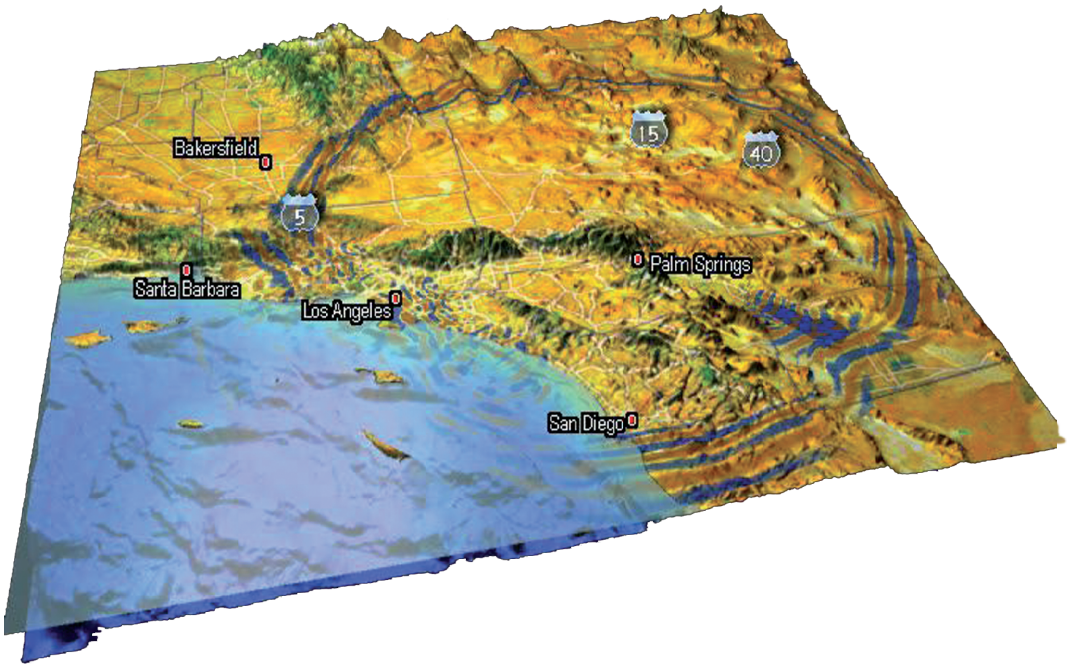
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)
PRINCETON UNIVERSITY (USA)
CNRS and UNIVERSITY OF MARSEILLE (FRANCE)
ETH ZÜRICH (SWITZERLAND)

SPECFEM 3D Cartesian

User Manual
Version 2.1



SPECFEM3D Cartesian
User Manual

Version 2.1

October 7, 2014

Authors

The SPECFEM3D Cartesian 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 April 4, 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 Legendre spectral-element method with diagonal mass matrix that he had used for other equations. We are deeply indebted and thankful to him for that. That followed a visit by Dimitri Komatitsch to OGS (Istituto Nazionale di Oceanografia e di Geofisica Sperimentale) in Trieste, Italy, in February 1995 to meet with Géza Seriani and Enrico Priolo, who introduced him to their 2D Chebyshev version of the spectral-element method with a non-diagonal mass matrix. We are deeply indebted and thankful to them for that.

Since then it has been developed and maintained by a development team: in alphabetical order, Jean-Paul (Pablo) Ampuero, Kangchen Bai, Piero Basini, Céline Blitz, Ebru Bozdağ, Emanuele Casarotti, Joseph Charles, Min Chen, Percy Galvez, Dominik Göddeke, Vala Hjörleifsdóttir, Sue Kientz, Dimitri Komatitsch, Jesús Labarta, Nicolas Le Goff, Pieyre Le Loher, Matthieu Lefebvre, Qinya Liu, Yang Luo, Alessia Maggi, Federica Magnoni, Roland Martin, René Matzen, Dennis McRitchie, Matthias Meschede, Peter Messmer, David Michéa, Surendra Nadh Somala, 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 Cartesian simulates seismic wave propagation at the local or regional scale 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 [**?????**].

For a detailed introduction to the SEM as applied to regional seismic wave propagation, please consult **?????** and in particular **??????**. 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, topography and bathymetry are included. The package can accommodate full 21-parameter anisotropy (see **?**) as well as lateral variations in attenuation [**?**]. Adjoint capabilities and finite-frequency kernel simulations are 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 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 won the Gordon Bell award for best performance at the SuperComputing 2003 conference in Phoenix, Arizona (USA) (see **?** and `www.sc-conference.org/sc2003/nrfinalaward.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).

This new release of the code (version 2.1) includes Convolution or Auxiliary Differential Equation Perfectly Matched absorbing Layers (C-PML or ADE-PML) [**?????**]. It also includes support for GPU graphics card acceleration [**????**].

The next release of the code will use the PT-SCOTCH parallel and threaded version of SCOTCH for mesh partitioning instead of the serial version.

SPECFEM3D Cartesian includes coupled fluid-solid domains and adjoint capabilities, which enables one to address seismological inverse problems, but for linear rheologies only so far. To accommodate visco-plastic or non-linear rheologies, the reader can refer to the GeoELSEsoftware package [**??**].

1.1 Citation

If you use SPECFEM3D Cartesian for your own research, please cite at least one of the following articles:

Numerical simulations in general

Forward and adjoint simulations are described in detail in **??????????????** or **?**. Additional aspects of adjoint simulations are described in **??????**. Domain decomposition is explained in detail in **?**, and excellent scaling up to 150,000 processor cores is shown for instance in **?????**,

GPU computing

Computing on GPU graphics cards for acoustic or seismic wave propagation applications is described in detail in **????**.

If you use this new version 2.1, which has non blocking MPI for much better performance for medium or large runs, please cite at least one of these six articles, in which results of non blocking MPI runs are presented: **??????**.

If you work on geophysical applications, you may be interested in citing some of these application articles as well, among others:

Southern California simulations

???.

If you use the 3D southern California model, please cite **?** (Los Angeles model), **?** (Salton Trough), and **?** (southern California). The Moho map was determined by **?**. The 1D SoCal model was developed by **?**.

Anisotropy

??????.

Attenuation

???.

Topography

?????.

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 USA 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 USA National Science Foundation, CNRS, INRIA, ANR or the European Marie Curie program.

Chapter 2

Getting Started

To download the SPECFEM3D_Cartesian software package, type this:

```
git clone -recursive -branch devel
https://github.com/geodynamics/specfem3d.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
```

Note that MPI must be installed with MPI-IO enabled because parts of SPECFEM3D perform I/Os through MPI-IO.

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. **In the case of that compiler, we have noticed that versions dot zero sometimes have bugs or issues that can lead to wrong results when running the code, thus we *strongly* recommend using versions dot one or above (for instance version 13.1 instead of 13.0, version 14.1 instead of 14.0 and so on)**.

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

`Mac OS` You will probably need to install `XCODE`. In addition, the `clock_gettime` routine, which is used by the `SCOTCH` library that we use, does not exist in Mac OS. You may need to replace it with `clock_get_time` if you want to use `SCOTCH`.

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"
-with-scotch-dir=....
```

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

You can add -DFORCE_VECTORIZATION to the compiler options in `flags.guess` (for all compilers, except for IBM for which the syntax is -WF,-DFORCE_VECTORIZATION) to speed up the code in the fluid (acoustic) parts (only; FORCE_VECTORIZATION support for elastic parts has been discontinued in the source code). This works fine

if (and only if) your computer always allocates a contiguous memory block for each allocatable array; this is the case for most machines and most compilers, but not all. For more details see https://github.com/geodynamics/specfem3d/issues/81 . To check if that option works fine on your machine, run the code with and without it for a model containing a significant fluid layer (or entirely fluid) and make sure the seismograms are identical.

Note that we use CUBIT (now called Trelis) to create meshes of hexahedra, but other packages can be used as well, for instance GiD from `http://gid.cimne.upc.es` or Gmsh from `http://geuz.org/gmsh` [**?**]. Even mesh creation packages that generate tetrahedra, for instance TetGen from `http://tetgen.berlios.de`, can be used because each tetrahedron can then easily be decomposed into four hexahedra as shown in the picture of the TetGen logo at `http://tetgen.berlios.de/figs/Delaunay-Voronoi-3D.gif`; while this approach does not generate hexahedra of optimal quality, it can ease mesh creation in some situations and it has been shown that the spectral-element method can very accurately handle distorted mesh elements [**?**].

The SPECFEM3D Cartesian software package relies on the SCOTCH library to partition meshes created with CUBIT. METIS [**???**] can also be used instead of SCOTCH if you prefer, by editing file src/decompose_mesh/decompose_mesh.F90 and uncommenting the flag USE_METIS_INSTEAD_OF_SCOTCH. You will also then need to install and compile Metis version 4.0 (do *NOT* install Metis version 5.0, which has incompatible function calls) and edit `src/decompose_mesh/Make` and uncomment the METIS link flag in that file before running `configure`.

The SCOTCH library [**?**] provides efficent static mapping, graph and mesh partitioning routines. SCOTCH is a free software package developed by François Pellegrini et al. from LaBRI and INRIA in Bordeaux, France, downloadable from the web page `https://gforge.inria.fr/projects/scotch/`. In case no SCOTCH libraries can be found on the system, the configuration will bundle the version provided with the source code for compilation. The path to an existing SCOTCH installation can to be set explicitly with the option `--with-scotch-dir`. Just as an example:

```
./configure FC=ifort MPIFC=mpif90 --with-scotch-dir=/opt/scotch
```

If you use the Intel ifort compiler to compile the code, we recommend that you use the Intel icc C compiler to compile Scotch, i.e., use:

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

When compiling the SCOTCH source code, if you get a message such as: "ld: cannot find -lz", the Zlib compression development library is probably missing on your machine and you will need to install it or ask your system administrator to do so. On Linux machines the package is often called "zlib1g-dev" or similar. (thus "sudo apt-get install zlib1g-dev" would install it)

To compile a serial version of the code for small meshes that fits on one compute node and can therefore be run serially, run `configure` with the `--without-mpi` option to suppress all calls to MPI.

A summary of the most important configuration variables follows.

`F90` Path to the Fortran compiler.

`MPIF90` Path to MPI Fortran.

`MPI_FLAGS` Some systems require this flag to link to MPI libraries.

`FLAGS_CHECK` Compiler flags.

The configuration script automatically creates for each executable a corresponding `Makefile` in the `src/` subdirectoy. The `Makefile` contains a number of suggested entries for various compilers, e.g., Portland, Intel, Absoft, NAG, and Lahey. 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. Select the compiler you wish to use on your system and choose the related optimization flags. Note that the default flags in the `Makefile` are undoubtedly not optimal for your system, so we encourage you to experiment with these flags and to solicit advice from your systems administrator. Selecting the right compiler and optimization flags can make a tremendous difference in terms of performance. We welcome feedback on your experience with various compilers and flags.

Now that you have set the compiler information, you need to select a number of flags in the `constants.h` file depending on your system:

LOCAL_PATH_IS_ALSO_GLOBAL Set to `.false.` on most cluster applications. For reasons of speed, the (parallel) distributed database generator typically writes a (parallel) database for the solver on the local disks of the compute nodes. Some systems have no local disks, e.g., BlueGene or the Earth Simulator, and other systems have a fast parallel file system, in which case this flag should be set to `.true.`. Note that this flag is not used by the database generator or the solver; it is only used for some of the post-processing.

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. Select your preference by selecting the appropriate setting in the `constants.h` file:

CUSTOM_REAL Set to `SIZE_REAL` for single precision and `SIZE_DOUBLE` for double precision.

In the `precision.h` file:

CUSTOM_MPI_TYPE Set to `MPI_REAL` for single precision and `MPI_DOUBLE_PRECISION` for double precision.

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.

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.

2.1 Adding OpenMP support in addition to MPI

NOTE FROM JULY 2013: OpenMP support is maybe / probably not maintained any more. Thus the section below is maybe obsolete.

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.

2.2 Compiling on an IBM BlueGene

More recent installation instruction for IBM BlueGene, from April 2013:

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
```

In order for the SCOTCH domain decomposer to compile, on some (but not all) Blue Gene systems you may need to run `configure` with `CC=gcc` instead of `CC=bgxlc_r`.
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`. Also, `AR=ar, ARFLAGS=cru` and `RANLIB=ranlib` are hardwired in all `Makefile.in` files by default, but to cross-compile on BlueGene/P one needs to change these values to `AR=bgar, ARFLAGS=cru` and `RANLIB=bgranlib`. Thus the easiest thing to do is to modify all `Makefile.in` files and the `configure` script to set them automatically by `configure`. 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 AR=bgar ARFLAGS=cru
RANLIB=bgranlib 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 Visualizing the subroutine calling tree of the source code

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

To do your own call graphs, you can follow these simple steps below.

0. Install `Doxygen` and `graphviz` (the two are usually in the package manager of classic Linux distribution).
1. Run in the terminal : `doxygen -g`, which creates a `Doxyfile` that tells doxygen what you want it to do.
2. Edit the Doxyfile. Two Doxyfile-type files have been already committed in the directory `specfem3d/doc/Call_trees`:
   - `Doxyfile_truncated_call_tree` will generate call graphs with maximum 3 or 4 levels of tree structure,
   - `Doxyfile_complete_call_tree` will generate call graphs with complete tree structure.

The important entries in the Doxyfile are:

PROJECT_NAME

OPTIMIZE_FOR_FORTRAN Set to YES

EXTRACT_ALL Set to YES

EXTRACT_PRIVATE Set to YES

EXTRACT_STATIC Set to YES

INPUT From the directory `specfem3d/doc/Call_trees`, it is `"../../src/"`

FILE_PATTERNS In SPECFEM case, it is `*.f90* *.F90* *.c* *.cu* *.h*`

HAVE_DOT Set to YES

CALL_GRAPH Set to YES

CALLER_GRAPH Set to YES

DOT_PATH The path where is located the dot program graphviz (if it is not in your $PATH)

RECURSIVE This tag can be used to turn specify whether or not subdirectories should be searched for input files as well. In the case of SPECFEM, set to YES.

EXCLUDE Here, you can exclude:
`"../../src/specfem3D/older_not_maintained_partial_OpenMP_port"`
`"../../src/decompose_mesh/scotch"`
`"../../src/decompose_mesh/scotch_5.1.12b"`

DOT_GRAPH_MAX_NODES to set the maximum number of nodes that will be shown in the graph. If the number of nodes in a graph becomes larger than this value, doxygen will truncate the graph, which is visualized by representing a node as a red box. Minimum value: 0, maximum value: 10000, default value: 50.

MAX_DOT_GRAPH_DEPTH to set the maximum depth of the graphs generated by dot. A depth value of 3 means that only nodes reachable from the root by following a path via at most 3 edges will be shown. Using a depth of 0 means no depth restriction. Minimum value: 0, maximum value: 1000, default value: 0.

3. Run : `doxygen Doxyfile`, HTML and LaTeX files created by default in `html` and `latex` subdirectories.
4. To see the call trees, you have to open the file `html/index.html` in your browser. You will have many informations about each subroutines of SPECFEM (not only call graphs), you can click on every boxes / subroutines. It show you the call, and, the caller graph of each subroutine : the subroutines called by the concerned subroutine, and the previous subroutines who call this subroutine (the previous path), respectively. In the case of a truncated calling tree, the boxes with a red border indicates a node that has more arrows than are shown (in other words: the graph is truncated with respect to this node).

Finally, some useful links:

- a good and short summary for the basic utilisation of Doxygen:
  `http://www.softeng.rl.ac.uk/blog/2010/jan/30/callgraph-fortran-doxygen/` ,
- to configure the diagrams :
  `http://www.stack.nl/ dimitri/doxygen/manual/diagrams.html` ,
- the complete alphabetical index of the tags in Doxyfile:
  `http://www.stack.nl/ dimitri/doxygen/manual/config.html` ,
- more generally, the Doxygen manual:
  `http://www.stack.nl/ dimitri/doxygen/manual/index.html` .

2.4 Using the ADIOS library for I/O

Regular POSIX I/O can be problematic when dealing with large simulations one large clusters (typically more than $10,000$ processes). SPECFEM3D use the ADIOS library **?** to deal transparently take advantage of advanced paralllel file system features. To enable ADIOS, the following steps should be done:

1. Install ADIOS (available from `https://www.olcf.ornl.gov/center-projects/adios/`). Make sure that your environment variables reference it.
2. You may want to change ADIOS related values in the `constants.h.in` file. The default values probably suit most cases.
3. Configure using the `-with-adios` flag.

ADIOS is currently only usable for meshfem3D generated mesh (i.e. not for meshes generated with CUBTI). Additional control parameters are discussed in section 4.1.

2.5 Becoming a developer of the code, or making small modifications in the source code

If you want to develop new features in the code, and/or if you want to make small changes, improvements, or bug fixes, you are very welcome to contribute. To do so, i.e. to access the development branch of the source code with read/write access (in a safe way, no need to worry too much about breaking the package, there is a robot called BuildBot that is in charge of checking and validating all new contributions and changes), please visit this Web page: `https://github.com/geodynamics/specfem3d/wiki/Using-Hub`.

Chapter 3

Mesh Generation

The first step in running a spectral-element simulation consists of constructing a high-quality mesh for the region under consideration. We provide two possibilities to do so: (1) relying on the external, hexahedral mesher CUBIT, or (2) using the provided, internal mesher `xmeshfem3D`. In the following, we explain these two approaches.

3.1 Meshing with `CUBIT`

CUBIT is a meshing tool suite for the creation of finite-element meshes for arbitrarily shaped models. It has been developed and maintained at Sandia National Laboratories and can be purchased for a small academic institutional fee at `http://cubit.sandia.gov`. Our experience showed that using CUBIT greatly facilitates and speeds up the generation and preparation of hexahedral, conforming meshes for a variety of geophysical models with increasing complexity.

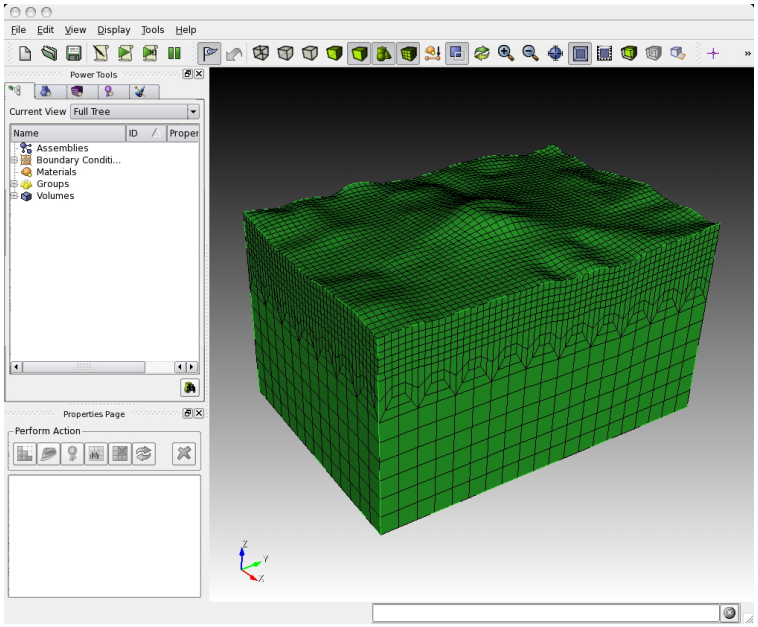


Figure 3.1: Example of the graphical user interface of CUBIT. The hexahedral mesh shown in the main display consists of a hexahedral discretization of a single volume with topography.

The basic steps in creating a load-balanced, partitioned mesh with CUBIT are:

1. setting up a hexahedral mesh with CUBIT,

2. exporting the CUBIT mesh into a SPECFEM3D Cartesian file format and

3. partitioning the SPECFEM3D Cartesian mesh files for a chosen number of cores.

Examples are provided in the SPECFEM3D Cartesian package in the subdirectory `EXAMPLES/`. We strongly encourage you to contribute your own example to this package by contacting the CIG Computational Seismology Mailing List (`cig-seismo@geodynamics.org`).

3.1.1 Creating the Mesh with CUBIT

For the installation and handling of the CUBIT meshing tool suite, please refer to the CUBIT user manual and documentation. In order to give you a basic understanding of how to use CUBIT for our purposes, examples are provided in the SPECFEM3D Cartesian package in the subdirectory `EXAMPLES/`:

`homogeneous_halfspace` Creates a single block model and assigns elastic material parameters.

`layered_halfspace` Combines two different, elastic material volumes and creates a refinement layer between the two. This example can be compared for validation against the solutions provided in subdirectory `VALIDATION_3D_SEM_SIMPLER_LAYER_SOURCE_DEPTH/`.

`waterlayered_halfspace` Combines an acoustic and elastic material volume as in a schematic marine survey example.

`tomographic_model` Creates a single block model whose material properties will have to be read in from a tomographic model file during the databases creation by `xgenerate_databases`.

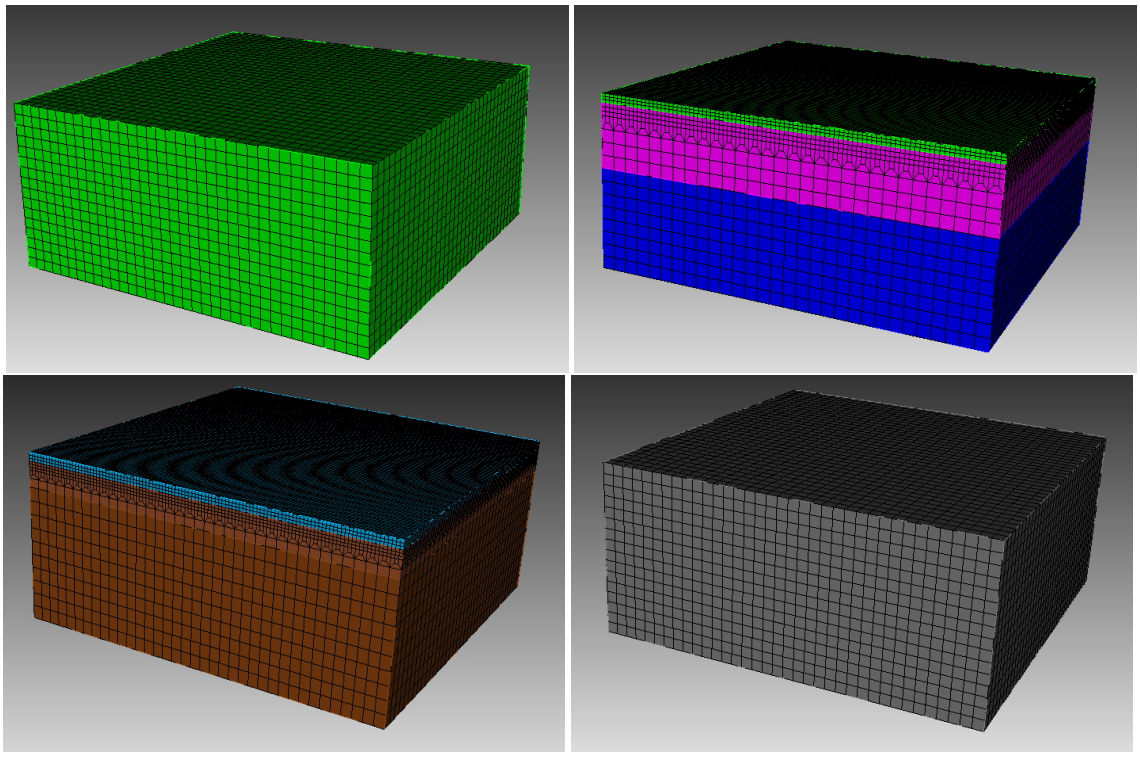


Figure 3.2: Screenshots of the CUBIT examples provided in subdirectory `EXAMPLES/`: homogeneous halfspace (top-left), layered halfspace (top-right), water layered halfspace (bottom-left) and tomographic model (bottom-right).

In each example subdirectory you will find a `README` file, which explains in a step-by-step tutorial the workflow for the example. Please feel free to contribute your own example to this package by contacting the CIG Computational Seismology Mailing List (`cig-seismo@geodynamics.org`).

In some cases, to re-create the meshes for the examples given, just type "claro ./create_mesh.py" or similar from the command line ("claro" is the command to run CUBIT from the command line).

IMPORTANT note: in order to correctly set up GEOCUBIT and run the examples, please read the file called "EXAMPLES/README"; in particular, please make sure you correctly set up the Python paths as indicated in that file.

3.1.2 Exporting the Mesh with `run_cubit2specfem3d.py`

Once the geometric model volumes in CUBIT are meshed, you prepare the model for exportation with the definition of material blocks and boundary surfaces. Thus, prior to exporting the mesh, you need to define blocks specifying the materials and absorbing boundaries in CUBIT. This process could be done automatically using the script `run_cubit2specfem3d.py` if the mesh meets some conditions or manually, following the block convention:

material_name Each material should have a specific block defined by a unique name. The name convention of the material is to start with either 'elastic' or 'acoustic'. It must be then followed by a unique identifier, e.g. 'elastic 1', 'elastic 2', etc. The additional attributes to the block define the material description.

For an elastic material:

material_id An integer value which is unique for this material.

Vp P-wave speed of the material (given in m/s).

Vs S-wave speed of the material (given in m/s).

rho density of the material (given in kg/m^3).

Q quality factor to use in case of a simulation with attenuation turned on. It should be between 1 and 9000. In case no attenuation information is available, it can be set to zero. Please note that your Vp- and Vs-speeds are given for a reference frequency. To change this reference frequency, you change the value of `ATTENUATION_f0_REFERENCE` in the main constants file `constants.h` found in subdirectory `src/shared/`.

anisotropic_flag Flag describing the anisotropic model to use in case an anisotropic simulation should be conducted. See the file `model_aniso.f90` in subdirectory `src/generate_databases/` for an implementation of the anisotropic models. In case no anisotropy is available, it can be set to zero.

Note that this material block has to be defined using all the volumes which belong to this elastic material. For volumes belonging to another, different material, you will need to define a new material block.

For an acoustic material:

material_id An integer value which is unique for this material.

Vp P-wave speed of the material (given in m/s).

0 S-wave speed of the material is ignored.

rho density of the material (given in kg/m^3).

face_topo Block definition for the surface which defines the free surface (which can have topography). The name of this block must be 'face_topo', the block has to be defined using all the surfaces which constitute the complete free surface of the model.

face_abs_xmin Block definition for the faces on the absorbing boundaries, one block for each surface with x=Xmin.

face_abs_xmax Block definition for the faces on the absorbing boundaries, one block for each surface with x=Xmax.

face_abs_ymin Block definition for the faces on the absorbing boundaries, one block for each surface with y=Ymin.

face_abs_ymax Block definition for the faces on the absorbing boundaries, one block for each surface with y=Ymax.

face_abs_bottom Block definition for the faces on the absorbing boundaries, one block for each surface with z=bottom.

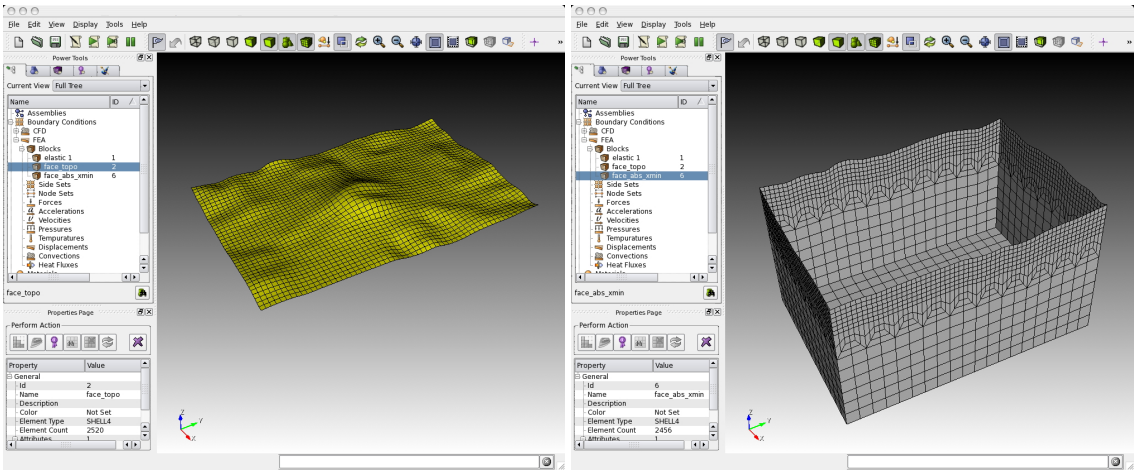


Figure 3.3: Example of the block definitions for the free surface 'face_topo' (left) and the absorbing boundaries, defined in a single block 'face_abs_xmin' (right) in CUBIT.

Optionally, instead of specifying for each surface at the boundaries a single block like mentioned above, you can also specify a single block for all boundary surfaces and name it as one of the absorbing blocks above, e.g. 'face_abs_xmin'.

After the block definitions are done, you export the mesh using the script `cubit2specfem3d.py` provided in each of the example directories (linked to the common script `CUBIT_GEOCUBIT/cubit2specfem3d.py`). If the export was successful, you should find the following files in a subdirectory `MESH/`:

absorbing_cpml_file (only needed in case of C-PML absorbing conditions) Contains on the first line the total number of C-PML spectral elements in the mesh, and then on the following line the list of all these C-PML elements with two numbers per line: first the spectral element number, and then a C-PML flag indicating to which C-PML layer(s) that element belongs, according to the following convention:

- Flag = 1 : element belongs to a X CPML layer only (either in Xmin or in Xmax),
- Flag = 2 : element belongs to a Y CPML layer only (either in Ymin or in Ymax),
- Flag = 3 : element belongs to a Z CPML layer only (either in Zmin or in Zmax),
- Flag = 4 : element belongs to a X CPML layer and also to a Y CPML layer,
- Flag = 5 : element belongs to a X CPML layer and also to a Z CPML layer,
- Flag = 6 : element belongs to a Y CPML layer and also to a Z CPML layer,
- Flag = 7 : element belongs to a X, to a Y and to a Z CPML layer, i.e., it belongs to a CPML corner.

Note that it does not matter whether an element belongs to a Xmin or to a Xmax CPML, the flag is the same in both cases; the same is true for Ymin and Ymax, and also Zmin and Zmax.

When you have an existing CUBIT (or similar) mesh stored in SPECFEM3D format, i.e., if you have existing "nodes_coords_file" and "mesh_file" files but do not know how to assign CPML flags to them, we have created a small serial Fortran program that will do that automatically for you, i.e., which will create the "absorbing_cpml_file" for you. That program is utils/CPML/convert_external_layers_of_a_given_mesh_to_CPML_layers.f90, and a small Makefile is provided in that directory (utils/CPML).

IMPORTANT: it is your responsibility to make sure that in the input CUBIT (or similar) mesh that this code will read in SPECFEM3D format from files "nodes_coords_file" and "mesh_file" you have created layers of elements that constitute a layer of constant thickness aligned with the coordinate grid axes (X, Y and/or Z), so that this code can assign CPML flags to them. This code does NOT check that (because it cannot, in any easy way). The mesh inside these CPML layers does not need to be structured nor regular, any non-structured mesh is fine as long as it has flat PML inner and outer faces, parallel to the axes, and thus of a constant thickness. The

thickness can be different for the X, Y and Z sides. But for X it must not vary, for Y it must not vary, and for Z it must not vary. If you do not know the exact thickness, you can use a slightly LARGER value in this code (say 2% to 5% more) and this code will fix that and will adjust it; never use a SMALLER value otherwise this code will miss some CPML elements.

Note: in a future release we will remove the constraint of having CPML layers aligned with the coordinate axes; we will allow for meshes that are titled by any constant angle in the horizontal plane. However this is not implemented yet.

Note: in the case of fluid-solid layers in contact, the fluid-solid interface currently needs to be flat and horizontal inside the CPML layer (i.e., bathymetry should become flat and horizontal when entering the CPML); this (small) constraint will probably remain in the code for a while because it makes fluid-solid matching inside the CPML much easier.

materials_file Contains the material associations for each element. The format is:

```
element_ID material_ID
```

where `element_ID` is the element identifier and `material_ID` is a unique identifier, positive (for materials taken from this list of materials, i.e. for which each spectral element has constant material properties taken from this list) or negative (for tomographic models, i.e. for spectral element whose real velocities and density will be assigned later by calling an external function to define model variations, for instance in the case of tomographic models; in such a case, material properties can vary inside each spectral element, i.e. be different at each of its Gauss-Lobatto-Legendre grid points).

nummaterial_velocity_file Defines the material properties.

- For classical materials (i.e., spectral elements for which the velocity and density model will not be assigned by calling an external function to define for instance a tomographic model), the format is:

  ```
  domain_ID material_ID rho vp vs Qkappa Qmu anisotropy_flag
  ```

  where **`domain_ID` is 1 for acoustic and 2 for elastic or viscoelastic materials,** `material_ID` a unique identifier, `rho` the density in $kg\,m^{-3}$, `vp` the P-wave speed in $m\,s^{-1}$, `vs` the S-wave speed in $m\,s^{-1}$, `Q` the quality factor and `anisotropy_flag` an identifier for anisotropic models. Note that the `Qkappa` value is ignored by the code unless `FULL_ATTENUATION_SOLID` is set. Note also that both `Qkappa` and `Qmu` are ignored by the code unless `ATTENUATION` is set. If you want a model with no `Qmu` attenuation, both set `ATTENUATION` to `.false.` in the `Par_file` and set `Qmu` to 9999 here. If you want a model with no `Qkappa` attenuation, both set `FULL_ATTENUATION_SOLID` to `.false.` in the `Par_file` and set `Qkappa` to 9999 here.
- For tomographic velocity models, please read Chapter 11 and Section 11.1 'Using external tomographic Earth models' for further details.

nodes_coords_file Contains the point locations in Cartesian coordinates of the mesh element corners.

mesh_file Contains the mesh element connectivity. The hexahedral elements can have 8 or 27 nodes.
See picture doc/mesh_numbering_convention/numbering_convention_27_nodes.jpg to see
in which (standard) order the points must be cited. In the case of 8 nodes, just include the first 8 points.

free_or_absorbing_surface_file_zmax Contains the free surface connectivity or
the surface connectivity of the absorbing boundary surface at the top (Zmax),
depending on whether the top surface is defined as free or absorbing (`STACEY_INSTEAD_OF_FREE_SURFACE` in `DATA/Par_file`).
You should put both the surface of acoustic regions and of elastic regions in that file; that is, list all the element faces that constitute the surface of the model in that file.

absorbing_surface_file_xmax Contains the surface connectivity of the absorbing boundary surface at Xmax
(also needed in the case of C-PML absorbing conditions, in order for the code to be able to impose Dirichlet conditions on their outer edge).

absorbing_surface_file_xmin Contains the surface connectivity of the absorbing boundary surface at Xmin (also needed in the case of C-PML absorbing conditions, in order for the code to be able to impose Dirichlet conditions on their outer edge).

absorbing_surface_file_ymax Contains the surface connectivity of the absorbing boundary surface at Ymax (also needed in the case of C-PML absorbing conditions, in order for the code to be able to impose Dirichlet conditions on their outer edge).

absorbing_surface_file_ymin Contains the surface connectivity of the absorbing boundary surface at Ymin (also needed in the case of C-PML absorbing conditions, in order for the code to be able to impose Dirichlet conditions on their outer edge).

absorbing_surface_file_bottom Contains the surface connectivity of the absorbing boundary surface at the bottom (Zmin)
(also needed in the case of C-PML absorbing conditions, in order for the code to be able to impose Dirichlet conditions on their outer edge).

These mesh files are needed as input files for the partitioner `xdecompose_mesh` to load-balance the mesh. Please see the next section for further details.

In directory "CUBIT_GEOCUBIT" we provide a script that can help doing the above tasks of exporting a CUBIT mesh to SPECFEM3D format automatically for you: "run_cubit2specfem3d.py". Just edit them to indicate the path to your local installation of CUBIT and also the name of the *.cub existing CUBIT mesh file that you want to export to SPECFEM3D format. These scripts will do the conversion for you automatically except assigning material properties to the different mesh layers. To do so, you will then need to edit the file called "nummaterial_velocity_file" that will have just been created and change it from the prototype created:

0 1 vol1 –> syntax: #material_domain_id #material_id #rho #vp #vs #Q_kappa #Q_mu #anisotropy 0 2 vol2 –> syntax: #material_domain_id #material_id #rho #vp #vs #Q_kappa #Q_mu #anisotropy

(where "vol1" and "vol2" here represent the volume labels that you have set while creating the mesh in CUBIT) to for instance

2 1 1500 2300 1800 9999.0 9999.0 0 2 2 1600 2500 20000 9999.0 9999.0 0

Checking the mesh quality

The quality of the mesh may be inspected more precisely based upon the serial code in the file `check_mesh_quality_CUBIT_Abaqus.f90` located in the directory `src/check_mesh_quality_CUBIT_Abaqus/`. Running this code is optional because no information needed by the solver is generated.

Prior to running and compiling this code, you have to export your mesh in CUBIT to an ABAQUS (.inp) format. For example, export mesh block IDs belonging to volumes in order to check the quality of the hexahedral elements. You also have to determine a number of parameters of your mesh, such as the number of nodes and number of elements and modify the header of the `check_mesh_quality_CUBIT_Abaqus.f90` source file. Then, in the main directory, type

```
make xcheck_mesh_quality_CUBIT_Abaqus
```

and use

```
./bin/xcheck_mesh_quality_CUBIT_Abaqus
```

to generate an OpenDX output file (`DX_mesh_quality.dx`) that can be used to investigate mesh quality, e.g. skewness of elements, and a Gnuplot histogram (`mesh_quality_histogram.txt`) that can be plotted with gnuplot (type '`gnuplot plot_mesh_quality_histogram.gnu`'). The histogram is also printed to the screen. Analyze that skewness histogram of mesh elements to make sure no element has a skewness above approximately 0.75, otherwise the mesh is of poor quality (and if even a single element has a skewness value above 0.80, then you must definitely improve the mesh). If you want to start designing your own meshes, this tool is useful for viewing your creations. You are striving for meshes with elements with 'cube-like' dimensions, e.g., the mesh should contain no very elongated or skewed elements.

3.1.3 Partitioning the Mesh with `xdecompose_mesh`

The SPECFEM3D Cartesian software package performs large scale simulations in a parallel 'Single Process Multiple Data' way. The spectral-element mesh created with CUBIT needs to be distributed on the processors. This partitioning is executed once and for all prior to the execution of the solver so it is referred to as a static mapping.

An efficient partitioning is important because it leverages the overall running time of the application. It amounts to balance the number of elements in each slice while minimizing the communication costs resulting from the placement of adjacent elements on different processors. `decompose_mesh` depends on the SCOTCH library [**?**], which provides efficent static mapping, graph and mesh partitioning routines. SCOTCH is a free software package developed by François Pellegrini et al. from LaBRI and INRIA in Bordeaux, France, downloadable from the web page `https://gforge.inria.fr/projects/scotch/`.

In most cases, the configuration with `./configure FC=ifort` should be sufficient. During the configuration process, the script tries to find existing SCOTCH installations. In case your system has no pre-existing SCOTCH installation, we provide the source code of SCOTCH, which is released open source under the French CeCILL-C version 1 license, in directory `src/decompose_mesh/scotch_5.1.12b`. This version gets bundled with the compilation of the SPECFEM3D Cartesian package if no libraries could have been found. If this automatic compilation of the SCOTCH libraries fails, please refer to file INSTALL.txt in that directory to see further details how to compile it on your system. In case you want to use a pre-existing installation, make sure you have correctly specified the path of the SCOTCH library when using the option `--with-scotch-dir` with the `./configure` script. In the future you should be able to find more recent versions at `http://www.labri.fr/perso/pelegrin/scotch/scotch_en.html` .

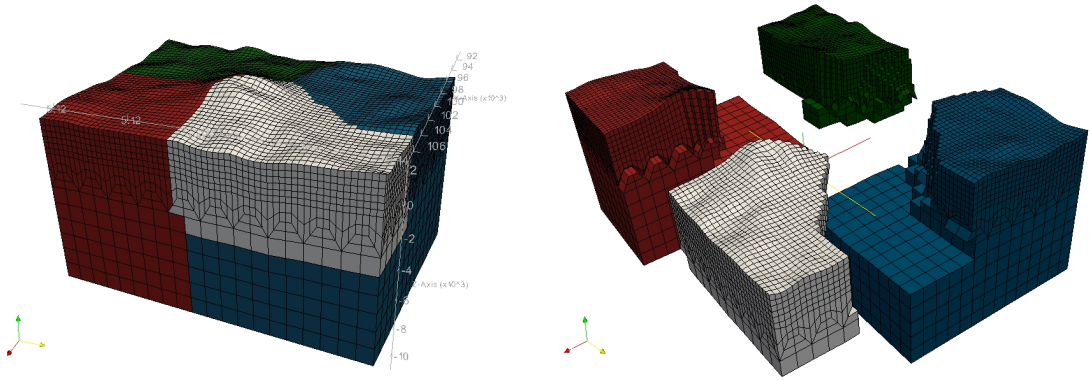


Figure 3.4: Example of a mesh partitioning onto four cores. Each single core partition is colored differently. The executable `xdecompose_mesh` can equally distribute the mesh on any arbitrary number of cores. Domain decomposition is explained in detail in **?**, and excellent scaling up to 150,000 processor cores in shown for instance in **?????**.

When you are ready to compile, in the main directory type

```
make xdecompose_mesh
```

If all paths and flags have been set correctly, the executable `bin/xdecompose_mesh` should be produced.

The partitioning is done in serial for now (in the next release we will provide a parallel version of that code). It needs to be run in the `bin/` directory because it expects the `../DATA/Par_file`. The synopsis is:

```
./xdecompose_mesh nparts input_directory output_directory
```

where

- `nparts` is the number of partitions, i.e., the number of cores for the parallel simulations,
- `input_directory` is the directory which holds all the files generated by the Python script `cubit2specfem3d.py` explained in the previous Section 3.1.2, e.g. `MESH/`, and

- `output_directory` is the directory for the output of this partitioner which stores ACII-format files named like `proc******_Database` for each partition. These files will be needed for creating the distributed databases, and have to reside in the directory `LOCAL_PATH` specified in the main `Par_file`, e.g. in directory `OUTPUT_FILES/DATABASES_MPI`. Please see Chapter 4 for further details.

Note that all the files generated by the Python script `cubit2specfem3d.py` must be placed in the `input_directory` folder before running the program.

3.2 Meshing with `xmeshfem3D`

In case you successfully ran the configuration script, you are also ready to compile the internal mesher. This is an alternative to CUBIT for the mesh generation of relatively simple geological models. The mesher is no longer dedicated to Southern California and more flexiblity is provided in this version of the package.

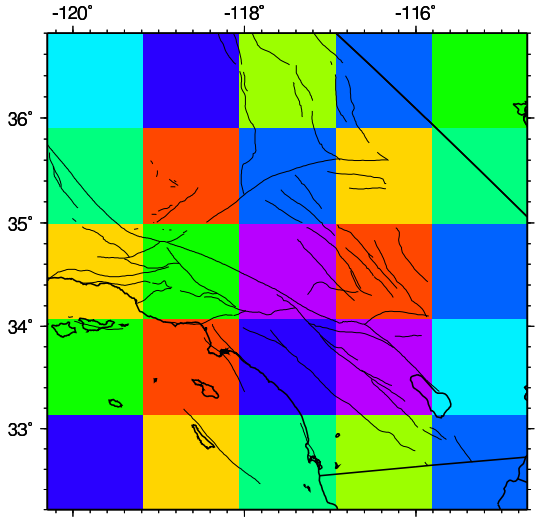


Figure 3.5: For parallel computing purposes, the model block is subdivided in `NPROC_XI` $\times$ `NPROC_ETA` slices of elements. In this example we use $5^2 = 25$ processors.

In the main directory, type

```
make xmeshfem3D
```

If all paths and flags have been set correctly, the mesher should now compile and produce the executable `bin/xmeshfem3D`. Please note that `xmeshfem3D` must be called directly from the `bin/` directory, as most of the binaries of the package.

Input for the mesh generation program is provided through the parameter file `Mesh_Par_file`, which resides in the subdirectory `DATA/meshfem3D_files/`. Before running the mesher, a number of parameters need to be set in the `Mesh_Par_file`. This requires a basic understanding of how the SEM is implemented, and we encourage you to read **??** and **?**.

The mesher and the solver use UTM coordinates internally, therefore you need to define the zone number for the UTM projection (e.g., zone 11 for Los Angeles). Use decimal values for latitude and longitude (no minutes/seconds). These values are approximate; the mesher will round them off to define a square mesh in UTM coordinates. When running benchmarks on rectangular models, turn the UTM projection off by using the flag `SUPPRESS_UTM_PROJECTION`, in which case all 'longitude' parameters simply refer to the x axis, and all 'latitude' parameters simply refer to the y axis. To run the mesher for a global simulation, the following parameters need to be set in the `Mesh_Par_file`:

`LATITUDE_MIN` Minimum latitude in the block (negative for South).

LATITUDE_MAX Maximum latitude in the block.

LONGITUDE_MIN Minimum longitude in the block (negative for West).

LONGITUDE_MAX Maximum longitude in the block.

DEPTH_BLOCK_KM Depth of bottom of mesh in kilometers.

UTM_PROJECTION_ZONE UTM projection zone in which your model resides, only valid when SUPPRESS_UTM_PROJECTION is .false.. Use a negative zone number for the Southern hemisphere: the Northern hemisphere corresponds to zones +1 to +60, the Southern hemisphere to zones -1 to -60. We use the WGS84 (World Geodetic System 1984) reference ellipsoid for the UTM projection. If you prefer to use the Clarke 1866 ellipsoid, edit file src/shared/utm_geo.f90, uncomment that ellipsoid and recompile the code. From http://en.wikipedia.org/wiki/Universal_Transverse_Mercator_coordinate_system : The Universal Transverse Mercator coordinate system was developed by the United States Army Corps of Engineers in the 1940s. The system was based on an ellipsoidal model of Earth. For areas within the contiguous United States the Clarke Ellipsoid of 1866 was used. For the remaining areas of Earth, including Hawaii, the International Ellipsoid was used. The WGS84 ellipsoid is now generally used to model the Earth in the UTM coordinate system, which means that current UTM northing at a given point can be 200+ meters different from the old one. For different geographic regions, other datum systems (e.g.: ED50, NAD83) can be used.

SUPPRESS_UTM_PROJECTION set to be .false. when your model range is specified in geographical coordinates, and needs to be .true. when your model is specified in Cartesian coordinates. UTM PROJECTION ZONE IN WHICH YOUR SIMULATION REGION RESIDES.

INTERFACES_FILE File which contains the description of the topography and of the interfaces between the different layers of the model, if any. The number of spectral elements in the vertical direction within each layer is also defined in this file.

NEX_XI The number of spectral elements along one side of the block. This number *must* be $8 \times$ a multiple of NPROC_XI defined below. Based upon benchmarks against semi-analytical discrete wavenumber synthetic seismograms [**?**], determined that a NEX_XI $= 288$ run is accurate to a shortest period of roughly 2 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 by

$$\text{shortest period (s)} = (288/\texttt{NEX\_XI}) \times 2. \tag{3.1}$$

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. We generally use NGLLX $= 5$, for a total of $5^3 = 125$ points per elements. We suggest not to change this value.

NEX_ETA The number of spectral elements along the other side of the block. This number *must* be $8 \times$ a multiple of NPROC_ETA defined below.

NPROC_XI The number of processors or slices along one side of the block (see Figure 3.5); we must have NEX_XI $= 8 \times c \times$ NPROC_XI, where $c \geq 1$ is a positive integer.

NPROC_ETA The number of processors or slices along the other side of the block; we must have NEX_ETA $= 8 \times c \times$ NPROC_ETA, where $c \geq 1$ is a positive integer.

USE_REGULAR_MESH set to be .true. if you want a perfectly regular mesh or .false. if you want to add doubling horizontal layers to coarsen the mesh. In this case, you also need to provide additional information by setting up the next three parameters.

NDOUBLINGS The number of horizontal doubling layers. Must be set to 1 or 2 if USE_REGULAR_MESH is set to .true..

NZ_DOUBLING_1 The position of the first doubling layer (only interpreted if USE_REGULAR_MESH is set to .true.).

NZ_DOUBLING_2 The position of the second doubling layer (only interpreted if USE_REGULAR_MESH is set to .true. and if NDOUBLINGS is set to 2).

CREATE_ABAQUS_FILES Set this flag to .true. to save Abaqus FEA (www.simulia.com) mesh files for subsequent viewing. Turning the flag on generates files in the LOCAL_PATH directory. See Section 9.1 for a discussion of mesh viewing features.

CREATE_DX_FILES Set this flag to .true. to save OpenDX (www.opendx.org) mesh files for subsequent viewing.

LOCAL_PATH Directory in which the partitions 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 partitions 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 partitions in parallel, one set for each of the NPROC_XI $\times$ NPROC_ETA slices that constitutes the mesh (see Figure 3.5). 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 files generated by the mesher. These files will be needed for creating the distributed databases, and have to reside in the directory LOCAL_PATH specified in the main Par_file, e.g. in directory OUTPUT_FILES/DATABASES_MPI. Please see Chapter 4 for further details.

NMATERIALS The number of different materials in your model. In the following lines, each material needs to be defined as :

```
material_ID rho vp vs Q anisotropy_flag domain_ID
```

where

- Q : quality factor (0=no attenuation)
- anisotropy_flag : 0=no anisotropy / 1,2,.. check with implementation in aniso_model.f90
- domain_id : 1=acoustic / 2=elastic

NREGIONS The number of regions in the mesh. In the following lines, because the mesh is regular or 'almost regular', each region is defined as :

```
NEX_XI_BEGIN NEX_XI_END NEX_ETA_BEGIN NEX_ETA_END NZ_BEGIN NZ_END material_ID
```

The INTERFACES_FILE parameter of Mesh_Par_File defines the file which contains the settings of the topography grid and of the interfaces grids. Topography is defined as a set of elevation values on a regular 2D grid. It is also possible to define interfaces between the layers of the model in the same way. The file needs to define several parameters:

- The number of interfaces, including the topography. This needs to be set at the first line. Then, from the bottom to the top of the model, you need to define the grids with:
- SUPPRESS_UTM_PROJECTION flag as described previously,
- number of points along x and y direction (NXI and NETA),
- minimal x and y coordinates (LONG_MIN and LAT_MIN),
- spacing between points along x and y (SPACING_XI and SPACING_ETA) and
- the name of the file which contains the elevation values (in $y.x$ increasing order).

At the end of this file, you simply need to set the number of spectral elements in the vertical direction for each layer. We provide a few models in the EXAMPLES/ directory.

Finally, depending on your system, you might 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 10 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 Mesh_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. See Chapter 10 for information about running the code on a system with a scheduler, e.g., LSF.

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. Please note that the mesher suggests a time step DT to run the solver with. The mesher output file also contains a table about the quality of the mesh to indicate possible problems with the distortions of elements. 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
```

To control the quality of the mesh, check the standard output (either on the screen or in the OUTPUT_FILES directory in output_mesher.txt) and analyze the skewness histogram of mesh elements to make sure no element has a skewness above approximately 0.75, otherwise the mesh is of poor quality (and if even a single element has a skewness value above 0.80, then you must definitely improve the mesh). To draw the skewness histogram on the screen, type gnuplot plot_mesh_quality_histogram.gnu.

Chapter 4

Creating the Distributed Databases

After using `xmeshfem3D` or `xdecompose_mesh`, the next step in the workflow is to compile `xgenerate_databases`. This program is going to create all the missing information needed by the SEM solver.

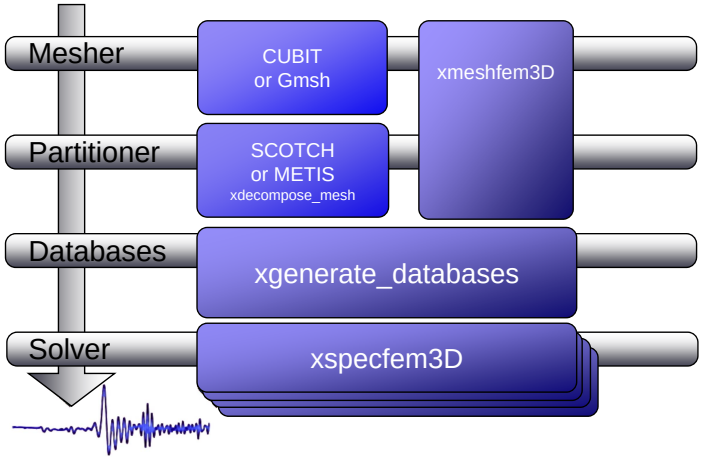


Figure 4.1: Schematic workflow for a SPECFEM3D Cartesian simulation. The executable `xgenerate_databases` creates the GLL mesh points and assigns specific model parameters.

In the main directory, type

```
make xgenerate_databases
```

Input for the program is provided through the main parameter file `Par_file`, which resides in the subdirectory `DATA`. Please note that `xgenerate_databases` must be called directly from the `bin/` directory, as most of the binaries of the package.

4.1 Main parameter file `Par_file`

Before running `xgenerate_databases`, a number of parameters need to be set in the main parameter `Par_file` located in the subdirectory `DATA`:

`SIMULATION_TYPE` is set to 1 for forward simulations, 2 for adjoint simulations (see Section 7.2) and 3 for kernel simulations (see Section 9.3).

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 9.3). For a regular forward simulation, leave `SIMULATION_TYPE` and `SAVE_FORWARD` at their default values.

UTM_PROJECTION_ZONE UTM projection zone in which your model resides, only valid when `SUPPRESS_UTM_PROJECTION` is `.false.`.

SUPPRESS_UTM_PROJECTION set to be `.false.` when your model range is specified in the geographical coordinates, and needs to be `.true.` when your model is specified in a cartesian coordinates. UTM PROJECTION ZONE IN WHICH YOUR SIMULATION REGION RESIDES.

NPROC The number of MPI processors, each one is assigned one slice of the whole mesh.

NSTEP The number of time steps of the simulation. This controls the length of the numerical simulation, i.e., twice the number of time steps requires twice as much CPU time. This feature is not used at the time of generating the distributed databases but is required for the solver, i.e., you may change this parameter after running `xgenerate_databases`.

DT The length of each time step in seconds. This feature is not used at the time of generating the distributed databases but is required for the solver. Please see also Section 4.2 for further details.

NGNOD The number of nodes for 2D and 3D shape functions for hexahedra. We use either 8-node mesh elements (bricks) or 27-node elements. If you use the internal mesher, the only option is 8-node bricks (27-node elements are not supported). `CUBIT` does not support HEX27 elements either (it can generate them, but they are flat, i.e. identical to HEX8). To generate HEX27 elements with curvature properly taken into account, you can use `Gmsh` `http://geuz.org/gmsh/`

MODEL Must be set to one of the following:

Models defined by mesh parameters:

default Uses model parameters as defined by meshing procedures described in the previous Chapter 3.

1D models with real structure:

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

1D_socal A standard isotropic 1D model for Southern California.

1D_cascadia Isotropic 1D profile for the Cascadia region.

Fully 3D models:

aniso For a user-specified fully anisotropic model. Parameters are set up in routines located in file `model_aniso.f90` in directory `src/generate_databases/`. See Chapter 11 for a discussion on how to specify your own 3D model.

external For a user-specified isotropic model which uses externally defined model parameters. Uses external model definitions set up in routines located in file `model_external_values.f90` in directory `src/generate_databases/`. Please modify these generic template routines to use your own model definitions.

gll For a user-specified isotropic model which uses external binary files for v_p, v_s and ρ. Binary files are given in the same format as when outputted by the `xgenerate_databases` executable when using option `SAVE_MESH_FILES`. These binary files define the model parameters on all GLL points which can be used for iterative inversion procedures.

salton_trough A 3D V_p model for Southern California. Users must provide the corresponding data file `regrid3_vel_p.bin` in directory `DATA/st_3D_block_harvard/`.

tomo For a user-specified 3D isotropic model which uses a tomographic model file `tomographic_model.xyz` in directory `DATA`. See Section 11.1, for a discussion on how to specify your own 3D tomographic model.

APPROXIMATE_OCEAN_LOAD Set to `.true.` if the effect of the oceans on seismic wave propagation should be incorporated based upon the (rough) 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. If you want to model the effect of a fluid-solid model at short periods, then set this flag to `.false.` and mesh the fluid layer explicitly in your mesher, so that it is computed accurately and without this approximation.

TOPOGRAPHY This feature is only effective if `APPROXIMATE_OCEAN_LOAD` is set to `.true.`. Set to `.true.` if topography and bathymetry should be read in based upon the topography file specified in the main constants file `constants.h` found in subdirectory `src/shared/` to evaluate elevations. If not set, elevations will be read from the numerical mesh.

ATTENUATION Set to `.true.` if attenuation should be incorporated. Turning this feature on increases the memory requirements significantly (roughly by a factor of 1.5), and is numerically fairly expensive. See **??** for a discussion on the implementation of attenuation based upon standard linear solids. Please note that the Vp- and Vs-velocities of your model are given for a reference frequency. To change this reference frequency, you change the value of `ATTENUATION_f0_REFERENCE` in the main constants file `constants.h` found in subdirectory `src/shared/`.

FULL_ATTENUATION_SOLID Set to `.true.` to add Q_κ attenuation to the simulations in addition to Q_μ attenuation when the `ATTENUATION` flag above is on. Should be off in almost all cases for seismic wave propagation, since Q_κ is almost always negligible in Earth models. However, this parameter can be useful for instance for ocean acoustics simulations, when Q_κ is not negligible in the solid layer at the sea bottom.

ANISOTROPY Set to `.true.` if you want to use an anisotropy model. Please see the file `model_aniso.f90` in subdirectory `src/generate_databases/` for the current implementation of anisotropic models.

TOMOGRAPHY_PATH Directory in which the tomography files are stored for using external tomographic Earth models (please read Chapter 11 and Section 11.1 'Using external tomographic Earth models' for further details.).

USE_OLSEN_ATTENUATION Set to `.true.` if you want to use the attenuation model that scaled from the S-wave speed model using Olsen's empirical relation (see **?**).

OLSEN_ATTENUATION_RATIO Determines the Olsen's constant in Olsen's empirical relation (see **?**).

PML_CONDITIONS Set to `.true.` to turn on C-PML boundary conditions for a regional simulation. Both fluids and elastic solids are supported. Note that xmeshfem3d generated meshes do not support C-PML yet.

PML_INSTEAD_OF_FREE_SURFACE Set to `.true.` to turn on C-PML boundary conditions on the top surface instead of the usual free surface.

f0_FOR_PML Determines the dominant frequency that will be used in the calculation of PML damping profiles; this should be set to the same (or similar) dominant frequency as that of the source that you will use in your simulation. If you plan to use a Dirac source, then use the dominant frequency of the source wavelet with which you plan to convolve your seismograms later on in post-processing.

STACEY_ABSORBING_CONDITIONS Set to `.true.` to turn on Clayton-Enquist absorbing boundary conditions (see **?**). In almost all cases it is much better to use CPML absorbing layers (see the options above) and leave this flag to `.false.`.

STACEY_INSTEAD_OF_FREE_SURFACE Set to `.true.` to turn on absorbing boundary conditions on the top surface which by default constitutes a free surface of the model.

CREATE_SHAKEMAP Set this flag to `.true.` to create a ShakeMap®, i.e., a peak ground velocity map of the maximum absolute value of the two horizontal components of the velocity vector.

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 9.2 for a discussion on the generation of movies. This feature is only relevant for the solver.

MOVIE_TYPE Set this flag to 1 to show the top surface (tomography + oceans) only, to 2 to show all external faces of the mesh (i.e. topography + vertical edges + bottom) in shakemaps and surface movies.

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 9.2 for a discussion on the generation of movies. This feature is only relevant for the solver.

SAVE_DISPLACEMENT Set this flag to `.true.` if you want to save the displacement instead of velocity for the movie frames.

USE_HIGHRES_FOR_MOVIES Set this flag to `.true.` if you want to save the values at all the NGLL grid points for the movie frames.

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 9.2 for a discussion on the generation of movies. This feature is only 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. Otherwise, it adds this half duration to the half duration specified in the source file `CMTSOLUTION`, thus simulates longer periods to make the movie images look smoother.

SAVE_MESH_FILES Set this flag to `.true.` to save ParaView (`www.paraview.org`) mesh files for subsequent viewing. Turning the flag on generates large (distributed) files in the `LOCAL_PATH` directory. See Section 9.1 for a discussion of mesh viewing features.

LOCAL_PATH Directory in which the distributed databases 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). `xgenerate_databases` generates the necessary databases in parallel, one set for each of the `NPROC` slices that constitutes the mesh (see Figure 3.4 and Figure 3.5). After the executable 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 `xgenerate_databases`. Please note that the `LOCAL_PATH` directory should already contain the output files of the partitioner, i.e. from `xdecompose_mesh` or `xmeshfem3D`.

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. 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 only relevant for the solver.

USE_FORCE_POINT_SOURCE Turn this flag on to use a (tilted) `FORCESOLUTION` force point source instead of a `CMTSOLUTION` moment-tensor source. When the force source does not fall exactly at a grid point, the solver interpolates the force between grid points using Lagrange interpolants. This can be useful e.g. for oil industry foothills simulations in which the source is a vertical force, normal force, tilted force, or an impact etc. Note that in the `FORCESOLUTION` file, you will need to edit the East, North and vertical components of an arbitrary (non-unitary) direction vector of the force vector; thus refer to Appendix A for the orientation of the reference frame. This vector is made unitary internally in the solver and thus only its direction matters here; its norm is ignored and the norm of the force used is the factor force source times the source time function. When using this option, by default the code can locate the force source anywhere between mesh points in order to honor its exact location; this is more precise than using the closest GLL mesh point, but it is also a bit slower. If needed, you can change that default behavior and force the code to use the closest GLL mesh point instead by setting flag `USE_BEST_LOCATION` to `.false.` instead of `.true.` in file `src/shared/constants.h.in` and running the `configure` script again and recompiling the code.

USE_RICKER_TIME_FUNCTION Turn this flag on to use a Ricker source time function instead of the source time functions set by default to represent a (tilted) FORCESOLUTION force point source or a CMTSOLUTION moment-tensor source. Originally, if a CMTSOLUTION moment-tensor source is used, a (pseudo) Heaviside step function with a very short half duration is defined for elastic cases to represent the permanent slip on the fault while in the acoustic case a Gaussian source time function with a similarly short half duration is defined to physically describe actions within the fluid. Otherwise, if a FORCESOLUTION force source is used, a (pseudo) Dirac delta source time function is defined by default. Any other source-time function may then be obtained by convolution.

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 only relevant for the solver.

GPU_MODE Turn this flag on to use GPUs.

ADIOS_ENABLED Turn this flag on to enable ADIOS. If set to .false., subsequent ADIOS parameters will not be considered.

ADIOS_FOR_DATABASES Turn this flag on to use ADIOS for xmeshfem3D output and xgenerate_database input.

ADIOS_FOR_MESH Turn this flag on to use ADIOS for generated databases.

ADIOS_FOR_FORWARD_ARRAYS Turn this flag on to read and write forward arrays using ADIOS.

ADIOS_FOR_KERNELS Turn this flag on to produce ADIOS kernels that can later be visualized with the ADIOS version of combine_vol_data.

If you use PML, the mesh elements that belong to the PML layers can be acoustic or elastic, but not viscoelastic nor poroelastic. Then, when defining your model, you should define these absorbing elements as either acoustic or elastic. In you forget to do that, the code will fix the problem by automatically converting the viscoelastic or poroelastic PML elements to elastic. This means that strictly speaking the PML layer will not be perfectly matched any more, since the physical model will change from viscoelastic or poroelastic to elastic at the entrance of the PML, but in practice this is sufficient and produces only tiny / negligible spurious reflections.

If you use PML and an external tomographic velocity and density model, you should be careful because mathematically a PML cannot handle heterogeneities along the normal to the PML edge inside the PML layer. This comes from the fact that the damping profile that is defined assumes a constant velocity and density model along the normal direction.

Thus, you need to modify your velocity and density model in order for it to be 1D inside the PML, as shown in Figure 4.2.

This applies to the bottom layer as well; there you should make sure that your model is 1D and thus constant along the vertical direction.

To summarize, only use a 3D velocity and density model inside the physical region, and in all the PML layers extend it by continuity from its values along the inner PML edge.

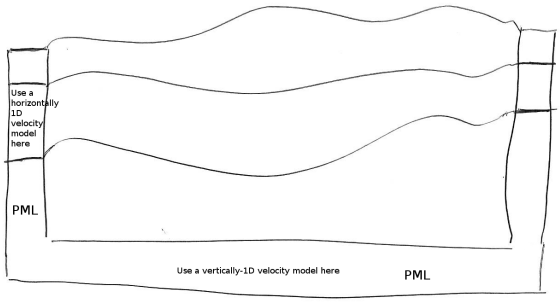


Figure 4.2: How to modify your external 3D velocity and density model in order to use PML. Such a modification is not needed when using Stacey absorbing boundary conditions (but such conditions are significantly less efficient).

4.2 Choosing the time step DT

The parameter DT sets the length of each time step in seconds. The value of this parameter is crucial for the stability of the spectral-element simulation. Your time step DT will depend on the minimum ratio between the distance h of neighboring mesh points and the wave speeds v defined in your model. The condition for the time step Δt is:

$$\Delta t < C \ \min_{\Omega}(\ h/v \ )$$

where C is the so-called Courant number and Ω denotes the model volume. The distance h depends on the mesh element size and the number of GLL points NGLL specified in the main constants file constants.h located in the src/shared/ subdirectory. The wave speed v is determined based on your model's P- (or S-) wave speed values.

The database generator xgenerate_databases, as well as the internal mesher xmeshfem3D, are trying to evaluate the value of Δt for empirically chosen Courant numbers $C \sim 0.3$. If you used the mesher xmeshfem3D to generate your mesh, you should set the value suggested in OUTPUT_FILES/output_mesher.txt file, which is created after the mesher completed. In case you used CUBIT to create the mesh, you might use an arbitrary value when running xgenerate_databases and then use the value suggested in the OUTPUT_FILES/output_mesher.txt file after the database generation completed. Note that the implemented Newmark time scheme uses this time step globally, thus your simulations become more expensive for very small mesh elements in high wave-speed regions. Please be aware of this restriction when constructing your mesh in Chapter 3.

Chapter 5

Running the Solver `xspecfem3D`

Now that you have successfully generated the databases, you are ready to compile the solver. In the main directory, type

```
make xspecfem3D
```

Please note that `xspecfem3D` must be called directly from the `bin/` directory, as most of the binaries of the package.

The solver needs three input files in the `DATA` directory to run:

`Par_file` the main parameter file which was discussed in detail in the previous Chapter 4,

`CMTSOLUTION` or `FORCESOLUTION` the earthquake source parameter file or the force source parameter file, and

`STATIONS` the stations file.

Most parameters in the `Par_file` should be set prior to running the databases generation. Only the following parameters may be changed after running `xgenerate_databases`:

- the simulation type control parameters: `SIMULATION_TYPE` and `SAVE_FORWARD`
- the time step parameters `NSTEP` and `DT`
- the absorbing boundary control parameter `PML_CONDITIONS` on condition that the `PML_INSTEAD_OF_FREE_SURFACE` flag remains unmodified after running the databases generation.
- the movie control parameters `MOVIE_SURFACE`, `MOVIE_VOLUME`, and `NTSTEPS_BETWEEN_FRAMES`
- the ShakeMap®option `CREATE_SHAKEMAP`
- the output information parameters `MOVIE_TYPE`, `NTSTEP_BETWEEN_OUTPUT_INFO` and `NTSTEP_BETWEEN_OUTPUT_SEISMOS`
- the `PRINT_SOURCE_TIME_FUNCTION` flags

Any other change to the `Par_file` implies rerunning both the database generator `xgenerate_databases` and the solver `xspecfem3D`.

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.seismology.harvard.edu`). It looks like this:

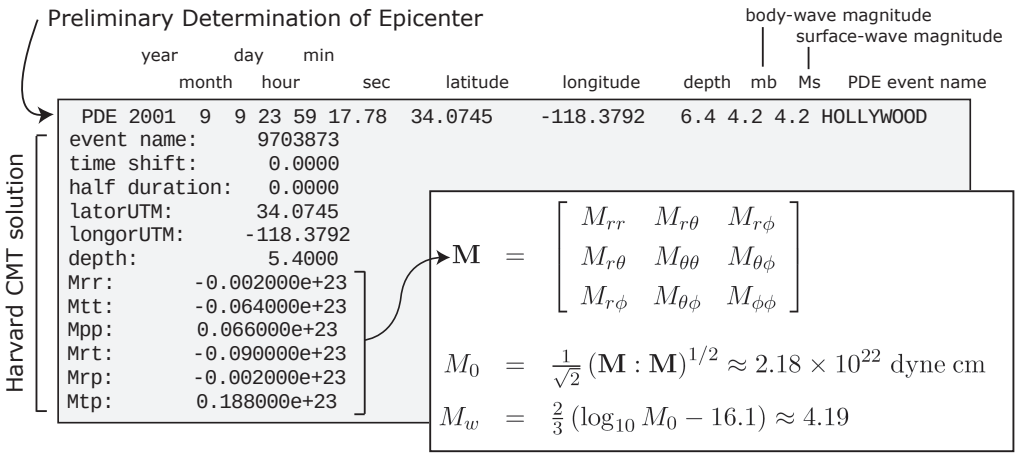


Figure 5.1: `CMTSOLUTION` file based on the format from the Harvard CMT catalog. $\mathbf{M}$ is the moment tensor, M_0 is the seismic moment, and M_w is the moment magnitude.

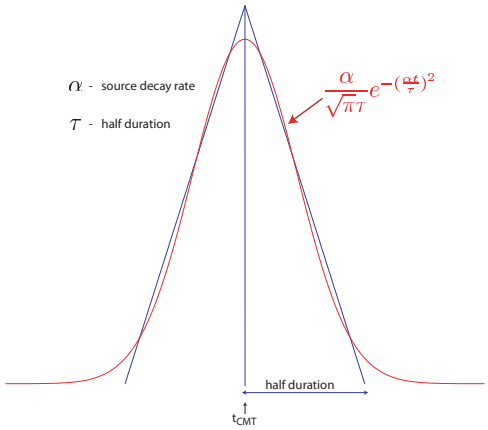


Figure 5.2: Comparison of the shape of a triangle and the Gaussian function actually used.

The CMTSOLUTION file should be edited in the following way:

- Set the latitude or UTM x coordinate, longitude or UTM y coordinate, depth of the source (in km).
- Set the time shift parameter equal to 0.0 (the solver will not run otherwise.) The time shift parameter would simply apply an overall time shift to the synthetics, something that can be done in the post-processing (see Section 12.1).
- For point-source simulations (see finite sources, page 33) 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 5.2) 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 12.1). **?** 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 serial code convolve_source_timefunction.f90 and the script convolve_source_timefunction.csh for this purpose, or alternatively use signal-processing software packages such as SAC (www.llnl.gov/sac). Type

  ```
  make xconvolve_source_timefunction
  ```

  to compile the code and then set the parameter hdur in 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.

If you know the earthquake source in strike/dip/rake format rather than in CMTSOLUTION format, use the C code SPECFEM3D_GLOBE/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.

To simulate a kinematic rupture, i.e., a finite-source event, represented in terms of $N_{\rm sources}$ point sources, provide a `CMTSOLUTION` file that has $N_{\rm sources}$ entries, one for each subevent (i.e., concatenate $N_{\rm sources}$ `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`. 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 5.3).

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 5.2. 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.

The `FORCESOLUTION` file should be edited in the following way:

- Set the `time shift` parameter equal to 0.0 (the solver will not run otherwise.) The time shift parameter would simply apply an overall time shift to the synthetics, something that can be done in the post-processing (see Section 12.1).
- Set the `half duration` parameter of the Ricker source time function when `USE_RICKER_TIME_FUNCTION` is turned on in the main parameter file `Par_file`. In case that the solver uses a (pseudo) Dirac delta source time function to represent a force point source, a very short `half duration` parameter (hdur = 5 * DT) is automatically set by default.
- Set the latitude or UTM x coordinate, longitude or UTM y coordinate, depth of the source (in km).
- Set the magnitude of the force source.
- Set the components of a (non-unitary) direction vector for the force source in the East/North/Vertical basis (see Appendix A for the orientation of the reference frame).

Where necessary, set a `FORCESOLUTION` file in the same way you configure a `CMTSOLUTION` file with $N_{\rm sources}$ entries, one for each subevent (i.e., concatenate $N_{\rm sources}$ `FORCESOLUTION` files to a single `FORCESOLUTION` 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`. Each subevent can have its own half latitude, longitude, depth, `half duration` and force parameters.

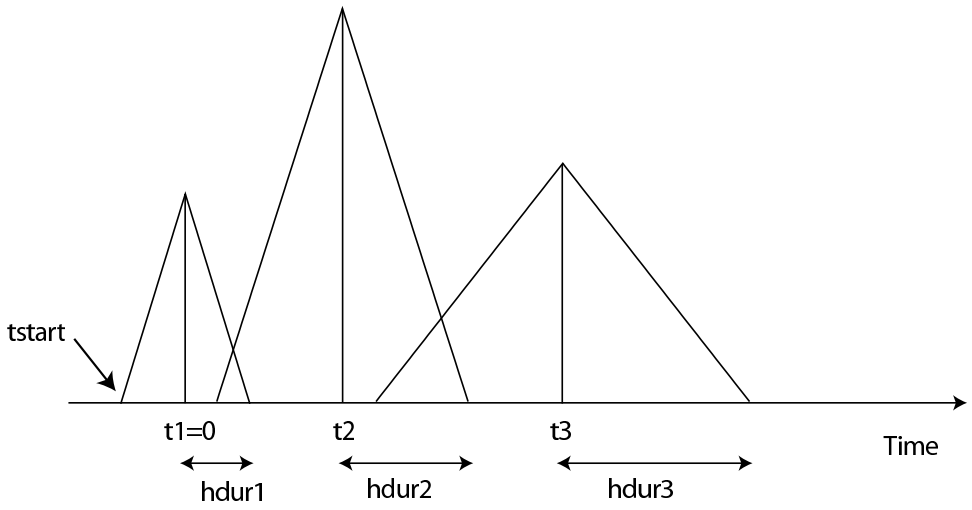


Figure 5.3: Example of timing for three sources. The center of the first source triangle is defined to be time zero. Note that this is NOT in general the hypocentral time, or the start time of the source (marked as tstart). The parameter `time shift` in the `CMTSOLUTION` file would be t1(=0), t2, t3 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)
ASBS	AZ	33.6208	-116.4664	0.0	0.0
BZN	AZ	33.4915	-116.6670	0.0	0.0
CRY	AZ	33.5654	-116.7373	0.0	0.0
ELKS	AZ	33.5813	-116.4496	0.0	0.0
AGA	CI	33.6384	-116.4011	0.0	0.0
AGO	CI	34.1465	-118.7670	0.0	0.0
ALP	CI	34.6870	-118.2995	0.0	0.0
BAK	CI	35.3444	-119.1044	0.0	0.0
BAR	CI	32.6801	-116.6722	0.0	0.0
BBA	CI	34.1955	-118.3534	0.0	0.0
BBB	CI	33.3526	-115.7332	0.0	0.0
BBR	CI	34.2623	-116.9207	0.0	0.0
BBS	CI	33.9214	-116.9805	0.0	0.0
⋮	⋮	⋮	⋮	⋮	⋮

Figure 5.4: 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)
```

The solver `xspecfem3D` filters the list of stations in file `DATA/STATIONS` to exclude stations that are not located within the region given in the `Par_file` (between `LATITUDE_MIN` and `LATITUDE_MAX` and between `LONGITUDE_MIN` and `LONGITUDE_MAX`). The filtered file is called `DATA/STATIONS_FILTERED`.

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
```

On PC clusters the seismogram files are generally written to the local disks (the path `LOCAL_PATH` in the `Par_file`) and need to be gathered at the end of the simulation.

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 #          10000
Time:     108.4890      seconds
Elapsed time in seconds =    1153.28696703911
Elapsed time in hh:mm:ss =     0 h 19 m 13 s
Mean elapsed time per time step in seconds =     0.115328696703911
Max norm displacement vector U in all slices (m) =    1.0789589E-02
```

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 all slices (m)`. If something is wrong with the model, the mesh, or the source, you will see the code become unstable through exponentially growing values of the displacement and fluid potential with time, and ultimately the run will be terminated by the program. 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` (or the `FORCESOLUTION`) and `STATIONS` files, you are now ready to launch the solver! This is most easily accomplished based upon the `go_solver` script (See Chapter 10 for information about running 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. The `runall` script compiles and runs both `xgenerate_databases` and `xspecfem3D` in sequence. This is a safe approach that ensures using the correct combination of distributed database 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` or `FORCESOLUTION` file, or for different stations by changing the `STATIONS` file. There is no need to rerun the `xgenerate_databases` executable. Of course it is best to include as many stations as possible, since this does not add to the cost of the simulation.

We have also added the ability to run several calculations (several earthquakes) in an embarrassingly-parallel fashion from within the same run; this can be useful when using a very large supercomputer to compute many earthquakes in a catalog, in which case it can be better from a batch job submission point of view to start fewer and much larger jobs, each of them computing several earthquakes in parallel. To turn that option on, set parameter NUMBER_OF_SIMULTANEOUS_RUNS to a value greater than 1 in file setup/constants.h.in before configuring and compiling the code. When that option is on, of course the number of processor cores used to start the code in the batch system must be a multiple of NUMBER_OF_SIMULTANEOUS_RUNS, all the individual runs must use the same number of processor cores, which as usual is NPROC in the input file DATA/Par_file, and thus the total number of processor cores to request from the batch system should be NUMBER_OF_SIMULTANEOUS_RUNS $\times$ NPROC. All the runs to perform must be placed in directories called run0001, run0002, run0003 and so on (with exactly four digits) and you must create a link from the root directory of the code to the first copy of the executable programs by typing " ln -s run0001/bin bin ".

Chapter 6

Kinematic and dynamic fault sources

SPECFEM3D can handle finite fault sources of two kinds:

1. *Kinematic*: the spatio-temporal distribution of slip rate is prescribed along the fault surface
2. *Dynamic*: a friction law and initial stresses are prescribed on the fault, a spontaneous rupture process is computed.

6.1 Mesh Generation with Split Nodes

Faults need to be handled in a special way during mesh generation. A fault surface must lie at the interface between elements (the mesh must honor the fault surfaces). Moreover, a fault is made of two surfaces in contact. Each of these two surfaces needs a separate set of nodes. This approach is known as "split nodes". Currently faults can only be run with `xdecompose_mesh` and `CUBIT. xmeshfem3D` is not yet ready to handle faults.
To facilitate the mesh generation with split nodes in CUBIT, we need to separate the two fault surfaces by a small distance, effectively creating a tiny opening of the fault (Figure 6.1, 6.2). Note that only the interior of the fault must be opened, its edges must remain closed (except the edge on the free surface). The fault is automatically closed later by SPECFEM3D.

Here is an example CUBIT script to generate a mesh with split nodes for a buried vertical strike-slip fault:

```
reset
brick x 10 y 10 z 10
webcut volume all with plane xplane
webcut volume all with plane yplane
webcut volume all with plane xplane offset 3
webcut volume all with plane zplane offset 3
webcut volume all with plane zplane offset -3
imprint all
merge all
unmerge surf 160
mesh vol all
set node constraint off
node in surf 168 move X 0 Y 0.01 Z 0
node in surf 160 move X 0 Y -0.01 Z 0
```

The CUBIT scripts (*.jou and *.py) in the directory `EXAMPLES` generate more complicated meshes. The *.py files are Python scripts that execute CUBIT commands and use the CUBIT-python interface for SPECFEM3D (see next

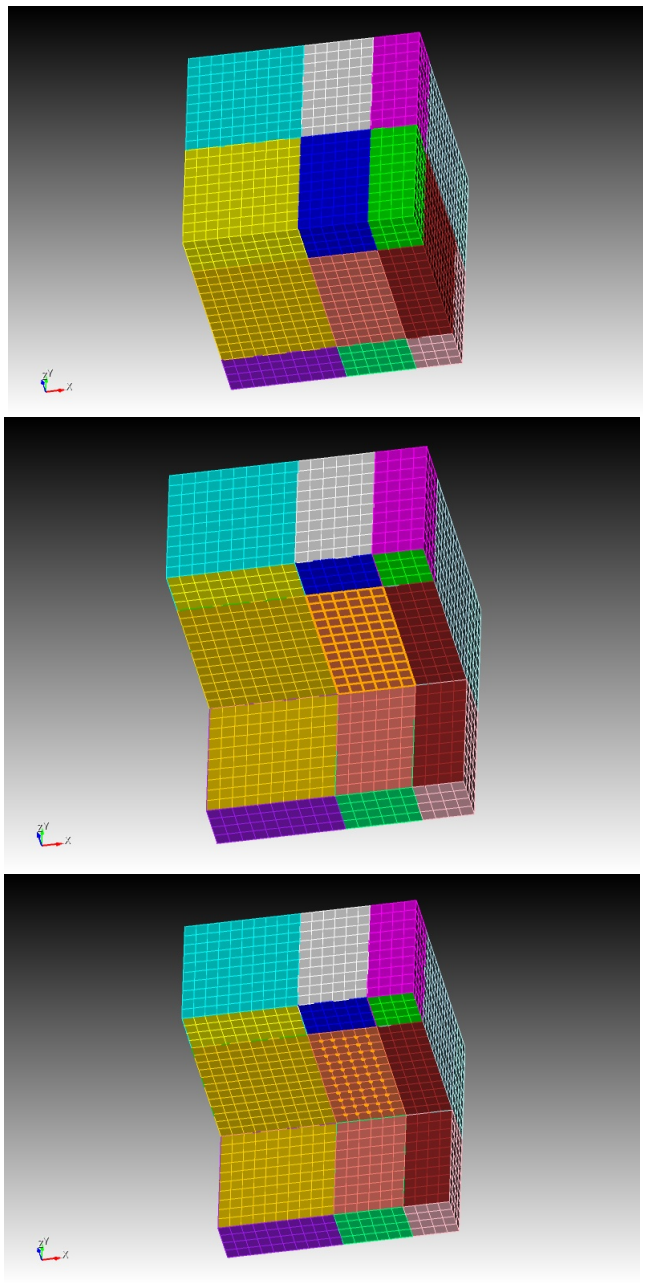


Figure 6.1: Screenshots of the CUBIT example for the embedded fault with split nodes: The entire mesh is decomposed into several volumes bounding the fault (top), the fault surface 168 is shown as orange squares (middle) and split nodes of the fault shown as orange dots(bottom). Note that only interior nodes of the fault are split while those on the edges of the fault surface are touching each other.

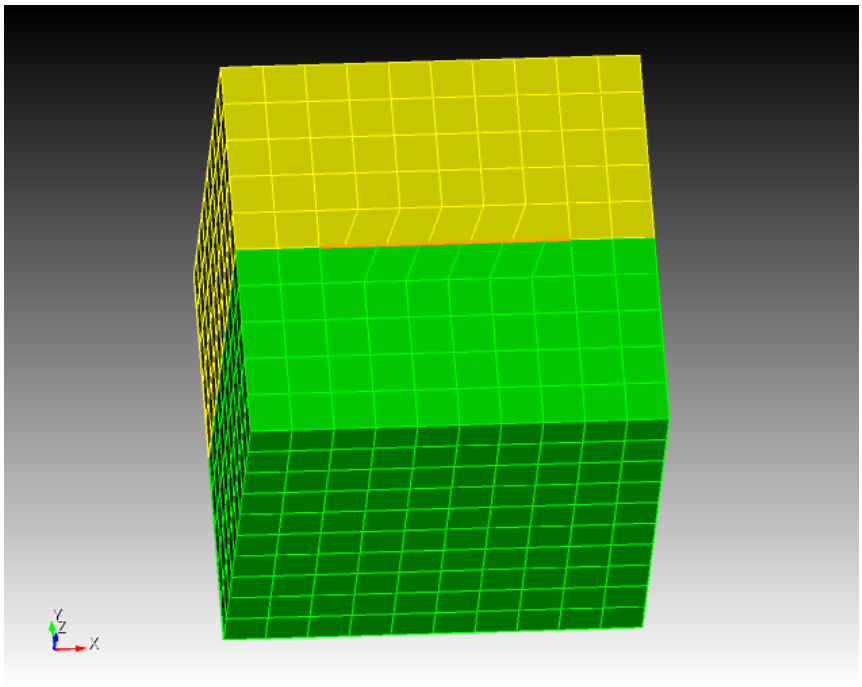


Figure 6.2: Screenshots of a CUBIT example showing split nodes for a fault reaching the surface. Surface trace of the fault is shown in orange. Note that the edges of the fault are not split while the interior nodes are offset by a small distance on either side of the fault

section). The Python language allows to define and manipulate variables to parameterize the mesh. Alternatively, the Python script can call a CUBIT journal file (*.jou), which looks like the example above. Variables can be defined and manipulated there using the `APREPRO` language built in CUBIT.

Note that you should avoid gaps in the list of indices of mesh objects with the following CUBIT command:

```
compress ids hex face edge node
```

(otherwise you will get a segmentation fault during domain decomposition)

6.2 CUBIT-Python Scripts for Faults

The mesh generated in CUBIT needs to be processed and exported in a format compatible with SPECFEM3D. This is achieved in the Python scripts by calling the Python-CUBIT interface functions defined in the CUBIT directory:

1. Function `define_bc` (or `boundary_definition.py`) must be called to set up the absorbing boundaries database
2. Function `fault_input` must be called once for each fault to set up the fault database
3. Function `cubit2specfem3d.export2SPECFEM3D` must be called at the very end of the script to export the mesh in a SPECFEM3D format.

The functions in #1 and #3 are for the bulk and are documented in Section 3.1.2. We focus here on #2:

Function: `fault_input`

Purpose: export a fault mesh database from CUBIT to a SPECFEM3D-compliant file

Syntax: fault_input(id_fault, ids_surf_1, ids_surf_2)

Inputs:

- id_fault integer index assigned to the fault. (The user must number all the faults, starting at 1, with unit increments)
- ids_surf_1 list of CUBIT ids of all surfaces that form side 1 of the fault.
- ids_surf_2 list of CUBIT ids of all surfaces that form side 2 of the fault. (The user must decide which side of the fault is side 1. This choice affects the sign conventions of fault quantities as explained in Section 6.4).

Outputs: file `fault_file_X.dat`, where X is the fault id (id_fault).

Example: For the example in Section 6.1:

```
A1 = [168]
A2 = [160]
fault_input(1,A1,A2)
```

6.3 Examples

The package includes examples, the SCEC benchmark problems:

- TPV5, a planar vertical strike-slip fault
- TPV14 and TPV15, vertical strike-slip fault system with a fault branch
- TPV102 and TPV103, rate-and-state benchmarks
- TPV22 and TPV23, step-over benchmarks
- TPV16, heterogenous initial stress

and

- Splay fault models from Wendt et al. (2009)

Read the documents in the directory `EXAMPLES/*/description`. They contain a description of the example and additional instructions to run it. Visualize the results with the Matlab scripts in the directory `EXAMPLES/*/post`

6.4 Sign Convention for Fault Quantities

During mesh generation, the fault is defined by two surfaces in contact. Let's denote as "*side 1*" the SECOND surface declared by the user in the call to the python function "*fault_input*", and the FIRST surface as "*side 2*". The local coordinate system on the fault is defined as the right-handed coordinate system defined by (strike, dip, normal), where "*normal*" is the normal vector outgoing from side 1, "*dip*" is parallel to the along-dip direction pointing downwards, and "*strike*" is the horizontal along-strike vector such that the system is right-handed.

Slip is defined as displacement on side 2 minus displacement on side 1. In the local coordinate system on the fault, positive along-strike slip is right-lateral and positive along-dip slip is thrust if side 1 is on the hanging wall (normal faulting if side 1 is on the foot wall).

Traction is defined as the stress induced on side 1 by side 2, which is the stress tensor times the normal vector outgoing from side 1. In the local coordinate system on the fault, the normal traction is negative in compression, positive along-strike traction generates right-lateral slip and positive along-dip traction generates thrust slip if side 1 is on the hanging wall (normal faulting if side 1 is on the foot wall).

6.5 Input Files

Three additional input files are required in the `DATA` directory for dynamic and kinematic faults. They are `Par_file_faults`, `FAULT_STATIONS` and `input_file.txt` (optional). If the former does not exist, the code assumes that there are no faults.

DATA/Par_file_faults contains parameters of the fault. The first part of this file has a strict format:

Line 1: Number of faults (NF)

Lines 2 to NF+1: Kelvin Voigt damping (in seconds) for each fault. (See below how to set this parameter)

Line NF+2: Type of simulation (1=dynamic , 2 = kinematic)

Line NF+3: Number of time steps between updates of the time series outputs at selected fault points (see DATA/FAULT_STATIONS), usually a large number (100s or 1000s). Note that the sampling rate of the time series is usually much higher.

Line NF+4: Number of time steps between fault snapshot outputs (quantities at every fault point exported at regular times), usually a large number (100s or 1000s).

Line NF+5: Slip velocity threshold below which frictional healing is set (friction coefficient is reset to its static value). If this value is negative healing is disabled.

Line NF+6: Slip velocity threshold to define the rupture front. Only used for outputs.

The rest of this file is made of namelist input blocks (see "namelist" in a Fortran 9x manual). The input for each fault has the following sequence (arguments in [brackets] are optional):

&**BEGIN_FAULT /**

&**INIT_STRESS** S1, S2, S3 [,n1, n2, n3] **/**

followed by (n1+n2+n3) &**DIST2D** blocks

&**SWF** mus, mud, dc [, nmus, nmud, ndc] **/**

followed by (nmus+nmud+ndc) &**DIST2D** blocks

&**INIT_STRESS** input block sets the initial fault stresses relative to the foot-wall side of the fault. Initial stresses are composed of a constant background value possibly overwritten in prescribed regions by heterogeneous distributions (see &**DIST2D** blocks below):

S1 = constant background value of along-strike shear stress (positive in the usual strike direction)

S2 = constant background value of along-dip shear (positive is down-dip, normal faulting)

S3 = constant background value of normal stress (negative in compression)

n1 = number of heterogeneous items for along-strike shear stress [default is 0]

n2 = number of heterogeneous items for along-dip shear stress [default is 0]

n3 = number of heterogeneous items for normal stress [default is 0]

&**SWF** input block sets the slip-weakening friction parameters of the fault:

mus = constant background value of static friction coefficient

mud = constant background value of dynamic friction coefficient

dc = constant background value of critical slip-weakening distance

nmus = number of heterogeneous items for static friction coefficient [default is 0]

nmud = number of heterogeneous items for dynamic friction coefficient [default is 0]

ndc = number of heterogeneous items for critical slip-weakening distance [default is 0]

&**DIST2D** input blocks modify (overwrite) the value of a fault parameter by a heterogeneous spatial distribution:

&**DIST2D** shape='`square`', val, xc, yc, zc, l /

sets a constant value (val) within a cube with center (xc,yc,zc) and edge size l.

&**DIST2D** shape='`rectangle`', val, xc, yc, zc, lx, ly, lz /

sets a constant value (val) within a rectangular prism with center (xc,yc,zc) and edge sizes (lx,ly,lz).

&**DIST2D** shape='`rectangle − taper`', val, valh, xc, yc, zc, lx, ly, lz /

sets a vertical linear gradient within a rectangular prism with center (xc,yc,zc) and edge sizes (lx,ly,lz). Values vary linearly as a function of vertical position z between value val at z = zc-lz/2 and value valh at z = zc+lz/2 .

&**DIST2D** shape='`circular`', val, xc, yc, zc, r /

sets a constant value (val) within a sphere with center (xc,yc,zc) and radius r.

DATA/FAULT_STATIONS Stations in the fault plane.

Line 1 : number of stations.

Line 2 to end: 5 columns: X, Y, Z (-depth), station name, fault-id

The fault-id identifies the fault that contains the station. It is the index of appearance in the faults list after line 2 of Par_file_faults

DATA/input_file.txt Heterogeneous stresses and friction parameters are documented in page 10 of

`EXAMPLES/tpv16/description/TPV16_17_Description_v03.pdf`

To activate this feature, in fault_solver_dynamic.f90 set **TPV16** = .true. then re-compile the code.

6.6 Setting the Kelvin-Voigt Damping Parameter

The purpose of the Kelvin-Voigt viscosity in the dynamic fault solver is to damp spurious oscillations generated by the fault slip at frequencies that are too high to be resolved by the mesh. The viscosity `eta` (in seconds) depends on the size of the elements on the fault. Here is how to set it:

1. Determine the average linear size of the elements on the fault plane, `h_fault`. Usually this value is prescribed by the user during mesh generation. Otherwise it can be found by inspection of the mesh inside the CUBIT GUI.
2. Use the Matlab function `utils/critical_timestep.m` to compute
   $\texttt{dtc\_fault} = \texttt{critical\_timestep}\left(\texttt{c}_\texttt{p}, \texttt{h\_fault}, \texttt{ngll}\right)$.
   This is the critical time step in an elastic medium for a hypothetical element of cubic shape with size equal to `h_fault`.
3. Set `eta` in `Par_file_faults` to (0.1 to 0.3)×`dtc_fault`. A larger `eta` damps high-frequencies more aggressively but it might also affect lower frequencies and rupture speed.

Viscosity reduces numerical stability: the critical timestep in a simulation with Kelvin-Voigt damping needs to be smaller than that in a purely elastic simulation. Here is how to set the time step accordingly:

1. Run a test simulation without viscosity (`eta`=0 and only a few time steps)
2. Look for the "maximum suggested time step" in `OUTPUT_FILES/output_mesher.txt`. This is the critical timestep of a purely elastic simulation, `dtc_bulk`.
3. Reset the timestep of the simulation with a Kelvin-Voigt material to a value smaller than
   $\texttt{dtc\_kv} = \texttt{eta}\left(\sqrt{1 + \texttt{dtc\_bulk}^2/\texttt{eta}^2} - 1\right)$

Note that in general `dtc_bulk` is smaller than `dtc_fault`, because elements off the fault might be smaller or more distorted than element faces on the fault.

6.7 Output Files

Several output files are saved in `OUTPUT_FILES`:

1. Seismograms for each station on the fault plane given in `DATA/FAUL_STATIONS`. One output file is generated for each station, named after the station. The files are ascii and start with a header (22 lines long) followed by a data block with the following format, one line per time sample:

   Column 1 = Time (s)

   Column 2 = horizontal right-lateral slip (m)

   Column 3 = horizontal right-lateral slip rate (m/s)

   Column 4 = horizontal right-lateral shear stress (MPa)

   Column 5 = vertical up-dip slip (m)

   Column 6 = vertical up-dip slip rate (m/s)

   Column 7 = vertical up-dip shear stress (MPa)

   Column 8 = normal stress (MPa)

   The stresses are relative to the footwall side of the fault (this convention controls their sign, but not their amplitude). Slip is defined as displacement of the hanging wall relative to the footwall.

2. Seismograms at stations in the bulk (out of the fault plane) given in `DATA/STATIONS`.

3. Rupture time files are named `Rupture_time_FAULT-id`. One file is generated for each fault. The files are ascii and start with a header (12 lines long) followed by a data block with the following format, one line per fault node:

   Column 1 = horizontal coordinate, distance along strike (m)

   Column 2 = vertical coordinate, distance down-dip (m)

   Column 3 = rupture time (s)

4. Fault quantities (slip, slip rate, stresses, etc) at regular times are stored in binary data files called `Snapshot#it#.bin`, where #it# is the timestep number. These can be read in Matlab with the function `utils/FSEM3D_snapshot.m`

6.8 Post-processing and Visualization

Some Matlab functions for post-processing and visualization are included in directory `utils`. The functions are internally documented (see their matlab help).

`FSEM3D_snapshot` reads a fault data snapshot

The directories `EXAMPLES/*/post` contain additional Matlab scripts to generate figures specific to each example.

Chapter 7

Adjoint Simulations

Adjoint simulations are generally performed for two distinct applications. First, they can be used in point source moment-tensor inversions, or source imaging for earthquakes with large ruptures such as the Lander's earthquake [**?**]. 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 7.1 and 7.2, respectively. The two related parameters in the `Par_file` are `SIMULATION_TYPE` (1, 2 or 3) and the `SAVE_FORWARD` (boolean).

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

When a specific misfit function between data and synthetics is minimized to invert for 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 as virtual sources in an adjoint simulation. 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 a sub-directory called `SEM/` in the root directory of the code, with the format `STA.NT.BX?.adj`, where `STA`, `NT` are the station name and network code given in the `DATA/STATIONS_ADJOINT` file, and `BX?` represents the component name of a particular adjoint seismogram. Please note that the band code can change depending on your sampling rate (see Appendix B for further details).
   - The adjoint seismograms are in the same format as the original seismogram (`STA.NT.BX?.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, collect the seismograms from `LOCAL_PATH`.

- 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.BX?.sem` for the three-component displacements (`BXN,BXE,BXZ`) 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.

7.2 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 implemented yet for the computation of finite-frequency kernels; therefore set `ATTENUATION = .false.` in the `Par_file`.
- We also suggest you modify the half duration of the `CMTSOLUTION` to be similar to the accuracy of the simulation (see Equation 3.1) 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 the Section 1.

- In the case of travel-time 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/xcreate_adjsrc_traveltime` to cut a certain portion of the original displacement seismograms and convert it 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. 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 travel-time 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 the `SEM/` directory, set `SIMULATION_TYPE = 3` and `SAVE_FORWARD = .false.` in the `Par_file` (you can use `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 `LOCAL_PATH` at the end of the kernel simulations, and can be collected to the local disk.
   - 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.
   - The arrays for density, P-wave speed and S-wave speed kernels are also saved in the `LOCAL_PATH` with the names `proc??????_rho(alpha,beta)_kernel.bin`, where `proc??????` represents the processor number, `rho(alpha,beta)` are the different types of kernels.

4. **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 cartesian coordinate system. This is done by setting `ANISOTROPIC_KL = .true.` in `constants.h` before compiling the package. 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 cartesian coordinates follows **?**. The 21 anisotropic kernels are saved in the `LOCAL_PATH` in one file with the name of `proc??????_cijkl_kernel.bin` (with `proc??????` the processor number). The output kernels correspond to absolute perturbations 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 for different parameterizations of anisotropy. This can be done, for example, when combining the kernel files from slices into one mesh file (see Section 9.3).

   If `ANISOTROPIC_KL = .true.` by additionally setting `ANISOTROPIC_KL = .true.` in `constants.h` the package will save anisotropic kernels paramterized as velocities related to transverse isotropy based on the the Chen and Tromp parameters **?**. The kernels are saved as relative perturbations for horizontal and vertical P and S velocities, $\alpha_v, \alpha_h, \beta_v, \beta_h$. Explicit relations can be found in appendix B. of **?**

   The anisotropic kernels are only currently available for CPU mode.

In general, the three steps need to be run sequentially to assure proper access to the necessary files. 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 dilemma are provided in Chapter 10. Visualization of the finite-frequency kernels is discussed in Section 9.3.

Chapter 8

Noise Cross-correlation Simulations

Besides earthquake simulations, SPECFEM3D Cartesian includes functionality for seismic noise tomography as well. In order to proceed successfully in this chapter, it is critical that you have already familiarized yourself with procedures for meshing (Chapter 3), creating distributed databases (Chapter 4), running earthquake simulations (Chapters 5) and adjoint simulations (Chapter 7). Also, make sure you read the article 'Noise cross-correlation sensitivity kernels' [**?**], in order to understand noise simulations from a theoretical perspective.

8.1 Input Parameter Files

As usual, the three main input files are crucial: `Par_file`, `CMTSOLUTION` and `STATIONS`. Unless otherwise specified, those input files should be located in directory `DATA/`.

`CMTSOLUTION` is required for all simulations. At a first glance, it may seem unexpected to have it here, since the noise simulations should have nothing to do with the earthquake – `CMTSOLUTION`. However, for noise simulations, it is critical to have no earthquakes. In other words, the moment tensor specified in `CMTSOLUTION` must be set to zero manually!

`STATIONS` remains the same as in previous earthquake simulations, except that the order of receivers listed in `STATIONS` is now important. The order will be used to determine the 'master' receiver, i.e., the one that simultaneously cross correlates with the others.

`Par_file` also requires careful attention. A parameter called `NOISE_TOMOGRAPHY` has been added which 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 it is 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 `Par_file` involves the parameter `NSTEP`. 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, i.e., $2\,\mathrm{NSTEP} - 1$. The code automatically makes the modification accordingly, if `NOISE_TOMOGRAPHY` is not zero.

There are other parameters in `Par_file` which should be given specific values. For instance, since the first two steps for calculating noise sensitivity 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.`.

Finally, for most system architectures, please make sure that LOCAL_PATH in Par_file is in fact local, not globally shared. Because we have to save the wavefields at the earth's surface at every time step, it is quite problematic to have a globally shared LOCAL_PATH, in terms of both disk storage and I/O speed.

8.2 Noise Simulations: Step by Step

Proper parameters in those parameter files are not enough for noise simulations to run. We have more parameters 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. In this section, standard procedures for noise simulations are described.

8.2.1 Pre-simulation

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

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

  Use the following if SCOTCH is needed:

  ```
  ./configure FC=ifort MPIFC=mpif90 --with-scotch-dir=/opt/scotch
  ```

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

  ```
  make xgenerate_databases
  make xspecfem3D
  ```

- Before we can run noise simulations, we have to specify the noise statistics, e.g., the ensemble-averaged noise spectrum. Matlab scripts are provided to help you to generate the necessary file.

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

  In Matlab, simply run:

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

  DT is given in Par_file, but NSTEP is NOT the one specified in Par_file. Instead, you have to feed $2\,\mathrm{NSTEP} - 1$ to account for the doubled length of cross correlations. Tmin and Tmax correspond to the period range you are interested in, whereas 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:
  OUTPUT_FILES/NOISE_TOMOGRAPHY/ in the SPECFEM3D Cartesian 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.

- Create a new directory in the SPECFEM3D Cartesian package, name it as OUTPUT_FILES/NOISE_TOMOGRAPHY/. We will add some parameter files later in this folder.

- Put the Matlab-generated-file `S_squared` in `OUTPUT_FILES/NOISE_TOMOGRAPHY/`.

  That's to say, you will have a file `OUTPUT_FILES/NOISE_TOMOGRAPHY/S_squared` in the SPECFEM3D Cartesian package.

- Create a file called `OUTPUT_FILES/NOISE_TOMOGRAPHY/irec_master_noise`. Note that this file is located in directory `OUTPUT_FILES/NOISE_TOMOGRAPHY/` as well. In general, all noise simulation related parameter files go into that directory. `irec_master_noise` contains only one integer, which is the ID of the 'master' receiver. For example, if this file contains 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 simulations, the `DATA/STATIONS` might contain receivers which are outside of our computational domains. Therefore, the integer in `irec_master_noise` is actually the ID in `DATA/STATIONS_FILTERED` (which is generated by `bin/xgenerate_databases`).

- Create a file called `OUTPUT_FILES/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 E/N/Z components respectively. Most often, the vertical component is used, and in those cases the three numbers should be 0, 0 and 1.

- Describe the noise direction and distributions in `src/shared/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 noise and a uniform distribution across the whole free surface of the model. It should be quite self-explanatory for modifications. Should you modify this part, you have to re-compile the source code.

8.2.2 Simulations

With all of the above done, we can finally launch our simulations. Again, please make sure that the `LOCAL_PATH` in `DATA/Par_file` is not globally shared. It is quite problematic to have a globally shared `LOCAL_PATH`, in terms of both disk storage and speed of I/O (we have to save the wavefields at the earth's surface at every time step).

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

- Step 1: simulation for generating wavefields
  `SIMULATION_TYPE=1`
  `NOISE_TOMOGRAPHY=1`
  `SAVE_FORWARD` not used, can be either .true. or .false.

- Step 2: simulation for ensemble forward wavefields
  `SIMULATION_TYPE=1`
  `NOISE_TOMOGRAPHY=2`
  `SAVE_FORWARD=.true.`

- Step 3: simulation for ensemble adjoint wavefields 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.

Note that it is better to run the three steps continuously within the same job, otherwise you have to collect the saved surface movies from the old nodes/CPUs to the new nodes/CPUs. This process varies from cluster to cluster and thus cannot be discussed here. Please ask your cluster administrator for information/configuration of the cluster you are using.

8.2.3 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` (which are 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 to other chapters 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 `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 `DATA/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 **?**.

8.3 Example

In order to illustrate noise simulations in an easy way, one example is provided in `EXAMPLES/noise_tomography/`. See `EXAMPLES/noise_tomography/README` for explanations.

Note, however, that they are created for a specific workstation (CLOVER@PRINCETON), which has at least 4 cores with 'mpif90' working properly.

If your workstation is suitable, you can run the example in `EXAMPLES/noise_tomography/` using:

```
./pre-processing.sh
```

Even if this script does not work on your workstation, the procedure it describes is universal. You may review the whole process described in the last section by following the commands in `pre-processing.sh`, which should contain enough explanations for all the commands.

Chapter 9

Graphics

9.1 Meshes

In case you used the internal mesher `xmeshfem3D` to create and partition your mesh, you can output mesh files in ABAQUS (.INP) and DX (.dx) format to visualize them. For this, you must set either the flag `CREATE_DX_FILES` or `CREATE_ABAQUS_FILES` to `.true.` in the mesher's parameter file `Mesh_Par_file` prior to running the mesher (see Chapter 3.2 for details). You can then use AVS (`www.avs.com`) or OpenDX (`www.opendx.org`) to visualize the mesh and MPI partition (slices).

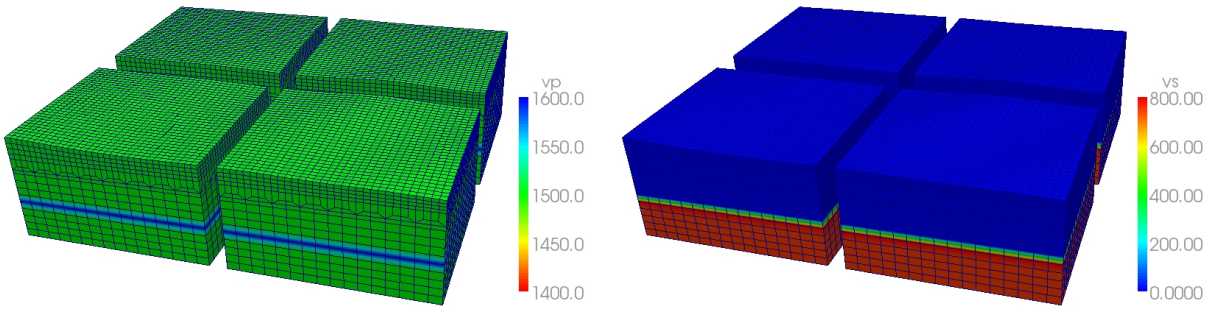


Figure 9.1: Visualization using Paraview of VTK files created by `xgenerate_databases` showing P- and S-wave velocities assigned to the mesh points. The mesh was created by `xmeshfem3D` for 4 processors.

You have also the option to visualize the distributed databases produced by `xgenerate_databases` using Paraview (`www.paraview.org`). For this, you must set the flag `SAVE_MESH_FILES` to `.true.` in the main parameter file `Par_file` (see Chapter 4.1 for details). This will create VTK files for each single partition. You can then use Paraview (`www.paraview.org`) to visualized these partitions.

9.2 Movies

To make a surface or volume movie of the simulation, set parameters `MOVIE_SURFACE`, `MOVIE_VOLUME`, `MOVIE_TYPE`, 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 displacement field if the parameter `SAVE_DISPLACEMENT` is set, otherwise the velocity field is saved. 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 × smallest period (see figure 5.1). When MOVIE_SURFACE = .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 × smallest period. As usual, for a point source one can set half duration in the CMTSOLUTION file to be 0.0 and HDUR_MOVIE = 0.0 in the Par_file to get the highest frequencies resolved by the simulation, but for a finite source one would keep all the half durations as prescribed by the finite source model and set HDUR_MOVIE = 0.0.

9.2.1 Movie Surface

When running xspecfem3D with the MOVIE_SURFACE flag turned on, the code outputs moviedata?????? files in the OUTPUT_FILES directory. There are several flags in the main parameter file Par_file that control the output of these movie data files (see section 4.1 for details): NTSTEP_BETWEEN_FRAMES to set the timesteps between frames, SAVE_DISPLACEMENT to save displacement instead of velocity, USE_HIGHRES_FOR_MOVIES to save values at all GLL point instead of element edges. In order to output additionally shakemaps, you would set the parameter CREATE_SHAKEMAP to .true..

The files are in a fairly complicated binary format, but there is a program provided to convert the output into more user friendly formats:

xcreate_movie_shakemap_AVS_DX_GMT From create_movie_shakemap_AVS_DX_GMT.f90, it outputs data in ASCII, OpenDX, or AVS format (also readable in ParaView). Before compiling the code, make sure you have the file surface_from_mesher.h in the OUTPUT_FILES/ directory. This file will be created by the solver run. Then type

```
make xcreate_movie_shakemap_AVS_DX_GMT
```

and run the executable xcreate_movie_shakemap_AVS_DX_GMT in the bin/ subdirectory. It will create visualization files in your format of choice. The code will prompt the user for input parameters.

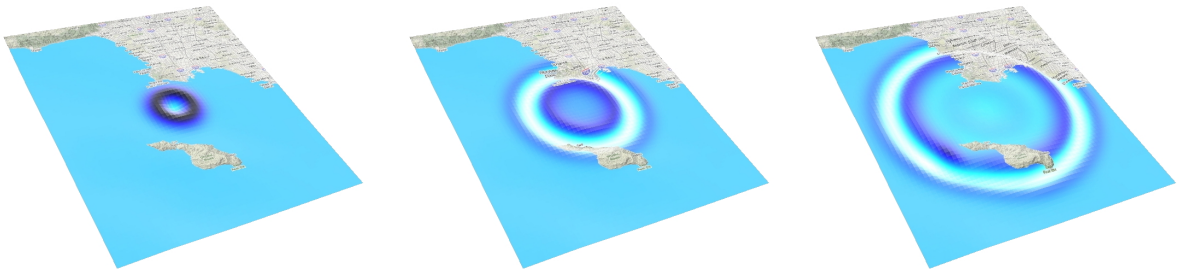


Figure 9.2: Visualization using AVS files created by xcreate_movie_shakemap_AVS_DX_GMT showing movie snapshots of vertical velocity components at different times.

The SPECFEM3D Cartesian code is running in near real-time to produce animations of southern California earthquakes via the web; see Southern California ShakeMovie®(www.shakemovie.caltech.edu).

9.2.2 Movie Volume

When running xspecfem3D with the MOVIE_VOLUME flag turned on, the code outputs several files in LOCAL_PATH specified in the main Par_file, e.g. in directory OUTPUT_FILES/DATABASES_MPI. The output is saved by each processor at the time interval specified by NTSTEP_BETWEEN_FRAMES. For all domains, the velocity field is output to files: proc??????_velocity_X_it??????.b proc??????_velocity_Y_it??????.bin and proc??????_velocity_Z_it??????.bin. For elastic domains,

the divergence and curl taken from the velocity field, i.e. $\nabla \cdot \mathbf{v}$ and $\nabla \times \mathbf{v}$, get stored as well: `proc??????_div_it??????.bin`, `proc??????_curl_X_t??????.bin`, `proc??????_curl_Y_it??????.bin` and `proc??????_curl_Z_it??????.bin`. The files denoted `proc??????_div_glob_it??????.bin` and `proc??????_curl_glob_it??????.bin` are stored on the global points only, all the other arrays are stored on all GLL points. Note that the components X/Y/Z can change to E/N/Z according to the `SUPPRESS_UTM_PROJECTION` flag (see also Appendix A and B).

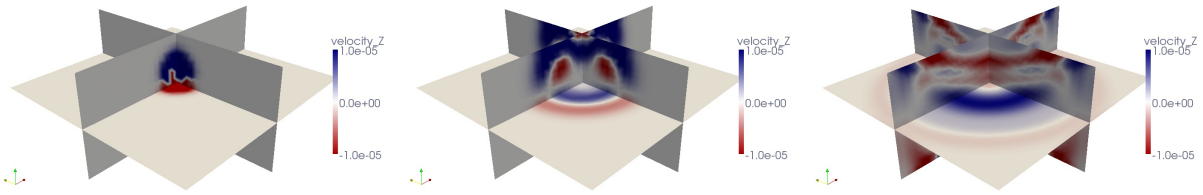


Figure 9.3: Paraview visualization using movie volume files (converted by `xcombine_vol_data` and `mesh2vtu.pl`) and showing snapshots of vertical velocity components at different times.

To visualize these files, we use an auxilliary program `combine_vol_data.f90` to combine the data from all slices into one mesh file. To compile it in the root directory, type:

```
make xcombine_vol_data
```

which will create the executable `xcombine_vol_data` in the directory `bin/`. To output the usage of this executable, type

```
./bin/xcombine_vol_data
```

without arguments.

`xcombine_vol_data` will combine the data on the different processors (located in `OUTPUT_FILES/DATABASES_MPI`), for a given quantity and a given iteration, to one file. For example, if you want to combine `velocity_Z`, for the iteration 400 on 4 processors, i.e. if you want to combine these files :

```
proc000000_velocity_Z_it000400.bin
proc000001_velocity_Z_it000400.bin
proc000002_velocity_Z_it000400.bin
proc000003_velocity_Z_it000400.bin
```

You have to go in the directory of the concerned example (where are the directories `DATA`, `OUTPUT_FILES`, etc.). Then, you can launch `xcombine_vol_data`, specifying the path where it is located. As an example, if the `DATA` and `OUTPUT_FILES` directories of the concerned example are in the root specfem3d directory, `xcombine_vol_data` is in `./bin`, and you have to type:

```
./bin/xcombine_vol_data 0 3 velocity_Z_it000400 ./OUTPUT_FILES/DATABASES_MPI ./OUTPUT_FILES 0
```

Here, 0 is the number of the first processor, 3 the number of the last one, `velocity_Z_it000400` the name of the files we want to combine without the prefix `proc000000*_`, `./OUTPUT_FILES/DATABASES_MPI` the directory where the files are located, `./OUTPUT_FILES` the directory where the combined file will be stored, and 0 is the parameter to create a low-resolution mesh file (1 for a high-resolution).

At the beginning of the source file `combine_vol_data.f90` (located in `src/auxiliaries/`), there is a logical constant parameter `USE_VTK_OUTPUT` whose default value is `.true.`, that which will provide the output combined file directly in the VTK format and you can use Paraview to visualize it.

If the value of this parameter is `.false.`, the output mesh file will have the name `velocity_Z_it000400.mesh`. Then we next have to convert the `.mesh` file into the VTU (Unstructured grid file) format which can be viewed in ParaView. For this task, you can use and modify the script `mesh2vtu.pl` located in directory `utils/Visualization/Paraview/`, for example:

```
mesh2vtu.pl -i velocity_Z_it000400.mesh -o velocity_Z_it000400.vtu
```

Notice that this Perl script uses a program `mesh2vtu` in the `utils/Visualization/Paraview/mesh2vtu` directory, which further uses the VTK (`http://www.vtk.org/`) run-time library for its execution. Therefore, make sure you have them properly set in the script according to your system.

Then, to do a movie with several iterations, you have to repeat this process for each iteration you want to put in your movie.

9.3 Finite-Frequency Kernels

The finite-frequency kernels computed as explained in Section 7.2 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.

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 on 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 `slice_number.pl` located in directory `utils/Visualization/Paraview/` can help to figure out the slice numbers that lie along the great circle path. It applies to meshes created with the internal mesher `xmeshfem3D`.

   (a) On machines where you have access to the script, copy the `Mesh_Par_file`, and `output_solver` files, and run:

   ```
   slice_number.pl Mesh_Par_file output_solver.txt slice_file
   ```

   which will generate a `slices_file`.

   (b) 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) You can use or modify the script `utils/copy_basin_database.pl` to accomplish this:

   ```
   utils/copy_database.pl slice_file lsf_machine_file filename [jobid]
   ```

   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.

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

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

   We use an auxilliary program `combine_vol_data.f90` to combine the kernel files from all slices into one mesh file.

   (a) Compile it in the root directory:

   ```
   make xcombine_vol_data
   ./bin/xcombine_vol_data slice_list filename input_dir output_dir high/low-resolution
   ```

   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.

(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.

(c) The output mesh file will have the name `filename_rho(alpha,beta).mesh`

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 this task, you can use and modify the script `mesh2vtu.pl` located in directory `utils/Visualization/Paraview/`, for example:

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

(b) Notice that this Perl script uses a program `mesh2vtu` in the `utils/Visualization/Paraview/mesh2vtu` directory, which further uses the VTK (`http://www.vtk.org/`) run-time library for its execution. Therefore, make sure you have them properly set in the script according to your system.

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 file `sr.vtk` located in the `OUTPUT_FILES/` directory to describe the source and receiver locations, which can also 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 bounding box 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 `global_slice_number.pl` script (either from the standard output or from `normal_plane.txt` file).

(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`).

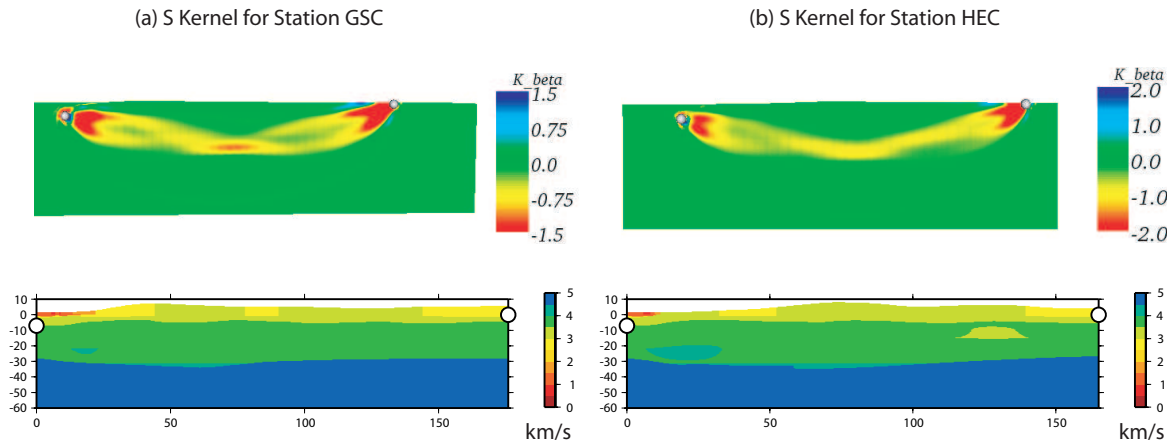


Figure 9.4: (a) Top Panel: Vertical source-receiver cross-section of the S-wave finite-frequency sensitivity kernel K_β for station GSC at an epicentral distance of 176 km from the September 3, 2002, Yorba Linda earthquake. Lower Panel: Vertical source-receiver cross-section of the 3D S-wave speed model used for the spectral-element simulations [**?**]. (b) The same as (a) but for station HEC at an epicentral distance of 165 km [**?**].

Chapter 10

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 `xgenerate_databases` and the `xspecfem3D` executables (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 `xgenerate_databases` and the `xspecfem3D` executables.
- 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.

Examples of job scripts can be found in the `utils/Cluster/` directory and can straightforwardly be modified and adapted to meet more specific running needs.

Chapter 11

Changing the Model

In this section we explain how to change the velocity model used for your simulations. These changes involve contributing specific subroutines that replace existing subroutines in the SPECFEM3D Cartesian package. Note that SPECFEM3D Cartesian can handle Earth models with material properties that vary within each spectral element.

11.1 Using external tomographic Earth models

To implement your own external tomographic model(s), you must provide your own external tomography file(s), and choose between two possible options:

(1) set in the Par_file the model parameter MODEL = tomo,
or for more user control:

(2) set in the Par_file the model parameter MODEL = default, define the negative material_ID identifier for each element in the file MESH/materials_file and use the following format in the file MESH/nummaterial_velocity_ when using CUBIT to construct your mesh (see also section 3.1.2):

```
domain_ID material_ID tomography elastic file_name 1
```

where:

domain_ID is 1 for acoustic or 2 for elastic materials,
material_ID a negative, unique identifier (i.e., -1,-2,...),
tomography keyword for tomographic material definition,
elastic keyword for elastic material definition,
file_name the name of the tomography file and 1 a positive unique identifier.

The external tomographic model is represented by a grid of points with assigned material properties and homogeneous resolution along each spatial direction x, y and z. The ASCII file file_name that describe the tomography should be located in the TOMOGRAPHY_PATH directory, set in the Par_file. The format of the file, as read from model_tomography.f90 located in the src/generate_databases directory, looks like Figure 11.1, where

ORIG_X, END_X are, respectively, the coordinates of the initial and final tomographic grid points along the x direction (in the mesh units, e.g., m);

ORIG_Y, END_Y respectively the coordinates of the initial and final tomographic grid points along the y direction (in the mesh units, e.g., m);

ORIG_Z, END_Z respectively the coordinates of the initial and final tomographic grid points along the z direction (in the mesh units, e.g., m);

SPACING_X, SPACING_Y, SPACING_Z the spacing between the tomographic grid points along the x, y and z directions, respectively (in the mesh units, e.g., m);

NX, NY, NZ the number of grid points along the spatial directions x, y and z, respectively; NX is given by [(END_X - ORIG_X)/SPACING_X]+1; NY and NZ are the same as NX, but for the y and z directions, respectively;

VP_MIN, VP_MAX, VS_MIN, VS_MAX, RHO_MIN, RHO_MAX the minimum and maximum values of the wave speed vp and vs (in $m\,s^{-1}$) and of the density rho (in $kg\,m^{-3}$); these values could be the actual limits of the tomographic parameters in the grid or the minimum and maximum values to which we force the cut of velocity and density in the model.

After the first four lines, in the file file_name the tomographic grid points are listed with the corresponding values of vp, vs and rho, scanning the grid along the x coordinate (from ORIG_X to END_X with step of SPACING_X) for each given y (from ORIG_Y to END_Y, with step of SPACING_Y) and each given z (from ORIG_Z to END_Z, with step of SPACING_Z).

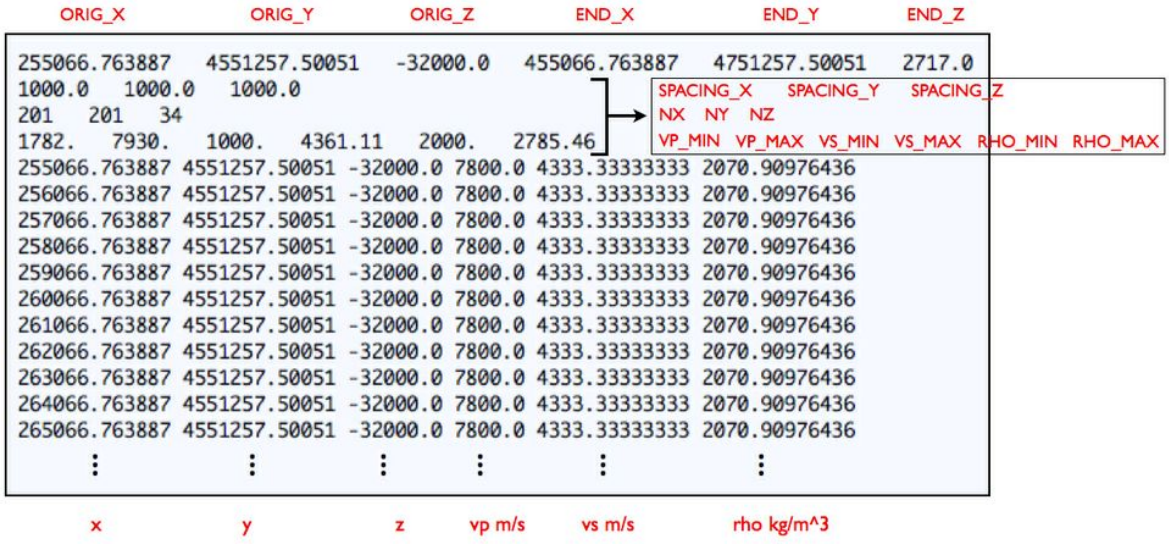


Figure 11.1: Tomography file file_name that describes an external Earth model. The coordinates x, y and z of the grid, their limits ORIG_X, ORIG_Y, ORIG_Z, END_X, END_Y, END_Z and the grid spacings SPACING_X, SPACING_Y, SPACING_Z should be in the units of the constructed mesh (e.g., UTM coordinates in meters).

The user can implement his own interpolation algorithm for the tomography model by changing the routine model_tomography.f90 located in the src/generate_databases/ directory. Moreover, for models that involve both fully defined materials and a tomography description, the nummaterial_velocity_file has multiple lines each with the corresponding suitable format described above.

Example: External tomography file with variable discretization intervals

The example in Figure 11.1 is for an external tomography file with uniform spacing in the x, y, and z directions. (Note that Δx, Δy, and Δz need not be the same as in the example.) In that example, the nummaterial_velocity_file is

```
2 -1 tomography elastic tomography_model.xyz 1
```

and the file tomography_model.xyz will need to reside in DATA/. All entries in the second column of materials_file will be -1, which means that each element in the mesh will be interpolated accordind to the values in tomography_model.xyz.

In some cases it may be desirable to have an external tomography model that is described in more than one file. For example, in cases like southern California, the length scale of variation in the structure of the wave speed model is much shorter in the sedimentary basin models within the upper 15 km. Therefore one might want to use an external tomography file that is sampled with, say, $\Delta x = 1000$ m, $\Delta y = 1000$ m, and $\Delta z = 250$ m in the uppermost 15 km, and then use $\Delta x = 2000$ m, $\Delta y = 2000$ m, and $\Delta z = 1000$ m below a depth of 15 km. If these intervals are chosen

appropriately, then it will result in a pair of external tomography file that is much smaller than the alternative of having a single file with the fine discretization. In this case `nummaterial_velocity_file` is

```
2 -1 tomography elastic file_above_15km.xyz 1
2 -2 tomography elastic file_below_15km.xyz 1
```

and the files `file_above_15km.xyz` and `file_below_15km.xyz` will need to reside in `DATA/`. All entries in the second column of `materials_file` will need to be either `-1` or `-2`, depending on whether the element is above or below 15 km depth. In this sense, the construction of the mesh and the discretization of the external model will generally be done in tandem.

Other more complicated discretizations can be used following the same procedure. (In particular, there is no limit on the number of different external files that are used.)

11.2 External (an)elastic Models

To use your own external model, you can set in the `Par_file` the model parameter `MODEL = external`. Three-dimensional acoustic and/or (an)elastic (attenuation) models may then be superimposed onto the mesh based upon your subroutine in `model_external_values.f90` located in directory `src/generate_databases`. The call to this routine would be as follows

```
call model_external_values(xmesh, ymesh, zmesh,  &
           rho, vp, vs, qmu_atten, iflag_aniso, idomain_id)
```

Input to this routine consists of:

`xmesh,ymesh,zmesh` location of mesh point

Output to this routine consists of:

`rho,vp,vs` isotropic model parameters for density ρ (kg/m^3), v_p (m/s) and v_s (m/s)

`qmu_atten` Shear wave quality factor: $0 < Q_\mu < 9000$

`iflag_aniso` anisotropic model flag, 0 indicating no anisotropy or 1 using anisotropic model parameters as defined in routine file `model_aniso.f90`

`idomain_id` domain identifier, 1 for acoustic, 2 for elastic, 3 for poroelastic materials.

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 (9000), consult `shared/constants.h`.

11.3 Anisotropic Models

To use your anisotropic models, you can either set in the `Par_file` the model parameter `ANISOTROPY` to `.true.` or set the model parameter `MODEL = aniso`. Three-dimensional anisotropic models may be superimposed on the mesh based upon the subroutine in file `model_aniso.f90` located in directory `src/generate_databases`. The call to this subroutine is of the form

```
call model_aniso(iflag_aniso, rho, vp, vs, &
           c11,c12,c13,c14,c15,c16,c22,c23,c24,c25,c26, &
           c33,c34,c35,c36,c44,c45,c46,c55,c56,c66)
```

Input to this routine consists of:

`iflag_aniso` flag indicating the type of the anisotropic model, i.e. 0 for no superimposed anisotropy, 1 or 2 for generic pre-defined anisotropic models.

rho,vp,vs reference isotropic model parameters for density ρ, v_p and v_s.

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

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

You can replace the `model_aniso.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.

11.4 Using external SEP models.

SEP models consists on two files per variables, one header file (`.H`) and one binary file (`.H@`). Among other information, the header file indicates, for each dimension, the offset (`o1`, `o2` and `o3` fields), the spatial step size (`d1`, `d2` and `d3` fields) and the number of samples (`n1`, `n2` and `n3` necessary to read the linked binary file (`in` field).

SEP model reading occures during database generation. It it the user responsability to ensure that proper interfaces between acoustic and elastic domains are set up during the meshing phase. For more information, refer to section 3.2.

In `DATA/Par_file`, **MODEL** should be set to **sep**, and **SEP_MODEL_DIRECTORY** should point to wherever your sep model files reside.

Note that in order not to surcharge the parameter file, SEP header files should be named `vp.H`, `vs.H` and `rho.H`. The `in` field in these binary files can be any path as long as it is relative to the previously set `SEP_MODEL_DIRECTORY`. We also assume that any dimensional value inside SEP header files is given in meters and that binary files are single precision floating point numbers stored in small endian format. There is currently only support for `vp`, `vs` and `rho` variables.

An example on how to use SEP models is given in: `EXAMPLES/meshfem3D_examples/sep_bathymetry`.

Chapter 12

Post-Processing Scripts

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

12.1 Process Data and Synthetics

In many cases, the SEM synthetics are calculated and compared to data 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 provided in the `utils/seis_process/` directory:

12.1.1 Data processing script `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*.BH?.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 the SAC and/or IASP91 to do the core operations; therefore make sure that the SAC and IASP91 packages are installed properly on your system, and that all the environment variables are set properly before running these scripts.

12.1.2 Synthetics processing script `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/*.BX?.semd
```

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 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 a detailed usage.

In order to convert between SAC format and ASCII files, useful scripts are provided in the subdirectories utils/sac2000_alpha_convert/ and utils/seis_process/asc2sac/.

12.1.3 Script `rotate.pl`

The original data and synthetics have three components: vertical (BHZ resp. BXZ), north (BHN resp. BXN) and east (BHE resp. BXE). However, for most seismology applications, transverse and radial components are also desirable. Therefore, we need to rotate the horizontal components of both the data and the synthetics to the transverse and radial direction, and rotate.pl can be used to accomplish this:

```
rotate.pl -l 0 -L 180 -d DATA/*.BHE.SAC.bp
rotate.pl -l 0 -L 180 SEM/*.BXE.semd.sac.bp
```

where the first command performs rotation on the SAC data obtained through Seismogram Transfer Program (STP) (http://www.data.scec.org/STP/stp.html), while the second command rotates the processed SEM synthetics.

12.2 Collect Synthetic Seismograms

The forward and adjoint simulations generate synthetic seismograms in the OUTPUT_FILES/ directory by default. For the forward simulation, the files are named like STA.NT.BX?.semd for two-column time series, or STA.NT.BX?.semd.sac for ASCII SAC format, where STA and NT are the station name and network code, and BX? stands for the component name. Please see the Appendix A and B for further details.

The adjont simulations generate synthetic seismograms with the name S?????.NT.S??.sem (refer to Section 7.1 for details). The kernel simulations output the back-reconstructed synthetic seismogram in the name STA.NT.BX?.semd, mainly for the purpose of checking the accuracy of the reconstruction. Refer to Section 7.2 for further details.

You do have further options to change this default output behavior, given in the main constants file constants.h located in src/shared/ directory:

SEISMOGRAMS_BINARY set to .true. to have seismograms written out in binary format.

WRITE_SEISMOGRAMS_BY_MASTER Set to .true. to have only the master process writing out seismograms. This can be useful on a cluster, where only the master process node has access to the output directory.

USE_OUTPUT_FILES_PATH Set to .false. to have the seismograms output to LOCAL_PATH directory specified in the main parameter file DATA/Par_file. In this case, you could collect the synthetics onto the frontend using the collect_seismo_lsf_multi.pl script located in the utils/Cluster/lsf/ directory. The usage of the script would be e.g.:

```
collect_seismo.pl machines DATA/Par_file
```

where machines is a file containing the node names and DATA/Par_file the parameter file used to extract the LOCAL_PATH directory used for the simulation.

12.3 Clean Local Database

After all the simulations are done, the seismograms are collected, and the useful database files are copied to the frontend, you may need to clean the local scratch disk for the next simulation. This is especially important in the case of kernel simulation, 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/:

```
cleanbase.pl machines
```

where machines is a file containing the node names.

12.4 Plot Movie Snapshots and Synthetic Shakemaps

12.4.1 Script movie2gif.gmt.pl

With the movie data saved in OUTPUT_FILES/ at the end of a movie simulation (MOVIE_SURFACE=.true.), you can run the 'create_movie_shakemap_AVS_DX_GMT' code to convert these binary movie data into GMT xyz files for futher processing. A sample script movie2gif.gmt.pl is provided to do this conversion, and then plot the movie snapshots in GMT, for example:

```
movie2gif.gmt.pl -m CMTSOLUTION -g -f 1/40 -n -2 -p
```

which for the first through the 40th movie frame, converts the moviedata files into GMT xyz files, interpolates them using the 'nearneighbor' command in GMT, and plots them on a 2D topography map. Note that '-2' and '-p' are both optional.

12.4.2 Script plot_shakemap.gmt.pl

With the shakemap data saved in OUTPUT_FILES/ at the end of a shakemap simulation (CREATE_SHAKEMAP=.true.), you can also run 'create_movie_shakemap_AVS_DX_GMT' code to convert the binary shakemap data into GMT xyz files. A sample script plot_shakemap.gmt.pl is provided to do this conversion, and then plot the shakemaps in GMT, for example:

```
plot_shakemap.gmt.pl data_dir type(1,2,3) CMTSOLUTION
```

where type=1 for a displacement shakemap, 2 for velocity, and 3 for acceleration.

12.5 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 10).

```
run_lsf.bash --gm-no-shmem --gm-copy-env remap_database old_machines 150
```

where old_machines is the LSF machine file used in the previous simulation, and 150 is the number of processors in total.

Chapter 13

Tomographic inversion using sensitivity kernels

One of the fundamental reasons for computing sensitivity kernels (Section 7.2) is to use them within a tomographic inversion. In other words, use recorded seismograms, make measurements with synthetic seismograms, and use the misfit between the two to iteratively improve the model described by (at least) $V_{\rm p}$, $V_{\rm s}$, and ρ as a function of space.

Whatever misfit function you use for the tomographic inversion (several examples in **?** and **?**), you will weight the sensitivity kernels with measurements. Furthermore, you will use as many measurements (stations, components, time windows) as possible per event; hence, we call these composite kernels "event kernels," which are volumetric fields representing the gradient of the misfit function with respect to one of the variables (e.g.,$V_{\rm s}$). The basic features of an adjoint-based tomographic inversion were illustrated in **?** and **?** using a conjugate-gradient algorithm; there are dozens of versions of gradient-based inversion algorithms that could alternatively be used. The tomographic inversion of **??** used SPECFEM3D Cartesian as well as several additional components which are also stored on the CIG svn server, described next.

The directory containing utilities for tomographic inversion using SPECFEM3D Cartesian (or other packages that evaluate misfit functions and gradients) is here on the CIG svn server:

```
 /cig/seismo/3D/ADJOINTTOMO/
flexwin/ -- FLEXWIN algorithm for automated picking of time windows
measureadj/ -- reads FLEXWIN output file and makes measurements, with
the option for computing adjoint sources iterateadj/ -- various tools
for iterative inversion (requires pre-computed \textquotedbl{}event
kernels\textquotedbl{})
```

This directory also contains a brief `README` file indicating the role of the three subdirectories, `flexwin` [**?**], `measure_adj`, and `iterate_adj`. The components for making the model update are there; however, there are no explicit rules for computing the model update, just as with any optimization problem. There are options for computing a conjugate gradient step, as well as a source subspace projection step. The workflow could use substantial "stitching together," and we welcome suggestions; undoubtedly, the tomographic inversion procedure will improve with time. **This is a work in progress.**

The best single file to read is probably: `ADJOINT_TOMO/iterate_adj/cluster/README`.

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 & Acknowledgments

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 (`http://www.opendx.org`) is open-source based on IBM Data Explorer, AVS (`http://www.avs.com`) is a trademark of Advanced Visualization Systems, and ParaView (`http://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:

MPI v 2.1, July 2012: Max Rietmann, Peter Messmer, Daniel Peter, Dimitri Komatitsch, Joseph Charles, Zhinan Xie: support for CUDA GPUs, better CFL stability for the Stacey absorbing conditions.

MPI v. 2.0, November 2010: Dimitri Komatitsch, Nicolas Le Goff, Roland Martin and Pieyre Le Loher, University of Pau, France, Daniel Peter, Jeroen Tromp and the Princeton group of developers, Princeton University, USA, and Emanuele Casarotti, INGV Roma, Italy: support for CUBIT meshes decomposed by SCOTCH; much faster solver using Michel Deville's inlined matrix products.

MPI v. 1.4 Dimitri Komatitsch, University of Pau, Qinya Liu and others, Caltech, September 2006: better adjoint and kernel calculations, faster and better I/Os on very large systems, many small improvements and bug fixes.

MPI v. 1.3 Dimitri Komatitsch, University of Pau, and Qinya Liu, Caltech, July 2005: serial version, regular mesh, adjoint and kernel calculations, ParaView support.

MPI v. 1.2 Min Chen and Dimitri Komatitsch, Caltech, July 2004: full anisotropy, volume movie.

MPI v. 1.1 Dimitri Komatitsch, Caltech, October 2002: Zhu's Moho map, scaling of V_s with depth, Hauksson's regional model, attenuation, oceans, movies.

MPI v. 1.0 Dimitri Komatitsch, Caltech, USA, May 2002: first MPI version based on global code.

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

Source and receiver locations

The SPECFEM3D Cartesian software code internally uses Cartesian coordinates. The given locations for sources and receiver locations thus may get converted. Sources and receiver locations are read in from the `CMTSOLUTION` (or `FORCESOLUTION`) and `STATIONS` files. Note that e.g. the `CMTSOLUTION` and `FORCESOLUTION` files denotes the location by "longitude/latitude/depth". We read in longitude as x coordinate, latitude as y coordinate.

In case the flag `SUPPRESS_UTM_PROJECTION` is set to `.false.` in the main parameter file (see Chapter 4), the x/y coordinates have to be given in degrees and are converted to Cartesian coordinates using a UTM conversion with the specified UTM zone.

The value for depth (given in km in `CMTSOLUTION` and `FORCESOLUTION`) or burial depth (given in m in `STATIONS`) is evaluated with respect to the surface of the mesh at the specified x/y location to find the corresponding z coordinate. It is possible to use this depth value directly as z coordinate by changing the flag `USE_SOURCES_RECVS_Z` to `.true.` in the file `constants.h` located in the `src/shared/` subdirectory.

Seismogram outputs

The seismogram output directions are given in Cartesian x/y/z directions and not rotated any further. Changing flags in `constants.h` in the `src/shared/` subdirectory only rotates the seismogram outputs if receivers are forced to be located at the surface (`RECVS_CAN_BE_BURIED_EXT_MESH` set to `.false.`) and the normal to the surface at the receiver location should be used (`EXT_MESH_RECV_NORMAL` set to `.true.`) as vertical. In this case, the outputs are rotated to have the vertical component normal to the surface of the mesh, x and y directions are somewhat arbitrary orthogonal directions along the surface.

For the labeling of the seismogram channels, see Appendix B. Additionally, we add labels to the synthetics using the following convention: For a receiver station located in an

elastic domain:

- `semd` for the displacement vector
- `semv` for the velocity vector
- `sema` for the acceleration vector

acoustic domain:

(please note that receiver stations in acoustic domains must be buried. This is due to the free surface condition which enforces a zero pressure at the surface.)

- `semd` for the displacement vector
- `semv` for the velocity vector
- `sema` for pressure which will be stored in the vertical component `Z`. Note that pressure in the acoustic media is isotropic and thus the pressure record would be the same in the other two component directions. We therefore use the other two seismogram components to store the acoustic potential in component `X` (or `N`) and the first time derivative of the acoustic potential in component `Y` (or `E`).

The seismograms are by default written out in ASCII-format to the `OUTPUT_FILES/` subdirectory by each parallel process. You can change this behavior by changing the following flags in the `constants.h` file located in the `src/shared/` subdirectory:

`SEISMOGRAMS_BINARY` set to `.true.` to have seismograms written out in binary format.

`WRITE_SEISMOGRAMS_BY_MASTER` Set to `.true.` to have only the master process writing out seismograms. This can be useful on a cluster, where only the master process node has access to the output directory.

`USE_OUTPUT_FILES_PATH` Set to `.false.` to have the seismograms output to `LOCAL_PATH` directory specified in the main parameter file `DATA/Par_file`.

Appendix B

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 format for channel codes, which are represented by three letters, such as `LHN`, `BHZ`, etc. In older versions of the SPECFEM3D Cartesian package, a common format was used for the channel codes of all seismograms, which was `BHE/BHN/BHZ` for three components. To avoid confusion when comparison are made to observed data, we are now using the FDSN convention (`http://www.fdsn.org/`) for SEM seismograms. In the following, we give a brief explanation of the FDSN convention used by IRIS, 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 B.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. IRIS 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.

`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. SPECFEM3D Cartesian uses the traditional orientation code `E/N/Z` (East-West, North-South, Vertical) for three components when a UTM projection is used. If the UTM conversion is suppressed, i.e. the flag `SUPPRESS_UTM_PROJECTION` is set to `.true.`, then the three components are labelled `X/Y/Z` according to the Cartesian reference frame.

`EXAMPLE:` The sampling rate is given by `DT` in the main parameter file `Par_file` located in the `DATA/` subdirectory. Depending on the resolution of your simulations, if you choose a sampling rate greater than 0.01 s and less than 1 s, a seismogram recording displacements on the vertical component of a station `ASBS` (network `AZ`) will be named `ASBS.AZ.MXZ.semd.sac`, whereas it will be `ASBS.AZ.BXZ.semd.sac`, if the sampling rate is greater than 0.0125 and less equal to 0.1 s.

Band code	Band type	Sampling rate (sec)	Corner period (sec)
F	...	> 0.0002 to <= 0.001	≥ 10 sec
G	...	> 0.0002 to <= 0.001	< 10 sec
D	...	> 0.001 to <= 0.004	< 10 sec
C	...	> 0.001 to <= 0.004	≥ 10 sec
E	Extremely Short Period	<= 0.0125	< 10 sec
S	Short Period	<= 0.1 to > 0.0125	< 10 sec
H	High Broad Band	<= 0.0125	>= 10 sec
B	Broad Band	<= 0.1 to > 0.0125	>= 10 sec
M	Mid Period	< 1 to > 0.1	
L	Long Period	1	
V	Very Long Period	10	
U	Ultra Long Period	100	
R	Extremely Long Period	1000	
P	On the order of 0.1 to 1 day	<= 100000 to > 10000	
T	On the order of 1 to 10 days	<= 1000000 to > 100000	
Q	Greater than 10 days	> 1000000	
A	Administrative Instrument Channel	variable	NA
O	Opaque Instrument Channel	variable	NA

Figure B.1: The 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 SPECFEM3D Cartesian, and the band codes used to name SEM seismograms are denoted in red.

Appendix C

Troubleshooting

FAQ

configuration fails:

Examine the log file 'config.log'. It contains detailed informations. In many cases, the paths to these specific compiler commands F90, CC and MPIF90 will not 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 stating:

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

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

after executing `xmeshfem3D` I've got elements with skewness of 81% percent, what does this mean: Look at the skewness table printed in the `output_mesher.txt` file after executing `xmeshfem3D` for the example given in `EXAMPLES/meshfem3D_examples/simple_model/`:

```
...
 histogram of skewness (0. good - 1. bad):
 0.0000000E+00 - 5.0000001E-02 27648 81.81818 %
 5.0000001E-02 - 0.1000000 0 0.0000000E+00 %
 ...
```

The first line means that you have 27,648 elements with a skewness value between 0 and 0.05 (which means the element is basically not skewed, just plain regular hexahedral element). The total number of elements you have in this mesh is (see in the `output_mesher.txt` file a bit further down):

```
...
 total number of elements in entire mesh: 33792
 ...
```

which gives you that: 27,648 / 33,792 $\sim$ 81.8 % of all elements are not skewed, i.e. regular elements. a fantastic value :)

The histogram lists for this mesh also some stronger skewed elements, for example the worst ones belong to:

```
...
 0.6000000 - 0.6500000 2048 6.060606 %
 ...
```

about 6 % of all elements have distortions with a skewness value between 0.6 and 0.65. The skewness values give you a hint of how good your mesh is. In an ideal world, you would want to have no distortions, just like the 81% from above. Those elements give you the best approximate values by the GLL quadrature used in the spectral-element method. However, having weakly distorted elements is still fine and the solutions are still accurate enough. So empirically, values up to around 0.7 are tolerable, above that you should consider remeshing...

To give you an idea why some of the elements are distorted, see the following figure C.1 of the mesh you obtain in the example `EXAMPLES/meshfem3D_examples/simple_model/`.

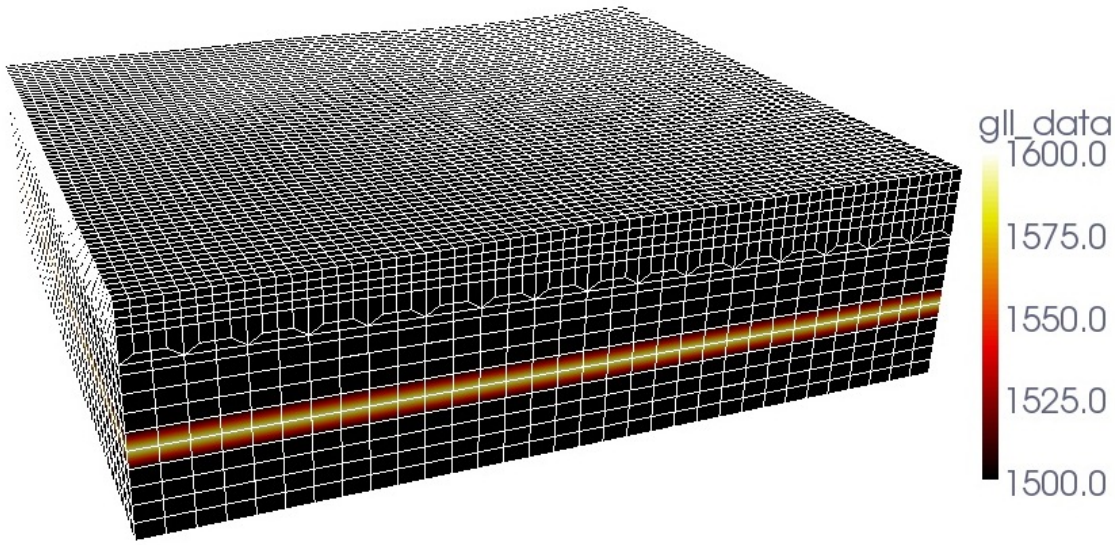


Figure C.1: Paraview visualization using the mesh vtk-files for the example given in `EXAMPLES/meshfem3D_examples/simple_model/`.

You will see that the mesh contains a doubling layer, where we stitch elements together such that the size of two elements will transition to the size of one element (very useful to keep the ratio of wavespeed / element_size about constant). Those elements in this doubling layer have higher skewness values and make up those 6 % in the histogram.

the code gives following error message "need at least one receiver": This means that no stations given in the input file `DATA/STATIONS` could be located within the dimensions of the mesh. This can happen for example when the mesh was created with the in-house mesher `xmeshfem3D` while using the Universal Transverse Mercator

(UTM) projection but the simulation with xspecfem3D was suppressing this projection from latitude/longitude to x/y/z coordinates.

In such cases, try to change your DATA/Par_file and set e.g.:

```
SUPPRESS_UTM_PROJECTION = .false.
```

to be the same in Mesh_Par_file and Par_file. This flag should be identical when using the in-house mesher xmeshfem3D, xgenerate_databases and xspecfem3D together to run simulations.

The flag determines if the coordinates you specify for your source and station locations are given as lat/lon degrees and must be converted to UTM coordinates. As an example, if you use .false. within Mesh_Par_file then you create a mesh with xmeshfem3D using the UTM projection from lat/lon as input format to UTM projected coordinates to store the mesh point positions, which is fine. The error then may occur if in the Par_file you have this set to .true. so that the xgenerate_databases and xspecfem3D suppress the UTM projection and assume that all coordinates you use now for source and receiver locations are given in meters (that is, converted) already. So it won't find the specified locations in the used mesh. As a solutions, just change the flag in Par_file to be the same as in Mesh_Par_file and rerun xgenerate_databases and xspecfem3D to make sure that your simulation works fine.

I get the following error message "forward simulation became unstable and blew up": In most cases this means that your time step size DT is chosen too big. Look at your files output_mesher.txt or output_solver.txt created in the folder OUTPUT_FILES. In these output files, find the section:

```
...
 *********************************************
 *** Verification of simulation parameters ***
 *********************************************
 ...
 *** Minimum period resolved = 4.308774
 *** Maximum suggested time step = 6.8863556E-02
 ...
 ...
```

then change DT in the DATA/Par_file to be somewhere close to the maximum suggested time step. In the example above:
DT = 0.05d0
would (most probably) work fine. It could be also bigger than the 0.068 s suggested. This depends a bit on the distortions of your mesh elements. The more regular they are, the bigger you can choose DT. Just play with this value a bit and see when the simulation becomes stable ...

Appendix D

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.