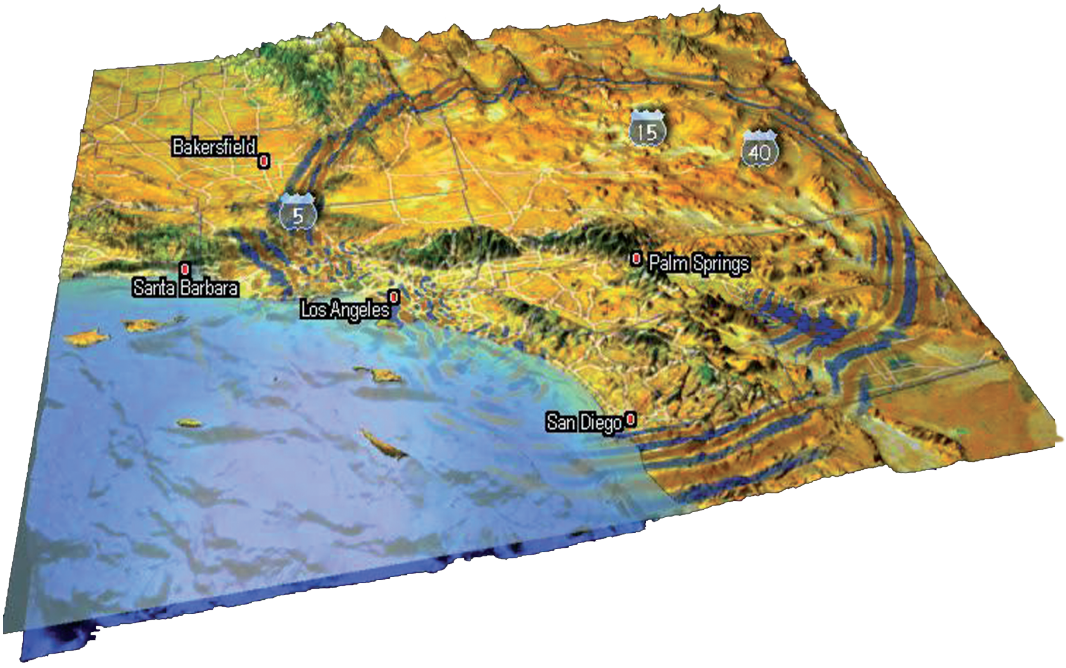
COMPUTATIONAL INFRASTRUCTURE FOR GEODYNAMICS (CIG)
PRINCETON UNIVERSITY (USA)
UNIVERSITY OF PAU, CNRS and INRIA (FRANCE)

SPECFEM 3D

User Manual
Version 2.0.0

Piero Basini
Céline Blitz
Ebru Bozdağ
Emanuele Casarotti
Min Chen
Vala Hjörleifsdóttir
Emiljana Jorgji
Sue Kientz
Dimitri Komatitsch
Jesús Labarta
Nicolas Le Goff
Pieyre Le Loher
Qinya Liu
Yang Luo
Alessia Maggi
Federica Magnoni
Roland Martin
Dennis McRitchie
Matthias Meschede
David Michéa
Hom Nath Gharti
Tarje Nissen-Meyer
Daniel Peter
Brian Savage
Bernhard Schuberth
Anne Sieminski
Leif Strand
Carl Tape
Jeroen Tromp
Hejun Zhu



SPECFEM3D
User Manual

Version 2.0.0
"Sesame"

December 6, 2010

Authors

The SPECFEM3D package was first developed by Dimitri Komatitsch and Jeroen Tromp at Harvard University and Caltech, USA, starting in 1998, based in part on earlier work by Dimitri Komatitsch and Jean-Pierre Vilotte at Institut de Physique du Globe (IPGP) in Paris, France from 1994 to 1997.

Since then it has been developed and maintained by a development team: in alphabetical order, Piero Basini, Céline Blitz, Ebru Bozdağ, Emanuele Casarotti, Min Chen, Hom Nath Gharti, Vala Hjörleifsdóttir, Emiljana Jorgji, Sue Kientz, Dimitri Komatitsch, Jesús Labarta, Nicolas Le Goff, Pieyre Le Loher, Qinya Liu, Yang Luo, Alessia Maggi, Federica Magnoni, Roland Martin, Dennis McRitchie, Matthias Meschede, David Michéa, Tarje Nissen-Meyer, Daniel Peter, Brian Savage, Bernhard Schuberth, Anne Sieminski, Leif Strand, Carl Tape, Jeroen Tromp, 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`).

Contents

Chapter 1

Introduction

The software package SPECFEM3D 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 [Bernardi et al., 1994, Kopriva et al., 2002, Chaljub et al., 2003, Kopriva, 2006, Wilcox et al., 2010]; it is then a particular case of the discontinuous Galerkin technique [Reed and Hill, 1973, Arnold, 1982, Falk and Richter, 1999, Hu et al., 1999, Cockburn et al., 2000, Giraldo et al., 2002, Rivière and Wheeler, 2003, Monk and Richter, 2005, Grote et al., 2006, Ainsworth et al., 2006, Bernacki et al., 2006, Dumbser and Käser, 2006, De Basabe et al., 2008, Wilcox et al., 2010, De Basabe and Sen, 2010, Étienne et al., 2010], with optimized efficiency because of its tensorized basis functions. In particular, it can accurately handle very distorted mesh elements [Oliveira and Seriani, 2011]. Note that in most geological models in the context of seismic wave propagation studies (except for fault dynamic rupture studies) a discontinous mesh is not needed because material property contrasts are not drastic and thus a continuous formulation is sufficient; conforming mesh doubling bricks can efficiently handle mesh size variations [Komatitsch and Tromp, 2002a, Komatitsch et al., 2004].

For a detailed introduction to the SEM as applied to regional seismic wave propagation, please consult Komatitsch and Vilotte [1998], Komatitsch and Tromp [1999], Chaljub et al. [2007], Tromp et al. [2008] and in particular Komatitsch et al. [2004]. A detailed theoretical analysis of the dispersion and stability properties of the SEM is available in Cohen [2002], De Basabe and Sen [2007] and Seriani and Oliveira [2007].

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 Chen and Tromp [2007]) as well as lateral variations in attenuation [Savage et al., 2010]. Adjoint capabilities and finite-frequency kernel simulations are included [Liu and Tromp, 2006, Tromp et al., 2008, Fichtner et al., 2009, Virieux and Operto, 2009].

All SPECFEM3D software is written in Fortran90 with full portability in mind, and conforms strictly to the Fortran95 standard. It uses no obsolete or obsolescent features of Fortran77. The package uses parallel programming based upon the Message Passing Interface (MPI) [Gropp et al., 1994, Pacheco, 1997].

SPECFEM3D won the Gordon Bell award for best performance at the SuperComputing 2003 conference in Phoenix, Arizona (USA) (see Komatitsch et al. [2003] and `www.sc-conference.org/sc2003/nr_finalaward.html`). It was a finalist again in 2008 for a run at 0.16 petaflops (sustained) on 149,784 processors of the 'Jaguar' Cray XT5 system at Oak Ridge National Laboratories (USA) [Carrington et al., 2008]. It also won the BULL Joseph Fourier supercomputing award in 2010.

The next release of the code will include support for GPU graphics card acceleration [Komatitsch et al., 2009, 2010a, Michéa and Komatitsch, 2010, Komatitsch, 2011] as well as Convolutional or Auxiliary Differential Equation Perfectly Matched absorbing Layers (C-PML or ADE-PML) [Komatitsch and Martin, 2007, Martin et al., 2008b,c, Martin and Komatitsch, 2009, Martin et al., 2010]. It will also use the PT-SCOTCH parallel library for mesh partitioning.

1.1 Citation

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

Numerical simulations in general

Forward simulations are described in detail in Tromp et al. [2008], Vai et al. [1999], Komatitsch et al. [2009, 2010a,b], Chaljub et al. [2007], Madec et al. [2009], Komatitsch et al. [2010c], Carrington et al. [2008], Tromp et al. [2010a], Komatitsch et al. [2002], Komatitsch and Tromp [2002a,b, 1999] or Komatitsch and Vilotte [1998]. Additional aspects dealing with adjoint simulations are described in Tromp et al. [2005], Liu and Tromp [2006], Tromp et al. [2008], Liu and Tromp [2008], Tromp et al. [2010a]. Domain decomposition is explained in detail in Martin et al. [2008a], and excellent scaling up to 150,000 processor cores is shown for instance in Carrington et al. [2008], Komatitsch et al. [2008], Martin et al. [2008a], Komatitsch et al. [2010a,b], Komatitsch [2011].

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

Southern California simulations

Komatitsch et al. [2004], Krishnan et al. [2006a,b].

If you use the 3D southern California model, please cite Süss and Shaw [2003] (Los Angeles model), Lovely et al. [2006] (Salton Trough), and Hauksson [2000] (southern California). The Moho map was determined by Zhu and Kanamori [2000]. The 1D SoCal model was developed by Dreger and Helmberger [1990].

Anisotropy

Chen and Tromp [2007], Ji et al. [2005], Chevrot et al. [2004], Favier et al. [2004], Ritsema et al. [2002], Tromp and Komatitsch [2000].

Attenuation

Savage et al. [2010], Komatitsch and Tromp [2002a, 1999].

Topography

Lee et al. [2009b,a, 2008], Godinho et al. [2009], van Wijk et al. [2004].

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

The SPECFEM3D software package comes in a gzipped tar ball. In the directory in which you want to install the package, type

```
tar -zxvf SPECFEM3D_V2.0.0.tar.gz
```

The directory `SPECFEM3D` will then contain the source code. 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 Fortran90 compiler and MPI package:

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

The SPECFEM3D software package relies on the SCOTCH library to partition meshes created with CUBIT. Note that we use CUBIT 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 [Oliveira and Seriani, 2011].

The SCOTCH library [Pellegrini and Roman, 1996] 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/`. It is more recent than METIS, actively maintained and performs better in many cases. A recent version of its source code is provided in directory `src/decompose_mesh_SCOTCH/scotch_5.1.10b`. In case no SCOTCH libraries can be found on the system, the configuration will bundle this version 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
```

To compile a serial version of the code for small meshes that fit 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 Fortran90 compiler.

`MPIF90` Path to MPI Fortran90.

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

`FLAGS_CHECK` Compiler flag for non-critical subroutines.

`FLAGS_NO_CHECK` Compiler flag for creating fast, production-run code for critical subroutines.

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. The default is single precision mode because this 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 a new system, it is definitely worth experimenting with single versus double precision simulations to determine which is faster. Note that on many current processors (e.g., Intel, AMD, IBM Power), single precision calculations are often significantly faster; the difference can typically be 10% to 25%. It is therefore often worth using single precision if you can. We recommend running the same calculation once in single precision and in double precision on your system and then comparing the seismograms. If they are identical, you should probably select single precision for your future runs.

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.

Note that if you run very large meshes on a relatively small number of processors, the memory size needed on each processor might become greater than 2 gigabytes, which is the upper limit for 32-bit addressing; in this case, on some compilers you may need to add "`-mcmodel=medium`" to the compiler options otherwise the compiler will display an error message; on an IBM machine with the `xlf` and `xlc` compilers, using `-q64` is usually sufficient.

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.

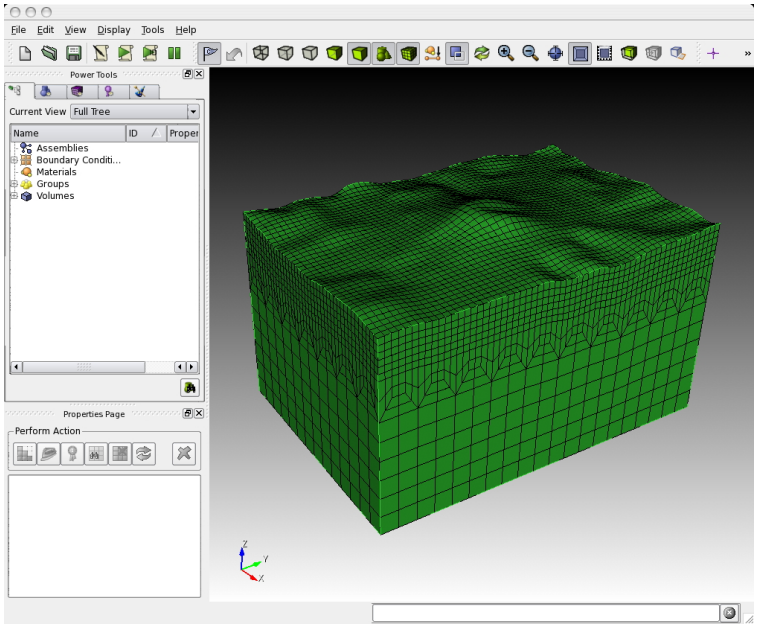


Figure 3.1: Example of the graphical user interface of CUBIT. The hexahedral mesh shown in the main display consists of a hexahedral discretization of a single volume with topography.

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 file format and

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

Examples are provided in the SPECFEM3D 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 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`.

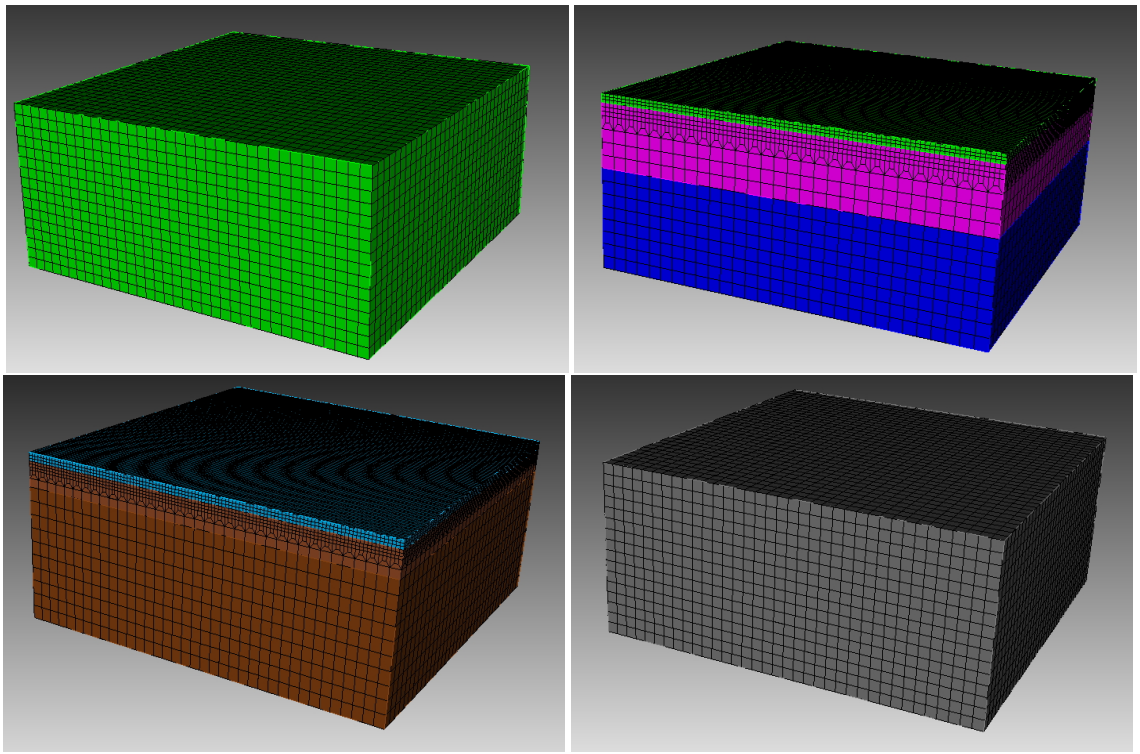


Figure 3.2: Screenshots of the CUBIT examples provided in subdirectory `examples/`: homogeneous halfspace (top-left), layered halfspace (top-right), water layered halfspace (bottom-left) and tomographic model (bottom-right).

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`).

3.1.2 Exporting the Mesh with `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 `boundary_definition.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.

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. If the export was successful, you should find the following files in a subdirectory `MESH/`:

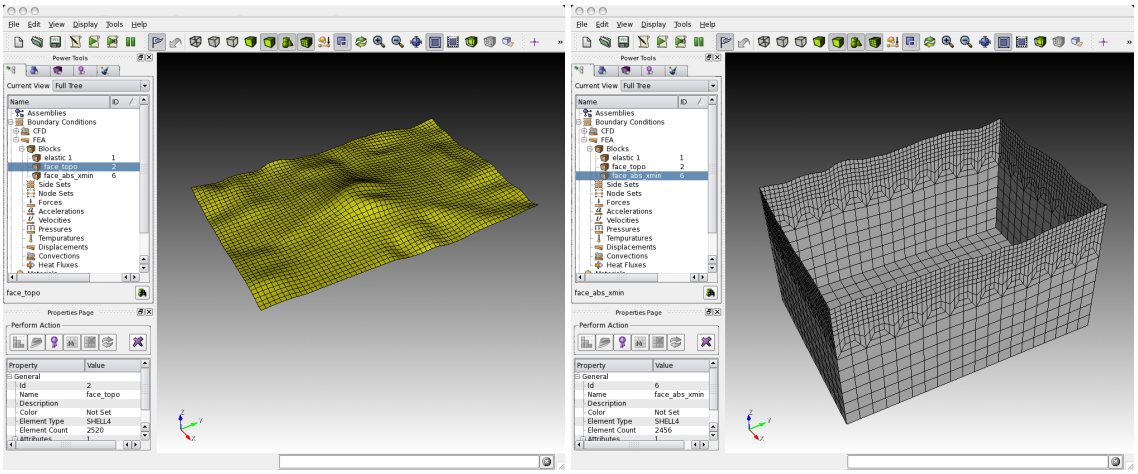


Figure 3.3: Example of the block definitions for the free surface 'face_topo' (left) and the absorbing boundaries, defined in a single block 'face_abs_xmin' (right) in CUBIT.

nummaterial_velocity_file Defines the material properties. For fully defined materials, the formats is:

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

where `domain_ID` is 1 for acoustic or 2 for elastic 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. For a tomographic model, the material definition format is:

```
domain_ID material_ID tomography name
```

where `domain_ID` is 1 for acoustic or 2 for elastic materials, `material_ID` a negative, unique identifier, `tomography` keyword for tomographic material definition and `name` the name of the tomography file. The name is not used so far, rather change the filename defined in the file `model_tomography.f90` located in the `src/generate_databases/` directory.

materials_file Contains the material associations for each element.

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

mesh_file Contains the mesh element connectivity.

free_surface_file Contains the free surface connectivity.

absorbing_surface_file_xmax Contains the surface connectivity of the absorbing boundary surface at the Xmax.

absorbing_surface_file_xmin Contains the surface connectivity of the absorbing boundary surface at the Xmin.

absorbing_surface_file_ymax Contains the surface connectivity of the absorbing boundary surface at the Ymax.

absorbing_surface_file_ymin Contains the surface connectivity of the absorbing boundary surface at the Ymin.

absorbing_surface_file_bottom Contains the surface connectivity of the absorbing boundary surface at the bottom.

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

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`. 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. 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 directory `src/check_mesh_quality_CUBIT_Abaqus`, type

```
./compile_all.csh
```

and use

```
xcheck_mesh_quality_CUBIT_Abaqus
```

to generate an AVS output file (`AVS_meshquality.inp` in AVS UCD format) or OpenDX output file (`DX_meshquality.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. 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_SCOTCH`

The SPECFEM3D 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_SCOTCH` depends on the SCOTCH library [Pellegrini and Roman, 1996], 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/`. It is more recent than METIS, actively maintained and performs better in many cases. A recent version of its source code is provided in directory `src/decompose_mesh_SCOTCH/scotch_5.1.10b`.

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/scotch_5.1.10b`. This version gets bundled with the compilation of the SPECFEM3D 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` .

When you are ready to compile, in the main directory type '`make decompose_mesh_SCOTCH`'. If all paths and flags have been set correctly, the executable `bin/xdecompose_mesh_SCOTCH` should be produced.

The partitioning is done in serial for now (in the next release we will provide a parallel version of that code), the synopsis is:

```
./bin/xdecompose_mesh_SCOTCH nparts input_directory output_directory
```

where

- `nparts` is the number of partitions, i.e., the number of cores for the parallel simulations,

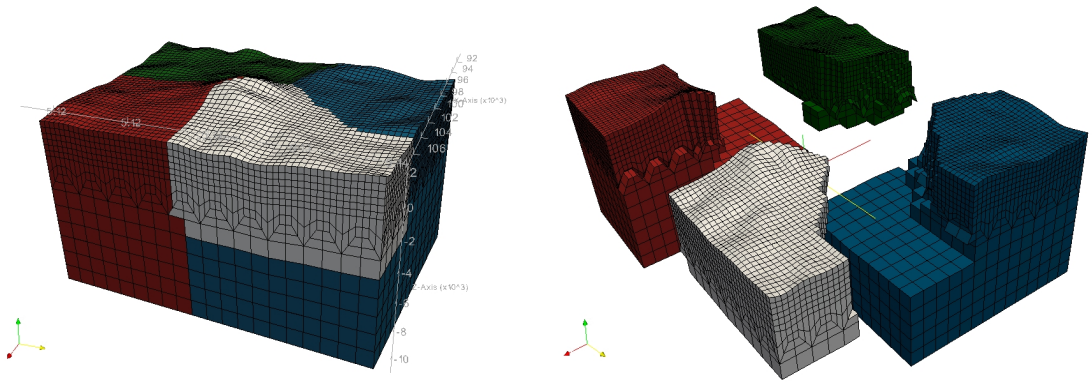


Figure 3.4: Example of a mesh partitioning onto four cores. Each single core partition is colored differently. The executable `xdecompose_mesh_SCOTCH` can equally distribute the mesh on any arbitrary number of cores. Domain decomposition is explained in detail in Martin et al. [2008a], and excellent scaling up to 150,000 processor cores in shown for instance in Carrington et al. [2008], Komatitsch et al. [2008], Martin et al. [2008a], Komatitsch et al. [2010a,b], Komatitsch [2011].

- `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 `in_out_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.

In the main directory type '`make meshfem3D`'. 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 `in_data_files/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 Komatitsch and Vilotte [1998], Komatitsch and Tromp [1999] and Komatitsch et al. [2004].

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).

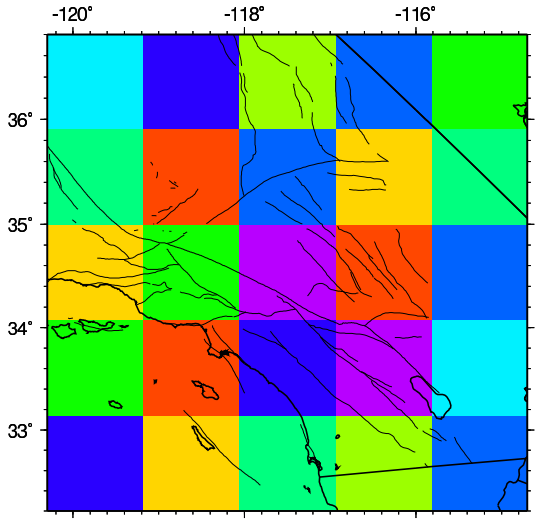


Figure 3.5: For parallel computing purposes, the model block is subdivided in NPROC_XI × NPROC_ETA slices of elements. In this example we use $5^2 = 25$ processors.

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..

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 × a multiple of NPROC_XI defined below. Based upon benchmarks against semi-analytical discrete wavenumber synthetic seismograms [Komatitsch et al., 2004], 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 × 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 8.1 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 in_out_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

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

```
XI_begin XI_end ETA_begin ETA_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 9 for information about running the code on a system with a scheduler, e.g., LSF.

Now that you have set the appropriate parameters in the `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 9 for information about running the code on a system with a scheduler, e.g., LSF.

Mesher output is provided in the `in_out_files/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
```

Chapter 4

Creating the Distributed Databases

After using `xmeshfem3D` or `xdecompose_mesh_SCOTCH`, 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.

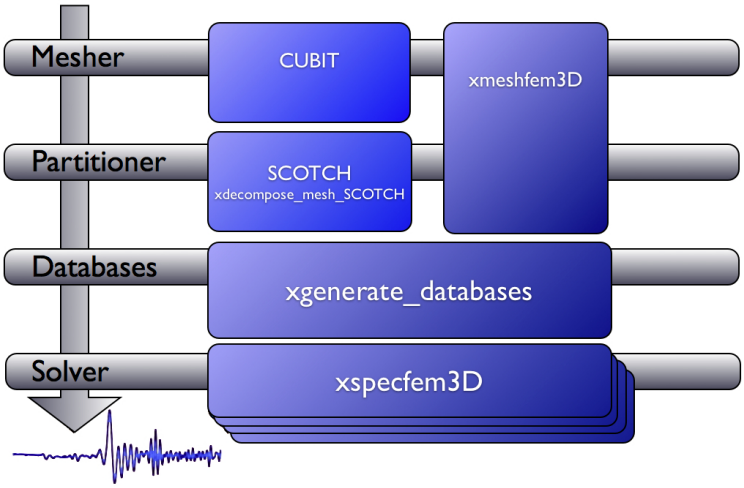


Figure 4.1: Schematic workflow for a SPECFEM3D simulation. The exectable `xgenerate_databases` creates the GLL mesh points and assigns specific model parameters.

In the main directory type '`make generate_databases`'. Input for the program is provided through the main parameter file `Par_file`, which resides in the subdirectory `in_data_files/`. 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 `in_data_files/`:

`SIMULATION_TYPE` is set to 1 for forward simulations, 2 for adjoint simulations (see Section 6.2) and 3 for kernel simulations (see Section 8.3).

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

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.

OCEANS Set to .true. if the effect of the oceans on seismic wave propagation should be incorporated based upon the approximate treatment discussed in Komatitsch and Tromp [2002b]. 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.

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 Komatitsch and Tromp [1999, 2002a] 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/.

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 et al. [2003]).

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.

ABSORBING_CONDITIONS Set to .true. to turn on Clayton-Enquist absorbing boundary conditions (see Komatitsch and Tromp [1999]).

MOVIE_SURFACE Set to .false., unless you want to create a movie of seismic wave propagation on the Earth's surface. Turning this option on generates large output files. See Section 8.2 for a discussion on the generation of movies. This feature is only relevant for the solver.

MOVIE_VOLUME Set to .false., unless you want to create a movie of seismic wave propagation in the Earth's interior. Turning this option on generates huge output files. See Section 8.2 for a discussion on the generation of movies. This feature is only relevant for the solver.

NTSTEP_BETWEEN_FRAMES Determines the number of timesteps between movie frames. Typically you want to save a snapshot every 100 timesteps. The smaller you make this number the more output will be generated! See Section 8.2 for a discussion on the generation of movies. This feature is only relevant for the solver.

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.

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.

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 8.1 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_SCOTCH 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 in_out_files/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.

PRINT_SOURCE_TIME_FUNCTION Turn this flag on to print information about the source time function in the file in_out_files/OUTPUT_FILES/plot_source_time_function.txt. This feature is only relevant for the solver.

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 in_out_files/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 in_out_files/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 specfem3D'. 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 in_data_files/ directory to run:

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

CMTSOLUTION the earthquake 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 ABSORBING_CONDITIONS
- the movie control parameters MOVIE_SURFACE, MOVIE_VOLUME, and NTSTEPS_BETWEEN_FRAMES
- the ShakeMap®option CREATE_SHAKEMAP
- the output information parameters 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:

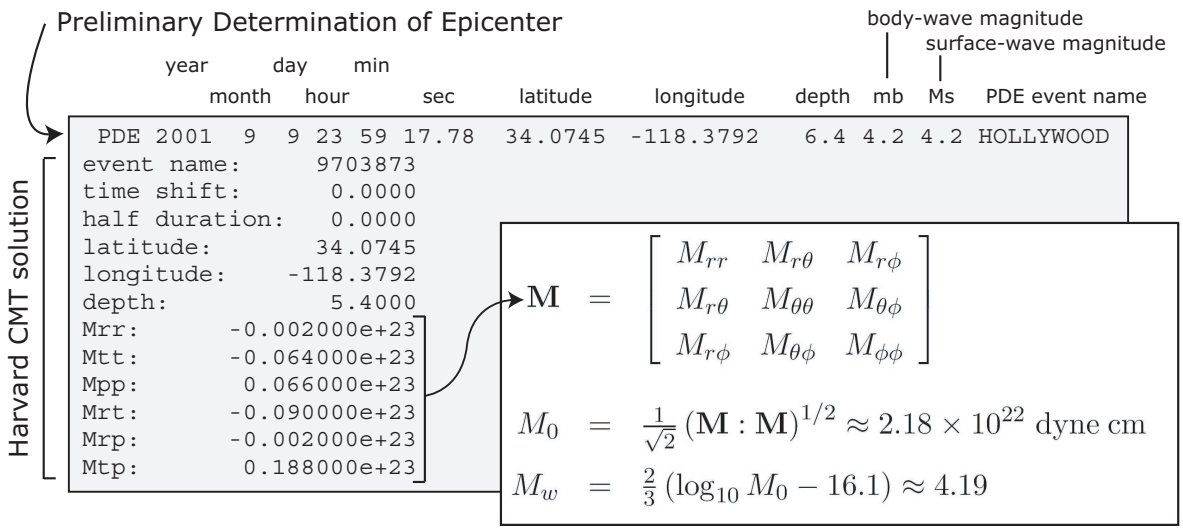


Figure 5.1: `CMTSOLUTION` file based on the format from the Harvard CMT catalog. $\mathbf{M}$ is the moment tensor, M_0 is the seismic moment, and M_w is the moment magnitude.

The `CMTSOLUTION` 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 10.1).
- For point-source simulations (see finite sources, page 22) we recommend setting the source half-duration parameter `half duration` equal to zero, which corresponds to simulating a step source-time function, i.e., a moment-rate function that is a delta function. If `half duration` is not set to zero, the code will use a Gaussian (i.e., a signal with a shape similar to a 'smoothed triangle', as explained in Komatitsch and Tromp [2002a] 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 10.1). Komatitsch and Tromp [2002a] 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 convolve_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.

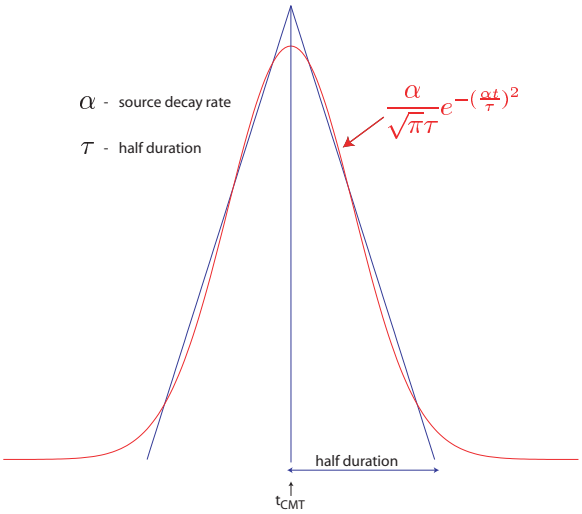


Figure 5.2: Comparison of the shape of a triangle and the Gaussian function actually used.

Centroid latitude and longitude should be provided in geographical coordinates. The code converts these coordinates to geocentric coordinates [Dahlen and Tromp, 1998]. 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.

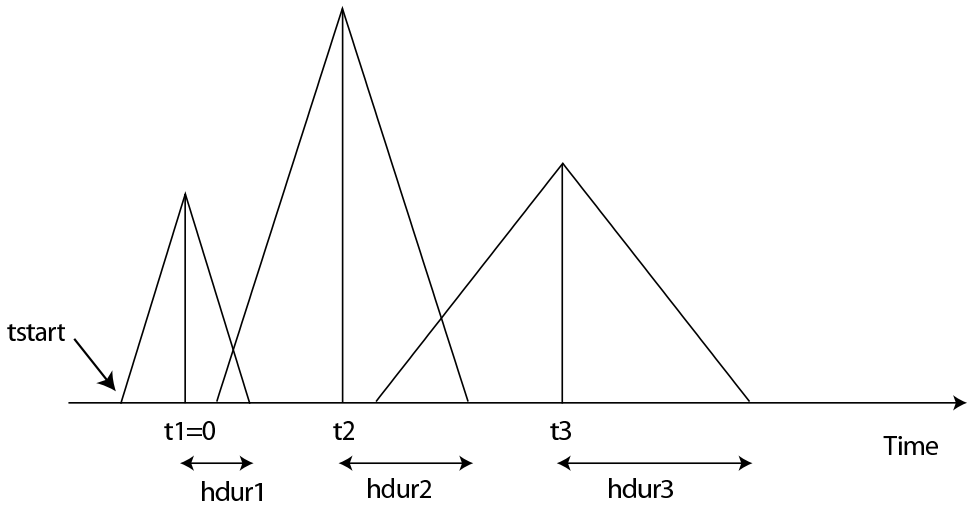


Figure 5.3: Example of timing for three sources. The center of the first source triangle is defined to be time zero. Note that this is NOT in general the hypocentral time, or the start time of the source (marked as tstart). The parameter `time shift` in the `CMTSOLUTION` file would be t1(=0), t2, t3 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:

```
                Network          Longitude (deg)          Burial (m)
Station         Latitude (deg)                  Elevation (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 `in_data_files/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 `in_data_files/STATIONS_FILTERED`.

Solver output is provided in the `in_out_files/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 `in_out_files/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` and `STATIONS` files, you are now ready to launch the solver! This is most easily accomplished based upon the `go_solver` script (See Chapter 9 for information about running 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` 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.

Chapter 6

Adjoint Simulations

Adjoint simulations are generally performed for two distinct applications. First, they can be used for earthquake source inversions, especially earthquakes with large ruptures such as the Lander's earthquake [Wald and Heaton, 1994]. Second, they can be used to generate finite-frequency sensitivity kernels that are a critical part of tomographic inversions based upon 3D reference models [Tromp et al., 2005, Liu and Tromp, 2006, Tromp et al., 2008, Liu and Tromp, 2008]. In either case, source parameter or velocity structure updates are sought to minimize a specific misfit function (e.g., waveform or traveltime differences), and the adjoint simulation provides a means of computing the gradient of the misfit function and further reducing it in successive iterations. Applications and procedures pertaining to source studies and finite-frequency kernels are discussed in Sections 6.1 and 6.2, respectively. The two related parameters in the `Par_file` are `SIMULATION_TYPE` (1 or 2) and the `SAVE_FORWARD` (boolean).

6.1 Adjoint Simulations for Sources

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

1. **Prepare the adjoint sources**

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

   ```
   utils/change_simulation_type.pl -f
   ```

   and then collect the recorded seismograms at all the stations given in `in_data_files/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 `in_data_files/STATIONS_ADJOINT` file, which has the same format as the regular `in_data_files/STATIONS` file.

   - Depending on what type of misfit function is used for the source inversion, adjoint sources need to be computed from the original recorded seismograms for the selected stations and saved in the `in_out_files/SEM/` directory with the format `STA.NT.BX?.adj`, where `STA`, `NT` are the station name and network code given in the `in_data_files/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 `in_data_files/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 `in_data_files/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.

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

Finite-frequency sensitivity kernels are computed in two successive simulations (please refer to Liu and Tromp [2006] and Tromp et al. [2008] 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/cut_velocity/xcut_velocity` to cut a certain portion of the original displacement seismograms and convert it into the proper adjoint source to compute the finite-frequency kernel.

```
xcut_velocity 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.

3. **Run the kernel simulation**

   With the successful forward simulation and the adjoint source ready in the `in_out_files/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.

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 9. Visualization of the finite-frequency kernels is discussed in Section 8.3.

Chapter 7

Noise Cross-correlation Simulations

Besides earthquake simulations, SPECFEM3D 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 6). Also, make sure you read the article 'Noise cross-correlation sensitivity kernels' [Tromp et al., 2010b], in order to understand noise simulations from a theoretical perspective.

7.1 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 `in_data_files/`.

`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 Tromp et al. [2010b] 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.

7.2 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.

7.2.1 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 generate_databases
  make specfem3D
  ```

- 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:
  in_out_files/NOISE_TOMOGRAPHY/ in the SPECFEM3D 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 package, name it as in_out_files/NOISE_TOMOGRAPHY/. We will add some parameter files later in this folder.

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

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

- Create a file called `in_out_files/NOISE_TOMOGRAPHY/irec_master_noise`. Note that this file is located in directory `in_out_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 `in_data_files/STATIONS` becomes the 'master'. That's why we mentioned previously that the order of receivers in `in_data_files/STATIONS` is important.

  Note that in regional simulations, the `in_data_files/STATIONS` might contain receivers which are outside of our computational domains. Therefore, the integer in `irec_master_noise` is actually the ID in `in_data_files/STATIONS_FILTERED` (which is generated by `bin/xgenerate_databases`).

- Create a file called `in_out_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 Tromp et al. [2010b]. 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.

7.2.2 Simulations

With all of the above done, we can finally launch our simulations. Again, please make sure that the `LOCAL_PATH` in `in_data_files/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 Tromp et al. [2010b], 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.

7.2.3 Post-simulation

After those simulations, you have all stuff you need, either in the `in_out_files/OUTPUT_FILES/` or in the directory specified by `LOCAL_PATH` in `in_data_files/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 `in_out_files/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 `in_data_files/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 Tromp et al. [2010b].

7.3 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 8

Graphics

8.1 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).

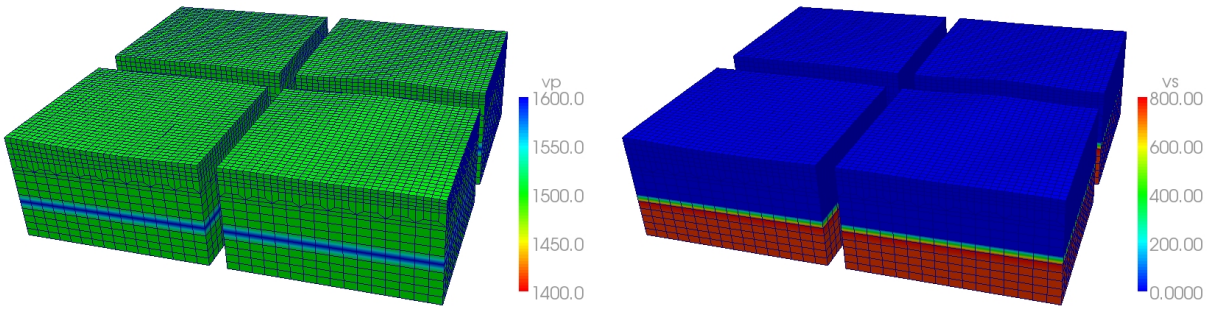


Figure 8.1: Visualization using Paraview of VTK files created by `xgenerate_databases` showing P- and S-wave velocities assigned to the mesh points. The mesh was created by `xmeshfem3D` for 4 processors.

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.

8.2 Movies

To make a surface or volume movie of the simulation, set parameters `MOVIE_SURFACE`, `MOVIE_VOLUME`, and `NTSTEP_BETWEEN_FRAMES` in the `Par_file`. Turning on the movie flags, in particular `MOVIE_VOLUME`, produces large output files. `MOVIE_VOLUME` files are saved in the `LOCAL_PATH` directory, whereas `MOVIE_SURFACE` output files are saved in the `in_out_files/OUTPUT_FILES` directory. We save the velocity field. The look of a movie is determined by the half-duration of the source. The half-duration should be large enough so that the

movie does not contain frequencies that are not resolved by the mesh, i.e., it should not contain numerical noise. This can be accomplished by selecting a CMT HALF_DURATION > 1.1 × 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 Par_file to be 0.0 and HDUR_MOVIE = 0.0 to get the highest frequencies resolved by the simulation, but for a finite source one would keep all the HALF_DURATIONs as prescribed by the finite source model and set HDUR_MOVIE = 0.0.

8.2.1 Movie Surface

When running xspecfem3D with the MOVIE_SURFACE flag turned on, the code outputs moviedata?????? files in the in_out_files/OUTPUT_FILES directory. 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 in_out_files/OUTPUT_FILES/ directory. This file will be created by the solver run. Then type 'make create_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.

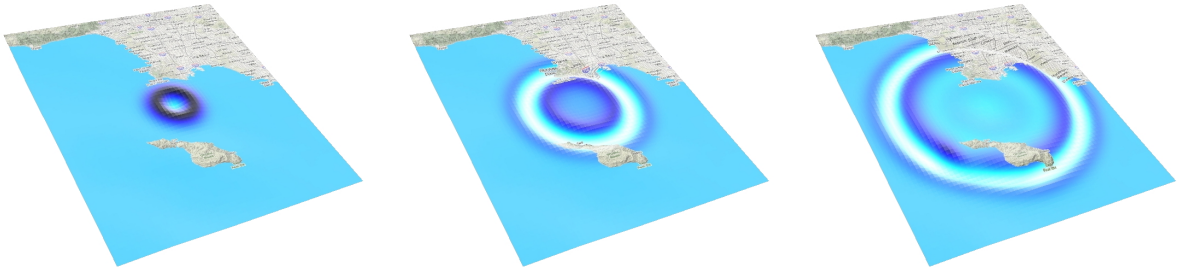


Figure 8.2: Visualization using AVS files created by xcreate_movie_shakemap_AVS_DX_GMT showing movie snapshots of vertical velocity components at different times.

The SPECFEM3D 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).

8.3 Finite-Frequency Kernels

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

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 combine_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 `in_out_files/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`).

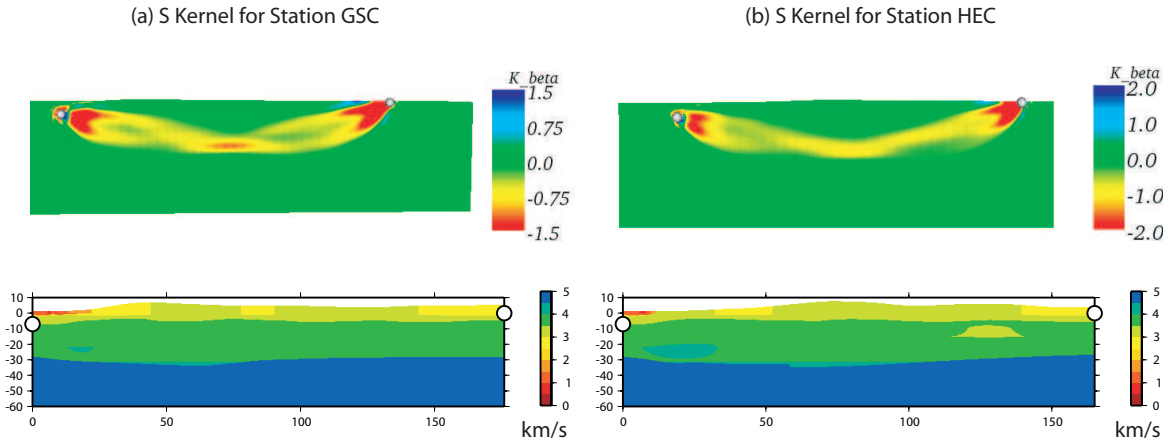


Figure 8.3: (a) Top Panel: Vertical source-receiver cross-section of the S-wave finite-frequency sensitivity kernel K_β for station GSC at an epicentral distance of 176 km from the September 3, 2002, Yorba Linda earthquake. Lower Panel: Vertical source-receiver cross-section of the 3D S-wave speed model used for the spectral-element simulations [Komatitsch et al., 2004]. (b) The same as (a) but for station HEC at an epicentral distance of 165 km [Liu and Tromp, 2006].

Chapter 9

Running through a Scheduler

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

- The processors/nodes to be used for each run are assigned dynamically by the scheduler, based on availability. Therefore, in order for the `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.

We describe here in some detail a job submission procedure for the Caltech 1024-node cluster, CITerra, under the LSF scheduling system. We consider the submission of a regular forward simulation using the internal mesher to create mesh partitions. The two main scripts are `run_lsf.bash`, which compiles the Fortran code and submits the job to the scheduler, and `go_mesher_solver_lsf_basin.forward`, which contains the instructions that make up the job itself. These scripts can be found in `utils/Cluster/lsf/` directory

9.1 Job submission `run_lsf.bash`

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

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

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

```
make meshfem3D
make generate_databases
make specfem3D
d=`date`
echo "Finished compilation $d"

# get total number of nodes needed for solver
NPROC=`grep NPROC in_data_files/Par_file | cut -c 34- `

# compute total number of nodes needed for mesher
NPROC_XI=`grep NPROC_XI in_data_files/meshfem3D_files/Mesh_Par_file | cut -c 34- `
NPROC_ETA=`grep NPROC_ETA in_data_files/meshfem3D_files/Mesh_Par_file | cut -c 34- `
# total number of nodes is the product of the values read
numnodes=$(( $NPROC_XI * $NPROC_ETA ))

# checks total number of nodes
if [ $numnodes -neq $NPROC ]; then
        echo "error number of procs mismatch"
        exit
fi

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

9.2 Job script `go_mesher_solver_lsf.forward`

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

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

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

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

4. The final portion of the script collects the seismograms and performs clean up on the nodes, using the Perl scripts `collect_seismo_lsf_multi.pl` and `cleanmulti.pl`.

```
#!/bin/bash -v
#BSUB -o in_out_files/OUTPUT_FILES/%J.o
#BSUB -a mpich_gm
#BSUB -J go_mesher_solver_lsf

# set up local scratch directories
```

```
BASEMPIDIR=/scratch/$USER/DATABASES_MPI
mkdir -p in_out_files/OUTPUT_FILES
echo "$LSB_MCPU_HOSTS" > in_out_files/OUTPUT_FILES/lsf_machines
echo "$LSB_JOBID" > in_out_files/OUTPUT_FILES/jobid
remap_lsf_machines.pl in_out_files/OUTPUT_FILES/lsf_machines >
in_out_files/OUTPUT_FILES/machines
shmux -M50 -Sall -c "rm -r -f /scratch/$USER; \
      mkdir -p /scratch/$USER; mkdir -p $BASEMPIDIR" \
      - < in_out_files/OUTPUT_FILES/machines >/dev/null

# run the specfem program
current_pwd=$PWD
cd bin/
run_lsf.bash --gm-no-shmem --gm-copy-env $current_pwd/xmeshfem3D
run_lsf.bash --gm-no-shmem --gm-copy-env $current_pwd/xgenerate_databases
run_lsf.bash --gm-no-shmem --gm-copy-env $current_pwd/xspecfem3D

# collect seismograms and clean up
cd current_pwd/
mkdir -p in_out_files/SEM
cd in_out_files/SEM/
collect_seismo.pl ../OUTPUT_FILES/lsf_machines
cleanbase.pl ../OUTPUT_FILES/machines
```

Chapter 10

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.

10.1 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:

10.1.1 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.

10.1.2 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/.

10.1.3 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.

10.2 Collect Synthetic Seismograms

The forward and adjoint simulations generate synthetic seismograms in the in_out_files/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 6.1 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 6.2 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 in_data_files/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 in_data_files/Par_file
```

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

10.3 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.

10.4 Plot Movie Snapshots and Synthetic Shakemaps

10.4.1 Script movie2gif.gmt.pl

With the movie data saved in in_out_files/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.

10.4.2 Script plot_shakemap.gmt.pl

With the shakemap data saved in in_out_files/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.

10.5 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 9).

```
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.

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`) or Jeroen Tromp (`jtromp-AT-princeton.edu`), and/or use our online bug tracking system Roundup (`www.geodynamics.org/roundup`).

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 Jeroen Tromp (`jtromp-AT-princeton.edu`) or to the CIG Computational Seismology Mailing List (`cig-seismo@geodynamics.org`).

Copyright

Main authors: Dimitri Komatitsch and Jeroen Tromp
Princeton University, USA, and University of Pau / CNRS / INRIA, France

Bibliography

M. Ainsworth, P. Monk, and W. Muniz. Dispersive and dissipative properties of discontinuous Galerkin finite element methods for the second-order wave equation. *Journal of Scientific Computing*, 27(1):5–40, 2006. doi: 10.1007/s10915-005-9044-x.

K. Aki and P. G. Richards. *Quantitative seismology, theory and methods*. W. H. Freeman, San Francisco, USA, 1980.

D. N. Arnold. An interior penalty finite element method with discontinuous elements. *SIAM Journal on Numerical Analysis*, 19(4):742–760, 1982. doi: 10.1137/0719052.

M. Bernacki, S. Lanteri, and S. Piperno. Time-domain parallel simulation of heterogeneous wave propagation on unstructured grids using explicit, nondiffusive, discontinuous Galerkin methods. *J. Comput. Acoust.*, 14(1):57–81, 2006.

C. Bernardi, Y. Maday, and A. T. Patera. A new nonconforming approach to domain decomposition: the Mortar element method. In H. Brezis and J. L. Lions, editors, *Nonlinear partial differential equations and their applications*, Séminaires du Collège de France, pages 13–51, Paris, 1994. Pitman.

L. Carrington, D. Komatitsch, M. Laurenzano, M. Tikir, D. Michéa, N. Le Goff, A. Snavely, and J. Tromp. High-frequency simulations of global seismic wave propagation using SPECFEM3D_GLOBE on 62 thousand processor cores. *Proceedings of the ACM/IEEE Supercomputing SC'2008 conference*, pages 1–11, 2008. doi: 10.1145/1413370.1413432. Article #60, Gordon Bell Prize finalist article.

E. Chaljub, Y. Capdeville, and J. P. Vilotte. Solving elastodynamics in a fluid-solid heterogeneous sphere: a parallel spectral-element approximation on non-conforming grids. *J. Comput. Phys.*, 187(2):457–491, 2003.

E. Chaljub, D. Komatitsch, J. P. Vilotte, Y. Capdeville, B. Valette, and G. Festa. Spectral element analysis in seismology. In R.-S. Wu and V. Maupin, editors, *Advances in wave propagation in heterogeneous media*, volume 48 of *Advances in Geophysics*, pages 365–419. Elsevier - Academic Press, London, UK, 2007.

M. Chen and J. Tromp. Theoretical and numerical investigations of global and regional seismic wave propagation in weakly anisotropic earth models. *Geophys. J. Int.*, 168(3):1130–1152, 2007. doi: 10.1111/j.1365-246X.2006.03218.x.

S. Chevrot, N. Favier, and D. Komatitsch. Shear wave splitting in three-dimensional anisotropic media. *Geophys. J. Int.*, 159(2):711–720, 2004. doi: 10.1111/j.1365-246X.2004.02432.x.

B. Cockburn, G. E. Karniadakis, and C.-W. Shu. *Discontinuous Galerkin Methods: Theory, Computation and Applications*. Springer, Heidelberg, Germany, 2000.

G. Cohen. *Higher-order numerical methods for transient wave equations*. Springer-Verlag, Berlin, Germany, 2002.

F. A. Dahlen and J. Tromp. *Theoretical Global Seismology*. Princeton University Press, Princeton, New-Jersey, USA, 1998.

J. D. De Basabe and M. K. Sen. Grid dispersion and stability criteria of some common finite-element methods for acoustic and elastic wave equations. *Geophysics*, 72(6):T81–T95, 2007. doi: 10.1190/1.2785046.

J. D. De Basabe and M. K. Sen. Stability of the high-order finite elements for acoustic or elastic wave propagation with high-order time stepping. *Geophys. J. Int.*, 181(1):577–590, 2010. doi: 10.1111/j.1365-246X.2010.04536.x.

J. D. De Basabe, M. K. Sen, and M. F. Wheeler. The interior penalty discontinuous Galerkin method for elastic wave propagation: grid dispersion. *Geophys. J. Int.*, 175(1):83–93, 2008. doi: 10.1111/j.1365-246X.2008.03915.x.

D. S. Dreger and D. V. Helmberger. Broadband modeling of local earthquakes. *Bull. Seismol. Soc. Am.*, 80:1162–1179, 1990.

M. Dumbser and M. Käser. An arbitrary high-order discontinuous Galerkin method for elastic waves on unstructured meshes-II. The three-dimensional isotropic case. *Geophys. J. Int.*, 167(1):319–336, 2006. doi: 10.1111/j.1365-246X.2006.03120.x.

V. Étienne, E. Chaljub, J. Virieux, and N. Glinsky. An hp-adaptive discontinuous Galerkin finite-element method for 3-D elastic wave modelling. *Geophys. J. Int.*, 183(2):941–962, 2010. doi: 10.1111/j.1365-246X.2010.04764.x.

R. S. Falk and G. R. Richter. Explicit finite element methods for symmetric hyperbolic equations. *SIAM Journal on Numerical Analysis*, 36(3):935–952, 1999. doi: 10.1137/S0036142997329463.

N. Favier, S. Chevrot, and D. Komatitsch. Near-field influences on shear wave splitting and traveltime sensitivity kernels. *Geophys. J. Int.*, 156(3):467–482, 2004. doi: 10.1111/j.1365-246X.2004.02178.x.

A. Fichtner, H. Igel, H.-P. Bunge, and B. L. N. Kennett. Simulation and inversion of seismic wave propagation on continental scales based on a spectral-element method. *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 4(1-2):11–22, 2009.

F. X. Giraldo, J. S. Hesthaven, and T. Warburton. Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. *J. Comput. Phys.*, 181(2):499–525, 2002. doi: 10.1006/jcph.2002.7139.

L. Godinho, P. A. Mendes, A. Tadeu, A. Cadena-Isaza, C. Smerzini, F. J. Sánchez-Sesma, R. Madec, and D. Komatitsch. Numerical simulation of ground rotations along 2D topographical profiles under the incidence of elastic plane waves. *Bull. Seismol. Soc. Am.*, 99(2B):1147–1161, 2009. doi: 10.1785/0120080096.

W. Gropp, E. Lusk, and A. Skjellum. *Using MPI, portable parallel programming with the Message-Passing Interface*. MIT Press, Cambridge, USA, 1994.

M. J. Grote, A. Schneebeli, and D. Schötzau. Discontinuous Galerkin finite element method for the wave equation. *SIAM Journal on Numerical Analysis*, 44(6):2408–2431, 2006. doi: 10.1137/05063194X.

E. Hauksson. Crustal structure and seismicity distribution adjacent to the Pacific and North America plate boundary in Southern California. *J. Geophys. Res.*, 105:13875–13903, 2000.

F. Q. Hu, M. Y. Hussaini, and P. Rasetarinera. An analysis of the discontinuous Galerkin method for wave propagation problems. *J. Comput. Phys.*, 151(2):921–946, 1999. doi: 10.1006/jcph.1999.6227.

C. Ji, S. Tsuboi, D. Komatitsch, and J. Tromp. Rayleigh-wave multipathing along the west coast of north america. *Bull. Seismol. Soc. Am.*, 95(6):2115–2124, 2005. doi: 10.1785/0120040180.

D. Komatitsch. Fluid-solid coupling on a cluster of GPU graphics cards for seismic wave propagation. *Comptes Rendus de l'Académie des Sciences - Mécanique*, 2011. in press.

D. Komatitsch and R. Martin. An unsplit convolutional Perfectly Matched Layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5):SM155–SM167, 2007. doi: 10.1190/1.2757586.

D. Komatitsch and J. Tromp. Spectral-element simulations of global seismic wave propagation-I. Validation. *Geophys. J. Int.*, 149(2):390–412, 2002a. doi: 10.1046/j.1365-246X.2002.01653.x.

D. Komatitsch and J. Tromp. Spectral-element simulations of global seismic wave propagation-II. 3-D models, oceans, rotation, and self-gravitation. *Geophys. J. Int.*, 150(1):303–318, 2002b. doi: 10.1046/j.1365-246X.2002.01716.x.

D. Komatitsch and J. Tromp. Introduction to the spectral-element method for 3-D seismic wave propagation. *Geophys. J. Int.*, 139(3):806–822, 1999. doi: 10.1046/j.1365-246x.1999.00967.x.

D. Komatitsch and J. P. Vilotte. The spectral-element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bull. Seismol. Soc. Am.*, 88(2):368–392, 1998.

D. Komatitsch, J. Ritsema, and J. Tromp. The spectral-element method, Beowulf computing, and global seismology. *Science*, 298(5599):1737–1742, 2002. doi: 10.1126/science.1076024.

D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp. A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the Earth Simulator. *Proceedings of the ACM/IEEE Supercomputing SC'2003 conference*, pages 4–11, 2003. doi: 10.1109/SC.2003.10023. Gordon Bell Prize winner article.

D. Komatitsch, Q. Liu, J. Tromp, P. Süss, C. Stidham, and J. H. Shaw. Simulations of ground motion in the Los Angeles basin based upon the spectral-element method. *Bull. Seismol. Soc. Am.*, 94(1):187–206, 2004. doi: 10.1785/0120030077.

D. Komatitsch, J. Labarta, and D. Michéa. A simulation of seismic wave propagation at high resolution in the inner core of the Earth on 2166 processors of MareNostrum. *Lecture Notes in Computer Science*, 5336:364–377, 2008.

D. Komatitsch, D. Michéa, and G. Erlebacher. Porting a high-order finite-element earthquake modeling application to NVIDIA graphics cards using CUDA. *Journal of Parallel and Distributed Computing*, 69(5):451–460, 2009. doi: 10.1016/j.jpdc.2009.01.006.

D. Komatitsch, G. Erlebacher, D. Göddeke, and D. Michéa. High-order finite-element seismic wave propagation modeling with MPI on a large GPU cluster. *J. Comput. Phys.*, 229(20):7692–7714, 2010a. doi: 10.1016/j.jcp.2010.06.024.

D. Komatitsch, D. Göddeke, G. Erlebacher, and D. Michéa. Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs. *Computer Science Research and Development*, 25(1-2):75–82, 2010b. doi: 10.1007/s00450-010-0109-1.

D. Komatitsch, L. P. Vinnik, and S. Chevrot. SHdiff/SVdiff splitting in an isotropic Earth. *J. Geophys. Res.*, 115(B7):B07312, 2010c. doi: 10.1029/2009JB006795.

D. A. Kopriva. Metric identities and the discontinuous spectral element method on curvilinear meshes. *Journal of Scientific Computing*, 26(3):301–327, 2006. doi: 10.1007/s10915-005-9070-8.

D. A. Kopriva, S. L. Woodruff, and M. Y. Hussaini. Computation of electromagnetic scattering with a non-conforming discontinuous spectral element method. *Int. J. Numer. Meth. Eng.*, 53(1):105–122, 2002. doi: 10.1002/nme.394.

S. Krishnan, C. Ji, D. Komatitsch, and J. Tromp. Case studies of damage to tall steel moment-frame buildings in Southern California during large San Andreas earthquakes. *Bull. Seismol. Soc. Am.*, 96(4A):1523–1537, 2006a. doi: 10.1785/0120050145.

S. Krishnan, C. Ji, D. Komatitsch, and J. Tromp. Performance of two 18-story steel moment-frame buildings in Southern California during two large simulated San Andreas earthquakes. *Earthquake Spectra*, 22(4):1035–1061, 2006b. doi: 10.1193/1.2360698.

S. J. Lee, H. W. Chen, Q. Liu, D. Komatitsch, B. S. Huang, and J. Tromp. Three-dimensional simulations of seismic wave propagation in the Taipei basin with realistic topography based upon the spectral-element method. *Bull. Seismol. Soc. Am.*, 98(1):253–264, 2008. doi: 10.1785/0120070033.

S. J. Lee, Y. C. Chan, D. Komatitsch, B. S. Huang, and J. Tromp. Effects of realistic surface topography on seismic ground motion in the Yangminshan region of Taiwan based upon the spectral-element method and LiDAR DTM. *Bull. Seismol. Soc. Am.*, 99(2A):681–693, 2009a. doi: 10.1785/0120080264.

S. J. Lee, D. Komatitsch, B. S. Huang, and J. Tromp. Effects of topography on seismic wave propagation: An example from northern Taiwan. *Bull. Seismol. Soc. Am.*, 99(1):314–325, 2009b. doi: 10.1785/0120080020.

Q. Liu and J. Tromp. Finite-frequency kernels based on adjoint methods. *Bull. Seismol. Soc. Am.*, 96(6):2383–2397, 2006. doi: 10.1785/0120060041.

Q. Liu and J. Tromp. Finite-frequency sensitivity kernels for global seismic wave propagation based upon adjoint methods. *Geophys. J. Int.*, 174(1):265–286, 2008. doi: 10.1111/j.1365-246X.2008.03798.x.

P. Lovely, J. Shaw, Q. Liu, and J. Tromp. A structural model of the Salton Trough and its implications for seismic hazard. *Bull. Seismol. Soc. Am.*, 96:1882–1896, 2006.

R. Madec, D. Komatitsch, and J. Diaz. Energy-conserving local time stepping based on high-order finite elements for seismic wave propagation across a fluid-solid interface. *Comput. Model. Eng. Sci.*, 49(2):163–189, 2009.

R. Martin and D. Komatitsch. An unsplit convolutional perfectly matched layer technique improved at grazing incidence for the viscoelastic wave equation. *Geophys. J. Int.*, 179(1):333–344, 2009. doi: 10.1111/j.1365-246X.2009.04278.x.

R. Martin, D. Komatitsch, C. Blitz, and N. Le Goff. Simulation of seismic wave propagation in an asteroid based upon an unstructured MPI spectral-element method: blocking and non-blocking communication strategies. *Lecture Notes in Computer Science*, 5336:350–363, 2008a.

R. Martin, D. Komatitsch, and A. Ezziani. An unsplit convolutional perfectly matched layer improved at grazing incidence for seismic wave equation in poroelastic media. *Geophysics*, 73(4):T51–T61, 2008b. doi: 10.1190/1.2939484.

R. Martin, D. Komatitsch, and S. D. Gedney. A variational formulation of a stabilized unsplit convolutional perfectly matched layer for the isotropic or anisotropic seismic wave equation. *Comput. Model. Eng. Sci.*, 37(3):274–304, 2008c.

R. Martin, D. Komatitsch, S. D. Gedney, and E. Bruthiaux. A high-order time and space formulation of the unsplit perfectly matched layer for the seismic wave equation using Auxiliary Differential Equations (ADE-PML). *Comput. Model. Eng. Sci.*, 56(1):17–42, 2010.

D. Michéa and D. Komatitsch. Accelerating a 3D finite-difference wave propagation code using GPU graphics cards. *Geophys. J. Int.*, 182(1):389–402, 2010. doi: 10.1111/j.1365-246X.2010.04616.x.

P. Monk and G. R. Richter. A discontinuous Galerkin method for linear symmetric hyperbolic systems in inhomogeneous media. *Journal of Scientific Computing*, 22-23(1-3):443–477, 2005. doi: 10.1007/s10915-004-4132-5.

S. P. Oliveira and G. Seriani. Effect of element distortion on the numerical dispersion of spectral element methods. *Communications in Computational Physics*, 9(4):937–958, 2011.

K. B. Olsen, S. M. Day, and C. R. Bradley. Estimation of Q for long-period (>2 sec) waves in the Los Angeles basin. *Bull. Seismol. Soc. Am.*, 93(2):627–638, 2003.

P. S. Pacheco. *Parallel programming with MPI*. Morgan Kaufmann Press, San Francisco, 1997.

F. Pellegrini and J. Roman. SCOTCH: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs. *Lecture Notes in Computer Science*, 1067:493–498, 1996.

W. H. Reed and T. R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, USA, 1973.

J. Ritsema, L. A. Rivera, D. Komatitsch, J. Tromp, and H. J. van Heijst. The effects of crust and mantle heterogeneity on PP/P and SS/S amplitude ratios. *Geophys. Res. Lett.*, 29(10):1430, 2002. doi: 10.1029/2001GL013831.

B. Rivière and M. F. Wheeler. Discontinuous finite element methods for acoustic and elastic wave problems. *Contemporary Mathematics*, 329:271–282, 2003.

B. Savage, D. Komatitsch, and J. Tromp. Effects of 3D attenuation on seismic wave amplitude and phase measurements. *Bull. Seismol. Soc. Am.*, 100(3):1241–1251, 2010. doi: 10.1785/0120090263.

G. Seriani and S. P. Oliveira. Optimal blended spectral-element operators for acoustic wave modeling. *Geophysics*, 72(5):SM95–SM106, 2007. doi: 10.1190/1.2750715.

M. P. Süss and J. H. Shaw. P wave seismic velocity structure derived from sonic logs and industry reflection data in the Los Angeles basin, California. *J. Geophys. Res.*, 108(B3):2170, 2003. doi: 10.1029/2001JB001628.

J. Tromp and D. Komatitsch. Spectral-element simulations of wave propagation in a laterally homogeneous Earth model. In E. Boschi, G. Ekström, and A. Morelli, editors, *Problems in Geophysics for the New Millennium*, pages 351–372. INGV, Roma, Italy, 2000.

J. Tromp, C. Tape, and Q. Liu. Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels. *Geophys. J. Int.*, 160(1):195–216, 2005. doi: 10.1111/j.1365-246X.2004.02453.x.

J. Tromp, D. Komatitsch, and Q. Liu. Spectral-element and adjoint methods in seismology. *Communications in Computational Physics*, 3(1):1–32, 2008.

J. Tromp, D. Komatitsch, V. Hjoerleifsdottir, Q. Liu, H. Zhu, D. Peter, E. Bozdag, D. McRitchie, P. Friberg, C. Trabant, and A. Hutko. Near real-time simulations of global CMT earthquakes. *Geophys. J. Int.*, 183(1):381–389, 2010a. doi: 10.1111/j.1365-246X.2010.04734.x.

J. Tromp, Y. Luo, S. Hanasoge, and D. Peter. Noise cross-correlation sensitivity kernels. *Geophys. J. Int.*, 183: 791–819, 2010b. doi: 10.1111/j.1365-246X.2010.04721.x.

R. Vai, J. M. Castillo-Covarrubias, F. J. Sánchez-Sesma, D. Komatitsch, and J. P. Vilotte. Elastic wave propagation in an irregularly layered medium. *Soil Dynamics and Earthquake Engineering*, 18(1):11–18, 1999. doi: 10.1016/S0267-7261(98)00027-X.

K. van Wijk, D. Komatitsch, J. A. Scales, and J. Tromp. Analysis of strong scattering at the micro-scale. *J. Acoust. Soc. Am.*, 115(3):1006–1011, 2004. doi: 10.1121/1.1647480.

J. Virieux and S. Operto. An overview of full-waveform inversion in exploration geophysics. *Geophysics*, 74(6): WCC1–WCC26, 2009. doi: 10.1190/1.3238367.

D. J. Wald and T. H. Heaton. Spatial and temporal distribution of slip for the 1992 Landers, California earthquake. *Bull. Seismol. Soc. Am.*, 84:668–691, 1994.

L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas. A high-order discontinuous Galerkin method for wave propagation through coupled elastic-acoustic media. *J. Comput. Phys.*, 229(24):9373–9396, 2010. doi: 10.1016/j.jcp.2010.09.008.

L. Zhu and H. Kanamori. Moho depth variation in southern California from teleseismic receiver functions. *J. Geophys. Res.*, 105:2969–2980, 2000.

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 Aki and Richards [1980] 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 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` and `STATIONS` file. Note that e.g. the `CMTSOLUTION` file 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`) 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 `in_out_files/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 `in_data_files/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 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 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 `in_data_files/` 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, 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 path's to these specific compiler commands F90, CC and MPIF90 won't be correct if './configure' fails.

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

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

compilation fails stating:

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

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

Normally, this message will appear when you're 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

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.