Par_file structure.

v. 3.4 Dimitri Komatitsch and Jeroen Tromp, Caltech, August 2003: merged global and regional codes, no iterations in fluid, better movies.

v. 3.3 Dimitri Komatitsch, Caltech, September 2002: flexible mesh doubling in outer core, inlined code, OpenDX support.

v. 3.2 Jeroen Tromp, Caltech, July 2002: multiple sources and flexible PREM reading.

v. 3.1 Dimitri Komatitsch, Caltech, June 2002: vectorized loops in solver and merged central cube.

v. 3.0 Dimitri Komatitsch and Jeroen Tromp, Caltech, May 2002: ported to SGI and Compaq, double precision solver, more general anisotropy.

v. 2.3 Dimitri Komatitsch and Jeroen Tromp, Caltech, August 2001: gravity, rotation, oceans and 3-D models.

v. 2.2 Dimitri Komatitsch and Jeroen Tromp, Caltech, USA, March 2001: final MPI package.

v. 2.0 Dimitri Komatitsch, Harvard, USA, January 2000: MPI code for the globe.

v. 1.0 Dimitri Komatitsch, UNAM, Mexico, June 1999: first MPI code for a chunk.

Jeroen Tromp and Dimitri Komatitsch, Harvard, USA, July 1998: first chunk solver using OpenMP on a Sun machine.

Dimitri Komatitsch, IPG Paris, France, December 1996: first 3-D solver for the CM-5 Connection Machine, parallelized on 128 processors using Connection Machine Fortran.

Appendix A

Reference Frame Convention

The code uses the following convention for the Cartesian reference frame:

- the x axis points East
- the y axis points North
- the z axis points up

Note that this convention is different from both the **?** convention and the Harvard Centroid-Moment Tensor (CMT) convention. The Aki & Richards convention is

- the x axis points North
- the y axis points East
- the z axis points down

and the Harvard CMT convention is

- the x axis points South
- the y axis points East
- the z axis points up

Appendix B

Non-Dimensionalization Conventions

All physical parameters used are non-dimensionalized internally in the code in order to work in a reference Earth of radius 1. In Table B.1 in the right column are the values by which we divide the parameters of the left column internally to perform the calculations. The values output and saved by the code (seismograms, sensitivity kernels...) are then scaled back to the right physical dimensions before being saved.

quantity (units)	divided internally by
distance (m)	R_EARTH
time (s)	$1/\sqrt{\texttt{PI} \times \texttt{GRAV} \times \texttt{RHOAV}}$
density (kg/m^3)	RHOAV

Table B.1: Non-dimensionalization convention employed internally by the code. The constants R_EARTH (the radius of the Earth), PI (the number π), GRAV (the universal gravitational constant), and RHOAV (the Earth's average density) are defined in the constants.h file.

Appendix C

Benchmarks

?? carefully benchmarked the spectral-element simulations of global seismic waves against normal-mode seismograms. Version 4.0 of SPECFEM3D_GLOBE has been benchmarked again following the same procedure.

In this appendix we present two tests: a 'long-period' (periods longer than 17 s) simulation of a shallow event in isotropic PREM [**?**] without the ocean layer, without attenuation but including the effects of self-gravitation (in the Cowling approximation) (Figures C.1 and C.2), and a 'short-period' (periods longer than 9 s) simulation of a deep event in transversely isotropic PREM without the ocean layer and including the effects of self-gravitation and attenuation (Figures C.3, C.4 and C.5).

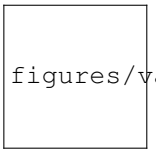


Figure C.1: Normal-mode (blue) and SEM (red) vertical displacements in isotropic PREM considering the effects of self-gravitation but not attenuation for 13 stations at increasing distance from the 1999 November 26th Vanuatu event located at 15 km depth. The SEM computation is accurate for periods longer than 17 s. The seismograms have been filtered between 50 s and 500 s. The station names are indicated on the left.

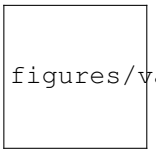


Figure C.2: Same as in Figure C.1 for the transverse displacements.

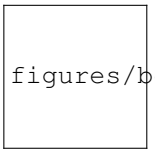


Figure C.3: Normal-mode (blue) and SEM (red) vertical displacements in transversely isotropic PREM considering the effects of self-gravitation and attenuation for 12 stations at increasing distance from the 1994 June 9th Bolivia event located at 647 km depth. The SEM computation is accurate for periods longer than 9 s. The seismograms have been filtered between 10 s and 500 s. The station names are indicated on the left.

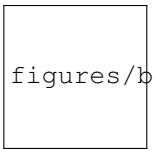


Figure C.4: Same as in Figure C.3 for the transverse displacements.

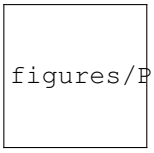


Figure C.5: Seismograms recorded between 130 degrees and 230 degrees, showing in particular the good agreement for core phases such as PKP. This figure is similar to Figure 24 of **?**. The results have been filtered between 15 s and 500 s.

The normal-mode synthetics are calculated with the code `QmXD` using mode catalogs with a shortest period of 8 s generated by the code `OBANI`. No free-air, tilt, or gravitational potential corrections were applied [**?**]. We also turned off the effect of the oceans in `QmXD`.

The normal-mode and SEM displacement seismograms are first calculated for a step source-time function, i.e., setting the parameter `half duration` in the `CMTSOLUTION` file to zero for the SEM simulations. Both sets of seismograms are subsequently convolved with a triangular source-time function using the processing script `UTILS/seis_process/process_syn.pl`. They are also band-pass filtered and the horizontal components are rotated to the radial and transverse directions (with the script `UTILS/seis_process/rotate.pl`).

The match between the normal-mode and SEM seismograms is quite remarkable for the experiment with attenuation, considering the very different implementations of attenuation in the two computations (e.g., frequency domain versus time domain, constant Q versus absorption bands).

Further tests can be found in the `EXAMPLES` directory. It contains the normal-mode and SEM seismograms, and the parameters (`STATIONS`, `CMTSOLUTION` and `Par_file`) for the SEM simulations.

Important remark: when comparing SEM results to normal mode results, one needs to convert source and receiver coordinates from geographic to geocentric coordinates, because on the equator the geographic and geocentric latitude are identical but not elsewhere. Even for spherically-symmetric simulations one must perform this conversion because the source and receiver locations provided by globalCMT.org and IRIS involve geographic coordinates.

Appendix D

SAC Headers

Information about the simulation (i.e., event/station information, sampling rate, etc.) is written in the header of the seismograms in SAC format. The list of values and related explanation are given in Figure D.1. Please check the SAC webpages (`www.iris.edu/software/sac/`) for further information. Please note that the reference time KZTIME is the centroid time ($t_{\mathrm{CMT}} = t_{\mathrm{PDE}} + \mathtt{time\ shift}$) which corresponds to zero time in the synthetics. For kinematic rupture simulations, KZTIME is equal to the CMT time of the source having the minimum time-shift in the `CMTSOLUTION` file, and coordinates, depth and half-duration of the event are not provided in the headers.

FILE: AAK.II.MXZ.sem.sac	
NPTS = 37200	number of points per data component
B = -2.250000e+00	beginning value of time array
E = 6.005389e+03	end value of time array
IFTYPE = TIME SERIES FILE	type of file
LEVEN = TRUE	TRUE if data is evenly spaced
DELTA = 1.615000e-01	sampling rate (s)
IDEP = DISPLACEMENT (NM)	type of seismograms*
DEPMIN = -5.038710e-07	minimum displacement value
DEPMAX = 5.296865e-07	maximum displacement value
DEPMEN = -6.698920e-10	mean displacement value
OMARKER = 0	reference time in synthetics
KZDATE = FEB 04 (035), 2010	event date
KZTIME = 17:48:15.599	event origin time (centroid time)
IZTYPE = EVENT ORIGIN TIME	reference time
KSTNM = AAK	station name
CMPAZ = 0.000000e+00	component azimuth (degrees clockwise from north)
CMPINC = 0.000000e+00	component incident angle (degrees from vertical)
STLA = 4.263900e+01	station latitude (degrees, north positive)
STLO = 7.449400e+01	station longitude (degrees, east positive)
STEL = 1.645000e+03	station elevation (meters)
STDP = 3.000000e+01	station depth below surface (meters)
KEVNM = 201002041748A	event name
EVLA = -1.950000e+01	event CMT latitude (degrees, north positive)
EVLO = -1.732400e+02	event CMT longitude (degrees, east positive)
EVDP = 2.500000e+01	event CMT depth (km)
IEVTYP = EARTHQUAKE	event type
DIST = 1.326017e+04	great circle distance between event and station (km)
AZ = 3.085366e+02	event to station azimuth (degrees)
BAZ = 9.023685e+01	station to event azimuth (backazimuth, degrees)
GCARC = 1.191897e+02	great circle distance between event and station (degrees)
LOVROK = TRUE	TRUE if it is ok to write the file on disk
USER0 = 1.500000e+00	source half-duration (s)
USER1 = 1.700000e+01	shortest period at which simulations are accurate (s)
USER2 = 5.000000e+02	longest period at which simulations are accurate (s)
KUSER0 = SEM	method used to compute synthetic seismograms
KUSER1 = v5.1.0	version of the SEM code
KUSER2 = Tiger	
NNVHDR = 6	header version number
SCALE = 1.000000e+09	scale factor to convert the unit of the synthetics from meters to nanometers
LPSPOL = TRUE	TRUE if station components have positive polarity
LCALDA = TRUE	TRUE if DIST, AZ, BAZ and GCARC are calculated from station and event coordinates
KCMPNM = MXZ	station component name
KNETWK = II	station network name

* the unit of synthetic seismograms is “meters”. Seismograms should be scaled by the header SCALE to obtain units of nanometers.

Figure D.1: List of SAC headers and their explanations for a sample seismogram.

Appendix E

Channel Codes of Seismograms

Seismic networks, such as the Global Seismographic Network (GSN), generally involve various types of instruments with different bandwidths, sampling properties and component configurations. There are standards to name channel codes depending on instrument properties. IRIS (`www.iris.edu`) uses SEED/FDSN format for channel codes, which are represented by three letters, such as `LHN`, `BHZ`, etc. In older versions of the SPECFEM package, a common format was used for the channel codes of all seismograms, which was `LHE/LHN/LHZ` for three components. To avoid confusion when comparisons are made to observed data, we are now using the FDSN convention for SEM seismograms. In the following, we give a brief explanation of the FDSN convention and how it is adopted in SEM seismograms. Please visit `www.iris.edu/manuals/SEED_appA.htm` for further information.

`Band code:` The first letter in the channel code denotes the band code of seismograms, which depends on the response band and the sampling rate of instruments. The list of band codes used by IRIS is shown in Figure E.1. The sampling rate of SEM synthetics is controlled by the resolution of simulations rather than instrument properties. However, for consistency, we follow the FDSN convention for SEM seismograms governed by their sampling rate. For SEM synthetics, we consider band codes for which $dt \leq 1$ s. The FDSN convention also considers the response band of instruments. For instance, short-period and broad-band seismograms with the same sampling rate correspond to different band codes, such as S and B, respectively. In such cases, we consider SEM seismograms as broad band, ignoring the corner period (≥ 10 s) of the response band of instruments (note that at these resolutions, the minimum period in the SEM synthetics will be less than 10 s). Accordingly, when you run a simulation the band code will be chosen depending on the resolution of the synthetics, and channel codes of SEM seismograms will start with either `L`, `M`, `B`, `H`, `C` or `F`, shown by red color in the figure.

Figure E.1: The FDSN band code convention is based on the sampling rate and the response band of instruments. Please visit `www.iris.edu/manuals/SEED_appA.htm` for further information. Grey rows show the relative band-code range in SPECFEM, and the band codes used to name SEM seismograms are denoted in red.

`Instrument code:` The second letter in the channel code corresponds to instrument codes, which specify the family to which the sensor belongs. For instance, `H` and `L` are used for high-gain and low-gain seismometers, respectively. The instrument code of SEM seismograms will always be `X`, as assigned by FDSN for synthetic seismograms.

`Orientation code:` The third letter in channel codes is an orientation code, which generally describes the physical configuration of the components of instrument packages. SPECFEM uses the traditional orientation code `E/N/Z` (East-West, North-South, Vertical) for three components.

`EXAMPLE:` Depending on the resolution of your simulations, if the sampling rate is greater than 0.1 s and less than 1 s,

a seismogram recorded on the vertical component of station `AAK` will be named `AAK.IU.MXZ.sem.sac`, whereas it will be `AAK.IU.BXZ.sem.sac`, if the sampling rate is greater than 0.0125 and less equal to 0.1 s.

Appendix F

Troubleshooting

FAQ

configuration fails: Examine the log file 'config.log'. It contains detailed informations. In many cases, the path's to these specific compiler commands F90, CC and MPIF90 won't be correct if './configure' fails.

Please make sure that you have a working installation of a Fortran compiler, a C compiler and an MPI implementation. You should be able to compile this little program code:

```
program main
include 'mpif.h'
integer, parameter ::  CUSTOM_MPI_TYPE = MPI_REAL
integer ier
call MPI_INIT(ier)
call MPI_BARRIER(MPI_COMM_WORLD,ier)
call MPI_FINALIZE(ier)
end
```

compilation fails: In case a compilation error like the following occurs, stating

```
...
obj/meshfem3D.o:  In function `MAIN__':
meshfem3D.f90:(.text+0x14):  undefined reference to `_gfortran_set_std'
...
```

make sure you're pointing to the right 'mpif90' wrapper command.

Normally, this message will appear when you are mixing two different Fortran compilers. That is, using e.g. gfortran to compile non-MPI files and mpif90, wrapper provided for e.g. ifort, to compile MPI-files.

fix: e.g. specify > ./configure FC=gfortran MPIF90=/usr/local/openmpi-gfortran/bin/mpif90

changing PPM model routines fails: In case you want to modify the PPM-routines in file `model_ppm.f90`, please consider the following points:

1. Please check in file `get_model_parameter.f90` that the entry for PPM models looks like:

   ```
   ...
   else if(MODEL_ROOT == 'PPM') then
   !  overimposed based on isotropic-prem
   CASE_3D = .true.
   CRUSTAL = .true.
   ISOTROPIC_3D_MANTLE = .true.
   ONE_CRUST = .true.
   THREE_D_MODEL = THREE_D_MODEL_PPM
   TRANSVERSE_ISOTROPY = .true.  !  to use transverse-isotropic prem
   ...
   ```

   You can set `TRANSVERSE_ISOTROPY` to `.false.` in case you want to use the isotropic PREM as 1D background model.

2. Transverse isotropy would mean different values for horizontal and vertically polarized wave speeds, i.e. different for vph and vpv, vsh and vsv, and it includes an additional parameter eta. By default, we take these wave speeds from PREM and add your model perturbations to them. For the moment, your model perturbations are added as isotropic perturbations, using the same dvp for vph and vpv, and dvs for vsh and vsv, see in `meshfem3D_models.f90`:

```
...
case(THREE_D_MODEL_PPM )
!  point profile model
call model_PPM(r_used,theta,phi,dvs,dvp,drho,PPM_V)
vpv=vpv*(1.0d0+dvp)
vph=vph*(1.0d0+dvp)
vsv=vsv*(1.0d0+dvs)
vsh=vsh*(1.0d0+dvs)
rho=rho*(1.0d0+drho)
...
```

   You could modify this to add different perturbations for vph and vpv, resp. vsh and vsv. This would basically mean that you add transverse isotropic perturbations. You can see how this is done with e.g. the model `s362ani`, following the flag `THREE_D_MODEL_S362ANI` on how to modify accordingly the file `meshfem3D_models.f90`.

3. In case you modify the structure `PPM_V` in file `model_ppm.f90`, also check that in file `meshfem3D_models.f90`, you use your updated `PPM_V` structure (for example including dvp):

```
...
!  point profile model_variables
type model_ppm_variables
sequence
double precision,dimension(:),pointer ::  dvs,dvp,lat,lon,depth
double precision ::  maxlat,maxlon,minlat,minlon,maxdepth,mindepth
double precision ::  dlat,dlon,ddepth,max_dvs,min_dvs,max_dvp,min_dvp
integer ::  num_v,num_latperlon,num_lonperdepth
integer ::  dummy !  padding 4 bytes to align the structure
end type model_ppm_variables
type (model_ppm_variables) PPM_V
...
```

   Together with these modifications in file `meshfem3D_models.f90`, your code should work fine. In case you want to add more specific model routines, follow the code sections starting with:

```
!--
!
!  ADD YOUR MODEL HERE
!
!--
```

   to see code sections sensitive to model updates.

Appendix G

License

GNU GENERAL PUBLIC LICENSE Version 2, June 1991. Copyright (C) 1989, 1991 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software – to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. Copyright the software, and

2. Offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program" below refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification.") Each licensee is addressed as "you."

   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

   These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

   Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

   In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

(a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

(c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version," you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found. For example:

> One line to give the program's name and a brief idea of what it does. Copyright © (year) (name of author)
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright © year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’. This is free software, and you are welcome to redistribute it under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than ‘show w’ and ‘show c’; they could even be mouse-clicks or menu items – whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a “copyright disclaimer” for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program ‘Gnomovision’ (which makes passes at compilers) written by James Hacker.
>
> (signature of Ty Coon)
> 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.