SES3D version 5.0

Documentation - Work in progress.

Andreas Fichtner

Department of Earth Sciences, ETH Zurich, Switzerland

Alexey Gokhberg

Department of Earth Sciences, ETH Zurich, Switzerland

Lion Krischer

Department of Earth Sciences, ETH Zurich, Switzerland

14 February 2014

Contents

Foreword

SES3D is a programme package for the simulation of elastic wave propagation and waveform inversion in a spherical section. The package is based on a spectral-element discretisation of the seismic wave equation combined with adjoint techniques.

SES3D supports 3D heterogeneous visco-elastic rheologies with radial anisotropy. Anisotropic perfectly matched layers are implemented to avoid reflections from the unphysical boundaries of the spherical section.

SES3D operates in the natural spherical coordinates, which is untypical for spectral-element approaches. The advantages are a compact programme code, fast computations for spherical sections that are sufficiently far from the poles and the core, and the easy implementation of 3D models.

SES3D is fully parallelised, meaning that the computational domain is partitioned into subdomains, each of which is assigned to one compute core. Communication between subdomains is based on MPI.

SES3D has been developed for continental-scale full seismic waveform inversion. It is, however, applicable to a wide range of local- to continental-scale wave propagation problems.

SES3D is deliberately puristic. This is intended to (1) make the code easily adaptable to particular problems, (2) facilitate the implementation of 3D models, (3) reduce the likelihood of programming errors, and (4) allow for an easy adaptation to new hardware architectures. The last point becomes particularly relevant in times when hardware architecture changes rapidly.

This tutorial is split into a description of the programme code and an introduction to the mathematical background of SES3D. Reading the mathematical part is not required to successfully run SES3D. It is, however, strongly recommended. The chance of using SES3D incorrectly or inefficiently is high when its mathematical background is not known. This is true for any numerical method.

The description of the programme package is centred around realistic examples that a new user may want to reproduce in order to become familiar with SES3D.

SES3D is hosted on Github and can be obtained from https://github.com/echolite/ses3d.

Zurich, February 2014 *Andreas Fichtner*

Part I
Code description

Chapter 1
Code structure

1.1 Directory structure

Figure 1.1 illustrates the directory structure of `SES3D`. The contents of the different directories are as follows:

`ADJOINT`: Adjoint source time functions and a list of adjoint source locations.

`DATA/COORDINATES`: ASCII files containing the grid point coordinates for each model subdomain.

`DATA/LOGFILES`: Logfiles for each model subdomain. Logfiles document the geometrical setup of each subdomain and the iteration progress. They are mostly used for debugging.

`DATA/OUTPUT`: Directory reserved for output such as seismograms, 3D wavefield snapshots and 3D sensitivity kernels.

`INPUT`: Input files: `setup` (model geometry and parallelisation), `event_1, event_2, ...` (event files, one for each earthquake to be modelled, source parameters, output directory, time stepping variables), `event_list` (list of events to be modelled), `recfile` (list of receiver locations), `stf` (source time function)

`MAIN`: Executables for `SES3D` wave propagation.

`SOURCE`: Fortran source files for `SES3D` wave propagation.

`MODELS/MAIN`: Executables for the generation of Earth models.

`MODELS/MODELS`: Physical model parameters. One file for each compute core. Read by the wave propagation executables.

`MODELS/MODELS_1D`: Fortran codes for the implementation of a selection of 1D Earth models.

`MODELS/MODELS_3D`: 3D Earth models or Earth model perturbations.

`MODELS/SOURCE`: Source code for Earth model generation.

`TOOLS`: Collection of Fortran and Python tools for visualisation, model manipulation and the computation of adjoint sources.

`SCENARIOS`: Input files for the `SES3D` scenarios described in this tutorial.

1.2 Source files for wave propagation

The principal source files for the wave propagation simulation and the computation of Fréchet kernels can be found in the directory `SOURCE`. These are:

`ses3d_main.f90`: Initialisation of MPI, open logfiles, call subroutines for initialisation, call subroutines to

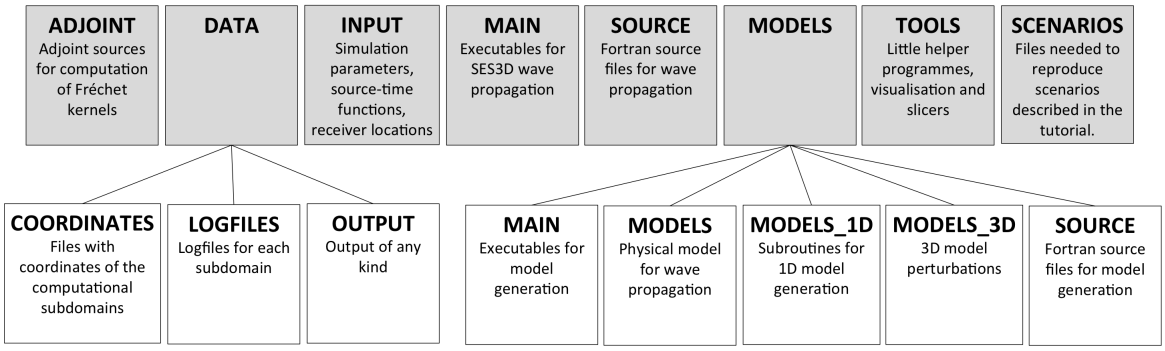

Fig. 1.1 `SES3D` directory structure. See the text for detailed descriptions.

iteratively advance the wavefield, monitor PML stability, write intermediate wave field in adjoint runs, clean up MPI and close files at the end of the iteration.

`ses3d_modules.f90`: Definition of variables and parameters.

`ses3d_input.f90`: Read parameter files `Par_*`, read `boxfile` (contains information concerning the parallelisation), read 3D model parameters, read source time functions (`stf_*`), read receiver locations (`recfile_*`), read adjoint source locations (`ad_srcfile`).

`ses3d_init.f90`: Set up grid point geometry, make mass matrix, compute receiver locations in unit cube coordinates, compute point source location in unit cube coordinates, read adjoint source time functions, initialise PML damping profiles.

`ses3d_evolution.f90`: Propagate dynamic fields one time step forward.

`ses3d_grad.f90`: Compute Fréchet kernels.

`ses3d_output.f90`: Collection of subroutines to write seismograms, store intermediate wave fields, write wave field snapshots and write Fréchet kernels.

`ses3d_miscellaneous.f90`: Subroutines to add external forces (single force, moment tensor source, adjoint sources) and for the communication between compute cores.

The above source files must be compiled together, e.g. by running the script `s_make` located in the `SOURCE` directory. **A recompilation is necessary after a change of the parallelisation scheme, i.e. a change of the division of the spherical section into subsections.**

The script `s_make` compiles all source files and combines them into the executables `ses3d.exe` located in the directory `MAIN`. Executing `ses3d.exe` starts the forward or adjoint wave propagation.

1.3 Variable style and nomenclature of 3D fields

In `SES3D`, dynamics fields (velocity, stress, strain, material parameters, etc.) are implemented in the form of six-dimensional arrays. This is illustrated in figure 1.2 with the example of the θ-component velocity field. The first three dimensions of the array correspond to the element indices within a spherical subsection in θ-, ϕ- and radial directions. Note that the radial index increases from the surface, where it is equal to 0, towards greater depth. The last three indices are used to address the nodes within one element.

Throughout `SES3D`, θ-components (colatitude) are labelled with `x`, ϕ-components (longitude) with `y` and radial components with `z`. This is intended to keep the variable names short.

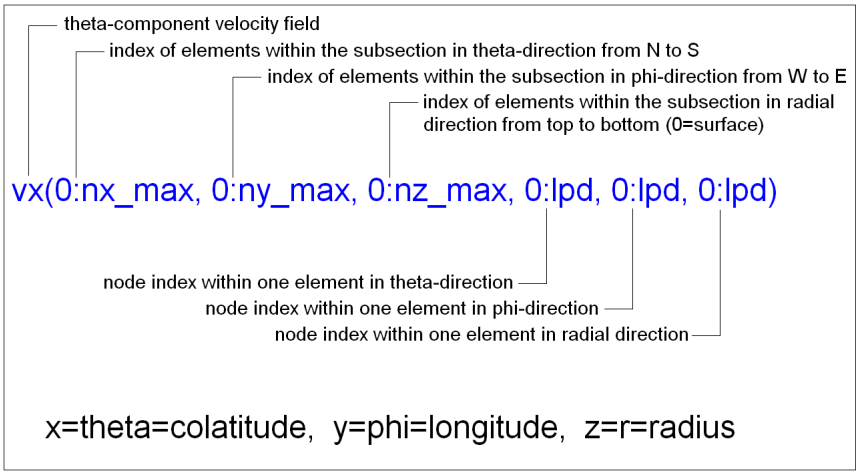

Fig. 1.2 Structure and nomenclature of the dynamic fields in `SES3D`. The example is for the velocity field in colatitudinal (θ) direction.

1.4 Calls to caution

Each numerical method needs to be handled with care, and `SES3D` is no exception. The following paragraphs are concerned with some of the difficulties that a user may encounter when working with `SES3D`. It is generally recommended to assess the accuracy of numerical solutions by comparing them to semi-analytical solutions that exist for simplified models, e.g. radially symmetric Earth models.

1.4.1 Absorbing boundaries

`SES3D` avoids unphysical reflections from the lateral and lower boundaries of the spherical section using a variant of the perfectly matched layers technique, called the method of anisotropic perfectly matched layers (APML, e.g. Teixeira & Chew, 1997). It consists in the modification of the elastic wave equation within a narrow region along the unrealistic boundaries of the spherical section. The mathematical details of this method are explained in section 6.

The width of the absorbing boundary region, in terms of the number of elements, is specified by the parameter `pml` in the `ses3d_modules.f90` source file. A good choice is `pml=3`, i.e. an absorbing boundary layer that is three elements wide.

Within the absorbing layer, incoming waves are attenuated. This means that one should place neither receivers nor sources within the boundary layer. It is recommended to have at least two elements between the absorbing layer and receivers and sources.

Contrary to what their name suggests, perfectly matched layers are not perfect. The imperfection comes in the form of two undesirable phenomena: (1) Incident waves are not completely absorbed by the absorbing layers. This means that small unphysical reflections will always be present. These can be minimised by placing sources and receivers further away from the boundaries. (2) All variants of the perfectly matched layers technique are long-term unstable. This means that the wavefield amplitudes may grow indefinitely for very long simulations, where the term *very long* is not very well defined.

1.4.2 Seismic discontinuities and the crust

Realistic Earth models typically contain discontinuities. Spectral-element solutions are correct only when discontinuities coincide with the edges of elements, such that the shared boundary nodes take the different values from each side of the discontinuity. The inflexible grid of `SES3D` is not always capable of accounting for discontinuities in the exact way. This means that discontinuities of the material properties may be located in the interior of elements. As a result, the numerical solutions may not be as exact as they would be in the case of perfectly honoured discontinuities. This effect is difficult to quantify, and it mostly concerns surface waves.

A related problem is the implementation of thin crustal layers that may be thinner than a layer of elements. This can also produce inaccurate numerical solutions.

Difficulties with thin crustal layers and discontinuities can most easily be avoided by the implementation of long wavelength equivalent models (e.g. Capdeville & Marigo, 2007, 2008; Fichtner & Igel, 2008).

1.4.3 The poles and the core

Since `SES3D` operates in the natural spherical coordinate system, one must exclude the poles and the core from the computational domain. A the centre of the Earth, spatial derivatives in spherical coordinates are singular. Also, the elements become very small near the poles and near the core, so that the time step `dt` must be chosen very small. To run `SES3D` efficiently, the spherical sections should not be deeper than ~ 3000 km and not closer than $\sim 20^\circ$ to one of the poles.

Chapter 2
Scenarios

The practical part of this tutorial is built around realistic scenarios that introduce various aspects of SES3D, ranging from basic the input to the implementation of 3D models, the compuation of sensitivity kernels, and rotations of the computational domain. For each of the scenarios, various source and input files must be changed. Those files can be found in the SCENARIOS folder.

2.1 Regional-scale wave propagation: Anatolia

In our first scenario, we work with an earthquake that occurred on August 25, 2007 in eastern Turkey. The hypocentre location is: latitude: 39.26°, longitude: 41.04°, depth: 5.0 km. Our goal is to introduce the input files of SES3D, implement a 3D heterogeneous model, and compute sensitivity kernels.

The default setup of SES3D is made to fully reproduce this example. All input and source files that are specific to this scenario can also be found in the SCENARIOS/ANATOLIA/ folder.

2.1.1 Input

2.1.1.1 Model setup

The geometrical setup of the model is described in the file setup in the directory INPUT. In our specific example, the computational domain ranges from $47.1^\circ - 55.9^\circ$ colatitude (colatitude$= 90^\circ -$latitude), from $23.1^\circ - 42.9^\circ$ longitude, and from a radius of $5,900,000.0 - 6,371,000.0$ m, i.e. from 471 km depth to the surface of the Earth. For the moment, we ignore visco-elastic dissipation (is_diss=0), and we construct a homogeneous model where all velocities and density are set to zero (model_type=1). More on the generation of specific Earth models can be found in section 2.1.2.

We parallelise the computations by dividing the computational domain into 3 subdomains in colatitudinal direction (px=3), 4 subdomains in longitudinal direction (py=4) and 4 subdomains in depth direction (pz=4). Thus, we have a total of 48 subdomains, each of which is assigned to one compute core. The parallelisation in SES3D is shown schematically in figure 2.1.

Within each of the subdomains, the finite elements are numbered from 0 to 66/px$= 22$ in colatitudinal direction (nx_global=66), from 0 to 108/py$= 27$ in longitudinal direction (ny_global=108), and from 0 to 28/pz$= 7$ in radial direction (nz_global=28). The total number of elements in the complete computational domain is (nx_global+px)(ny_global+py)(nz_global+pz)$= 247,296$. Note that the ratios

nx_max=nx_global/px, ny_max=ny_global/py and nz_max=nz_global/pz

must all be integers. The values of `nx_max`, `ny_max` and `nz_max` must be set in the first lines of the source file `ses3d_modules.f90` in the directory `SOURCE`, **and the code must be recompiled after changing these numbers.**

Within each element, the dynamic fields (e.g. stress tensor, displacement field, ...) are represented by Lagrange polynomials of degree 4 (`lpd`=4). One element therefore comprises $(4+1)^3 = 125$ grid points, meaning that the complete computational domain contains $247,296 \cdot 125 = 30,912,000$ grid points.

For the moment, we perform a pure forward simulation, and therefore set the parameter `adjoint_flag` to 0. The remaining parameters in the `setup` file, as well as the actual meaning of the `adjoint_flag` will be discussed in section 2.1.4.

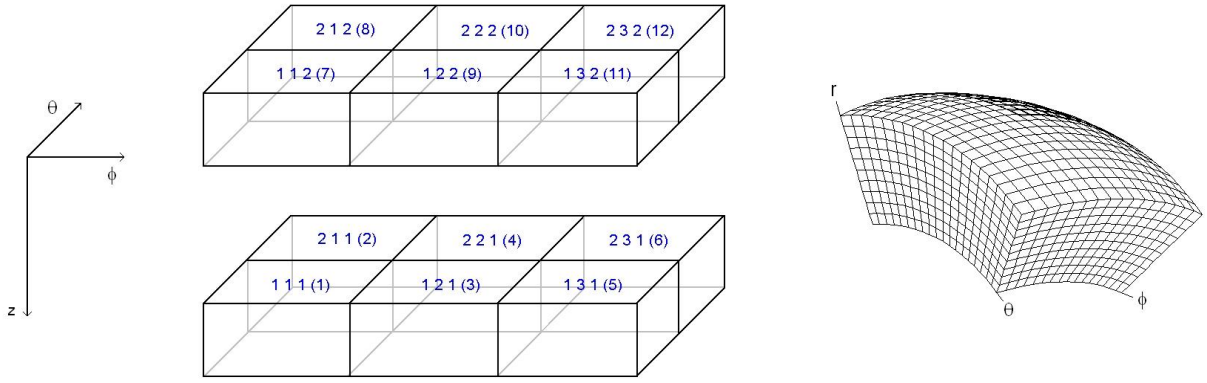

Fig. 2.1 Schematic representation of the parallelisation of `SES3D`. The spherical section is subdivided into subsections, and each subsection is assigned to one compute core. A triple index and a single index (in parenthesis) is assigned to a subsection.

2.1.1.2 Event information

`SES3D` can model a sequence of earthquakes (events), each labelled by an integer number. The event information for the event numbered x is contained in the file `event_x`. A list of all events to be modelled must be provided in the file `event_list` (number of events followed by their label).

In this example, we label our event with 1. The event information in the file `event_1` are the colatitude (`xxs`= 50.740°), longitude (`yys`= 41.040°) and depth (`zzs`= 5,000 m) of the source. The source type (`srctype`) is set to 3, which indicates a moment tensor source. (The options `srctype`=1,2,3 correspond to vector forces in the colatitudinal, longitudinal and radial directions, respectively.) The moment tensor in our simulation corresponds to an explosion:

$$\mathbf{M} = \begin{pmatrix} M_{\theta\theta} & M_{\theta\phi} & M_{\theta r} \\ M_{\theta\phi} & M_{\phi\phi} & M_{\phi r} \\ M_{\theta r} & M_{\phi r} & M_{rr} \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 1.0 \end{pmatrix} \cdot 10^{16}\,\mathrm{N\,m}. \tag{2.1}$$

In our simulation we perform `nt`= 4,000 time steps, each of which is `dt`= 0.13 s long. As output directory for the synthetic seismograms we set `../DATA/OUTPUT/1.8s/`.

2.1.1.3 Receiver locations

A list of receivers must be provided in the file `recfile_*` located in the `INPUT` directory. The * is to be replaced by the event number, i.e. we have `recfile_1` in our case. Following the number of receivers in the list, each receiver is listed with its name and, in the line below, colatitude (°), longitude (°) and depth (m). The depth must be larger or equal to zero, i.e., no receivers are allowed above the surface. Each receiver name is exactly 12 characters long. These characters are typically occupied by the actual station name, the network and any type of additional information.

2.1.1.4 Source time function

The source time function `stf` is given in the `INPUT` directory in the form of an ASCII list. Each entry is a sample of the source time function, and the time spacing must equal the one in the `event_*` files. (This time spacing is not given explicity in `stf`. In our case it is `dt`$= 0.13$ s.)

The source time function should ideally be a bandpass filtered Heaviside function. Periods that are too long cannot be modelled because the computational domain is finite, and periods that are too short cannot be modelled accurately because of discretisation errors. (Furthermore, short periods tend to produce artefacts when they interact with the absorbing boundaries.) As a rule of thumb, the shortest period in the source time function should be such that the corresponding wavelength is not shorter than 1.5 to 2 elements. In our case, one element is around 14 km wide (e.g. 69 elements over a distance of $55.9° - 47.1° = 8.8°$), meaning that the minimum wavelength should be 21 to 28 km. Assuming a minimum propagation velocity of around 3 km/s, this translates to a minimum period of 7 to 9 s. For our example we use a Heaviside function filtered between 8 s and 100 s (see figure 2.2).

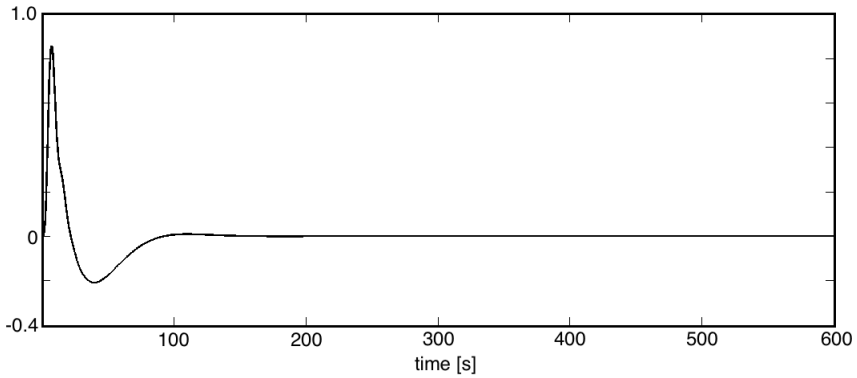

Fig. 2.2 Source time function. We use a Heaviside function bandpass filtered between 8 s and 100 s. The number of samples in the source time function file `stf` must be at least the number of time steps `nt` given in the event files `event_*`.

2.1.2 Model construction

The construction of 3D Earth models in `SES3D`is proceeds in two steps: (1) The construction of a 1D, i.e. radially symmetric, Earth model, and (2) the addition of 3D perturbations to the 1D model. To facilitate the solution of tomographic inverse problems, model construction in `SES3D`is completely decoupled from the actual wave propagation. The necessary executables can be generated by compiling the source code located in `MODELS/SOURCE`, e.g. by running the script `s_make`. The executables are located in `MODELS/MAIN`.

2.1.2.1 Step 1: Constructing 1D Earth models

We construct 1D Earth models by running `generate_models.exe`, located in `MODELS/MAIN`. This programme reads the geometrical setup provided in the `setup` file. The type of 1D Earth model is specified by the parameter `model_type` in the `setup` file. In our case, `model_type`= 1, which generates a homogeneous model with all velocities and density set to zero.

The following alternative Earth models are currently implemented:

1. `model_type=2`: Isotropic version of PREM (Dziewonski and Anderson, 1981).
2. `model_type=3`: All-zero elastic model with a smoothed version of the Q model QL6 (Durek & Ekström).
3. `model_type=4`: Modified version of the isotropic PREM with the 220 km discontinuity replaced by a linear gradient.
4. `model_type=7`: AK135 (Kennett et al., 1995).

The detailed setups of these 1D model can be found in `MODELS/MODELS_1D`.

Running `generate_models.exe` produces files containing the physical model parameters (λ, μ, A, B, C and $1/\rho$) for each of the computational subdomains (48 in our case). These are located in `MODELS/MODELS`. Furthermore, `generate_models.exe` writes the `boxfile` which summarises the geometrical setup and the parallelisation of the computational domain. All these files serve as input for the actual wave propagation.

For debugging purposes, `generate_models.exe` also writes human-readable vertical profiles through each of the subdomains. These are named `prof_*`, where * denotes the index of the subdomain.

2.1.2.2 Step 2: Adding 3D heterogeneity

Description of 3D heterogeneous models

Following the construction of a 1D model, we add 3D perturbations, located in `MODELS/MODELS_3D`. The term *3D perturbations* is loosely defined. In our specific case where the 1D model is the homogeneous model with all parameters set to zero, the perturbations are in fact the absolute 3D model itself.

The 3D perturbations of P velocity (in km/s, file `dvp`), SH velocity (in km/s, file `dvsh`), SV velocity (in km/s, file `dvsv`) and density (in g/cm^3, file `drho`) are parametrised in discrete regular blocks. The geometry of the blocks, i.e. the locations of their bounding grid points, is descibed in the files `block_x` (colatitudinal direction), `block_y` (longitudinal direction) and `block_z` (radial direction).

In our specific example, the horizontal grid spacing is 0.25° from 5871 km to 6266 km radius, and 0.1° from 6266 km to 6371 km radius. The radial grid spacing is 5 km throughout the model. This variable grid spacing roughly reflects the tomographic resolution that we expect in different depth ranges. Figure 2.3 illustrates how the variable grid spacing within the two subdomains is represented in the `block_*` files. A schematic representation of the grid spacing is shown in figure 2.4.

Note that the grid on which the 3D model is described, is completely independent of the numerical grid used in the wave propagation machinery or for the generation of the 1D model. This independence allows us to add 3D model perturbations to any previously constructed 1D model - regardless of its specific setup.

The files containing the actual model perturbations (`dvp, dvsh, dvsv, drho`) are organised as shown in figure 2.5. Note that the velocity and density values are given for the volumetric blocks bounded by the grid points. The number of blocks is therefore smaller than the number of grid points.

block_x (colatitudinal grid spacing)	
2	subdomains
37	grid points in 1. subdomain
47.00	1. grid point location (°)
47.25	2. grid point location (°)
...	
56.00	37. grid point location (°)
91	grid points in 2. subdomain
47.00	1. grid point location (°)
47.10	2. grid point location (°)
...	
56.00	91. grid point location (°)

block_y (longitudinal grid spacing)	
2	subdomains
81	grid points in 1. subdomain
23.00	1. grid point location (°)
23.25	2. grid point location (°)
...	
43.00	81. grid point location (°)
201	grid points in 2. subdomain
23.00	1. grid point location (°)
23.10	2. grid point location (°)
...	
43.00	201. grid point location (°)

block_z (radial grid spacing)	
2	subdomains
80	grid points in 1. subdomain
5871	1. grid point location (km)
5876	2. grid point location (km)
...	
6266	80. grid point location (km)
22	grid points in 2. subdomain
6266	1. grid point location (km)
6271	2. grid point location (km)
...	
6371	22. grid point location (km)

Fig. 2.3 Organisation of the `block_*` files that describe the geometric parametrisation of the 3D heterogeneities.

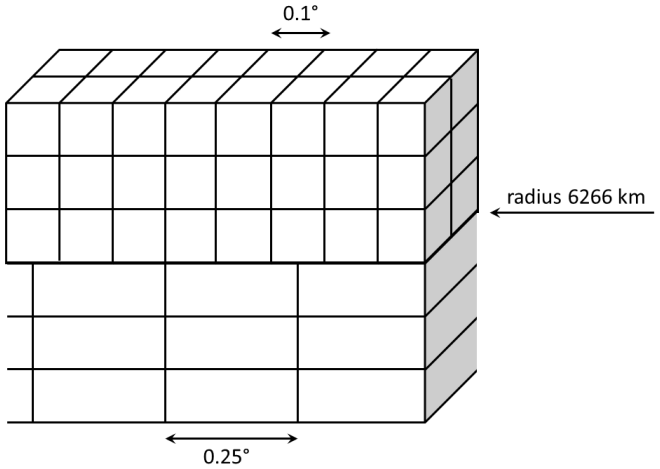

Fig. 2.4 Schematic representation of the variable grid spacing of the 3D heterogeneities. Above a radius of 6266 km, the horizontal grid spacing is 0.1°. Below, it is 0.25°.

dvp (colatitudinal grid spacing)	
2	subdomains
227520	grid points in 1. subdomain (227,520=36·80·79)
9.57198	P velocity (km/s) within 1. block
9.55128	P velocity (km/s) within 2. block
...	
378000	grid points in 2. subdomain (378,000=90·200·21)
...	

Fig. 2.5 Organisation of the P velocity perturbations in the file `dvp`. The number of subdomains (2) is followed by the number of blocks in the first subdomain (227,520). The actual values of P velocity (perturbations) are given as a list that results from looping over the 3D volume. The outer loop if over colatitude, the intermediate loop over longitude, and the inner loop over radius. This list is then followed by a similar list for the second subvolume. The files `dvsh`, `dvsv` and `drho` are organised analogously.

Adding 3D heterogeneity to the 1D model

We can add 3D perturbations in v_P, v_{SV}, v_{SH} and ρ by running the executable `add_perturbation.exe`, located in `MODELS/MAIN`. Using geometric information from the `setup` file and the `boxfile`, the programme `add_perturbation.exe` reads `dvp`, `dvsv`, `dvsh` and `drho` and add these 3D perturbations to the pre-existing model files in `MODELS/MODELS`. For our example we use a 3D model of the Anatolian region, described in Fichtner et al., 2013a and Fichtner et al., 2013b. The v_{SV} distribution in this model is shown in figure 2.6.

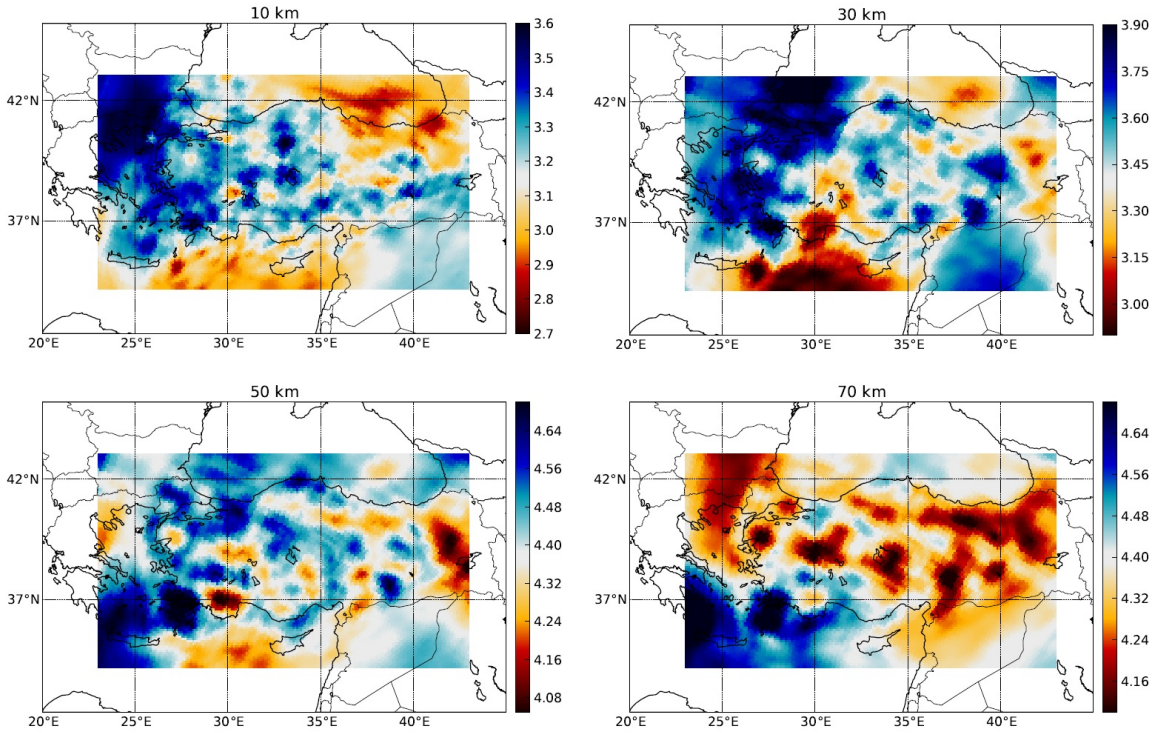

Fig. 2.6 Distribution of v_{SV} in the Anatolian model at 20 km, 50 km and 100 km depth. The depth slices in this figure were generated using the Python tool `models.py` in the `TOOLS` directory.

2.1.3 Simulation of wave propagation and output

The actual wave propagation executable is `ses3d.exe`, located in `MAIN`. Upon running `ses3d.exe`, the geometrical information in `setup`, the event information in `event_x` and `event_list`, as well as the physical model parameters in `MODELS/MODELS` are read. The seismic wavefield is the propagated forward in time, for `nt` time steps. After the last time step, synthetic seismograms are written to the output directory, in our case `DATA/OUTPUT/1.8s/`. The three-component synthetic seismograms for station BALB are shown in figure 2.7.

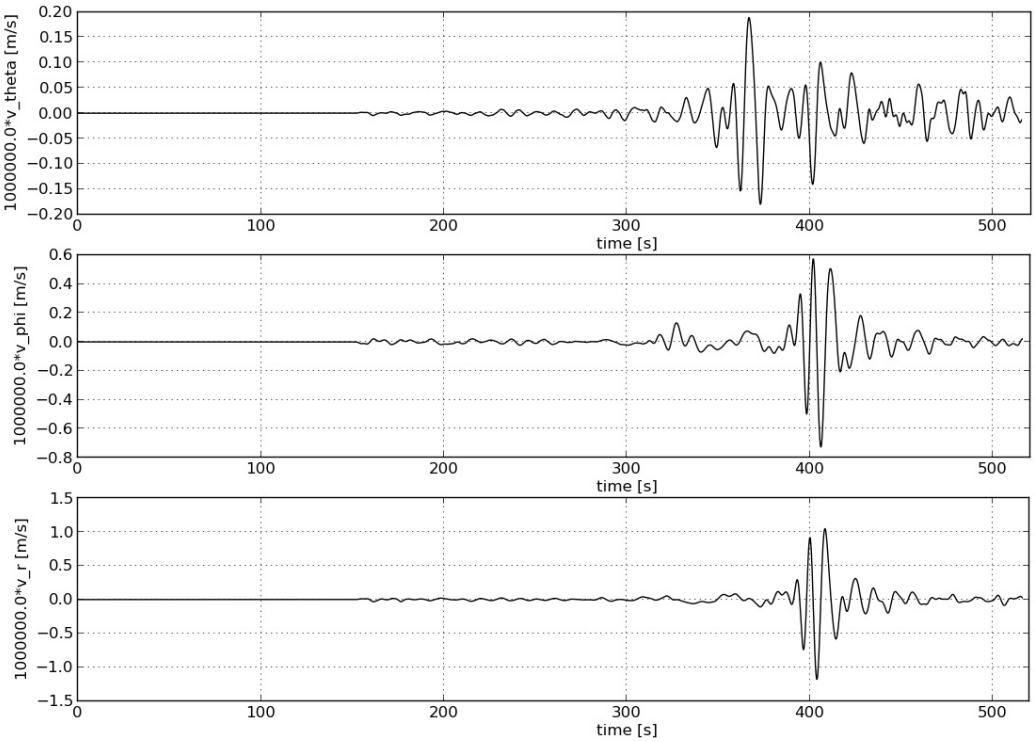

Fig. 2.7 Three-component synthetic seismograms for station BALB.

2.1.4 Computing sensitivity kernels

The computation of sensitivity kernels proceeds in several steps. First of all, a forward simulation must be run with the `adjoint_flag` in the `setup` file set to `1`. This ensures that the forward field is stored in the directory which appears in the last line of the `setup` file. Make sure enough storage is actually available.
One the forward field is computed, adjoint sources for a variety of measurements can be computed using Python tools in the `TOOLS` directory. An example of this procedure is given below:

`run seismograms.py`: Compile Python tool for reading and plotting seismograms.

`run adjoint_source.py`: Compile Python tool for the computation of adjoint sources.

`s=ses3d_seismogram()`: Make empty seismogram structure.

`s.read('../DATA/OUTPUT/1.8s/','BALB_.KO.___')`: Read seismograms for station BALB.

`s.plot(1e6)`: Plot seismograms, scaled by a factor of $1 \cdot 10^6$.

`a=adjoint_source()`: Make empty adjoint source structure.

`a.fetch_seismogram(s)`: Load seismograms into the adjoint source structure.

`a.make_adsrc_mttime('z',391.0,412.0,0.1,True)`: Compute adjoint source for a multitaper measurement of a time delay on the vertical component for the window from 391.0 s to 412.0 s at the frequency of 0.1 Hz.

`a.plot()`: Plot adjoint source.

`a.write('../ADJOINT/1/','ad_src_1')`: Write adjoint source to file.

It is important that the adjoint sources for `event_1` are written to the directory `ADJOINT/1/`. For an event

with event file `event_102`, the adjoint sources would be expected to be located in `ADJOINT/102/`.
For each event, multiple adjoint sources may be computed. In our example, we use only one. The second adjoint source time function would be named `ad_src_2`, and it would be located in the same directory. The file `ad_srcfile` contains a list of all adjoint source time functions used for this specific event. In our case, `ad_srcfile` contains only 2 lines:

`1` (indicates that one adjoint source is used)
`50.36 27.88 1000.0` (colatitude, longitude and depth of the adjoint source)

Once the adjoint sources are computed, simply set the `adjoint_flag` to `2` and re-run the wave propagation code. Sensitivity kernels will then be computed automatically, and stored in the output directory specified in the event file. Then, using `project_kernels.exe` in the `TOOLS` directory, the kernels can be projected onto the grid used to represent 3D models. This is particularly helpful in the context of tomographic inverse problems, but also for plotting purposes. Since the projected kernels have the same format as a 3D model, we can use the Python tool `models.py` to read and plot the projected kernels. The result is shown in figure 2.8.

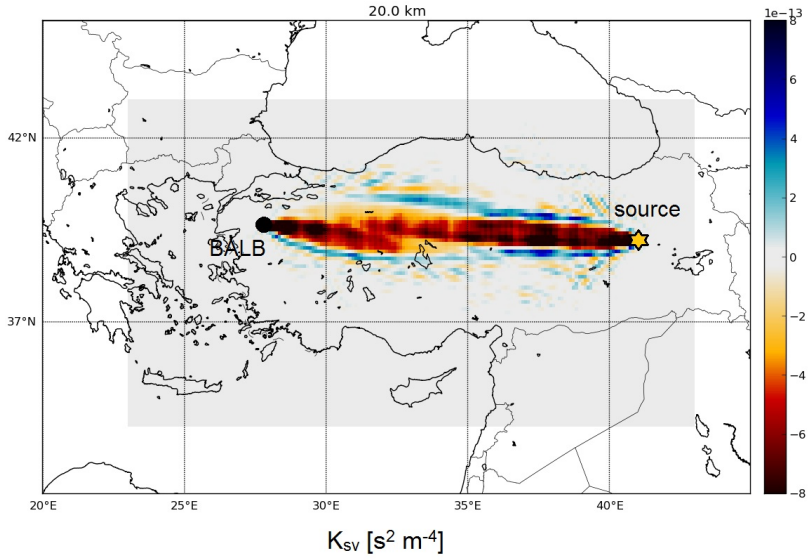

Fig. 2.8 Sensitivity kernel with respect to the SV velocity at 20 km depth. The measurement is a multitaper measurement of a time delay on the vertical component of station BALB for the window from 391.0 s to 412.0 s at the frequency of 0.1 Hz.

2.2 Continental-scale wave propagation: North America

Seismic wave propagation in `SES3D` may become inefficient when the computational domain is too close to the poles where the elements become very small, thus enforcing a small time step. This problem is particularly relevant for large domains that approach or even include the poles. To avoid excessively small time steps, the computational domain can be rotated away from the poles towards the equator.

This rotation procedure will play a central role in the following scenario where we consider seismic wave propagation across North America and the North Atlantic. The scenario-specific input and source files are located in the directory `SCENARIOS/NORTH_AMERICA`.

2.2.1 Rotating the computational domain

We are interested in the computational domain shown in figure 2.9 that covers the North American continent and the North Atlantic. Since the domain closely approaches the North pole, where elements become very small, it should ideally be rotated southwards to avoid excessively small time steps. In this example, we rotate the complete domain southwards by 30° around an axis given in terms of the unit vector

$$\mathbf{n} = \begin{pmatrix} \cos(40^\circ) \\ \sin(40^\circ) \\ 0.0 \end{pmatrix} \tag{2.2}$$

The rotated domain is shown in figure 2.10. In rotated coordinates, the computational domain falls between the following colatitudes and longitudes that are specified in the `setup` file: `theta_min=45.2`, `theta_max=114.8`, `phi_min=-119.8`, `phi_max=-0.2`.

Since the actual computations are not performed in the true physical but in the rotated domain, various input parameters must be adapted. The input parameters include the station and event coordinates, as well as the moment tensor. Python tools for the rotation of source/receiver locations and moment tensors can be found in `TOOLS/rotation.py`. These tools are also described in section 3. While the original epicentral coordinates of our event are $\theta = 34.08^\circ$, $\phi = -17.80^\circ$, the rotated cooridnates are $\theta' = 61.30^\circ$, $\phi' = -30.10^\circ$. Similarly the moment tensor is rotated as follows:

$$\mathbf{M} = \begin{pmatrix} M_{tt} & M_{tp} & M_{tr} \\ M_{tp} & M_{pp} & M_{pr} \\ M_{tr} & M_{pr} & M_{rr} \end{pmatrix} = \begin{pmatrix} -3.794 & 1.978 & -1.145 \\ 1.977 & 4.004 & -0.514 \\ -1.145 & -0.514 & -0.211 \end{pmatrix}$$

$$\rightarrow$$

$$\mathbf{M}' = \begin{pmatrix} M'_{tt} & M'_{tp} & M'_{tr} \\ M'_{tp} & M'_{pp} & M'_{pr} \\ M'_{tr} & M'_{pr} & M'_{rr} \end{pmatrix} = \begin{pmatrix} -4.220 & -0.644 & -0.935 \\ -0.644 & 4.430 & -0.838 \\ -0.935 & -0.838 & -0.211 \end{pmatrix} \tag{2.3}$$

The rotated event parameters are specified in `event_1`.

Most plotting routines for models and seismograms that are located in the `TOOLS` directory, correctly account for the rotation. All that needs to be done is to set the rotation vector and the rotation angle in the file `TOOLS/rotation_parameters.txt`.

2.2.2 Source time function

While the computational domain in this scenario is comparatively large, its discretisation with ~ 1.8 elements per degree (specified in the `setup` file) is rather coarse. We are thus limited to the modelling of long-period wave propagation. To construct a suitable source time function, we can make use of the `make_stf.py` scripts in the `TOOLS` directory. The function call

```
make_stf(0.2,12000,1.0/200.0,1.0/100.0,'../INPUT/stf')
```

Produces a source time function that is a bandpass-filtered Heaviside function with cutoffs at $1/200$ Hz and $1/100$ Hz. The time step is equal to 0.2 s, identical to the time step `dt` specified in the `setup` file. The number of time steps is 12000, and the output file is written to `../INPUT/stf`. The source time function in the time and frequency domain is displayed in figure 2.11.

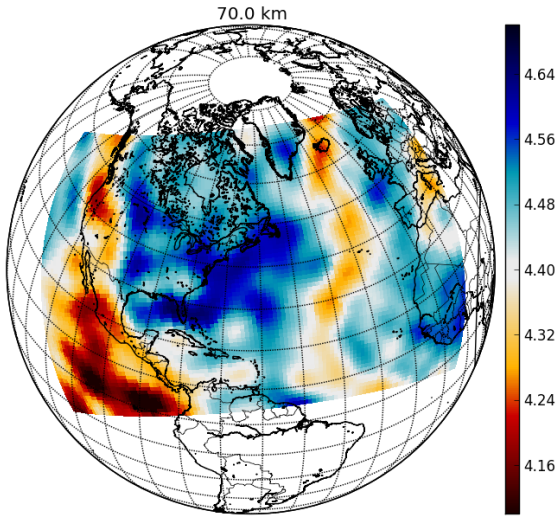

Fig. 2.9 Horizontal slice through the SV velocity in model S40RTS (Ritsema et al., 2011) at 70 km depth beneath North America and the North Atlantic. The computational domain approaches the North pole, and therefore needs to be rotated southwards. This plot was made with the tools provided in `TOOLS/models.py`.

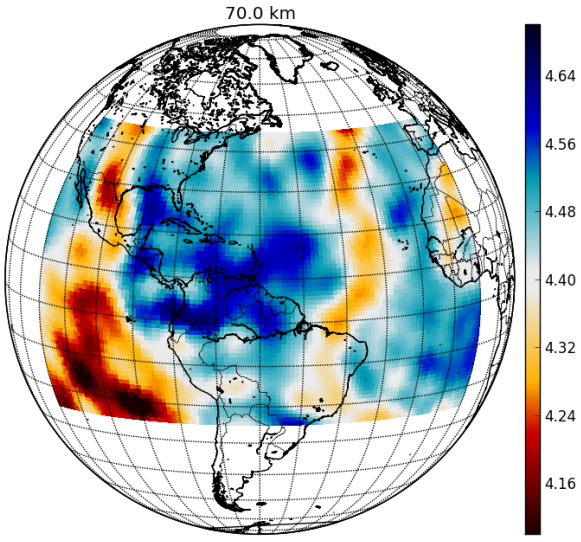

Fig. 2.10 Rotated North America model. All computations are performed in this rotated domain. This plot was made with the tools provided in `TOOLS/models.py`.

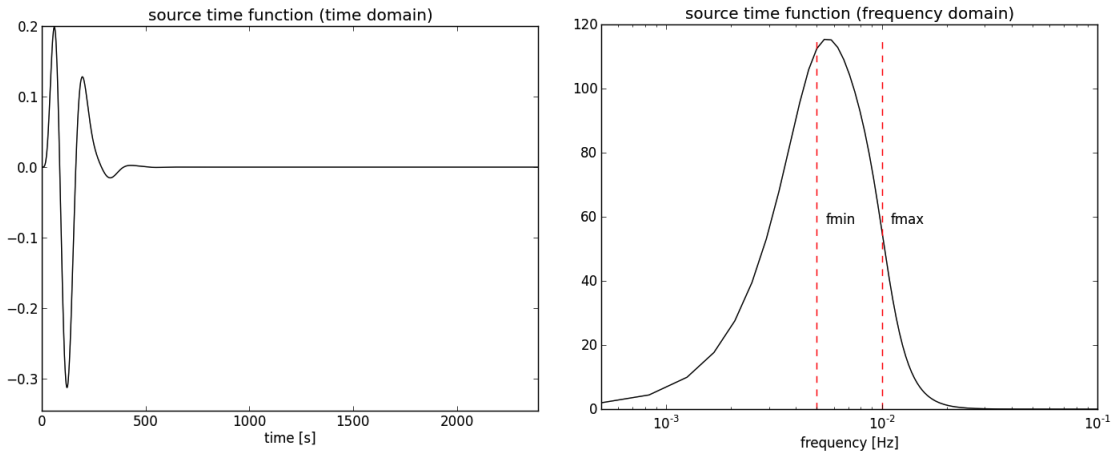

Fig. 2.11 Time- and frequency-domain representation of the source time function produced by `make_stf(0.2,12000,1.0/200.0,1.0/100.0,'../INPUT/stf')`

2.2.3 Constructing the 3D model

The construction of a 3D Earth model for this scenario is similar to the previous one described in section 2.1. The only difference is that the coordinates in the `block_x`, `block_y` and `block_z` files must of course be rotated towards the equator.

In the first step, we again generate a 1D model using `generate_models`. As 1D model we choose one where all elastic parameters are set to zero, and where a smoothed version of the Q model QL6 (Durek & Ekström, 1996) is implemented. To use this model, set `model_type` to `3` in the `setup` file. More details on visco-elastic dissipation for this specific scenario and from a more mathematical perspective can be found in sections 2.2.4 and 5.1, respectively.

Following this initial step, we add 3D heterogeneities, again using `add_perturbation`. To ensure that the heterogeneities are properly implemented, we can visualise the material parameters on the spectral-element grid using thy Python tool `ses3d_fields.py` in the `TOOLS` directory (see also section 3). The resulting plot is shown in figure 2.12.

2.2.4 Visco-elastic dissipation

To enable visco-elastic dissipation in `SES3D`, the `is_diss` flag in the `setup` file must be set to 1. The mathematical description and numerical implementation of visco-elastic dissipation is described in chapter 5. The visco-elastic properties of the Earth model are encoded in a set of relaxation parameters that can be computed using the Python code `Q_discrete.py` located in the `TOOLS` directory. Requesting a constant Q for periods between 80 s and 250 s, `Q_discrete.py` produces the relaxation weights $1.684, 0.838, 1.357$, and the corresponding relaxation times $3.200, 17.692, 74.504$. These values must be written into the `relax` file in the `INPUT` directory. An illustration of the constant target Q and the numerical approximation, as well as of the corresponding phase velocity dispersion, is provided in figure 2.13.

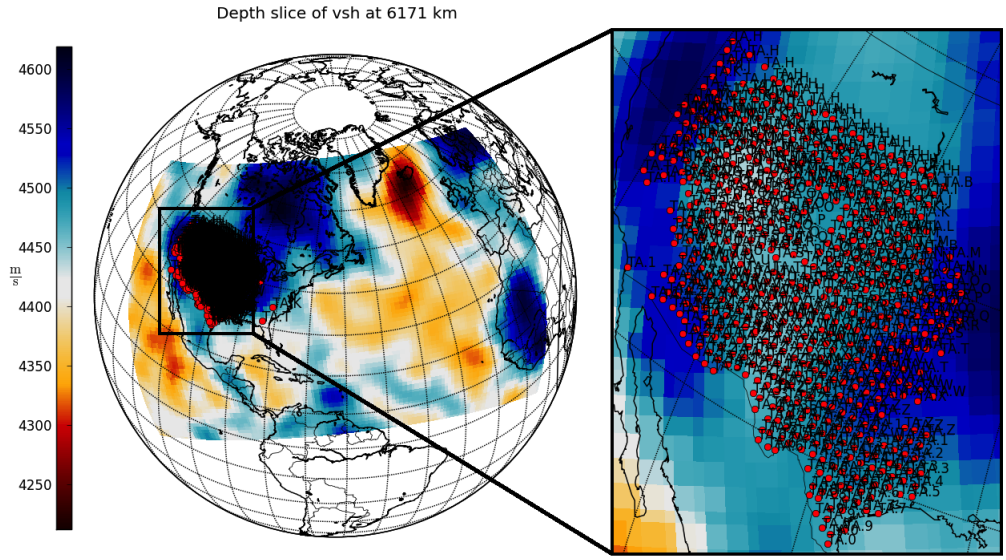

Fig. 2.12 Rotated North America model (v_{SV}) as implemented on the spectral-element grid. A zoom into the western part of USArray is shown to the right. This plot was made using `ses3d_fields.py` from the `TOOLS` directory.

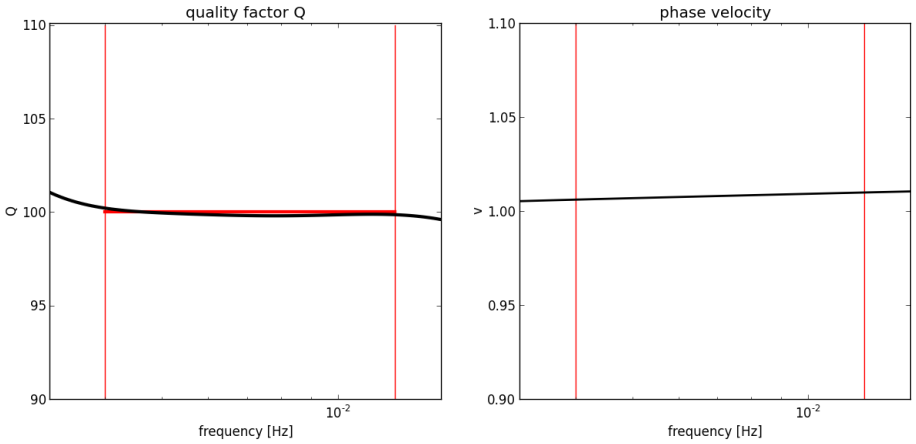

Fig. 2.13 **Left**: Comparison of the constant target shear Q with value 100 and the corresponding numerical approximation, as computed by `Q_discrete.py`. **Right**: Phase velocity dispersion.

2.2.5 Wiggly lines

To plot seismograms, we can use the Python class `ses3d_seismogram` in the `TOOLS` directory. Simply run `seismograms.py` and then initiate a class member, e.g. by typing

```
s=ses3d_seismogram().
```

Using the `read` function, we can read a seismogram by giving the directory and station name:

```
s.read('../DATA/OUTPUT/1/','TA.L16A_.___').
```

Typing

[s.plot()]

will produce the plot shown in figure 2.14.

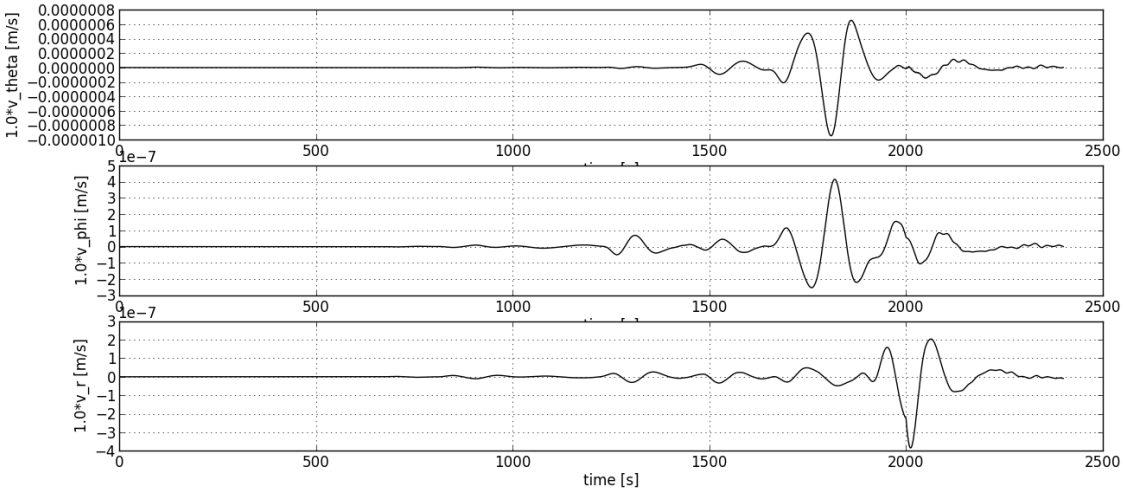

Fig. 2.14 Three-component velocity seismograms at station TA.L16A.

Chapter 3
Python tools

SES3D comes with a collection of Python tools for visualisation, computing source time functions and adjoint sources, and various other purposes. All of these tools are located in the TOOLS directory. Being within python or ipython, a instructions for the use of these tools can be obtained by simply typing help(name of the tool). Required Python packages include Numpy, Scipy, Matplotlib, Basemap and Obspy.The following is a commented list of currently available tools:

3.1 Visualising and editing Earth models, snapshots and kernels

3.1.1 Visualising 3-D fields defined in terms of constant-property blocks

Three-dimensional fields, including Earth models, velocity snapshots and sensitivity kernels, can be visualised with the help of models.py, provided that they are parametrised in the form described in section 2.1.2.2. An instance of the class ses3d_model can use the function read to read a 3-D field from a file given in terms of the path and filename. For this, the appropriate block_* files must be in that path as well. A vertical slice can be plotted using the function plot_slice.

3.1.2 Writing 3-D fields defined in terms of constant-property blocks

After reading a 3-D field as described in the previous paragraph, the field may be manipulated and then written again using the write function of the ses3d_model class.

3.1.3 Converting 3-D fields defined in terms of constant-property blocks into vtk format

For more elaborate visualisation, for instance in Paraview or VisIt, the 3-D fields need to be converted into the vtk format. The function convert_to_vtk of the ses3d_model class performs this conversion task.

3.1.4 Visualising Earth models and velocity snapshots on the spectral-element grid

The Python class `ses3d_fields` contains tools for the visualisation of Earth models and velocity field snapshots defined on the spectral-element grid of SES3D. To plot, for instance, v_{SH} at 20 km depth with a colourbar ranging from 2698.0 m/s to 3972.0 m/s with a high-resolution coastline, just type

```
run ses3d_fields.py
field=ses3d_fields('../','/MODELS/MODELS/','earth_model')
field.plot_depth_slice('vsh',20.0,2698.0,3872.0,res='h')
```

The first path '../' denotes the parent directory of the SES3D project, and the second '/MODELS/MODELS/' is the location of the Earth model parameters. Similarly, for plotting velocity field snapshots for v_x at 100 km depth and at iteration 1000, type

```
run ses3d_fields.py
field=ses3d_fields('../','/DATA/OUTPUT/','velocity_snapshot')
field.plot_depth_slice('vx',100.0,-9e-8,9e-8,iteration=1000,res='h')
```

3.2 Computing source time functions

Source time functions can be computed and stored using `make_stf.py`. Source time functions are computed as bandpass filtered Heaviside functions. `make_stf.py` is deliberately simplistic. For real-data applications you must ensure that the bandpass applied in `make_stf.py` is *exactly* the bandpass applied to the data. Thus, you may want to adjust `make_stf.py` to your specific needs. An example for the computation of a source time function can be found in section 2.2.2.

3.3 Computing adjoint sources

Code for the computation of adjoint sources for some simple measurements can be found in `adjoint_source.py`. This tools fetches an `SES3D` seismogram, makes a measurement in a pre-defined time window, and then computes the corresponding adjoint source. In the absence of real data, only data-independent adjoint sources can be computed. The currently implemented measurements are the cross-correlation traveltime shift, relative amplitude differences, multi-taper phase shifts and multi-taper amplitude differences. A concrete example for the computation of adjoint sources is given in section 2.1.4.

3.4 Computing relaxation parameters of Q models

As described in chapter 5, visco-elastic dissipation in `SES3D` is described by a superposition of a finite number of relaxation mechanisms. The weights of these mechanisms and their respective relaxation times determine the behavior of Q within a pre-defined frequency band. The Python code `Q_discrete.py` computes these weights and relaxation times. The input parameters (frequency band, number of mechanisms, ...) must be given directly in the input section of the code, i.e. they are not passed as parameters to a function (because there are just too many parameters). A concrete example for the computation of relaxation weights and times can be found in section 2.2.4.

3.5 Rotating coordinates and moment tensors

When working with a rotated coordinate system, a little set of tools for rotating coordinates and moment tensors is useful. These can be found in `rotation.py`. `rotate_coordinates` rotates coordinates (colatitude and longitude) and `rotate_moment_tensor` rotates moment tensors. A detailed description is provided in the Python code itself.

3.6 Reading and plotting seismograms

The code `seismograms.py` contains the definition of the `ses3d_seismogram` class that can be used to read and visualise synthetic seismograms produced by `SES3D`. A detailed example can be found in section 2.2.5.

Part II
Mathematical background

Chapter 4
The seismic wave equation

This chapter is concerned with the mathematical description of seismic wave propagation in the Earth and the derivation of equations that can be solved numerically in an efficient way. Following a brief review of the elastic wave equation and its subsidiary conditions in section 4.1, we elaborate on the simulation of anisotropy (section 4.2) and attenuation (section 5.1). We also cover the implementation and analysis of absorbing boundaries within this chapter on the analytical setup, because their properties are relatively independent from a particular numerical scheme.

4.1 The elastic wave equation

The propagation of seismic waves in the Earth can be modelled with the elastic wave equation

$$\rho(\mathbf{x})\ddot{\mathbf{u}}(\mathbf{x},t) - \nabla\cdot\boldsymbol{\sigma}(\mathbf{x},t) = \mathbf{f}(\mathbf{x},t)\,, \tag{4.1}$$

that relates the displacement field $\mathbf{u}$ to the mass density ρ, the stress tensor $\boldsymbol{\sigma}$ and an external force density $\mathbf{f}$ (e.g. Dahlen & Tromp, 1998; Kennett, 2001; Aki & Richards, 2002). The elastic wave equation as presented above is a linearised version of the momentum balance equation, i.e., of Newton's second law. It is valid under the assumption that deviations from the reference configuration of the Earth are small. Furthermore, the rotation of the Earth and its self-gravitation are omitted. These effects are negligible when oscillation periods are shorter than ~ 200 s. At the surface $\delta\oplus$ of the Earth $\oplus$ the normal components of the stress tensor $\boldsymbol{\sigma}$ vanish:

$$\boldsymbol{\sigma}\cdot\mathbf{e}_r|_{\mathbf{x}\in\partial\oplus} = \mathbf{0}\,. \tag{4.2}$$

Equation (4.2) is the free surface boundary condition. Both the displacement field $\mathbf{u}$ and the velocity field $\mathbf{v} = \dot{\mathbf{u}}$ are required to vanish prior to $t = t_0$ when the external force $\mathbf{f}$ starts to act, i.e.,

$$\mathbf{u}|_{t<t_0} = \mathbf{v}|_{t<t_0} = \mathbf{0}\,. \tag{4.3}$$

For convenience we will mostly choose $t_0 = 0$. To obtain a complete set of equations, the stress tensor $\boldsymbol{\sigma}$ has to be related to the displacement field $\mathbf{u}$. It is usually assumed that the rheology is visco-elastic, meaning that the current stress tensor $\boldsymbol{\sigma}$ depends linearly on the history of the strain tensor $\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^T)$:

$$\boldsymbol{\sigma}(\mathbf{x},t) = \int_{-\infty}^{\infty} \dot{\mathbf{C}}(\mathbf{x},t-t') : \boldsymbol{\varepsilon}(\mathbf{x},t')\,dt'\,. \tag{4.4}$$

The fourth order tensor $\mathbf{C}$ is the elastic tensor. In the case of a perfectly elastic medium the elastic tensor is of the form $\mathbf{C}(\mathbf{x},t) = \mathbf{C}(\mathbf{x})H(t)$, where H is the Heaviside function. We then have $\boldsymbol{\sigma} = \mathbf{C}(\mathbf{x}) : \boldsymbol{\varepsilon}(\mathbf{x},t)$. The

symmetry of ε, the conservation of angular momentum and the relation of $\mathbf{C}$ to the internal energy (e.g. Dahlen & Tromp, 1998) require that the components of $\mathbf{C}$ satisfy the following symmetry relations:

$$C_{ijkl} = C_{klij} = C_{jikl}\,. \tag{4.5}$$

Moreover, the elastic tensor is causal:

$$\mathbf{C}(t)|_{t<t_0} = \mathbf{0}\,. \tag{4.6}$$

Equation (4.4) is − unlike the wave equation within its limits of validity − not a fundamental law of physics but an empirical relation that has been found to describe a wide range of phenomena very well. It can be regarded as a linearisation of a more general non-linear constitutive relation. Its validity is, in this sense, restricted to scenarios where the strain tensor ε is a small quantity. The particular choices of $\mathbf{C}$ and its time dependence will be the subjects of the following sections on attenuation and anisotropy.

4.2 Anisotropy

Anisotropy is the dependence of the elastic tensor on the orientation of the coordinate system. Its most direct seismological expression is the dependence of seismic velocities on the propagation and polarisation directions of elastic waves. It is thought to play a major role in the Earth's crust and upper mantle.
Besides the splitting of shear waves, the Love wave-Rayleigh wave discrepancy is one of the principal seismic observations that is directly related to anisotropy: A Love wave and a Rayleigh wave travelling in the same direction usually exhibit different wave speeds as a consequence of their different polarisations. This led to the inclusion of anisotropy with radial symmetry axis in the global reference model PREM (Dziewonski & Anderson, 1981). In this model the anisotropy is limited to the upper 220 km.
Guided by these observations, we decided to implement anisotropy with radial symmetry axis. For such a medium, there are only 5 independent elastic tensor components that are different from zero. They can be summarised in a 6×6 matrix (e.g. Babuska & Cara, 1991):

$$\begin{aligned}
&\begin{pmatrix}
C_{rrrr} & C_{rr\phi\phi} & C_{rr\theta\theta} & C_{rr\phi\theta} & C_{rrr\theta} & C_{rrr\phi}\\
C_{\phi\phi rr} & C_{\phi\phi\phi\phi} & C_{\phi\phi\theta\theta} & C_{\phi\phi\phi\theta} & C_{\phi\phi r\theta} & C_{\phi\phi r\phi}\\
C_{\theta\theta rr} & C_{\theta\theta\phi\phi} & C_{\theta\theta\theta\theta} & C_{\theta\theta\phi\theta} & C_{\theta\theta r\theta} & C_{\theta\theta r\phi}\\
C_{\phi\theta rr} & C_{\phi\theta\phi\phi} & C_{\phi\theta\theta\theta} & C_{\phi\theta\phi\theta} & C_{\phi\theta r\theta} & C_{\phi\theta r\phi}\\
C_{r\theta rr} & C_{r\theta\phi\phi} & C_{r\theta\theta\theta} & C_{r\theta\phi\theta} & C_{r\theta r\theta} & C_{r\theta r\phi}\\
C_{r\phi rr} & C_{r\phi\phi\phi} & C_{r\phi\theta\theta} & C_{r\phi\phi\theta} & C_{r\phi r\theta} & C_{r\phi r\phi}
\end{pmatrix}\\
&= \begin{pmatrix}
\lambda+2\mu & \lambda+c & \lambda+c & 0 & 0 & 0\\
\lambda+c & \lambda+2\mu+a & \lambda+a & 0 & 0 & 0\\
\lambda+c & \lambda+a & \lambda+2\mu+a & 0 & 0 & 0\\
0 & 0 & 0 & \mu & 0 & 0\\
0 & 0 & 0 & 0 & \mu+b & 0\\
0 & 0 & 0 & 0 & 0 & \mu+b
\end{pmatrix}
\end{aligned} \tag{4.7}$$

Love (1892) proposed an alternative parametrisation

$$\begin{pmatrix} C_{rrrr} & C_{rr\phi\phi} & C_{rr\theta\theta} & C_{rr\phi\theta} & C_{rrr\theta} & C_{rrr\phi} \\ C_{\phi\phi rr} & C_{\phi\phi\phi\phi} & C_{\phi\phi\theta\theta} & C_{\phi\phi\phi\theta} & C_{\phi\phi r\theta} & C_{\phi\phi r\phi} \\ C_{\theta\theta rr} & C_{\theta\theta\phi\phi} & C_{\theta\theta\theta\theta} & C_{\theta\theta\phi\theta} & C_{\theta\theta r\theta} & C_{\theta\theta r\phi} \\ C_{\phi\theta rr} & C_{\phi\theta\phi\phi} & C_{\phi\theta\theta\theta} & C_{\phi\theta\phi\theta} & C_{\phi\theta r\theta} & C_{\phi\theta r\phi} \\ C_{r\theta rr} & C_{r\theta\phi\phi} & C_{r\theta\theta\theta} & C_{r\theta\phi\theta} & C_{r\theta r\theta} & C_{r\theta r\phi} \\ C_{r\phi rr} & C_{r\phi\phi\phi} & C_{r\phi\theta\theta} & C_{r\phi\phi\theta} & C_{r\phi r\theta} & C_{r\phi r\phi} \end{pmatrix}$$
$$= \begin{pmatrix} C_L & F_L & F_L & 0 & 0 & 0 \\ F_L & A_L & A_L - 2N_L & 0 & 0 & 0 \\ F_L & A_L - 2N_L & A_L & 0 & 0 & 0 \\ 0 & 0 & 0 & N_L & 0 & 0 \\ 0 & 0 & 0 & 0 & L_L & 0 \\ 0 & 0 & 0 & 0 & 0 & L_L \end{pmatrix} \tag{4.8}$$

We chose to use the parametrisation from equation (4.7) because it easily allows us to model isotropy by simply setting $a = b = c = 0$. Remember, that all components of the elastic tensor are functions of time, t, and space, $\mathbf{x}$. Density, ρ and the elastic parameters λ, μ, a, b and c can be related to the wave speeds of SH, SV, PH and PV waves:

$$v_{\text{SV}} = \sqrt{\frac{\mu + b}{\rho}}, \quad v_{\text{SH}} = \sqrt{\frac{\mu}{\rho}}, \quad v_{\text{PV}} = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \quad v_{\text{PH}} = \sqrt{\frac{\lambda + 2\mu + a}{\rho}}. \tag{4.9}$$

The velocities of SH, SV, PH and PV waves specify only two of the three additional elastic parameters necessary for anisotropy with radial symmetry axis, namely a and b. Also radially propagating S waves do not allow us to find c, because they propagate with the velocity $\sqrt{(\mu + b)/\rho}$, just as SV waves. In fact, using plane waves, c can be determined only from P waves that do not travel in exactly radial or horizontal directions. Following Takeuchi & Saito (1972) and Dziewonski & Anderson (1981) we incorporate c into a dimensionless parameter

$$\eta := (\lambda + c)/(\lambda + a). \tag{4.10}$$

Chapter 5
Description and implementation of attenuation

5.1 The visco-elastic rheology and memory variables

The numerical implementation of attenuation is largely motivated by technical convenience and not so much by the true physics of seismic wave attenuation in the interior of the Earth. In our analysis we closely follows Robertsson et al. (1994), but introduce some modifications that facilitate the construction of constant-Q models. Assume that σ, C and ε are representative of some particular components of $\sigma, \mathbf{C}$ and ε, respectively. Then a scalar version of the stress-strain relation is given by

$$\dot{\sigma}(t) = (\dot{C} * \dot{\varepsilon})(t) = \int_{-\infty}^{\infty} \dot{C}(t-t')\dot{\varepsilon}(t')\,dt' . \tag{5.1}$$

The spatial dependence has been omitted for brevity. As already discussed, we choose the stress relaxation function or elastic tensor component C to be that of a superposition of N standard linear solids, weighted by coefficients D_p $(p = 1, ..., N)$, i.e.,

$$C(t) := C_r \left[1 - \sum_{p=1}^{N} D_p \left(1 - \frac{\tau_{\varepsilon p}}{\tau_{\sigma p}} \right) e^{-t/\tau_{\sigma p}} \right] H(t) , \tag{5.2}$$

where $\tau_{\varepsilon p}$ and $\tau_{\sigma p}$ are the strain and stress relaxation times of the pth standard linear solid, respectively. The symbol H denotes the Heaviside function and C_r is the relaxed modulus. Equation (5.2) is very general so that different sets of relaxation times can give almost the same relaxation function $C(t)$. To reduce this subjectively undesirable non-uniqueness we limit the number of free parameters. Following the τ-method introduced by Blanch et al. (1995) we determine the relaxation times $\tau_{\varepsilon p}$ by defining a dimensionless variable τ through

$$\tau := \frac{\tau_{\varepsilon p}}{\tau_{\sigma p}} - 1 . \tag{5.3}$$

This gives

$$C(t) = C_r \left[1 + \tau \sum_{p=1}^{N} D_p\, e^{-t/\tau_{\sigma p}} \right] H(t) . \tag{5.4}$$

Differentiating (5.4) with respect to t and introducing the result into equation (5.1) yields

$$\dot{\sigma}(t) = C_r(1 + s\,\tau)\,\dot{\varepsilon}(t) + C_r \sum_{p=1}^{N} M_p , \quad s := \sum_{p=1}^{N} D_p \tag{5.5}$$

where the memory variables M_p are defined by

$$M_p := -\frac{D_p\,\tau}{\tau_{\sigma p}} \int\limits_{-\infty}^{\infty} e^{-(t-t')/\tau_{\sigma p}}\, H(t-t')\,\dot{\varepsilon}(t')\,dt'\,. \tag{5.6}$$

The differentiation of (5.6) with respect to time yields a set of first-order differential equations for the memory variables:

$$\dot{M}_p = -\frac{D_p\,\tau}{\tau_{\sigma p}}\,\dot{\varepsilon} - \frac{1}{\tau_{\sigma p}}\,M_p\,. \tag{5.7}$$

Anelasticity can thus be modelled by simultaneously sloving the momentum equation, a modified stress-strain relation and a set of N ordinary differential equations for the memory variables M_p. The memory variables are formally independent of the elastic parameter C_r. This formulation, proposed by Moczo & Kristek (2005), gives more accurate results in the case of strong attenuation heterogeneities than the classical formulation by Robertsson et al. (1994).
Generalising equations (5.5) and (5.7) to the case of a three-dimensional and anisotropic medium with radial symmetry axis is straightforward. The component-wise stress-strain relation in the absence of dissipation is given by the following set of equations:

$$\sigma_{rr} = (\kappa - 2\mu/3)(\varepsilon_{rr}+\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) + 2\mu\varepsilon_{rr} + c(\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) \tag{5.8a}$$
$$\sigma_{\phi\phi} = (\kappa - 2\mu/3)(\varepsilon_{rr}+\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) + 2\mu\varepsilon_{\phi\phi} + c\varepsilon_{rr} + a(\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) \tag{5.8b}$$
$$\sigma_{\theta\theta} = (\kappa - 2\mu/3)(\varepsilon_{rr}+\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) + 2\mu\varepsilon_{\theta\theta} + c\varepsilon_{rr} + a(\varepsilon_{\theta\theta}+\varepsilon_{\phi\phi}) \tag{5.8c}$$
$$\sigma_{r\phi} = 2(\mu+b)\varepsilon_{r\phi} \tag{5.8d}$$
$$\sigma_{r\theta} = 2(\mu+b)\varepsilon_{r\theta} \tag{5.8e}$$
$$\sigma_{\phi\theta} = 2\mu\varepsilon_{\phi\theta} \tag{5.8f}$$

We introduced the bulk modulus $\kappa = \lambda + \frac{2}{3}\mu$ because it is $-$ in contrast to the Lamé parameter λ $-$ physically interpretable.[1] The transition to the dissipative medium is now made by analogy:

[1] There is, to the best of my knowledge, no physical interpretation of λ. The unphysical nature of this parameter becomes most apparent through the fact that the Q factor associated with λ, denoted by Q_λ, can be negative for positive Q_μ and Q_κ. Thus, if there were a physical process, e.g. an elastic wave, that depended only on λ and ρ, then this process would go hand in hand with a continuously growing elastic energy.

$$\dot{\sigma}_{rr} = \left[\kappa_r(1+s_\kappa\tau_\kappa) - \frac{2}{3}\mu_r(1+s_\mu\tau_\mu)\right](\dot{\varepsilon}_{rr}+\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + 2\mu_r(1+s_\mu\,\tau_\mu)\dot{\varepsilon}_{rr} + c(\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + \kappa_r\sum_{p=1}^{N}\left(K_p^{rr}+K_p^{\theta\theta}+K_p^{\phi\phi}\right) - \frac{2}{3}\mu_r\sum_{p=1}^{N}\left(M_p^{\theta\theta}+M_p^{\phi\phi}\right) + \frac{4}{3}\mu_r\sum_{p=1}^{N}M_p^{rr} \tag{5.9a}$$

$$\dot{\sigma}_{\phi\phi} = \left[\kappa_r(1+s_\kappa\tau_\kappa) - \frac{2}{3}\mu_r(1+s_\mu\tau_\mu)\right](\dot{\varepsilon}_{rr}+\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + 2\mu_r(1+s_\mu\tau_\mu)\dot{\varepsilon}_{\phi\phi} + c\dot{\varepsilon}_{rr} + a(\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + \kappa_r\sum_{p=1}^{N}\left(K_p^{rr}+K_p^{\theta\theta}+K_p^{\phi\phi}\right) - \frac{2}{3}\mu_r\sum_{p=1}^{N}\left(M_p^{rr}+M_p^{\theta\theta}\right) + \frac{4}{3}\mu_r\sum_{p=1}^{N}M_p^{\phi\phi} \tag{5.9b}$$

$$\dot{\sigma}_{\theta\theta} = \left[\kappa_r(1+s_\kappa\tau_\kappa) - \frac{2}{3}\mu_r(1+s_\mu\tau_\mu)\right](\dot{\varepsilon}_{rr}+\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + 2\mu_r(1+s_\mu\tau_\mu)\dot{\varepsilon}_{\theta\theta} + c\dot{\varepsilon}_{rr} + a(\dot{\varepsilon}_{\theta\theta}+\dot{\varepsilon}_{\phi\phi}) + \kappa_r\sum_{p=1}^{N}\left(K_p^{rr}+K_p^{\theta\theta}+K_p^{\phi\phi}\right) - \frac{2}{3}\mu_r\sum_{p=1}^{N}\left(M_p^{rr}+M_p^{\phi\phi}\right) + \frac{4}{3}\mu_r\sum_{p=1}^{N}M_p^{\theta\theta} \tag{5.9c}$$

$$\dot{\sigma}_{r\phi} = 2\mu_r(1+s_\mu\tau_\mu)\dot{\varepsilon}_{r\phi} + 2b\dot{\varepsilon}_{r\phi} + 2\mu_r\sum_{p=1}^{N}M_p^{r\phi} \tag{5.9d}$$

$$\dot{\sigma}_{r\theta} = 2\mu_r(1+s_\mu\tau_\mu)\dot{\varepsilon}_{r\theta} + 2b\dot{\varepsilon}_{r\phi} + 2\mu_r\sum_{p=1}^{N}M_p^{r\theta} \tag{5.9e}$$

$$\dot{\sigma}_{\phi\theta} = 2\mu_r(1+s_\mu\tau_\mu)\dot{\varepsilon}_{\phi\theta} + 2\mu_r\sum_{p=1}^{N}M_p^{\phi\theta} \tag{5.9f}$$

For equations (5.18) we assumed that the parameters a, b and c are not involved in the dissipation of elastic energy. The memory variables associated with μ – denoted by M_p^{ij} – and the memory variables associated with κ – denoted by K_p^{ij} – are governed by the first-order differential equations

$$\dot{M}_p^{ij} = -\frac{D_{p,\mu}\,\tau_\mu}{\tau_{\sigma p,\mu}}\dot{\varepsilon}_{ij} - \frac{1}{\tau_{\sigma p,\mu}}M_p^{ij} \tag{5.10}$$

$$\dot{K}_p^{ij} = -\frac{D_{p,\kappa}\,\tau_\kappa}{\tau_{\sigma p,\kappa}}\dot{\varepsilon}_{ij} - \frac{1}{\tau_{\sigma p,\kappa}}K_p^{ij} \tag{5.11}$$

The above description of anelasticity is computationally inexpensive compared to the spatial discretisation of the equations of motion which account for the largest portion of the computational costs.

5.2 Q and phase velocity dispersion

In seismology there has traditionally been more emphasis on the quality factor Q than on particular stress or strain relaxation functions. The definition of Q is based on the definition of the complex modulus

$$\mathscr{C}(\nu) := \mathbf{i}\,\nu\int_0^\infty C(t)\,e^{-\mathbf{i}\nu t}\,dt\,, \tag{5.12}$$

with $\nu := \omega + \mathbf{i}\gamma$ and the damping factor $\gamma < 0$. Then

$$Q(\omega) := \frac{\mathfrak{Re}\,\mathscr{C}(\omega)}{\mathfrak{Im}\,\mathscr{C}(\omega)}\,. \tag{5.13}$$

For our stress relaxation function defined in equation (5.4) we find

$$\mathscr{C}(\omega) = C_r + \mathbf{i}\omega C_r \tau \sum_{p=1}^{N} \frac{D_p}{\mathbf{i}\omega + \tau_{\sigma p}^{-1}} = A(\omega) + \mathbf{i}B(\omega), \tag{5.14}$$

with the real and imaginary parts

$$A(\omega) = C_r \left[1 + \tau \sum_{p=1}^{N} \frac{D_p \omega^2 \tau_{\sigma p}^2}{1 + \omega^2 \tau_{\sigma p}^2} \right], \quad B(\omega) = C_r \tau \sum_{p=1}^{N} \frac{D_p \omega \tau_{\sigma p}}{1 + \omega^2 \tau_{\sigma p}^2}. \tag{5.15}$$

Therefore, Q is explicitly given by

$$Q(\omega) = \frac{\left(1 + \sum_{p=1}^{N} \frac{D_p \omega^2 \tau_{\sigma p}^2 \tau}{1 + \omega^2 \tau_{\sigma p}^2} \right)}{\sum_{p=1}^{N} \left(\frac{D_p \omega \tau_{\sigma p} \tau}{1 + \omega^2 \tau_{\sigma p}^2} \right)}. \tag{5.16}$$

Visco-elastic dissipation implies phase velocity dispersion, where the phase velocity v is defined by

$$v = \frac{\omega}{\mathfrak{Re}\, k}. \tag{5.17}$$

The wave number is determined by the dispersion relation

$$k^2 = \frac{\omega^2 \rho}{\mathscr{C}}, \tag{5.18}$$

which follows directly from the one-dimensional wave equation with a spatially invariable C. After some algebra, we find

$$(\mathfrak{Re}\, k)^2 = \frac{1}{2} \rho \omega^2 \frac{A + \sqrt{A^2 + B^2}}{A^2 + B^2}, \tag{5.19}$$

and

$$v^2 = \frac{2 (A^2 + B^2)}{\rho (A + \sqrt{A^2 + B^2})}. \tag{5.20}$$

5.3 Q and the loss of elastic energy

The quality factor Q and its most common interpretation is closely related to the loss of elastic energy, that finds its most prominent expression in the reduction of the displacement amplitude, as the wave progresses in time and space. To see this, we first write the solution of the scalar wave equation in terms of the real and imaginary parts of the complex wave number $k = k_r + \mathbf{i} k_i$:

$$u(x,t) = u_0 e^{\mathbf{i}(k_r x - \omega t)} e^{-k_i x}. \tag{5.21}$$

Equation (5.21) reveals that k_i describes the amplitde reduction with increasing propagation distance. Let us now determine k_i explicitly, first in terms of A and B. For k_i we find

$$k_i^2 = \frac{1}{2} \frac{\rho \omega^2}{A^2 + B^2} \left(\sqrt{A^2 + B^2} - A \right). \tag{5.22}$$

Working under the assumption that Q is large, i.e. B is small, we find the first-order expression

$$k_i^2 \approx \frac{1}{4} \frac{\rho \omega^2 A Q^{-1}}{A^2 + B^2}. \tag{5.23}$$

Furthermore, for large Q, we have $v^2 \approx A/\rho$, and equation (5.23) simplifies to

$$k_i = \frac{\omega}{2vQ}\,, \tag{5.24}$$

where we replaced $\approx$ by $=$ for clarity in the following expressions. Based on equation (5.24) we find that the amplitude reduction is proportional to $\exp\left(-\frac{\omega x}{2vQ}\right)$, and that the elastic energy reduction is proportional to $\exp\left(-\frac{\omega x}{vQ}\right)$. Thus, by progressing from position x to position $x+\lambda$, where λ is the wavelength, the fractional energy loss is, correct to first order, given by

$$\frac{E(x+\lambda)-E(x)}{E(x)} = -\frac{2\pi}{Q}\,. \tag{5.25}$$

Equation (5.25) can be interpreted as the definition of the *spatial* Q. Alternatively, we may substitute $x = vt$, and then ask how the energy is reduced during one temporal osciallation cycle, i.e., as we progress from time t to time $t+T$, where $T = \lambda v^{-1}$ is the period. Then we find, again correct to first order,

$$\frac{E(t+T)-E(t)}{E(t)} = -\frac{2\pi}{Q}\,. \tag{5.26}$$

This defines the *temporal* Q.

5.4 Constructing constant-Q models

5.4.1 Continuous case

We are interested in constructing Q models that are approximately independent of frequency for circular frequencies ω that are within the absorption band

$$\tau_1 \ll \omega^{-1} \ll \tau_2\,, \tag{5.27}$$

where τ_1 and τ_2 are stress-relaxation times that define the boundaries of the absorption band. Constructing such constant-Q models means to find a set of stress-relaxation times $\tau_{\sigma p}$ and weights D_p such that $Q(\omega) \approx \text{const}$ for ω satisfying (5.27). This is a difficult undertaking, which requires a non-linear optimisation of equation (5.2).

To facilitate our analysis, we make a detour via the case of a continuous distribution of standard-linear solids, i.e. we generalise equation (5.2) to

$$C(t) = C_r\left[1+\tau\int_0^\infty D(\tau_\sigma)\,e^{-t/\tau_\sigma}\,d\tau_\sigma\right]H(t)\,. \tag{5.28}$$

The complex modulus is then

$$\mathscr{C}(\omega) = A(\omega) + \mathbf{i}B(\omega)\,, \tag{5.29}$$

with

$$A(\omega) = C_r\left[1+\tau\int_0^\infty \frac{\omega^2\tau_\sigma^2}{1+\omega^2\tau_\sigma^2}\,D(\tau_\sigma)\,d\tau_\sigma\right], \quad B(\omega) = C_r\left[\tau\int_0^\infty \frac{\omega\tau_\sigma}{1+\omega^2\tau_\sigma^2}D(\tau_\sigma)\,d\tau_\sigma\right]. \tag{5.30}$$

Choosing

$$D(\tau_\sigma) = \begin{cases} \tau_\sigma^{-1}, & \text{for} \quad \tau_1 \le \tau_\sigma \le \tau_2 \\ 0, & \text{otherwise} \end{cases},$$

we obtain

$$A(\omega) = C_r \left[1 + \frac{1}{2} \tau \ln \left(\frac{1+\omega^2\tau_2^2}{1+\omega^2\tau_1^2} \right) \right], \tag{5.31}$$

$$B(\omega) = C_r \tau \left[\arctan(\omega\tau_2) - \arctan(\omega\tau_1) \right]. \tag{5.32}$$

For ω well within the absorption band, we obtain the useful approximations

$$Q^{-1} \approx \frac{1}{2} \tau\pi = \text{const.}, \tag{5.33}$$

and

$$v \approx \sqrt{\frac{C_r}{\rho}} \left[1 + \frac{1}{\pi Q} \ln(\omega\tau_2) \right]. \tag{5.34}$$

As desired, Q is approximately constant. Furthermore, the phase velocity increases with increasing frequency. This result implies that the weights of the relaxation mechanisms in the discrete case, D_p, should be proportional to $\tau_{\sigma p}^{-1}$. The parameter τ controls the numerical value of Q, and the relaxation times $\tau_{\sigma p}$ control the width of the absorption band. An example is shown in figure 5.1.

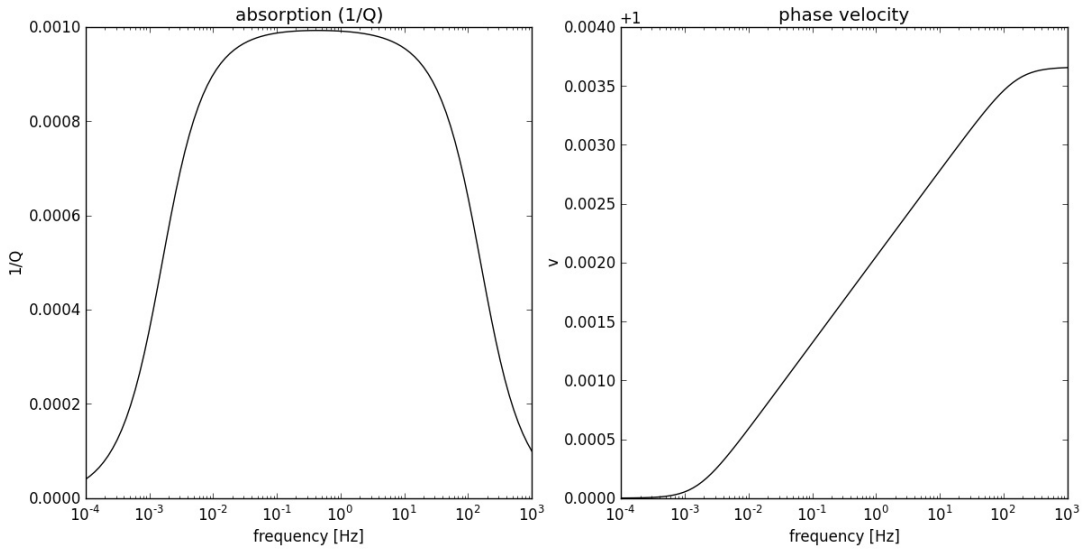

Fig. 5.1 Example of a continuous absorption-band model with $Q = 1000$, $\tau_1 = 1 \cdot 10^{-3}$ s and $\tau_2 = 1 \cdot 10^2$ s. The frequency-dependence of Q^{-1} and the phase velocity v are shown. This plot was produced using the Python tool `Q_continuous.py` in the `TOOLS` directory.

5.4.2 Discrete case

In the case of a discrete superposition of a finite number of relaxation mechanisms, the previous analysis is not sufficiently accurate because the frequency range considered is usually rather narrow. Therefore, we only take a

few relaxation mechanisms (typically 2 to 4) with relaxation times $\tau_{\sigma p}$ and assign initial weights as $D_p = \tau_{\sigma p}$. Using a non-linear optimisation we then find optimal $\tau_{\sigma p}$ and D_p such that

$$\sum_{p=1}^{N} \frac{D_p \omega \tau_{\sigma p}}{1 + \omega^2 \tau_{\sigma p}^2} = \frac{\pi}{2}\,, \tag{5.35}$$

within the desired frequency band. For large Q, the relation between Q and τ is then the same as for the continuous case:

$$Q^{-1} \approx \frac{1}{2}\,\tau\pi\,. \tag{5.36}$$

Figure 5.2 shows an example.

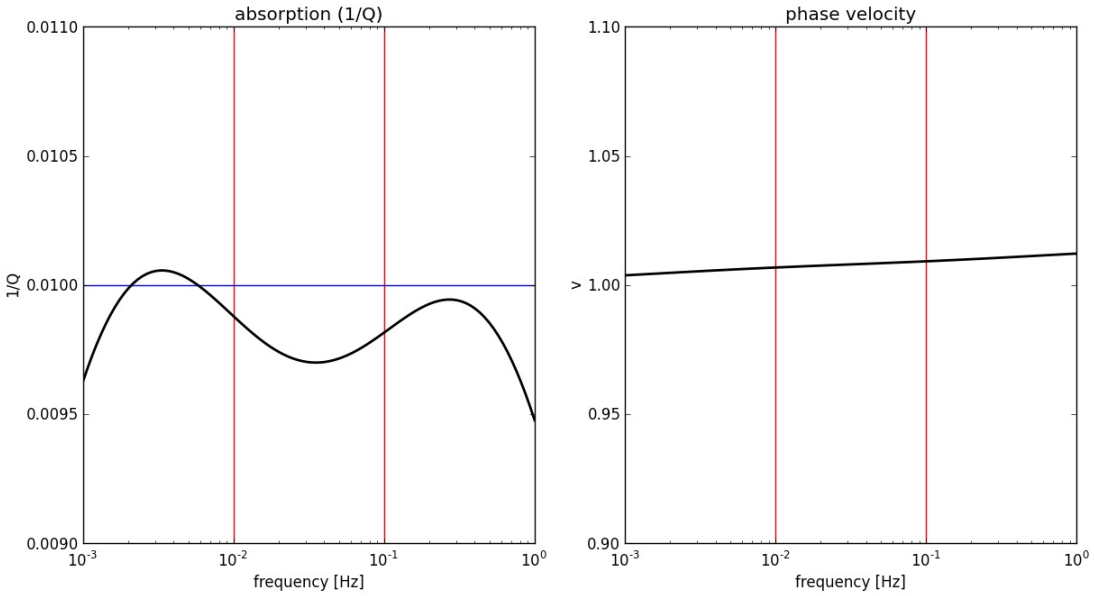

Fig. 5.2 Example of a discrete absorption-band model with a constant target $Q = 100$ in the frequency range $1 \cdot 10^{-3}$-$1 \cdot 10^{-2}$ Hz, which is indicated by red vertical lines. The number of relaxation mechanisms is $N = 3$. This plot was produced using the Python tool `Q_discrete.py` in the `TOOLS` directory.

Chapter 6
Absorbing boundaries

Restricting the considered spatial domain to only a part of the true physical domain in the interest of computational efficiency, introduces unrealistic boundaries. If not treated adequately, the reflections from the artificial boundaries dominate the numerical error. The most widely used solutions for this problem fall into two categories: *absorbing boundary conditions* and *absorbing boundary layers*.
Absorbing boundary conditions are usually based on paraxial approximations of the wave equation. Early applications of this technique to finite-difference modelling can be found in Clayton & Engquist (1977) or in Stacey (1988). Along the artificial boundary the wave equation is replaced by one of its paraxial approximations of order n, typically not higher than one or two. The reflection coefficient then behaves as $(\sin\phi)^n$, where ϕ is the angle of incidence. Absorbing boundary conditions therefore become inefficient for large angles of incidence in general and for surface waves in particular. Moreover, they suffer from numerical stability problems.
Absorbing boundary layers are regions near the unphysical boundaries where the wave field is artificially attenuated in order to prevent reflections. Cerjan et al. (1985) proposed to multiply the wave field in each time step with a Gaussian damping function. While this technique proves to be efficient for finite-difference methods, it leads to unacceptably large boundary layers when high-order methods such as the spectral-element method are used. Robertsson et al. (1994) suggested that the absorption could be improved through physical dissipation, i.e., a very low Q inside the boundary layers. This, however, leads to reflections from the boundary layer itself because low Q values effectively change the elastic properties of the medium.
A comparatively efficient boundary layer technique was introduced by Bérenger (1994), who proposed to modify the electrodynamic wave equation inside a *perfectly matched layer* (PML) such that the solutions decay exponentially with distance, without producing reflections from the boundary between the regular medium and the PML. Though originally designed for first-order systems of differential equations - such as the elastodynamic equations in stress-velocity formulation (see e.g. Collino & Tsogka, 2001 or Festa & Vilotte, 2005) - it has been shown that the method can be extended to second-order systems (Komatitsch & Tromp, 2003). The classical PML approach leads to a new and generally larger system of differential equations. Therefore, existing codes have to be substantially modified. This complication can be avoided by using *anisotropic perfectly matched layers* (APML), studied for example by Teixeira & Chew (1997) and Zheng & Huang (1997). `SES3D` implements the APML, which are described in the following paragraphs.

The elastic wave equation in the frequency domain is given by

$$\mathbf{i}\omega\rho\mathbf{v}(\mathbf{x},\omega) = \nabla\cdot\boldsymbol{\sigma}(\mathbf{x},\omega) + \mathbf{f}(\mathbf{x},\omega)\,, \tag{6.1a}$$

$$\mathbf{i}\omega\boldsymbol{\sigma}(\mathbf{x},\omega) = \Xi(\mathbf{x},\omega) : \nabla\mathbf{v}(\mathbf{x},\omega)\,, \tag{6.1b}$$

$$\mathbf{i}\omega\mathbf{u}(\mathbf{x},\omega) = \mathbf{v}(\mathbf{x},\omega)\,. \tag{6.1c}$$

We introduced the symbol Ξ for the rate of relaxation tensor $\dot{\mathbf{C}}$ in equations (6.1). In a homogeneous and isotropic medium and when external forces are absent, it is known to have plane wave solutions of the form

$$\mathbf{u}(\mathbf{x},\omega) = \mathbf{A}\,e^{-\mathbf{i}\mathbf{k}(\omega)\cdot\mathbf{x}}\,, \tag{6.2}$$

with either

$$\mathbf{A} \,||\, \mathbf{k}(\omega) \quad \text{and} \quad ||\mathbf{k}(\omega)||^2 = \frac{\rho\omega^2}{\lambda+2\mu} = \frac{\omega^2}{v_{\mathrm{P}}^2} \qquad \text{(P-wave)} \quad \text{or} \tag{6.3a}$$

$$\mathbf{A} \perp \mathbf{k}(\omega) \quad \text{and} \quad ||\mathbf{k}(\omega)||^2 = \frac{\rho\omega^2}{\mu} = \frac{\omega^2}{v_{\mathrm{S}}^2} \qquad \text{(S-wave)}. \tag{6.3b}$$

Our goal is to modify (6.1) such that it allows for plane wave solutions that decay exponentially with distance, say for example, in increasing z direction. For this, the cartesian coordinate z is replaced by a new variable $\tilde{z} := \tilde{z}(\mathbf{x}, \gamma_z(\mathbf{x}))$, such that the resulting solutions are

$$\mathbf{u}(\mathbf{x},\omega) = \mathbf{A}\, e^{-\mathbf{i}\mathbf{k}(\omega)\cdot\mathbf{x}}\, e^{-f(\gamma_z)z}, \tag{6.4}$$

for $z > 0$ and some function f that depends on the particular choice of γ_z. Choosing $f(\gamma_z)|_{z<0} = 0$ will leave the solution in $z < 0$ unchanged. We use the coordinate transformation

$$\tilde{z} := \int_0^z \gamma_z(x,y,z')dz'. \tag{6.5}$$

Outside the perfectly matched layer, that is in our case for $z < 0$, we require $\tilde{z} = z$ or equivalently $\gamma_z(\mathbf{x}) = 1$. To obtain exponentially decaying plane waves also in the remaining coordinate directions, the transformations $x \to \tilde{x}$ and $y \to \tilde{y}$ are defined analogously. Replacing the derivatives ∂_x, ∂_y and ∂_z in (6.1a) by the derivatives $\partial_{\tilde{x}}$, $\partial_{\tilde{y}}$ and $\partial_{\tilde{z}}$, gives the following set of equations,

$$\mathbf{i}\omega\rho v_x = \gamma_x^{-1}\partial_x\sigma_{xx} + \gamma_y^{-1}\partial_y\sigma_{yx} + \gamma_z^{-1}\partial_z\sigma_{zx}, \tag{6.6a}$$

$$\mathbf{i}\omega\rho v_y = \gamma_x^{-1}\partial_x\sigma_{xy} + \gamma_y^{-1}\partial_y\sigma_{yy} + \gamma_z^{-1}\partial_z\sigma_{zy}, \tag{6.6b}$$

$$\mathbf{i}\omega\rho v_z = \gamma_x^{-1}\partial_x\sigma_{xz} + \gamma_y^{-1}\partial_y\sigma_{yz} + \gamma_z^{-1}\partial_z\sigma_{zz}. \tag{6.6c}$$

Note that the divergence in equation (6.1a) is interpreted as a left-divergence in equations (6.6). Multiplication by $\gamma_x\gamma_y\gamma_z$ yields

$$\mathbf{i}\omega\gamma_x\gamma_y\gamma_z v_x = \gamma_y\gamma_z\partial_x\sigma_{xx} + \gamma_x\gamma_z\partial_y\sigma_{yx} + \gamma_x\gamma_y\partial_z\sigma_{zx}, \tag{6.7a}$$

$$\mathbf{i}\omega\gamma_x\gamma_y\gamma_z v_y = \gamma_y\gamma_z\partial_x\sigma_{xy} + \gamma_x\gamma_z\partial_y\sigma_{yy} + \gamma_x\gamma_y\partial_z\sigma_{zy}, \tag{6.7b}$$

$$\mathbf{i}\omega\gamma_x\gamma_y\gamma_z v_z = \gamma_y\gamma_z\partial_x\sigma_{xz} + \gamma_x\gamma_z\partial_y\sigma_{yz} + \gamma_x\gamma_y\partial_z\sigma_{zz}. \tag{6.7c}$$

The products $\gamma_i\gamma_j$ on the left-hand sides of (6.7) can be placed under the differentiation if γ_x depends only on x, γ_y only on y and γ_z only on z. We may then write the new version of (6.1a) as

$$\mathbf{i}\omega\gamma_x\gamma_y\gamma_z\mathbf{v} = \nabla\cdot\sigma^{\mathrm{pml}}, \tag{6.8}$$

with the definition

$$\sigma^{\mathrm{pml}} := \gamma_x\gamma_y\gamma_z \begin{pmatrix} \gamma_x^{-1} & 0 & 0 \\ 0 & \gamma_y^{-1} & 0 \\ 0 & 0 & \gamma_z^{-1} \end{pmatrix} \cdot\sigma = \gamma_x\gamma_y\gamma_z\Lambda\cdot\sigma. \tag{6.9}$$

Since (6.1a) and (6.8) are very similar, the anisotropic perfectly matched layers can be implemented with relative ease by slightly modifying pre-existing codes. It is noteworthy at this point that σ^{pml} is in general not symmetric.

In the next step we consider the constitutive relation. Based on (6.1b) and (6.1c) we find

$$\sigma_{ij}^{\mathrm{pml}} = \gamma_x\gamma_y\gamma_z\gamma_i^{-1}\sigma_{ij} = \gamma_x\gamma_y\gamma_z\gamma_i^{-1}\sum_{k,l=1}^{3}\Xi_{ijkl}\partial_{x_k}u_l. \tag{6.10}$$

Again replacing ∂_x, ∂_y and ∂_z by $\partial_{\tilde{x}}$, $\partial_{\tilde{y}}$ and $\partial_{\tilde{z}}$, yields

$$\sigma_{ij}^{\text{pml}} = \gamma_x\gamma_y\gamma_z\gamma_i^{-1}\sum_{k,l=1}^{3}\Xi_{ijkl}\gamma_k^{-1}\partial_{x_k}u_l\,. \tag{6.11}$$

For convenience we define a new strain tensor ε^{pml} as

$$\varepsilon_{kl}^{\text{pml}} := \gamma_x\gamma_y\gamma_z\gamma_k^{-1}\partial_{x_k}u_l\,, \tag{6.12}$$

which finally leads to our modified version of the wave equation:

$$\mathbf{i}\omega\rho\gamma_x\gamma_y\gamma_z v_i = \sum_{j=1}^{3}\partial_{x_j}\sigma_{ji}^{\text{pml}}\,, \tag{6.13a}$$

$$\gamma_i\sigma_{ij}^{\text{pml}} = \sum_{k,l=1}^{3}\Xi_{ijkl}\varepsilon_{kl}^{\text{pml}}\,, \tag{6.13b}$$

$$\varepsilon_{kl}^{\text{pml}} = \gamma_x\gamma_y\gamma_z\gamma_k^{-1}\partial_{x_k}u_l\,. \tag{6.13c}$$

The source term in (6.13) is omitted because the PML region will usually be chosen such that the sources are inside the regular medium and not inside the PML.

To demonstrate that exponentially decaying plane waves are indeed solutions of (6.13) we consider a homogeneous and isotropic medium. For simplicity, we restrict the analysis to the case $\gamma_x = \gamma_y = 1$, $\gamma_z = const \neq 1$. Assuming a plane wave solution of the form

$$\mathbf{u}(\mathbf{x},\omega) = \mathbf{A}\,e^{-\mathbf{i}(k_x x + k_z z)}\,, \tag{6.14}$$

the problem reduces to finding the dispersion relation $\mathbf{k} = \mathbf{k}(\omega)$. The Christoffel equation resulting from the combination of (6.13) with (6.14) is

$$\begin{pmatrix} v_{\text{P}}^2k_x^2 + v_{\text{S}}^2k_z^2\gamma_z^{-2} & 0 & (v_{\text{P}}^2 - v_{\text{S}}^2)(k_xk_z\gamma_z^{-1}) \\ 0 & v_{\text{S}}^2(k_x^2 + k_z^2\gamma_z^{-2}) & 0 \\ (v_{\text{P}}^2 - v_{\text{S}}^2)(k_xk_z\gamma_z^{-1}) & 0 & v_{\text{P}}^2k_z^2\gamma_z^{-2} + v_{\text{S}}^2k_x^2 \end{pmatrix}\begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix} = \omega^2\begin{pmatrix} A_x \\ A_y \\ A_z \end{pmatrix}. \tag{6.15}$$

Non-trivial solutions exist only if either

$$k_x^2 + k_z^2\gamma_z^{-2} = \omega^2 v_{\text{P}}^{-2} \tag{6.16a}$$

or

$$k_x^2 + k_z^2\gamma_z^{-2} = \omega^2 v_{\text{S}}^{-2} \tag{6.16b}$$

are satisfied. Equations (6.16a) and (6.16b) are the dispersion relations inside the PML. The resulting plane wave solutions are

$$\mathbf{u}_s(\mathbf{x},\omega) = \mathbf{A}_s\,e^{-\mathbf{i}\omega(x\sin\phi + \gamma_z z\cos\phi)/v_{\text{S}}}\,,\quad \mathbf{A}_s \perp (k_x, 0, k_z\gamma_z^{-1})^T\,, \tag{6.17a}$$

for the S wave and

$$\mathbf{u}_p(\mathbf{x},\omega) = \mathbf{A}_p\,e^{-\mathbf{i}\omega(x\sin\phi + \gamma_z z\cos\phi)/v_{\text{P}}}\,,\quad \mathbf{A}_p \,||\, (k_x, 0, k_z\gamma_z^{-1})^T\,, \tag{6.17b}$$

for the P wave. The variable ϕ denotes the angle of incidence. One may now define the coordinate stretching variable γ_z. There are in principle no restrictions other than $\gamma_z = 1$ outside the PML. We use

$$\gamma_z(\mathbf{x}) := 1 + \frac{a_z}{\mathbf{i}\omega}\,. \tag{6.18}$$

Inserting (6.18) into (6.17) yields

$$\mathbf{u}_{s/p}(\mathbf{x},\omega) = \mathbf{A}_{s/p}\, e^{-\mathbf{i}\omega(x\sin\phi + z\cos\phi)/v_{\mathrm{S/P}}}\, e^{-(a_z z\cos\phi)/v_{\mathrm{S/P}}}\,. \tag{6.19}$$

The exponential decay is therefore frequency-independent in the case of incident body waves. A frequency-dependent decay can be expected for dispersive waves such as surface waves. However, waves with an angle of incidence, ϕ, close to $\pm\pi/2$ will decay slower than waves with an angle of incidence close to 0. It should be noted that the example of a homogeneous and isotropic medium with constant γ_z is oversimplified. Since analytical solutions for more complex models are usually unavailable, the efficiency of the APML technique must be tested numerically.

With the exception of the corners of the model[1], the damping regions at the x, y and z boundaries do not overlap. We therefore find

$$a_i a_j = a_i^2\,\delta_{ij} \tag{6.20}$$

and

$$\gamma_x\gamma_y\gamma_z = \frac{\mathbf{i}\omega + a_x + a_y + a_z}{\mathbf{i}\omega}\,. \tag{6.21}$$

The resulting equations of motion in the frequency domain are now

$$\mathbf{i}\omega\rho\,(\mathbf{i}\omega + a_x + a_y + a_z)\,u_i = \sum_{j=1}^{3} \partial_{x_j}\sigma_{ji}^{\mathrm{pml}}\,, \tag{6.22a}$$

$$(\mathbf{i}\omega + a_i)\,\sigma_{ij}^{\mathrm{pml}} = \sum_{k,l=1}^{3} \mathbf{i}\omega\,\Xi_{ijkl}\,\varepsilon_{kl}^{\mathrm{pml}}\,, \tag{6.22b}$$

$$\mathbf{i}\omega\,\varepsilon_{kl}^{\mathrm{pml}} = (\mathbf{i}\omega + a_x + a_y + a_z - a_k)\,\partial_{x_k} u_l\,. \tag{6.22c}$$

Transforming into the time domain, finally yields

$$\rho\,\partial_t^2 u_i + \rho\,(a_x + a_y + a_z)\partial_t u_i = \sum_{j=1}^{3} \partial_{x_j}\sigma_{ji}^{\mathrm{pml}}\,, \tag{6.23a}$$

$$\partial_t\sigma_{ij}^{\mathrm{pml}} + a_i\sigma_{ij}^{\mathrm{pml}} = \sum_{k,l=1}^{3} \Xi_{ijkl} * \partial_t\varepsilon_{kl}^{\mathrm{pml}}\,, \tag{6.23b}$$

$$\partial_t\varepsilon_{kl}^{\mathrm{pml}} = \partial_t\partial_{x_k} u_l + (a_x + a_y + a_z - a_k)\,\partial_{x_k} u_l\,, \tag{6.23c}$$

or in tensor notation:

$$\rho\,\partial_t^2\mathbf{u} + \rho\,\mathrm{tr}(\mathbf{a})\,\partial_t\mathbf{u} = \nabla\cdot\sigma^{\mathrm{pml}}\,, \tag{6.24a}$$

$$\partial_t\sigma^{\mathrm{pml}} + \mathbf{a}\cdot\sigma^{\mathrm{pml}} = \int_{-\infty}^{t} \Xi(t-\tau) : \partial_t\varepsilon^{\mathrm{pml}}(\tau)\,d\tau\,, \tag{6.24b}$$

$$\partial_t\varepsilon^{\mathrm{pml}} = \partial_t\nabla\mathbf{u} + (\mathbf{I}\,\mathrm{tr}(\mathbf{a}) - \mathbf{a})\cdot\nabla\mathbf{u}\,, \tag{6.24c}$$

where $\mathbf{a}$ is defined as

$$\mathbf{a} := \begin{pmatrix} a_x & 0 & 0 \\ 0 & a_y & 0 \\ 0 & 0 & a_z \end{pmatrix}. \tag{6.25}$$

The gradients in equation (6.24c) are left-gradients. The choice of γ_i as inversely proportional to ω leads to relatively simple differential equations that can be marched forward in time explicitly. Different definitions of

[1] There is, to the best of my knowledge, no PML variant where the corners are treated adequately. In the corners two or more PMLs overlap, and it is not immediately clear what the consequences are. In fact, numerical experiments show that the instability tends to start in the corners. This suggests that PML methods might be improved substantially through the construction of absorbing corners where elastic waves do indeed decay.

γ_i generally result in temporal convolutions and therefore lead to integro-differential equations that are more difficult to solve, at least in the time domain.

Cited literature and further reading material

1. Aki, K. & Richards, P. G., 2002. *Quantitative Seismology, 2nd edition*, University Science Books.
2. Babuska, V., Cara, M., 1991. *Seismic anisotropy in the Earth*, Kluwer Acad. Pub.
3. Bérenger, J.-P., 1994. *A perfectly matched layer for the absorption of electromagnetic waves*, J. Comp. Phys., **114**, 185-200.
4. Blanch, J. O., Robertsson, J. O. A., Symes, W. W., 1995. *Modelling of a constant Q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique*, Geophysics, **60**(1), 176-184.
5. Capdeville, Y., Marigo, J.-J., 2007. Second order homogenization of the elastic wave equation for non-periodic layered media, *Geophys. J. Int.*, **170**, 823-838.
6. Capdeville, Y., Marigo, J.-J., 2008. *Shallow layer correction for spectral element like methods*, Geophys. J. Int., **172**, 1135-1150.
7. Cerjan, C., Kosloff, D., Kosloff, R., Reshef, M., 1985. *A non reflecting boundary condition for discrete acoustic and elastic wave calculations*, Geophysics, **50**, 705-708.
8. Clayton, R. & Engquist, B., 1977. *Absorbing boundary conditions for acoustic and elastic wave equations*, Bull. Seis. Soc. Am., **67**(6), 1529-1540.
9. Collino, F. & Tsogka, C., 2001. *Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media*, Geophysics, **66**(1), 294-307.
10. Dahlen, F. A., Tromp, J., 1998. *Theoretical global seismology*, Princeton University Press.
11. Durek, J. J. & Ekstroöm, G., 1996. *A radial model of anelasticity consistent with long-period surface-wave attenuation*, Bull. Seis. Soc. Am., **86**, 144-158.
12. Dziewonski, A. M., Anderson, D. L., 1981. *Preliminary reference Earth model*, Phys. Earth Planet. Int., **25**, 297-356.
13. Festa, G. & Vilotte, J.-P., 2005. *The Newmark scheme as velocity-stress time-staggering: an efficient PML implementation for spectral element simulations for elastodynamics*, Geophys. J. Int, **161**, 789-812.
14. Fichtner, A., Igel, H., 2008. *Efficient numerical surface wave propagation through the optimization of discrete crustal models - a technique based on non-linear dispersion curve matching (DCM)*, Geophys. J. Int., **173**(2), 519-533.
15. Fichtner, A., Saygin, E., Taymaz, T., Cupillard, P., Capdeville, Y., Trampert, J., 2013a. *The deep structure of the North Anatolian Fault zone*, Earth Planet. Sci. Lett., **373**, 109-117.
16. Fichtner, A., Trampert, J., Cupillard, P., Saygin, E., Taymaz, T., Capdeville, Y., Villasenor, A., 2013b. *Multi-scale full waveform inversion*, Geophys. J. Int., **194**, 534-556.
17. Kennett, B. L. N., Engdahl, E. R., Buland, R., 1995. *Constraints on seismic velocities in the Earth from traveltimes*, Geophys. J. Int., 122, 108-124.
18. Kennett, B. L. N., 2001. *The seismic wavefield: Volume 1. Introduction and theoretical development*, Cambridge University Press.
19. Komatitsch, D. & Tromp, J., 2003. *A perfectly matched layer absorbing boundary condition for the second-order seismic wave equation*, Geophys. J. Int., **154**, 146-153.
20. Love, A. E. H., 1892. *The mathematical theory of elasticity*, Cambridge Univ. Press.
21. Moczo, P. & Kristek, J., 2005. *On the rheological methods used for the time-domain methods of seismic wave propagation*, Geophys. Res. Lett., **32**(1), Art. No. L01306.
22. Ritsema, J., van Heijst, H. J., Deuss, A., Woodhouse, J. H., 2010. *S40RTS: a degree-40 shear-velocity model of the mantle from new Rayleigh wave dispersion, teleseismic traveltimes, and normal-mode splitting function measurements*, Geophys. J. Int., **184**, doi:10.1111/j.1365246X.2010.04884.x.
23. Robertsson, J. O. A., Blanch, J. O., Symes, W. W., 1994. *Viscoelastic finite-difference modeling*, Geophysics, **59**(9), 1444-1456.
24. Stacey, R., 1988. *Improved transparent boundary formulations for the elastic-wave equation*, Bull. Seis. Soc. Am., **78**(6), 2089-2097.
25. Takeuchi, H., Saito, M., 1972. *Seismic surface waves*, in *Methods in Computational Physics, Vol. 11*, 217-295, Academic Press.
26. Teixeira, F. L. & Chew, W. C., 1997. *Systematic derivation of anisotropic pml absorbing media in cylindrical and spherical coordinates*, IEEE Microwave and Guided Wave Letters, **7**(11), 371-373.
27. Zheng, Y. & Huang, X., 2002. *Anisotropic perfectly matched layers for elastic waves in cartesian and curvilinear coordinates*, MIT Earth Resources Laboratory, Consortium Report.