



**- SOFI -**  
**SEISMIC MODELING WITH FINITE**  
**DIFFERENCES**  
**2D (VISCO-)ELASTIC ANISOTROPIC VERSION**

**Users guide**

Department of Physics, Geophysical Institute (GPI)  
Hertzstraße 16, 76187 Karlsruhe, Germany

**Authors**

Thomas Bohlen  
Denise De Nil  
Daniel Köhn  
Stefan Jetschny  
Thomas Hertweck  
Lars Hout  
Sonia Şortan

Karlsruhe, Jan. 2023

# Contents

<b>1</b>	<b>Getting started</b>	<b>2</b>
1.1	Requirements . . . . .	2
1.2	Directory structure . . . . .	3
1.3	Installation . . . . .	3
1.4	First tests . . . . .	5
1.5	License . . . . .	6
<b>2</b>	<b>Introduction</b>	<b>6</b>
<b>3</b>	<b>Theory</b>	<b>7</b>
3.1	Equation of motion for an elastic medium . . . . .	7
3.2	Equation of motion for a viscoelastic medium . . . . .	8
3.3	Equation of motion for an anisotropic medium . . . . .	11
3.3.1	Seismic anisotropy and elasticity tensor . . . . .	11
3.3.2	Wave equations . . . . .	14
3.3.3	Numerical results . . . . .	16
3.4	Solution of the wave equation by finite differences . . . . .	16
3.4.1	Standard staggered grid . . . . .	17
3.4.2	Free surface and interfaces on a SSG . . . . .	18
3.4.3	Discretization of the wave equation . . . . .	20
3.4.4	Accuracy of FD operators . . . . .	20
3.4.5	Adams-Bashforth fourth-order accurate time integrators . . . . .	21
3.4.6	Absorbing boundary conditions . . . . .	23
3.5	Numerical artifacts and instabilities . . . . .	25
3.5.1	Grid dispersion . . . . .	25
3.5.2	The Courant instability . . . . .	26
<b>4</b>	<b>Definition of the modeling parameters</b>	<b>28</b>
4.1	Domain decomposition . . . . .	29
4.2	Order of the FD operator . . . . .	30
4.3	Space discretization . . . . .	31
4.4	Time stepping . . . . .	31
4.5	Wave equations . . . . .	31
4.6	Sources . . . . .	32
4.7	Generation of models . . . . .	37
4.8	Q-approximation . . . . .	38
4.9	Boundary conditions . . . . .	39
4.10	Wavefield snapshots . . . . .	40
4.11	Receivers . . . . .	40
4.12	Receiver array . . . . .	42
4.13	Seismograms . . . . .	42
4.14	Monitoring the simulation . . . . .	43
4.15	Postprocessing . . . . .	44
<b>A</b>	<b>Appendix: Parameters used in the code</b>	<b>46</b>

# 1 Getting started

## 1.1 Requirements

SOFI is written in the C programming language using the C99 standard. The software should therefore compile and work on many systems currently in use around the world. As the software is parallelised using the Message Passing Interface (MPI), you obviously need an MPI environment available at compile time and runtime. As only basic MPI functions are used, most MPI implementations should work out of the box; you may have to adapt the start procedure, though, in case your environment requires special settings. Our preferred platform is Linux, as most of the large-scale compute clusters around the world are based on this operating system, including our clusters at the KIT. Furthermore, Linux is our main development platform for the software. As we use openSUSE ourselves, this Linux distribution is perhaps the most widely tested platform for SOFI. However, we have also successfully tested and used SOFI on RedHat Enterprise Linux, SUSE Linux Enterprise Server, CentOS, Ubuntu and others. The following programs or libraries and environments should ideally be installed for running SOFI and working with its input and output data:

- **Make**, preferably GNU Make. We provide a `Makefile` for compilation of the software and documentation. However, you could in principle also compile the software yourself (see instructions in section 1.3), or set up your own build system using, for instance, CMake. The same holds for the documentation.  
⇒ <https://www.gnu.org/software/make/>
- **C compiler** supporting the C99 standard. We tend to use the standard GNU Compiler Collection (GCC) for development, but the software has also been successfully compiled using the Intel C/C++ compiler or Clang, the C-frontend for LLVM.  
⇒ <https://www.gnu.org/software/gcc/>
- **MPI environment**. We typically use OpenMPI (version 3) for our development, but the software has also been successfully compiled using the Intel MPI library. Other MPI libraries like MPICH should also work.  
⇒ <https://www.open-mpi.org/>
- **Seismic Unix (SU)**. While not strictly required to compile or run SOFI, SU is used as preferred output format for seismograms. SU's `xmovie` program is also used, for instance, to display wavefield snapshot movies.  
⇒ <https://wiki.seismic-unix.org/>
- **Matlab**, or its free alternative **GNU Octave**, and/or **Python**. We provide some scripts that help in setting up models or determine optimized visco-elastic parameters. Apart from that, none of these tools is strictly required to compile or run SOFI.  
⇒ <https://octave.org/> or <https://www.python.org/>
- **TeX environment**. The SOFI documentation contains not only basic instructions on how to run the software but also a lot of information on the scientific background. It is therefore written using TeX. In order to compile the documentation, you need a TeX environment and several macro packages. Recent TeX distributions like TeX Live or MikTeX should work just fine, provided you have all the required macro packages available. If you run into problems, check first that all required packages are installed.  
⇒ <https://www.tug.org/texlive/> or <https://miktex.org/>

## 1.2 Directory structure


We assume that you have successfully checked out the git-repository containing the software. Your local SOFI directory will contain the following subdirectories:

- **bin**: An empty directory in which executables will be installed by the build process, unless the default values are changed.
- **build**: A directory including a `Makefile` that can be used to build the software and documentation. We compile the software outside the source tree so you can have as many different build directories and therefore software builds as you like, for instance using different compilers or compiler settings.
- **doc**: A directory containing the source for the documentation.
- **examples**: A directory containing small examples so you can test the software after compilation. We also provide reference results so you can check your build against ours.
- **mfiles**: A directory containing Matlab/Octave support scripts.
- **pyfiles**: A directory containing Python support scripts.
- **src**: A directory containing the actual source code written in C and corresponding header files.
- **util**: A directory containing utility programs or scripts, primarily of interest for developers.

## 1.3 Installation

As mentioned above, we build the software outside the source tree. This has several reasons: Firstly, we keep the source tree clean without object or dependency files. Secondly and probably more importantly, this approach allows us to have several builds in parallel using different compilers and/or compiler settings, which is particularly useful during development.

If you would like to build the software and documentation in our standard `build` directory within the downloaded SOFI repository, just enter this directory. Otherwise, copy the `Makefile` you find in our `build` directory to any place where you would actually like to build the software on your system. If you use this approach, the entire downloaded git repository will be kept 'as is', i.e., in a clean state.

Once you are in the build directory (either the default one or your own one), you can type `$> make` 

to get some help about the `Makefile` targets and the current setup, including the installation directory and compiler settings. If you copied the `Makefile` to a different place and you are working in a build directory outside the downloaded git repository, you need to adjust the `Makefile` and its `BASEDIR` variable which specifies where the SOFI base directory is located. In this case, you might also check on the installation directory (variable `INSTDIR`) and adjust it according to your needs.

By default, the `Makefile` is set up such that it uses the standard `mpicc` command and standard MPI library found on your system. You may want to check the compiler and linker settings in the `Makefile` – the corresponding sections are documented in the `Makefile` and there are also exemplary alternative compiler/linker settings provided. Dependent on your computer platform and architecture, you may want to enable or disable certain features during compilation. By default, we create executables with optimization turned on.

Once you have checked that all settings are fine, simply run

```
$> make install
```

to compile the software and install it in the chosen installation directory. The installation process will automatically back up already existing programs with the same name, i.e., you always have a chance to go back to your previous executables if necessary (backup files have a suffix `.bck` in the installation directory).

If you would like to compile the software only locally in the build directory but not install it in the installation directory, you can simply run

```
$> make all
```

As can be seen, by default the full compilation commands etc. are not shown, only a summary of what `make` is currently doing. If you would like to see the full commands as they are executed by `make`, simply run

```
$> make V=1 install
```

or

```
$> make V=1 all
```

and the whole compilation command etc. is visible. This is particularly useful if the build process fails with an error.

Upon first compilation, hidden dependency files are generated which allow `make` to figure out the programs and object files that need to be regenerated in case certain source code files change.

Our `Makefile` supports parallel build processes. That means, on systems with more than one CPU core you can use, for instance,

```
$> make -j6 install
```

to compile the software in parallel using six processes which will significantly speed up the whole compilation process.

The documentation, provided  $\text{\LaTeX}$  and the required macro packages are available, can be built by

```
$> make doc
```

After a successful  $\text{\LaTeX}$  run, you will find a PDF file called `sofi2D_manual.pdf` in your build directory. Given that you are currently reading this text, you have either already successfully compiled the manual, or downloaded a pre-compiled PDF, or looked at the documentation source code.

You can force a recompilation and installation of all programs using

```
$> make force install
```

and the build directory can be cleaned up using either

```
$> make clean
```

or

```
$> make distclean
```

dependent on whether you would like to remove the object files and (local) executables only, or whether you would like to restore the original state of the build directory (i.e., dependency files and the local documentation are also removed).

At the time of writing, the source code should compile without any warnings with `-Wall` and `-Wextra` flags being set. Once you have successfully installed the software in your chosen installation directory, you should be good to give it a first go.

The `Makefile` is documented. In other words, in case there is a need to modify it, you should be able to understand the setup, variables and different parts of the `Makefile` reasonably well. On most systems there should be no need to make any significant modifications.

## 1.4 First tests

At this point, we assume you have successfully compiled and installed the software. You can now enter SOFI's `examples` directory where you will find scripts to run basic tests and compare your results to our reference results. All tests should run within a few seconds – they are not meant to stress-test the software or your system, their main purpose is to check that you have successfully compiled and installed the software, and that you can successfully launch MPI jobs. Each test will use four CPU cores, which should work fine on any reasonably modern system. The following scripts are provided:

- **run\_single\_test.sh**: A shell script to run a single test.
- **run\_all\_tests.sh**: A shell script to loop through all available individual short tests.
- **compare\_single\_test.sh**: A shell script to compare your results, once successfully created by one of the test runs, with our reference results available in the subdirectories ending in `_ref`. For this comparison to work, you must have a working Seismic Unix installation and the SU executables must be in your `PATH`.
- **clean.sh**: A simple shell script to remove all locally generated test files in the subdirectories. The reference files which are part of the git repository will not be touched.

Each script (apart from `clean.sh`), when called without any options or parameters, outputs some help about its use. By default, the `run_single_test.sh` scripts assume that the software was built using the standard `Makefile` in our build directory, and executables are in the relative path `../bin` as seen from the `examples` directory. If this is not the case (for instance, because you built the software completely outside the downloaded git repository), you may have to adapt the `run_single_test.sh` script.

Attention: It is mandatory that you use the same MPI environment at runtime that you used to build the software. MPI is quite sensitive in terms of version numbers, i.e., you cannot mix and match different MPI environments or versions.

In principle, each SOFI run looks as follows (the example here assumes you would like to run SOFI directly from the commandline):

```
$> mpirun -np N /path/to/sofi2D /path/to/sofi2D.json
```

Basically, you are instructing MPI to run `N` processes of `sofi2D` and use the file `sofi2D.json` as the only argument for the program. The `json` file contains all the parameters that SOFI requires. They are listed in detail in subsequent chapters of this manual.


In practice, you will most likely run SOFI on a compute cluster using a queuing system like, for instance, Slurm (<https://slurm.schedmd.com/>). In this case, you need to set up a shell script with instructions for Slurm and the `mpirun`-command for SOFI, and finally submit the shell script to the queuing manager using a command like `sbatch`. An exemplary, basic script to run SOFI via Slurm is shown below:

```
#!/bin/bash
#SBATCH --ntasks=96
#SBATCH --job-name="SOFI:MPI"
#SBATCH --partition=horeka
mpirun /path/to/sofi2D /path/to/sofi2D.json
```

Please check your local queuing manager documentation for details. Note: You need to make sure that the total number of FD modelling partitions (domain decomposition) matches the number of total MPI processes.

If wavefield snapshots are output, each MPI process will write to a shared directory but the

files are not merged during execution (neither by broadcasting individual results so the main MPI process could combine them nor by MPI I/O), as this could significantly slow down the program. Individual wavefield snapshots are merged after a successful SOFI run using a program called `snapmerge`. In principle, you simply run

```
$> /path/to/snapmerge /path/to/sofi2D.json 
```

in order to merge the files. Obviously, you need to pass the same json file to `snapmerge` that you used for `sofi2D` to create the wavefield snapshots in the first place, otherwise the process of merging will fail.

## 1.5 License

`sofi2D` 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, version 2.0 of the License only.

`sofi2D` 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 `sofi2D`. See file `COPYING` and/or <http://www.gnu.org/licenses/gpl-2.0.html>.

The authors of `sofi2D` are listed in file `AUTHORS`.

## 2 Introduction

In order to determine information about the 2D structure and composition of the subsurface from seismic observations, it is necessary to be able to predict how seismic wavefields are affected by complex structures. Since exact analytical solutions to the wave equations do not exist for arbitrary Earth models, the solutions can only be obtained by numerical methods. Various techniques for seismic wave modeling in realistic (complex) media have been developed over the years. Explicit finite-difference methods have been widely used to model seismic wave propagation in 2D elastic media because of their ability to accurately model seismic waves in arbitrary heterogeneous media. The FD software described here is based on the original work of Virieux (1986) and Levander (1988) who both formulated a staggered-grid, finite-difference scheme based on a system of first-order coupled elastic equations where the variables are stresses and velocities. This distribution of wavefield and material parameters is referred to as the standard staggered grid (SSG). Robertsson, J. O. Blanch, et al. (1994) extended the elastic SSG algorithms of Virieux and Levander to the viscoelastic case. To incorporate absorption he applied the “generalized standard linear solid” (GSLs) rheological model which was first proposed by Emmerich and Korn (1987).

Not only accurate seismic modeling of attenuation is essential for the understanding of wave propagation in the subsurface, but also the modeling of seismic anisotropy. Most real-world rocks exhibit a directional dependence of both wave velocities and attenuation. Therefore, in the past decades, anisotropy of seismic velocities and attenuation have been incorporated into seismic modeling and imaging practice (e.g., Carcione et al., 1988; Komatitsch and Tromp, 1999; Thomsen, 1986; Tsvankin, 2012; Bai and Tsvankin, 2016).

The main drawback of the FD method is that modeling of realistic models consumes vast quantities of computational resources. Such computational requirements are generally beyond the resources available on a single PC or workstation. Over the years, it has become fairly common to use clusters of compute nodes for scientific computing. FD modeling can benefit from this technique by using the Message Passing Interface (MPI) (Bohlen, 2002).

Using the free and portable MPI, the calculations are distributed on compute nodes which are connected by, ideally, a fast network. By using clusters of processors, wall-clock runtimes can be decreased and possible grid sizes (and therefore memory usage) can be increased significantly.

### 3 Theory

#### 3.1 Equation of motion for an elastic medium

The propagation of waves in a general elastic medium can be described by a system of coupled linear partial differential equations. They consist of the **equations of motion**

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i , \quad (1)$$

which simply states that the momentum of the medium, the product of density  $\rho$  and the derivative of displacement velocity  $v_i$ , i.e., acceleration, can be changed by surface forces, described by the stress tensor  $\sigma_{ij}$ , or body forces  $f_i$ . This is the **stress-velocity** formulation.

Another common form of the elastic equation of motion can be deduced by taking the time derivative of the displacement velocity ( $v_i = \frac{\partial u_i}{\partial t}$ ). Eq. 1 can then be transformed into a second-order partial differential equation:

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i . \quad (2)$$

This expression is called **stress-displacement** formulation. Both formulations describe a general medium, like gas, fluid, solid or plasma. The material-specific properties are introduced by additional equations which describe how the medium reacts when a certain force is applied.

The **generalized stress-strain relationship** (Hooke's law) for any elastic medium, valid for infinitesimally small deformations, is generally formulated as

$$\sigma_{ij}(t) = C_{ijkl}(t) \varepsilon_{kl}(t) , \quad (3)$$

where  $C_{ijkl}$  represents the elasticity (or stiffness) tensor, which is described in detail in subsection 3.3, and  $\varepsilon_{kl}$  the deformation tensor, given by

$$\varepsilon_{kl} = \frac{1}{2} \left( \frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k} \right) . \quad (4)$$

The variable  $\varepsilon_{kl}$  is a second-order symmetric tensor with  $u_k$  and  $u_l$  representing the displacement of a particle in the direction of the  $l$ -th and  $k$ -th dimension, respectively, of a Cartesian coordinate system. Its derivative has the form of

$$\frac{\partial \varepsilon_{kl}}{\partial t} = \frac{1}{2} \left( \frac{\partial v_k}{\partial x_l} + \frac{\partial v_l}{\partial x_k} \right) , \quad (5)$$

where  $v_k$  and  $v_l$  represent the  $l$ -th and  $k$ -th component, respectively, of the particle velocity. Not only  $\varepsilon_{kl}$  is a symmetric tensor, but also  $\sigma_{ij}$  and, consequently,  $C_{ijkl}$ .

In the special case of **isotropy**, the stress-strain relation in the modified form is given by

$$\sigma_{ij} = \lambda \theta \delta_{ij} + 2 \mu \varepsilon_{ij} , \quad (6)$$



where  $\theta$  denotes the cubic dilatation, i.e., the trace of  $\varepsilon_{ij}$ , and  $\delta_{ij}$  is the Kronecker delta;  $\lambda$  and  $\mu$  represent the two Lamé parameters, both of them time-invariant, with  $\mu$  also referred to as the shear modulus. By taking the derivative with respect to time, eq. 6 becomes

$$\frac{\partial \sigma_{ij}}{\partial t} = \lambda \frac{\partial v_k}{\partial x_k} \delta_{ij} + \mu \left( \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right). \quad (7)$$

### 3.2 Equation of motion for a viscoelastic medium

When propagating through the Earth, the amplitudes of seismic waves decrease due to a couple of different physical phenomena: geometrical spreading, intrinsic attenuation, and scattering. Of great importance is the **intrinsic (anelastic) attenuation**, which represents the actual dissipation of seismic wave mechanical energy that is then converted to heat (mainly due to changes in viscosity); it is often anisotropic (Bai, T. Zhu, et al., 2019). To quantify its propagation effects, a dimensionless frequency-dependent parameter must be introduced, namely the seismic quality factor  $Q$ , which physically describes the relative energy loss per period:  $Q(\omega) = 4\pi \frac{E}{\Delta E}$  (O’Connell and Budiansky, 1978).

The mathematical framework that allows us to describe the effects of attenuation is the viscoelastic model. In such a model, the stress-strain relation is governed by the presence of a time dependency referred to as the “**memory (or relaxation) effect**”, which means that the stress  $\sigma_{ij}$  depends on the entire history of the strain field, not only on its current value, which is the case for purely elastic media. Eq. 3 becomes

$$\sigma_{ij}(t) = C_{ijkl}(t) * \varepsilon_{kl}(t), \quad (8)$$

where  $C_{ijkl}$  represents the stress-impulse response function of the viscoelastic medium and the star represents convolution. With  $C_{ijkl}(t) = \dot{\Psi}_{ijkl}(t)$ , where  $\Psi_{ijkl}(t)$  is referred to as the **stress-relaxation function**, it becomes<sup>1</sup>

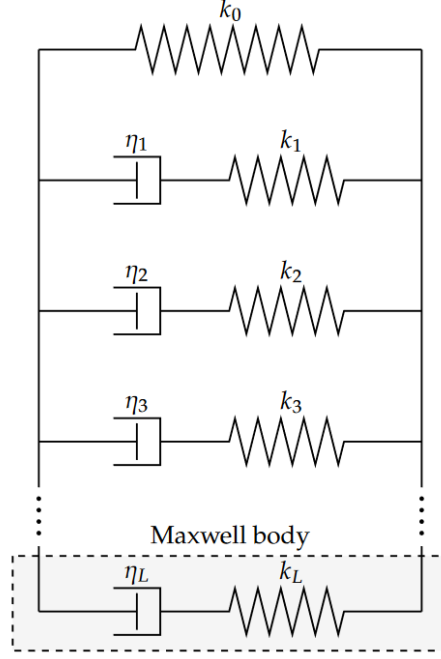
$$\sigma_{ij}(t) = \dot{\Psi}_{ijkl}(t) * \varepsilon_{kl}(t) = \Psi_{ijkl}(t) * \dot{\varepsilon}_{kl}(t). \quad (9)$$

In order to calculate  $\Psi_{ijkl}$ , a mathematical model that can explain the observations must be considered. The most common method to analyze attenuation is based on mechanical (rheological) models (Knopoff and MacDonald, 1958), where frequency-independent  $Q$  behavior of seismic wave propagation in the Earth’s interior is approximated by a superposition of mechanical elements, i.e., a **generalized standard linear solid** (GSLS), introduced by Liu et al. (1976). A GSLS consists of a superposition of  $L$  rheological (Maxwell) bodies arranged in parallel, where each one provides one relaxation mechanism consisting of two other mechanical systems: a Hooke element  $k_0$  (a single spring) for elastic behavior, and a Maxwell body (a spring  $k_i$  and a dashpot  $\eta_i$  in series) for absorption (J. Blanch et al., 1995) (see the schematic diagram in Fig. 1).

With such a setup, new parameters are introduced:

- $\tau^{\sigma l}$  and  $\tau^{\varepsilon l}$ : **stress-relaxation time** and **strain-retardation time**, respectively, of the  $l$ -th relaxation mechanism,
- $C^R$ : **relaxed (or equilibrium) modulus**, i.e.,  $\lim_{t \rightarrow \infty} C(t)$  (where  $C(t)$  represents the stress-impulse response function of the viscoelastic medium), corresponding to the low-frequency limit, i.e.,  $\omega = 0$ .

<sup>1</sup>For more details, see the paper of Bai and Tsvankin (2016).



**Figure 1:** Schematic diagram of a generalized standard linear solid (GSLS) composed of  $L$  relaxation mechanisms (or Maxwell bodies), highlighted by the dashed square.  $k_i$  ( $i = 1, \dots, L$ ) and  $\eta_i$  ( $i = 1, \dots, L$ ) represent the elastic moduli and Newtonian viscosities, respectively, and are connected with the stress relaxation times  $\tau^{\sigma l}$  and strain retardation times  $\tau^{\epsilon l}$  via  $\tau^{\sigma l} = \frac{\eta_l}{k_l}$  and  $\tau^{\epsilon l} = \frac{\eta_l}{k_0} + \frac{\eta_l}{k_l}$  (after Zener, 1948; Bohlen, 2002).

According to Bai and Tsvankin (2016), each relaxation process can be expressed in terms of the stress relaxation function  $\Psi_{ijkl}(t)$  of the GSLS as<sup>2</sup>

$$\Psi_{ijkl}(t) = C_{ijkl}^R \left[ 1 + \sum_{l=1}^L \left( \frac{\tau_{ij}^{\epsilon l}}{\tau^{\sigma l}} - 1 \right) \exp \left( -\frac{t}{\tau^{\sigma l}} \right) \right] H(t) . \quad (10)$$

To simplify equation 10, J. Blanch et al. (1995) introduced the  $\tau$ -method, which demonstrates that the magnitude of attenuation in anisotropic media is directly quantified by the dimensionless matrix  $\tau_{ij}$ , which generally describes the **attenuation parameters** (directional variation of attenuation):

$$\tau_{ij} = \frac{\tau_{ij}^{\epsilon l}}{\tau^{\sigma l}} - 1 . \quad (11)$$

While  $\tau_{ij}$  simply vanishes in the elastic case, as  $\tau^{\sigma l} = \tau^{\epsilon l}$ , it should remain constant for all  $L$  relaxation mechanisms in the viscoelastic case. In the previous studies, the attenuation was considered to be isotropic. We now included attenuation anisotropy as well.

Considering the  $\tau$ -method, the explicit form of equation 10 is given by

$$\Psi_{ijkl}(t) = C_{ijkl}^R \left[ 1 + \sum_{l=1}^L \tau_{ij} \exp \left( -\frac{t}{\tau^{\sigma l}} \right) \right] H(t) . \quad (12)$$

Note that this relaxation function differs from the definition given in Bai and Tsvankin

<sup>2</sup>The differential calculus is firstly done in the Laplace domain and then back-transformed to the time domain.

(2016). In the special case of isotropy,  $\Psi(t)$  reads

$$\Psi_P(t) = \pi \left[ 1 + \sum_{l=1}^L \tau_P \exp\left(-\frac{t}{\tau^{\sigma l}}\right) \right] H(t) \quad (13a)$$

$$\Psi_S(t) = \mu \left[ 1 + \sum_{l=1}^L \tau_S \exp\left(-\frac{t}{\tau^{\sigma l}}\right) \right] H(t) . \quad (13b)$$

Bai and Tsvankin (2016) relate the relaxed modulus  $C_{ijkl}^R$ , corresponding to the model velocities, to the  $C_{ijkl}$  modulus defined at the reference frequency  $\omega_0 = 2\pi f_0$ :

$$C_{ijkl}^R = C_{ijkl} \Big|_{\omega_0} \left( 1 + \sum_{l=1}^L \tau_{ij} \frac{(\omega_0 \tau^{\sigma l})^2}{1 + (\omega_0 \tau^{\sigma l})^2} \right)^{-1} . \quad (14)$$

which, in the isotropic case, becomes

$$\pi = V_P^2 \Big|_{\omega_0} \rho \left( 1 + \sum_{l=1}^L \tau_P \frac{(\omega_0 \tau^{\sigma l})^2}{1 + (\omega_0 \tau^{\sigma l})^2} \right)^{-1} , \quad (15a)$$

$$\mu = V_S^2 \Big|_{\omega_0} \rho \left( 1 + \sum_{l=1}^L \tau_S \frac{(\omega_0 \tau^{\sigma l})^2}{1 + (\omega_0 \tau^{\sigma l})^2} \right)^{-1} . \quad (15b)$$

Following Y. Zhu and Tsvankin (2006) and equation 14, the mathematical matrix representation of  $Q$  is given by

$$Q_{ij}^{-1}(\omega, \tau^{\sigma l}, \tau_{ij}) = \frac{\sum_{l=1}^L \tau_{ij} \frac{\omega \tau^{\sigma l}}{1 + (\omega \tau^{\sigma l})^2}}{1 + \sum_{l=1}^L \tau_{ij} \frac{(\omega \tau^{\sigma l})^2}{1 + (\omega \tau^{\sigma l})^2}} , \quad (16)$$

with the substitutions  $\tau = \tau_P$  and  $\tau = \tau_S$ , in the special case of isotropy, defining the level of attenuation for P- and S-waves, respectively. In general, the width of the frequency range in which a nearly constant  $Q_{ij}$ -spectrum can be simulated directly depends on the number of relaxation mechanisms (Bai and Tsvankin, 2016; Komatitsch and Tromp, 1999). When a GSLs consists of only one relaxation mechanism, i.e.,  $L = 1$ , a good estimate for  $\tau$  is  $\tau = \frac{2}{Q}$  (Bohlen, 2002).

At  $t = 0$ ,  $\Psi_{ijkl}$  yields the **unrelaxed (or instantaneous) modulus**  $C^U$ , i.e.,  $\lim_{t \rightarrow 0} C(t)$ , which corresponds to the high-frequency limit, i.e.,  $\omega = \infty$ .

$$C_{ijkl}^U \equiv \Psi_{ijkl} \Big|_{t=0} = C_{ijkl}^R (1 + L \tau_{ij}) , \quad (17)$$

which, in the special case of isotropy, becomes

$$\pi^U = \pi (1 + L \tau_P) \quad \text{and} \quad \mu^U = \mu (1 + L \tau_S) . \quad (18)$$

By substituting  $\Psi_{ijkl}(t)$  in the generalized stress-strain relation in viscoelastic media, and taking the time derivative on both sides, equation 9 becomes

$$\begin{aligned} \dot{\sigma}_{ij} &= \dot{\Psi}_{ijkl}(t) * \dot{\epsilon}_{kl} = \\ &= C_{ijkl}^U \dot{\epsilon}_{kl} - (C_{ijkl}^U - C_{ijkl}^R) \left[ \sum_{l=1}^L \frac{1}{\tau^{\sigma l}} \exp\left(-\frac{t}{\tau^{\sigma l}}\right) \right] H(t) * \dot{\epsilon}_{kl} . \end{aligned} \quad (19)$$

which, in isotropic anelastic media, is given by

$$\sigma_{ij} = (\dot{\Psi}_P - 2\dot{\Psi}_S) * \theta \delta_{ij} + 2\dot{\Psi}_S * \varepsilon_{ij} . \quad (20)$$

In order to replace the convolution terms, Carcione et al. (1988) and Robertsson, J. O. Blanch, et al. (1994) suggested that **memory variables**  $r_{ij}^l$ , corresponding to the  $ij$ -th component for the  $l$ -th relaxation mechanism should be introduced in such a way that

$$r_{ij}^l = -\frac{2}{L\tau^{\sigma l}} (C_{ijkl}^U - C_{ijkl}^R) \exp\left(-\frac{t}{\tau^{\sigma l}}\right) H(t) * \dot{\varepsilon}_{kl} , \quad (21)$$

which becomes by differentiating with respect to time and rearranging

$$\begin{aligned} \dot{r}_{ij}^l = & -\frac{2}{L\tau^{\sigma l}} \left[ -\frac{1}{\tau^{\sigma l}} (C_{ijkl}^U - C_{ijkl}^R) \exp\left(-\frac{t}{\tau^{\sigma l}}\right) H(t) * \dot{\varepsilon}_{kl} \right] - \\ & -\frac{2}{L\tau^{\sigma l}} (C_{ijkl}^U - C_{ijkl}^R) \exp\left(-\frac{t}{\tau^{\sigma l}}\right) \delta(t) * \dot{\varepsilon}_{kl} . \end{aligned} \quad (22)$$

Introducing the expression of  $r_{ij}^l$  at  $t = 0$  in equation 22, I get

$$\dot{r}_{ij}^l = -\frac{2}{L\tau^{\sigma l}} \left[ \frac{1}{2} (C_{ijkl}^U - C_{ijkl}^R) \left( \frac{\partial v_k}{\partial x_l} + \frac{\partial v_l}{\partial x_k} \right) + r_{ij}^l \right] . \quad (23)$$

Thus, equation 19 becomes

$$\dot{\sigma}_{ij} = C_{ijkl}^U \left( \frac{\partial v_k}{\partial x_l} + \frac{\partial v_l}{\partial x_k} \right) + \sum_{l=1}^L r_{ij}^l . \quad (24)$$

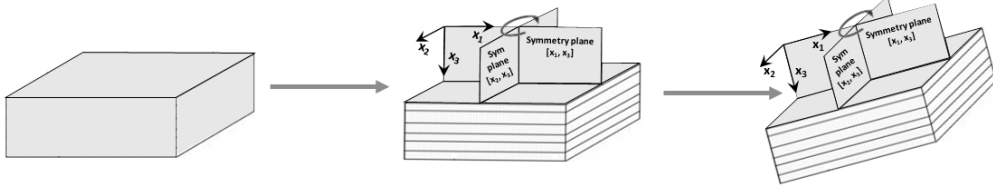
which, in the special case of isotropy, reads

$$\dot{\sigma}_{ij} = \left[ \pi (1 + L\tau_{\varepsilon l}^P) - 2\mu (1 + L\tau_{\varepsilon l}^S) \right] \frac{\partial v_k}{\partial x_k} \delta_{ij} + \mu (1 + L\tau_{\varepsilon l}^S) \left( \frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) + \sum_{l=1}^L r_{ij}^l . \quad (25)$$

### 3.3 Equation of motion for an anisotropic medium

#### 3.3.1 Seismic anisotropy and elasticity tensor

As opposed to isotropy, anisotropy is the property of directional dependency. In seismics, anisotropy describes the directional or angular dependence of the velocity of seismic waves in a medium within the Earth, which can help to characterize preferred orientation and in-situ stress conditions. If the rock properties are aligned over a greater volume extending several wavelengths, they can create certain symmetries of seismic anisotropy (Thomsen, 1986). Hexagonal (or cylindrical) symmetry is of particular interest in seismics, as it is relatively simple but can still approximate many actual situations in the Earth, for instance where sedimentary layers are neatly ordered. The most typical anisotropic medium that exhibits hexagonal symmetry is **transversely isotropic**, which means that the velocity normal to the bedding is slower than the one along it. Commonly investigated symmetries are **vertical-transverse isotropy** (VTI), with a vertical symmetry axis and infinite number of horizontal planes with isotropic behavior, and **tilted-transverse isotropy** (TTI), with a rotated symmetry axis, and thus properties changing with azimuth (Tsvankin, 2012) (see Fig. 2). Bai and Tsvankin (2016) developed a detailed numerical demonstration and analysis of vertical-transverse isotropic attenuation by 2D time-domain finite-difference modeling, while Oh et al. (2020) developed a similar tilted-transverse version.



**Figure 2:** From left to right: isotropic, VTI, and TTI media in a 3D Cartesian coordinate system with  $(x_1, x_2, x_3) = (x, y, z)$  (modified after TGS, 2017).

The structure of the **stiffness tensor**  $C_{ijkl}$ , previously introduced in section 3.1, is the factor that determines the velocity and polarization of seismic waves for any propagation direction (Tsvankin, 2012), thus uniquely describing the above-mentioned symmetries and their elastic moduli. In 3D, the fourth-order tensor has 36 independent components,

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{12} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{13} & c_{23} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{14} & c_{24} & c_{34} & c_{44} & c_{45} & c_{46} \\ c_{15} & c_{25} & c_{35} & c_{45} & c_{55} & c_{56} \\ c_{16} & c_{26} & c_{36} & c_{46} & c_{56} & c_{66} \end{pmatrix}, \quad (26)$$

which is expressed in the Voigt notation (Voigt, 1910).

For a polarization direction of the P-SV waves in 2D only, namely in the  $x_1x_2$ -plane, with the axes of the Cartesian coordinate system defined as  $x_1$  for horizontal axis and  $x_2$  for the vertical (depth) axis, the stiffness matrix becomes a  $3 \times 3$  matrix with elements defining the  $x_1x_1$ -,  $x_2x_2$ -, and  $x_1x_2$ -directions. The stiffness tensor is then reduced to

$$\mathbf{C} = \begin{pmatrix} c_{11} & c_{13} & c_{15} \\ c_{13} & c_{33} & c_{35} \\ c_{15} & c_{35} & c_{55} \end{pmatrix}. \quad (27)$$

In the case of isotropic media, where the elastic behavior is independent of direction, only four elastic constants are independent and they simply reduce to

$$c_{11} = c_{33} = \lambda + 2\mu = \pi, \quad (28a)$$

$$c_{13} = \lambda, \quad (28b)$$

$$c_{55} = \mu, \quad (28c)$$

which form the elements of the stiffness matrix in 2D isotropic media:

$$\mathbf{C}^{\text{iso}} = \begin{pmatrix} \pi & \lambda & 0 \\ \lambda & \pi & 0 \\ 0 & 0 & \mu \end{pmatrix}. \quad (29)$$

The P- and S-wave velocities will then read

$$v_{P,ver} = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \quad (30a)$$

$$v_{SV,ver} = \sqrt{\frac{\mu}{\rho}}. \quad (30b)$$

In the case of 2D VTI media, eq. 27 reads (Thomsen, 1986)

$$\mathbf{C}^{\text{VTI}} = \begin{pmatrix} c_{11} & c_{13} & 0 \\ c_{13} & c_{33} & 0 \\ 0 & 0 & c_{55} \end{pmatrix}. \quad (31)$$

Under the assumption of weak anisotropy, i.e., less than approx. 20%, these independent elastic constants have been combined by Thomsen (1986) into the so-called **Thomsen anisotropic parameters**  $\varepsilon$  and  $\delta$  (dimensionless), such that they can better describe the seismic wave propagation:

$$\varepsilon = \frac{c_{11} - c_{33}}{2c_{33}}, \quad (32a)$$

$$\delta = \frac{(c_{13} + c_{55})^2 - (c_{33} - c_{55})^2}{2c_{33}(c_{33} - c_{55})}. \quad (32b)$$

The vertical P- and S-wave velocities can also be expressed in terms of these elastic constants:

$$v_{P,ver} \equiv \sqrt{\frac{c_{33}}{\rho}} \quad \text{and} \quad v_{SV,ver} \equiv \sqrt{\frac{c_{55}}{\rho}}. \quad (33)$$

The angle-dependency of phase velocities, with the angle  $\theta$  made by the symmetry axis of the VTI medium with the normal of the wavefront, is then given by the following approximate expressions:

$$v_P(\theta) \approx v_{P,ver} (1 + \delta \sin^2 \theta \cos^2 \theta + \varepsilon \sin^4 \theta), \quad (34a)$$

$$v_{SV}(\theta) \approx v_{SV,ver} \left( 1 + \frac{v_{P,ver}^2}{v_{SV,ver}^2} (\varepsilon - \delta) \sin^2 \theta \cos^2 \theta \right), \quad (34b)$$

with the maximal P-wave velocity given for a horizontal direction of wave propagation, while the maximal S-wave is reached at  $45^\circ$ :

$$v_{P,hor} \equiv v_{P,ver} (1 + \varepsilon), \quad (35a)$$

$$v_{S,45^\circ} \equiv \frac{v_{P,hor}^2 (\varepsilon - \delta)}{4 v_{SV,ver}} + v_{SV,ver}. \quad (35b)$$

In the case of  $C_{ijkl}$  set along the symmetry axis, i.e.,  $\theta = 0^\circ$ , the elements of the stiffness matrix  $\mathbf{C}^{\text{VTI}}$  in eq. 31 can be expressed as

$$\begin{pmatrix} c_{11} \\ c_{33} \\ c_{13} \\ c_{55} \end{pmatrix} = \begin{pmatrix} \rho v_{P,ver}^2 (1 + 2\varepsilon) \\ \rho v_{P,ver}^2 \\ \rho \left( \sqrt{(v_{P,ver}^2 - v_{SV,ver}^2) [v_{P,ver}^2 (1 + 2\delta) - v_{SV,ver}^2]} - v_{SV,ver}^2 \right) \\ \rho v_{SV,ver}^2 \end{pmatrix}. \quad (36)$$

It can be observed that, while  $\varepsilon$  quantifies the velocity difference for wave propagation along and perpendicular to the symmetry axis,  $\delta$  controls the P-wave propagation for angles near the symmetry axis.

In the case of TTI media, the stiffness matrix can be derived from  $\mathbf{C}^{\text{VTI}}$  using the Bond transformation (Bond, 1943; Carcione, 2007) as suggested by Oh et al. (2020):

$$\tilde{\mathbf{C}}^{\text{TTI}} = D(\theta_T) \mathbf{C}^{\text{VTI}} D^T(\theta_T), \quad (37)$$

with

$$D(\theta_T) = \begin{pmatrix} \cos^2 \theta_T & \sin^2 \theta_T & 2 \sin \theta_T \cos \theta_T \\ \sin^2 \theta_T & \cos^2 \theta_T & -2 \sin \theta_T \cos \theta_T \\ -\sin \theta_T \cos \theta_T & \sin \theta_T \cos \theta_T & -\sin^2 \theta_T \end{pmatrix}. \quad (38)$$

The matrix  $D$ , rotated about the vertical axis in a counter-clockwise direction, represents a simple coordinate-rotation matrix, and the parameter  $\theta_T$  denotes the tilt angle of the symmetry axis measured from the vertical. With its general expression given by

$$\tilde{\mathbf{C}}^{\text{TTI}} = \begin{pmatrix} \tilde{c}_{11} & \tilde{c}_{13} & \tilde{c}_{15} \\ \tilde{c}_{13} & \tilde{c}_{33} & \tilde{c}_{35} \\ \tilde{c}_{15} & \tilde{c}_{35} & \tilde{c}_{55} \end{pmatrix}, \quad (39)$$

the  $\tilde{c}_{ij}$  elements can be explicitly defined as

$$\begin{pmatrix} \tilde{c}_{11} \\ \tilde{c}_{33} \\ \tilde{c}_{13} \\ \tilde{c}_{55} \\ \tilde{c}_{15} \\ \tilde{c}_{35} \end{pmatrix} = \begin{pmatrix} c_{11} \cos^4 \theta_T + c_{33} \sin^4 \theta_T + 2(c_{13} + 4c_{55}) \sin^2 \theta_T \cos^2 \theta_T \\ c_{11} \sin^4 \theta_T + c_{33} \cos^4 \theta_T + 2(c_{13} + 4c_{55}) \sin^2 \theta_T \cos^2 \theta_T \\ (c_{11} + c_{33} - 4c_{55}) \sin^2 \theta_T \cos^2 \theta_T + c_{13} (\sin^4 \theta_T + \cos^4 \theta_T) \\ (c_{11} + c_{33} - 2c_{13}) \sin^2 \theta_T \cos^2 \theta_T + c_{55} (\cos^2 \theta_T - \sin^2 \theta_T)^2 \\ (c_{13} - c_{11} + 2c_{55}) \cos^3 \theta_T \sin \theta_T - (c_{13} - c_{33} + 2c_{55}) \cos \theta_T \sin^3 \theta_T \\ (c_{13} - c_{11} + 2c_{55}) \sin^3 \theta_T \cos \theta_T - (c_{13} - c_{33} + 2c_{55}) \sin \theta_T \cos^3 \theta_T \end{pmatrix}. \quad (40)$$

### 3.3.2 Wave equations

In this section, the velocity-stress formulation of the system of first-order differential equations that are the basis for the 2D FD anisotropic implementation is given. The wave equation for the 2D viscoelastic TTI media has been implemented recently into SOFI2D in order to verify the consistency of our extended wave equation formulation for anisotropic media with our previous formulation for isotropic media.

For elastic media:

- isotropic wave equations

$$\dot{\sigma}_{xx} = \pi \frac{\partial v_x}{\partial x} + \lambda \frac{\partial v_z}{\partial z}, \quad (41a)$$

$$\dot{\sigma}_{zz} = \lambda \frac{\partial v_x}{\partial x} + \pi \frac{\partial v_z}{\partial z}, \quad (41b)$$

$$\dot{\sigma}_{xz} = \mu \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right). \quad (41c)$$

- VTI wave equations

$$\dot{\sigma}_{xx} = c_{11} \frac{\partial v_x}{\partial x} + c_{13} \frac{\partial v_z}{\partial z}, \quad (42a)$$

$$\dot{\sigma}_{zz} = c_{13} \frac{\partial v_x}{\partial x} + c_{33} \frac{\partial v_z}{\partial z}, \quad (42b)$$

$$\dot{\sigma}_{xz} = c_{55} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right). \quad (42c)$$

- TTI wave equations

$$\dot{\sigma}_{xx} = \tilde{c}_{11} \frac{\partial v_x}{\partial x} + \tilde{c}_{13} \frac{\partial v_z}{\partial z} + \tilde{c}_{15} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right), \quad (43a)$$

$$\dot{\sigma}_{zz} = \tilde{c}_{13} \frac{\partial v_x}{\partial x} + \tilde{c}_{33} \frac{\partial v_z}{\partial z} + \tilde{c}_{35} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right), \quad (43b)$$

$$\dot{\sigma}_{xz} = \tilde{c}_{15} \frac{\partial v_x}{\partial x} + \tilde{c}_{35} \frac{\partial v_z}{\partial z} + \tilde{c}_{55} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right). \quad (43c)$$

The model parameters in the implementation of viscoelastic modeling are represented by the velocities and attenuation at  $\omega_0$  and the relaxation frequencies  $\tau^{\sigma l}$ :

- isotropic wave equations

$$\dot{\sigma}_{xx} = \pi (1 + L \tau^P) \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) - 2 \mu (1 + L \tau^S) \frac{\partial v_z}{\partial z} + \sum_{l=1}^L r_{xx}^l, \quad (44a)$$

$$\dot{\sigma}_{zz} = \pi (1 + L \tau^P) \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) - 2 \mu (1 + L \tau^S) \frac{\partial v_x}{\partial x} + \sum_{l=1}^L r_{zz}^l, \quad (44b)$$

$$\dot{\sigma}_{xz} = \mu (1 + L \tau^S) \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + \sum_{l=1}^L r_{xz}^l, \quad (44c)$$

$$\dot{r}_{xx}^l = -\frac{1}{\tau^{\sigma l}} \left[ \pi \tau^P \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) - 2 \mu \tau^S \frac{\partial v_z}{\partial z} + r_{xx}^l \right], \quad (44d)$$

$$\dot{r}_{zz}^l = -\frac{1}{\tau^{\sigma l}} \left[ \pi \tau^P \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) - 2 \mu \tau^S \frac{\partial v_x}{\partial x} + r_{zz}^l \right], \quad (44e)$$

$$\dot{r}_{xz}^l = -\frac{1}{\tau^{\sigma l}} \left[ \mu \tau^S \left( \frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z} \right) + r_{xz}^l \right]. \quad (44f)$$

- VTI media

$$\dot{\sigma}_{xx} = c_{11}^U \frac{\partial v_x}{\partial x} + c_{13}^U \frac{\partial v_z}{\partial z} + \sum_{l=1}^L r_{xx}^l, \quad (45a)$$

$$\dot{\sigma}_{zz} = c_{13}^U \frac{\partial v_x}{\partial x} + c_{33}^U \frac{\partial v_z}{\partial z} + \sum_{l=1}^L r_{zz}^l, \quad (45b)$$

$$\dot{\sigma}_{xz} = c_{55}^U \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + \sum_{l=1}^L r_{xz}^l, \quad (45c)$$

$$\dot{r}_{xx}^l = -\frac{1}{\tau^{\sigma l}} \left( c_{11}^R \tau_{11} \frac{\partial v_x}{\partial x} + c_{13}^R \tau_{13} \frac{\partial v_z}{\partial z} + r_{xx}^l \right), \quad (45d)$$

$$\dot{r}_{zz}^l = -\frac{1}{\tau^{\sigma l}} \left( c_{13}^R \tau_{13} \frac{\partial v_x}{\partial x} + c_{33}^R \tau_{33} \frac{\partial v_z}{\partial z} + r_{zz}^l \right), \quad (45e)$$

$$\dot{r}_{xz}^l = -\frac{1}{\tau^{\sigma l}} \left[ c_{55}^R \tau_{55} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + r_{xz}^l \right]. \quad (45f)$$

- TTI media

$$\dot{\sigma}_{xx} = \tilde{c}_{11}^U \frac{\partial v_x}{\partial x} + \tilde{c}_{13}^U \frac{\partial v_z}{\partial z} + \tilde{c}_{15} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + \sum_{l=1}^L r_{xx}^l, \quad (46a)$$



$$\dot{\sigma}_{zz} = \tilde{c}_{13}^U \frac{\partial v_x}{\partial x} + \tilde{c}_{33}^U \frac{\partial v_z}{\partial z} + \tilde{c}_{35} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + \sum_{l=1}^L r_{zz}^l, \quad (46b)$$

$$\dot{\sigma}_{xz} = \tilde{c}_{15}^U \frac{\partial v_x}{\partial x} + \tilde{c}_{35}^U \frac{\partial v_z}{\partial z} + \tilde{c}_{55} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + \sum_{l=1}^L r_{xz}^l, \quad (46c)$$

$$-\tau_{ol} \dot{r}_{xx}^l = \tilde{c}_{11}^R \tau_{11} \frac{\partial v_x}{\partial x} + \tilde{c}_{13}^R \tau_{13} \frac{\partial v_z}{\partial z} + \tilde{c}_{15}^R \tau_{13} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + r_{xx}^l, \quad (46d)$$

$$-\tau_{ol} \dot{r}_{zz}^l = \tilde{c}_{33}^R \tau_{33} \frac{\partial v_z}{\partial z} + \tilde{c}_{13}^R \tau_{13} \frac{\partial v_x}{\partial x} + \tilde{c}_{35}^R \tau_{35} \left( \frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) + r_{zz}^l, \quad (46e)$$

$$-\tau_{ol} \dot{r}_{xz}^l = \tilde{c}_{55}^R \tau_{55} \left( \frac{\partial v_z}{\partial x} + \frac{\partial v_x}{\partial z} \right) + \tilde{c}_{15}^R \tau_{15} \frac{\partial v_x}{\partial x} + \tilde{c}_{35}^R \tau_{35} \frac{\partial v_z}{\partial z} + r_{xz}^l. \quad (46f)$$

### 3.3.3 Numerical results

In order to emphasize the pronounced effects of anisotropy on the wavefield, we show the directional variation of P- and S-waves in a highly anisotropic VTI medium (Fig. 3). The P-wave velocity increases significantly towards the horizontal direction ( $\theta = 90^\circ$ ). The angular variation of S-wave depends upon polarity. Horizontally polarized S-waves (SH) are fastest in the horizontal direction, whereas vertically polarized S-waves (SV) have a maximum velocity around  $\theta = 45^\circ$ .

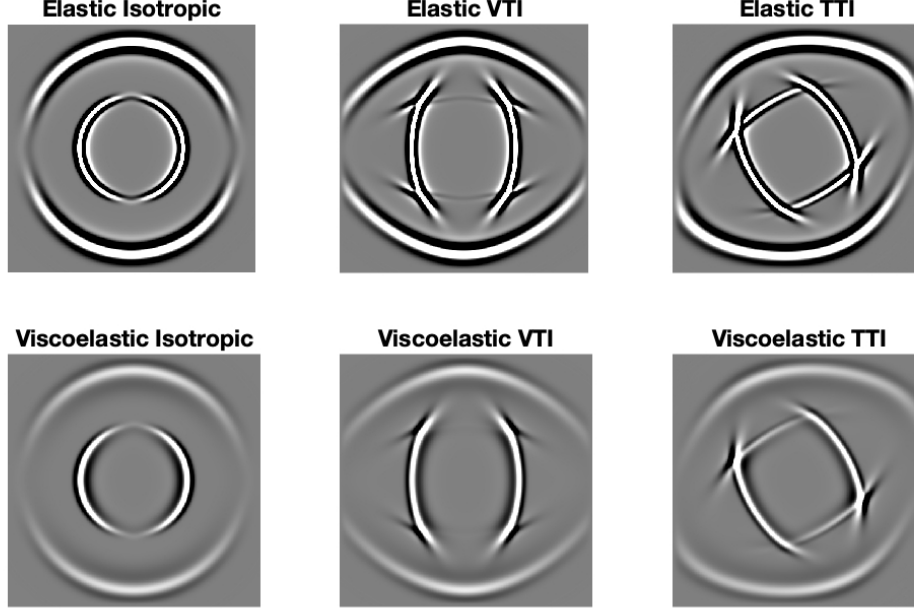
We also compare numerical results for the elastic and viscoelastic case to highlight the effects of attenuation. Whereas attenuation mainly leads to a frequency dependent decay of amplitudes with distance, anisotropy affects both traveltimes and amplitudes considerably. Note, for instance, the strong deformation of both the P- and SV-wave wavefronts in the VTI case according to the directional variation of phase velocities depicted in Fig. 3. Furthermore, in the VTI medium the SV-wave develops significant amplitude anomalies along the diagonal direction ( $\theta = 45^\circ$ ), known as cusps. The counter-clockwise rotation of the symmetry axis in the TTI case by  $\theta = 30^\circ$  is clearly visible and has a strong impact on the radiation pattern of the wavefield for both P- and SV-waves.

### 3.4 Solution of the wave equation by finite differences

Solving the above-mentioned wave equations analytically is only possible for very simple models. In more realistic cases, wave propagation can only be estimated using numerical approaches. One of them is the **finite-difference (FD) method**, in which spatial and temporal derivatives in the wave equations are approximated by FD operators, making it numerical efficient and relatively straightforward to implement. This approximation requires the discretization of the wave equation in both space and time using a Cartesian coordinate system of equidistant grid spacing. Thus, the continuous space coordinates  $x$  and  $z$  and the time vector  $t$  are replaced by their discrete form in a way that  $x = (i - 1) \Delta h$ ,  $z = (j - 1) \Delta h$ , and  $t = n \Delta t$ , with  $i, j$  representing the position of a specific grid point and  $n$  referring to a specific **time step**;  $\Delta h$  denotes the spatial distance between two adjacent grid points (i.e., the **grid spacing**) and  $\Delta t$  the difference between two successive time steps (i.e., the time sampling interval). Therefore, every grid point is located in the interval  $i \in N[1, NX]$ ,  $j \in N[1, NY]$  and  $n \in N[1, NT]$ , where  $NX$ ,  $NY$  and  $NT$  are the number of discrete spatial grid points and time steps, respectively.<sup>3</sup>

Two types of operators can be distinguished, forward and backward operators  $D^+$ ,  $D^-$ . The derivative of a function  $y$  with respect to a variable  $x$  can be approximated by the following

<sup>3</sup>Although SOFI is written in the C programming language, it internally uses one-based indices, hence the index range from  $[1, N]$  rather than  $[0, N - 1]$  as you might expect in C.



**Figure 3:** Snapshots of vertical component of particle velocity excited by a vertical point force. The snapshots show P- and SV-waves and the effects of attenuation and VTI and TTI anisotropy. The homogeneous anisotropic VTI model is that of Greenhorn shale. In the TTI model, the symmetry axis has been rotated counter-clockwise  $\theta = 30^\circ$ . Attenuation is assumed to be isotropic and described by  $\sigma_P = \sigma_S = 0.25$ ,  $L = 1$ .

operators:

$$D_x^+ y = \frac{y[i+1] - y[i]}{\Delta h}, \quad (47a)$$

$$D_x^- y = \frac{y[i] - y[i-1]}{\Delta h}. \quad (47b)$$

### 3.4.1 Standard staggered grid

The propagation of elastic waves in the stress-velocity formulation can be modeled on a standard staggered grid (SSG), which ensures stability and high accuracy (Virieux, 1986; Levander, 1988). On such a grid, different components of one physical parameter are defined at different staggered points. Fig. 4 shows the locations of viscoelastic wavefield parameters and material parameters on the SSG. Fig. 4 shows the locations of wavefield parameters: the two different components of the particle velocity  $v_i$  are distributed over two different staggered locations, at half grid points; the stress components  $\sigma_{ij}$  are distributed over two different locations: the diagonal stresses on full grid points, and the shear stresses on half grid points; the same holds for the memory variables.

The material parameters, namely  $\rho$ ,  $\mu$ ,  $\tau$ , and the  $c_{ij}$  components need to be locally averaged, i.e., calculated from neighboring grid points. The averaging of material parameters is critical for the accuracy at strong discontinuities (Zahradník et al., 1993; Falk, 1998; Moczo et al., 2002). To obtain stable results, arithmetic averaging must be used for the density  $\rho$  and attenuation parameter  $\tau^S$ , and harmonic averaging is required for the shear modulus (Fellinger et al., 1995; Graves, 1996; Falk, 1998; Vossen et al., 2002; Moczo et al., 2002):

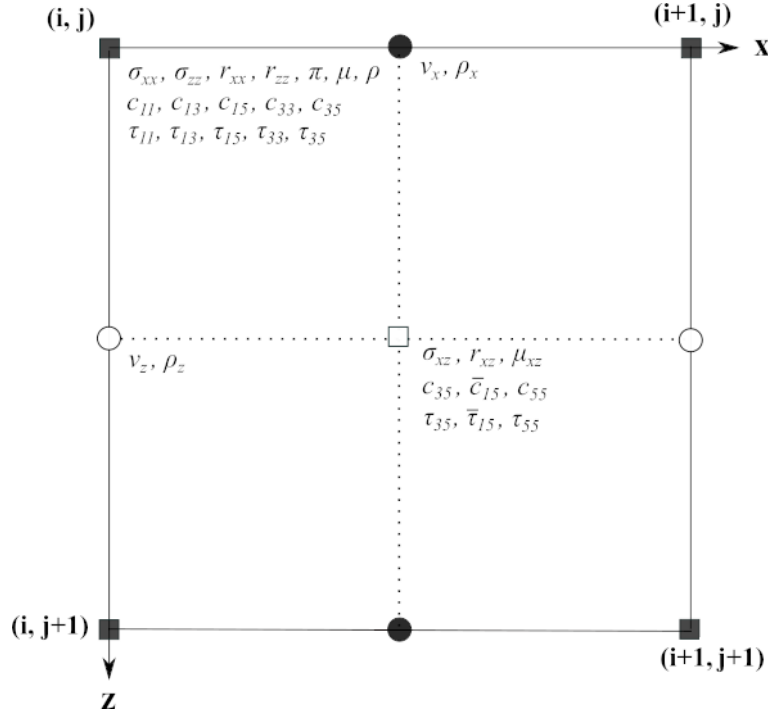
$$\rho_x[j, i + \frac{1}{2}] = \frac{\rho[j, i] + \rho[j, i+1]}{2}, \quad (48a)$$

$$\rho_y[j + \frac{1}{2}, i] = \frac{\rho[j, i] + \rho[j + 1, i]}{2}, \quad (48b)$$

$$\tau_{xy}^s[j + \frac{1}{2}, i + \frac{1}{2}] = \frac{\tau^s[j, i] + \tau^s[j, i + 1] + \tau^s[j + 1, i + 1] + \tau^s[j + 1, i]}{4}, \quad (48c)$$

$$\mu_{xy}[j + \frac{1}{2}, i + \frac{1}{2}] = \frac{4}{\mu^{-1}[j, i] + \mu^{-1}[j, i + 1] + \mu^{-1}[j + 1, i + 1] + \mu^{-1}[j + 1, i]}. \quad (48d)$$

This averaging procedure satisfies the condition of traction continuity across the interface between two media on standard staggered grids (Moczo et al., 2002). Harmonic averaging of the shear modulus  $\mu$  yields higher accuracy than arithmetic averaging for modeling Scholte waves propagating along fluid-solid interfaces (Falk, 1998). What matters even more, our numerical tests show that arithmetic averaging of the shear modulus across free surfaces leads to instabilities; harmonic averaging produces stable simulations.

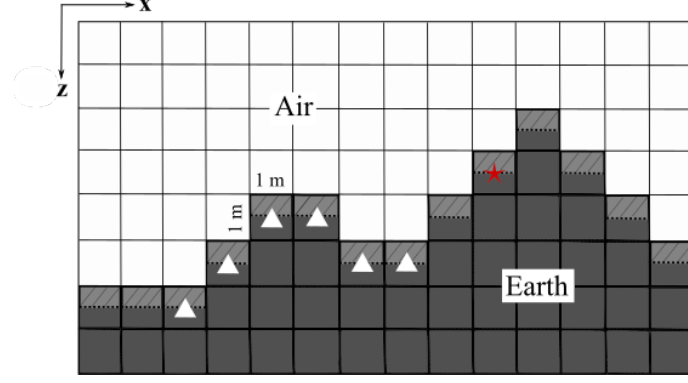


**Figure 4:** Distribution of viscoelastic anisotropic wavefield- and material parameters on an elementary finite-difference cell in the standard staggered grid (modified after Virieux, 1986). At half grid point, i.e.,  $(i + \frac{1}{2}, j + \frac{1}{2})$ ,  $\bar{c}_{15}$  and  $\bar{\tau}_{15}$  are harmonically- and arithmetically-averaged, respectively.

### 3.4.2 Free surface and interfaces on a SSG

The determination of the actual location of an interface between two media on the SSG is not a trivial task. Only in the case that interfaces are aligned with the grid the interface location can be determined. On curved interfaces, however, the interface location cannot be determined exactly due to the staggered grid representation of the model.

Varying topography in FD modeling can only be represented by staircases (Fig. 4). As the true topography can show strong variations in height, the topography on the SSG needs to be carefully handled. Following the distribution of physical parameters in an SSG cell, with the vertical component of the particle velocity calculated at half grid points, all vertical point sources and receivers must be located at half grid points, too, so that any numerical issues are avoided during modeling (Fig. 5).



**Figure 5:** Topography on the standard staggered grid. Red star: source point; white triangles: receivers. Note the staircases that appear due to quantization of the elevation values.

The interface between the elastic medium and air at the surface is very important when trying to model surface waves or multiple reflections in a marine environment. Since all stresses in the normal direction at this interface vanish in a way that  $\sigma_{xy} = \sigma_{yy} = 0.0$ ; this boundary condition is called (stress-)**free surface**. Two types of implementations are common. In the implicit definition of the free surface, a small layer with the acoustic parameters of air ( $v_p = 300$  m/s,  $v_s = 0$  m/s,  $\rho = 1.25$  kg/m<sup>3</sup>) is placed on top of the model. One advantage of the implicit definition of the free surface is the easy implementation of topography on the FD grid; however, to get accurate results for surface waves or multiples, this approach requires a fine spatial sampling of the FD grid near the free surface. An explicit free surface can be implemented by using the mirroring technique by Levander, which leads to stable and accurate solutions for plain interfaces (Levander, 1988, Robertsson, Levander, et al., 1995). If the planar free surface is located at grid point  $j = h$ , the stress at this point is set to zero and the stresses below the free surface are mirrored with an inverse sign:

$$\begin{aligned}
 \sigma_{yy}[h, i] &= 0, \\
 \sigma_{yy}[h-1, i] &= -\sigma_{yy}[h+1, i], \\
 \sigma_{xy}[h-\frac{1}{2}, i+\frac{1}{2}] &= -\sigma_{xy}[h+\frac{1}{2}, i+\frac{1}{2}], \\
 \sigma_{xy}[h-\frac{3}{2}, i+\frac{1}{2}] &= -\sigma_{xy}[h+\frac{3}{2}, i+\frac{1}{2}].
 \end{aligned} \tag{49}$$

When updating the stress component  $\sigma_{xx} = \Delta t \pi (v_{xx} + v_{yy}) - 2\Delta t \mu v_{yy}$  at the free surface only horizontal particle velocities should be used because vertical derivatives over the free surface lead to instabilities (Levander, 1988). The vertical derivative of  $v_{yy}$  can be replaced by using the boundary condition at the free surface:

$$\begin{aligned}
 \sigma_{yy} &= \Delta t \pi (v_{xx} + v_{yy}) - 2\Delta t \mu v_{xx} = 0, \\
 v_{yy} &= -\frac{\pi - 2\mu}{\pi} v_{xx}.
 \end{aligned} \tag{50}$$

Therefore,  $\sigma_{xx}$  can be written as

$$\sigma_{xx} = \Delta t \left( 4\mu - \frac{(2\mu)^2}{\pi} \right) v_{xx}. \tag{51}$$

### 3.4.3 Discretization of the wave equation

The discretization of the linear stress-strain relationship at time step  $n$  leads to the following system of equations:

$$\begin{aligned}
v_{xx} [j, i] &\approx \frac{v_x [j, i + \frac{1}{2}] - v_x [j, i - \frac{1}{2}]}{dh} , \\
v_{yy} [j, i] &\approx \frac{v_y [j + \frac{1}{2}, i] - v_y [j - \frac{1}{2}, i]}{dh} , \\
v_{yx} [j + \frac{1}{2}, i + \frac{1}{2}] &\approx \frac{v_y [j + \frac{1}{2}, i + 1] - v_y [j + \frac{1}{2}, i]}{\Delta h} , \\
v_{xy} [j + \frac{1}{2}, i + \frac{1}{2}] &\approx \frac{v_x [j + 1, i + \frac{1}{2}] - v_x [j, i + \frac{1}{2}]}{\Delta h} , \\
\sigma_{xy}^n [j + \frac{1}{2}, i + \frac{1}{2}] &= \sigma_{xy}^{n-1} [j + \frac{1}{2}, i + \frac{1}{2}] + \Delta t \cdot \mu_{xy} [j + \frac{1}{2}, i + \frac{1}{2}] \cdot \\
&\quad \cdot \left( v_{xy} [j + \frac{1}{2}, i + \frac{1}{2}] + v_{yx} [j + \frac{1}{2}, i + \frac{1}{2}] \right) , \\
\sigma_{xx}^n [j, i] &= \sigma_{xx}^{n-1} [j, i] + \Delta t \cdot \pi [j, i] \cdot \left( v_{xx} [j, i] - v_{yy} [j, i] \right) - \\
&\quad - 2 \cdot \Delta t \cdot \mu [j, i] \cdot v_{xx} [j, i] , \\
\sigma_{yy}^n [j, i] &= \sigma_{yy}^{n-1} [j, i] + \Delta t \cdot \pi [j, i] \cdot \left( v_{xx} [j, i] - v_{yy} [j, i] \right) - \\
&\quad - 2 \cdot dt \cdot \mu [j, i] \cdot v_{yy} [j, i] .
\end{aligned} \tag{52}$$

The discretization of the momentum equation leads to the following system of equations:

$$\begin{aligned}
vtt_x [j, i + \frac{1}{2}] &= \left( \sigma_{xx} [j, i + 1] - \sigma_{xx} [j, i] + \sigma_{xy} [j + \frac{1}{2}, i] - \sigma_{xy} [j - \frac{1}{2}, i] \right) , \\
vtt_y [j + \frac{1}{2}, i] &= \left( \sigma_{xy} [j, i + \frac{1}{2}] - \sigma_{xy} [j, i - \frac{1}{2}] + \sigma_{yy} [j + 1, i] - \sigma_{yy} [j, i] \right) , \\
v_x^n [j, i + \frac{1}{2}] &= v_x^{n-1} [j, i + \frac{1}{2}] + \frac{\Delta t}{\Delta h} \cdot \rho_x [j, i + \frac{1}{2}] \cdot vtt_x [j, i + \frac{1}{2}] , \\
v_y^n [j + \frac{1}{2}, i] &= v_y^{n-1} [j + \frac{1}{2}, i] + \frac{\Delta t}{\Delta h} \cdot \rho_y [j + \frac{1}{2}, i] \cdot vtt_y [j + \frac{1}{2}, i] .
\end{aligned} \tag{53}$$

### 3.4.4 Accuracy of FD operators

The derivation of the FD operators in the last section was a simple replacement of the partial derivatives by finite differences. In the following more systematic approach, the first derivative of a variable  $f$  at a grid point  $i$  is estimated by a Taylor series expansion (Jastram, 1992):

$$\begin{aligned}
(2k - 1) \frac{\partial f}{\partial x} \Big|_i &= \frac{1}{\Delta h} (f_{i+(k-1/2)} - f_{i-(k-1/2)}) + \\
&\quad + \frac{1}{\Delta h} \sum_{l=2}^N \frac{((k - \frac{1}{2})(\Delta h)^{2l-1})}{(2l - 1)!} \frac{\partial^{(2l-1)} f}{\partial x^{(2l-1)}} \Big|_i + O(\Delta h)^{2N} .
\end{aligned} \tag{54}$$

For an operator with length  $2N$ ,  $N$  equations are added with a weight  $\beta_k$ :

$$\left[ \sum_{k=1}^N \beta_k (2k-1) \right] \frac{\partial f}{\partial x} \Big|_i = \frac{1}{\Delta h} \sum_{k=1}^N \beta_k (f_{i+(k-1/2)} - f_{i-(k-1/2)}) + \frac{1}{\Delta h} \sum_{k=1}^N \sum_{l=2}^N \beta_k \frac{((k-\frac{1}{2})(\Delta h)^{2l-1})}{(2l-1)!} \frac{\partial^{(2l-1)} f}{\partial x^{(2l-1)}} \Big|_i + O(\Delta h)^{2N}. \quad (55)$$

The case  $N = 1$  leads to the FD operator derived in the last section, which has a length of  $2N = 2$ . The Taylor series is truncated after the first term ( $O(\Delta h)^2$ ). Therefore, this operator is called **2nd order FD operator** which refers to the truncation error of the Taylor series and not to the order of the approximated derivative. To understand equation better, we estimate a **4th order FD operator**. This operator has the length  $2N = 4$  or  $N = 2$ . The sums in Eq. lead to:

$$\begin{aligned} (\beta_1 + 3\beta_2) \frac{\partial f}{\partial x} \Big|_i &= \frac{1}{\Delta h} (\beta_1 (f_{i+1/2} - f_{i-1/2}) + \beta_2 (f_{i+3/2} - f_{i-3/2})) + \\ &+ \frac{\Delta h^3}{\Delta h} \left[ \beta_1 \frac{1}{8 \cdot 3!} + \beta_2 \frac{27}{8 \cdot 3!} \right] \frac{\partial^3 f}{\partial x^3} \Big|_i. \end{aligned} \quad (56)$$

The weights  $\beta_k$  can be calculated by the following approach: The factor in front of the partial derivative on the LHS of Eq. should equal 1, therefore  $\beta_1 + 3\beta_2 = 1$ . The coefficients in front of  $\frac{\partial^3 f}{\partial x^3} \Big|_i$  on the RHS of Eq. should vanish:  $\beta_1 + 27\beta_2 = 0$ . The weights  $\beta_k$  can be estimated by solving the matrix equation:

$$\begin{pmatrix} 1 & 3 \\ 1 & 27 \end{pmatrix} \cdot \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The resulting coefficients are  $\beta_1 = 9/8$  and  $\beta_2 = -1/24$ . Therefore, the 4th order backward- and forward operators are:

$$\begin{aligned} \frac{\partial f}{\partial x} \Big|_{i+1/2} &= \frac{1}{\Delta h} [\beta_1 (f_{i+1} - f_i) + \beta_2 (f_{i+2} - f_{i-1})] && \text{forward operator ,} \\ \frac{\partial f}{\partial x} \Big|_{i-1/2} &= \frac{1}{\Delta h} [\beta_1 (f_i - f_{i-1}) + \beta_2 (f_{i+1} - f_{i-2})] && \text{backward operator .} \end{aligned} \quad (57)$$

The coefficients  $\beta_i$  in the FD operator are called **Taylor coefficients**. The accuracy of higher order FD operators can be improved by seeking for FD coefficients  $\beta_k$  that approximate the first derivative in a certain frequency range (Holberg, 1987). These numerically optimized coefficients are called **Holberg coefficients**.

### 3.4.5 Adams-Bashforth fourth-order accurate time integrators

To improve the precision of the temporal discretization the Adams-Bashforth fourth-order accurate time integrator can be used. The staggered Adams-Bashforth method (ABS) is a multi-step method that uses previously calculated wavefields to increase the accuracy order in time. In Bohlen and Wittkamp (2015a) and Bohlen and Wittkamp (2015b) a detailed study of the performance of the ABS method in 1D and 3D is given. The analysis shows that the numerical dispersion is much lower than that of the widely used second-order leapfrog method. However numerical dissipation is introduced by the ABS method. In simulation

experiments the convincing improvements of accuracy of the fourth-order ABS method is verified. For instance, in a realistic elastic 3D scenario the computing time reduces by a factor of approximately 2.2, whereas the memory requirements increase by approximately the same factor. The ABS method thus provides an alternative strategy to increase the time accuracy by investing computer memory instead of computing time.

In the following we provide an extract of Bohlen and Wittkamp (2015a) to demonstrate the underlying theory. Additionally, seismograms of different temporal orders of accuracy and an analysis of the numerical simulation error are shown.

In SOFI2D, only a fourth-order accurate Adams-Bashforth time integrator is implemented, due to the superior performance over third-order.

**Theory.** For the sake of simplicity, we illustrate the staggered ABS method using the 1D acoustic wave equation in velocity-stress formulation

$$\begin{aligned}\frac{\partial p(x, t)}{\partial t} &= -\pi(x) \frac{\partial v(x, t)}{\partial x} , \\ \frac{\partial v(x, t)}{\partial t} &= -\rho^{-1}(x) \frac{\partial p(x, t)}{\partial x} .\end{aligned}\tag{58a}$$

The wavefield variables are the pressure  $p(x, t)$  and the particle velocity  $v(x, t)$ . The material is described by the P-wave modulus  $\pi(x)$  and the mass density  $\rho(x)$ . For simplicity we omit the temporal ( $t$ ) and spatial dependencies ( $x$ ) in the following.

Using the conventional second order staggered grid approximation to the first order time derivative we obtain

$$\frac{p|^{n+1/2} - p|^{n-1/2}}{\Delta t} = -\pi \left. \frac{\partial v}{\partial x} \right|^n + O(\Delta t^2) ,\tag{59a}$$

$$\frac{v|^n - v|^{n-1}}{\Delta t} = -\rho^{-1} \left. \frac{\partial p}{\partial x} \right|^{(n-1/2)} + O(\Delta t^2) .\tag{59b}$$

This results in the conventional explicit second order accurate time-stepping (leapfrog) scheme

$$p|^{n+1/2} = p|^{n-1/2} - \Delta t \pi \left. \frac{\partial v}{\partial x} \right|^n + O(\Delta t^2) ,\tag{60a}$$

$$v|^n = v|^{n-1} - \Delta t \rho^{-1} \left. \frac{\partial p}{\partial x} \right|^{(n-1/2)} + O(\Delta t^2) ,\tag{60b}$$

which requires no additional storage of the wavefield variables  $p$  and  $v$ .

With the ABS-method the order of the temporal integration can be increased by using previous time levels of the right hand sides of Eq. 59b (Ghrist et al., 2000). The ABS-method thus requires the storage of previous time levels of spatial derivatives of the pressure  $p$  and particle velocity  $v$ . The time update for the ABS-method thus reads

$$p|^{n+1/2} = p|^{n-1/2} - \Delta t \pi \sum_{k=0}^{M-1} a_k \left. \frac{\partial v}{\partial x} \right|^{n-k} + O(\Delta t^M) ,\tag{61a}$$

$$v|^n = v|^{n-1} - \Delta t \rho^{-1} \sum_{k=0}^{M-1} a_k \left. \frac{\partial p}{\partial x} \right|^{(n-1/2-k)} + O(\Delta t^M) .\tag{61b}$$

The weights for the time accuracy orders  $M = 2, 3, 4$  are given in Table 1.

**Table 1:** Weights used for the summation of previous time levels in the Adams-Bashforth method (Ghrist et al., 2000).

$M$	$a_0$	$a_1$	$a_2$	$a_3$
2	1	0	0	0
3	25/24	-1/12	1/24	0
4	13/12	-5/24	1/6	-1/24

**Accuracy in 1-D.** We first compare seismograms calculated with the 1D ABS-method using Eq. 61b for different orders of accuracy in time ( $M$ ). We use a 1D homogeneous acoustic medium with a wave velocity of  $c = 3000$  m/s and constant density of  $\rho = 2000$  kg/m<sup>3</sup>. The source signal is a Ricker signal with a center frequency of 60 Hz. We choose a large source-receiver distance of 120 dominant wavelength to emphasize the effects of numerical dispersion and numerical dissipation in the synthetic seismograms. The spatial derivatives are computed with high accuracy using a centered staggered FD stencil of 8th order accuracy. The spatial grid spacing is held constant at  $\Delta x = 0.4$  m corresponding to approximately 14 grid points per dominant wavelength. Discrepancies to the analytical solution (time-shifted Ricker signal) are thus mainly caused by the chosen time step interval  $\Delta t$  and the order of the temporal discretization ( $M = 2, 3, 4$ ). The numerical results for Courant numbers  $r = c\Delta t/\Delta x = 0.4, 0.2, 0.1$  and temporal orders of accuracy  $M = 2, 3, 4$  are compared with the analytical solution in Fig. 6. For a large Courant number of  $r = 0.4$  (corresponding to a large time step interval) and second order approximation of the time derivative ( $M = 2$ ) we can observe a large time dispersion error causing a leading phase with high amplitude (Fig. 6, top-left). Fig. 6 compares two ways to reduce this time discretization error. The conventional way is to stay with the given temporal accuracy order (typically  $M = 2$ ) and then reduce the time step interval, i.e., Courant number. The resulting waveforms are shown in the rows of Fig. 6.

Reducing the Courant number (time step interval) will increase the computation time proportional to  $1/r$ . The alternative way proposed in this study is to increase the order of the temporal discretization with the ABS-method which requires to store  $M - 1$  previously calculated spatial derivatives of wavefields. The resulting waveforms are shown in the columns of Fig. 6.

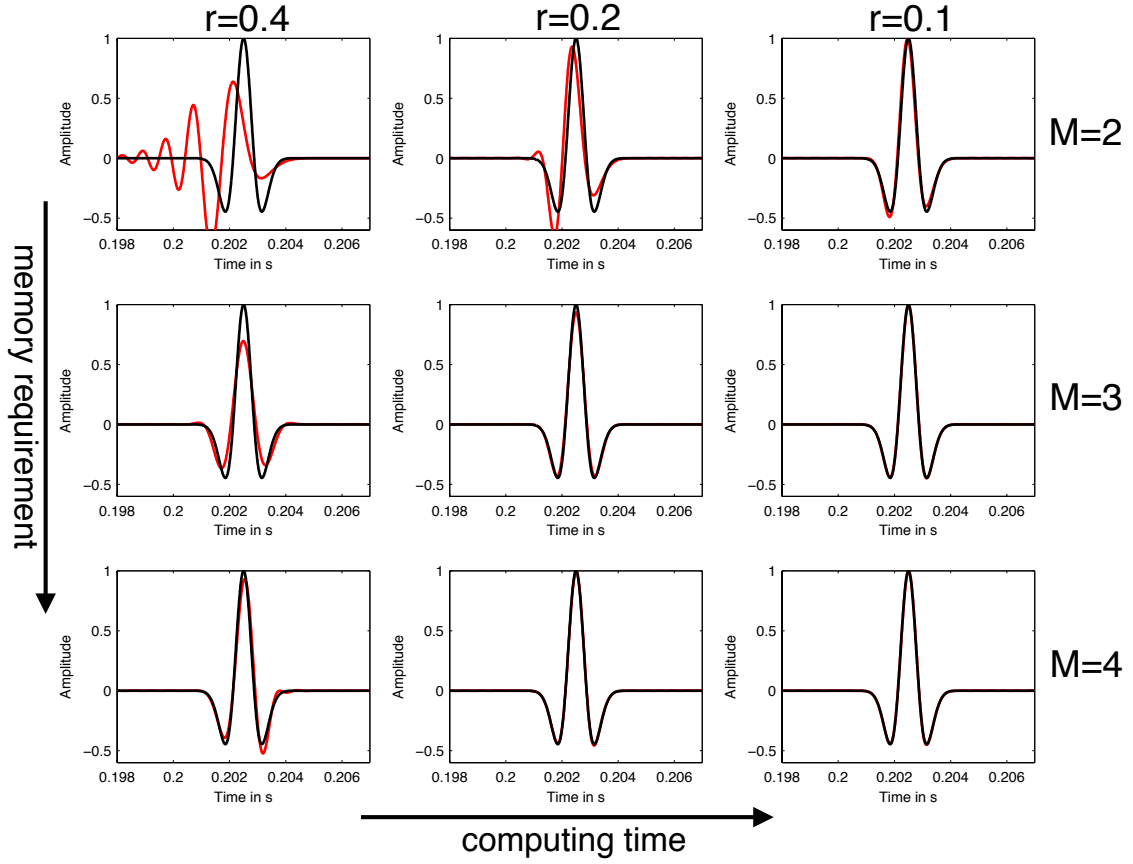
In Fig. 6 we see that the simulation accuracy increases with decreasing Courant number and increasing accuracy order. In order to quantify the corresponding numerical simulation error we calculated the normalized L2-misfit between the numerical and analytical solution for different Courant numbers and accuracy orders  $M = 2, 3, 4$ .

In Fig. 7 we plot the normalized L2-misfit over the number of time steps  $NT = T/\Delta t = Tc/(r\Delta x)$  where the propagation time of waves is held fixed at  $T = 0.24$  s. We can observe the expected higher order reduction of the simulation accuracy which reduces proportional to  $\Delta t^M$  or  $NT^{-M}$ . For a given accuracy level of  $E = 0.1\%$  (horizontal line in Fig. 7) the required numbers of time steps are 39233, 13938, and 8704 for  $M = 2, M = 3$  and  $M = 4$ , respectively. The number of time steps thus reduces to 36% and 22% when we increase the temporal order from  $M = 2$  to  $M = 3$  and  $M = 4$ , respectively (Fig. 7). The ABS methods thus allows to significantly decrease the number of time steps and thus to save computing time.

### 3.4.6 Absorbing boundary conditions

Due to limited computational resources, the FD grid has to be as small as possible. To model problems with an infinite extension in different directions, e.g., a full or half-space problem,





**Figure 6:** Seismograms (red lines) calculated for the 1-D case for different Courant numbers  $r$  and accuracy orders  $M$ .  $M = 2$  corresponds to the classical second order leapfrog scheme (Eq. 59b).  $M = 3, 4$  correspond to the multi-step ABS method (Eq. 61b). The analytical solution is plotted as a black line.

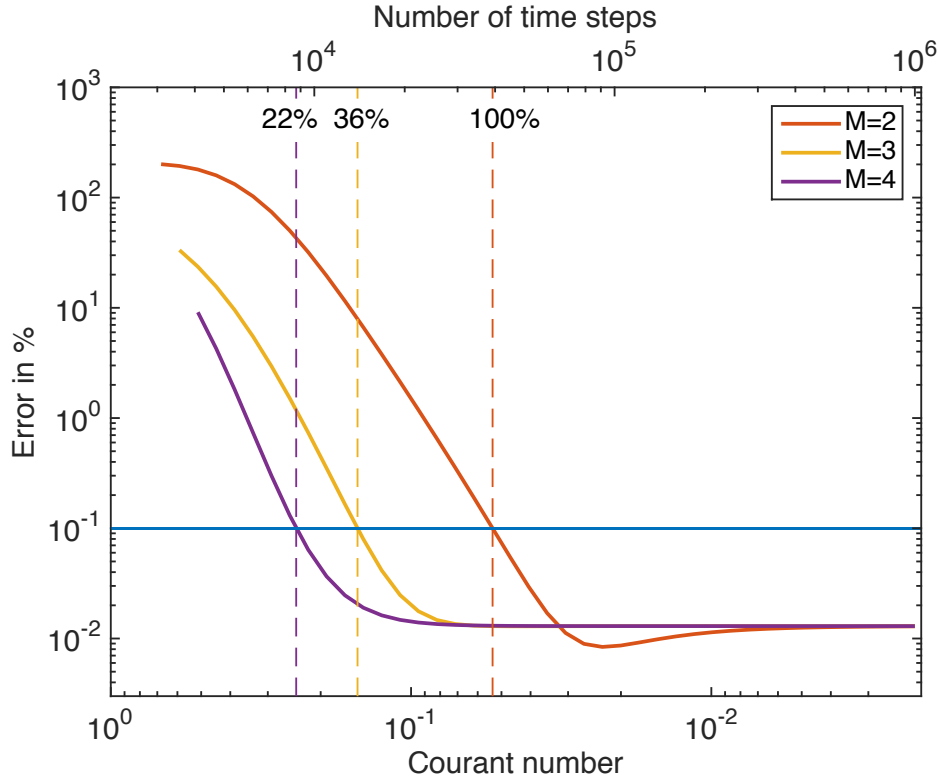
an artificial absorbing boundary condition has to be applied. A very effective way to damp the waves near the boundaries are **perfectly matched layers (PMLs)**. This can be achieved by a coordinate stretch of the wave equations in the frequency domain Komatitsch and Martin, 2007. The coordinate stretch creates exponentially decaying plane wave solutions in the absorbing boundary frame. The PMLs are only reflectionless if the exact wave equation is solved. As soon as the problem is discretized (for example using finite differences) you are solving an approximate wave equation and the analytical perfection of the PML is no longer valid. In SOFI2D, a damping profile  $d_x$  after Collino and Tsogka (2001) is implemented:

$$d_x(x) = d_0 \left( \frac{x}{L} \right)^n, \quad (62)$$

where  $L$  denotes the width of the absorbing frame and  $d_0$  is a function of the theoretical reflection coefficient  $R$ :

$$d_0 = -(n+1) V_{PML} \frac{\log(R)}{2L}, \quad (63)$$

where  $V_{PML}$  denotes the typical P-wave velocity of the medium in the absorbing boundary frame. A reflection coefficient of  $R = 1 \times 10^{-4}$  is assumed. The PMLs are damping the seismic waves by a factor 5-10 more effective than the conventional absorbing boundary frame.



**Figure 7:** Relative error of 1D simulations (Fig. 6). The normalized L2 norm is plotted over the required number of time steps ( $NT$ ). The temporal accuracy orders are  $M = 2, 3, 4$ . The spatial accuracy is fixed at order  $N = 8$ . The number of time steps to achieve a given accuracy of  $E = 0.01\%$  (horizontal line) reduce to 36% and 22% for the temporal orders  $M = 3$  and  $M = 4$ , respectively. For large  $NT$  (small time step intervals) the error converges to the fixed error of the spatial discretization.

### 3.5 Numerical artifacts and instabilities

To avoid numerical artifacts and instabilities during a FD modeling run, a spatial and temporal sampling condition for the wavefield has to be satisfied. These will be discussed in the following two sections.

#### 3.5.1 Grid dispersion

The first question when building a FD model is: What is the maximum spatial grid point distance  $\Delta h$ , for a correct sampling of the wavefield? To answer this question we take a look at this simple example: The particle displacement in  $x$ -direction is defined by a sine function:

$$v_x = \sin\left(2\pi \frac{x}{\lambda}\right), \quad (64)$$

where  $\lambda$  denotes the wavelength. When calculating the derivation of this function analytically at  $x = 0$  and setting  $\lambda = 1$  m we get:

$$\left. \frac{dv_x}{dx} \right|_{x=0} = \frac{2\pi}{\lambda} \cos\left(2\pi \frac{x}{\lambda}\right) \Big|_{x=0} = 2\pi. \quad (65)$$

In the next step the derivation is approximated numerically by a staggered 2nd order finite-difference operator:

$$\left. \frac{dv_x}{dx} \right|_{x=0} \approx \frac{v_x(x + \frac{1}{2}\Delta x) - v_x(x - \frac{1}{2}\Delta x)}{\Delta x} \Big|_{x=0} = \frac{\sin\left(\frac{2\pi(\frac{1}{2}\Delta x)}{\lambda}\right) - \sin\left(\frac{2\pi(\frac{1}{2}\Delta x)}{\lambda}\right)}{\Delta x}. \quad (66)$$

Using the Nyquist-Shannon sampling theorem it should be sufficient to sample the wavefield with  $\Delta x = \lambda/2$ . In Table 2 the numerical solutions (Eq. 66) and the analytical solution (Eq. 65) are compared for different sample intervals  $\Delta x = \lambda/n$ , where  $n$  is the number of gridpoints per wavelength. For the case  $n = 2$ , which corresponds to the Nyquist-Shannon theorem, the numerical solution is  $\left. \frac{dv_x}{dx} \right|_{x=0} = 4.0$ , which is not equal with the analytical solution  $2\pi$ . A refinement of the spatial sampling of the wavefield results in an improvement of the finite difference solution. For  $n = 16$  the numerical solution is accurate to the second decimal place. The effect of a sparsely sampled pressure field is illustrated in Fig. 8 for a homogeneous block model with stress free surfaces. The dimensions of the FD grid are fixed and the central frequency of the source signal is increased systematically. When using a spatial sampling of 16 grid points per minimum wavelength (8, top) the wavefronts are sharply defined. For  $n = 4$  grid points a slight numerical dispersion of the wave occurs (8, center). This effect is obvious when using the Nyquist criterion ( $n = 2$ ) (8, bottom). Since the numerical calculated wavefield seem to be dispersive this numerical artefact is called **grid dispersion**.

To avoid the occurrence of grid dispersion the following criteria for the spatial grid spacing  $\Delta h$  has to be satisfied:

$$\Delta h \leq \frac{\lambda_{min}}{n} = \frac{V_{min}}{n f_{max}}, \quad (67)$$

where  $\lambda_{min}$  denotes the minimum wavelength,  $V_{min}$  the minimum velocity in the model, and  $f_{max}$  is the maximum frequency of the source signal.

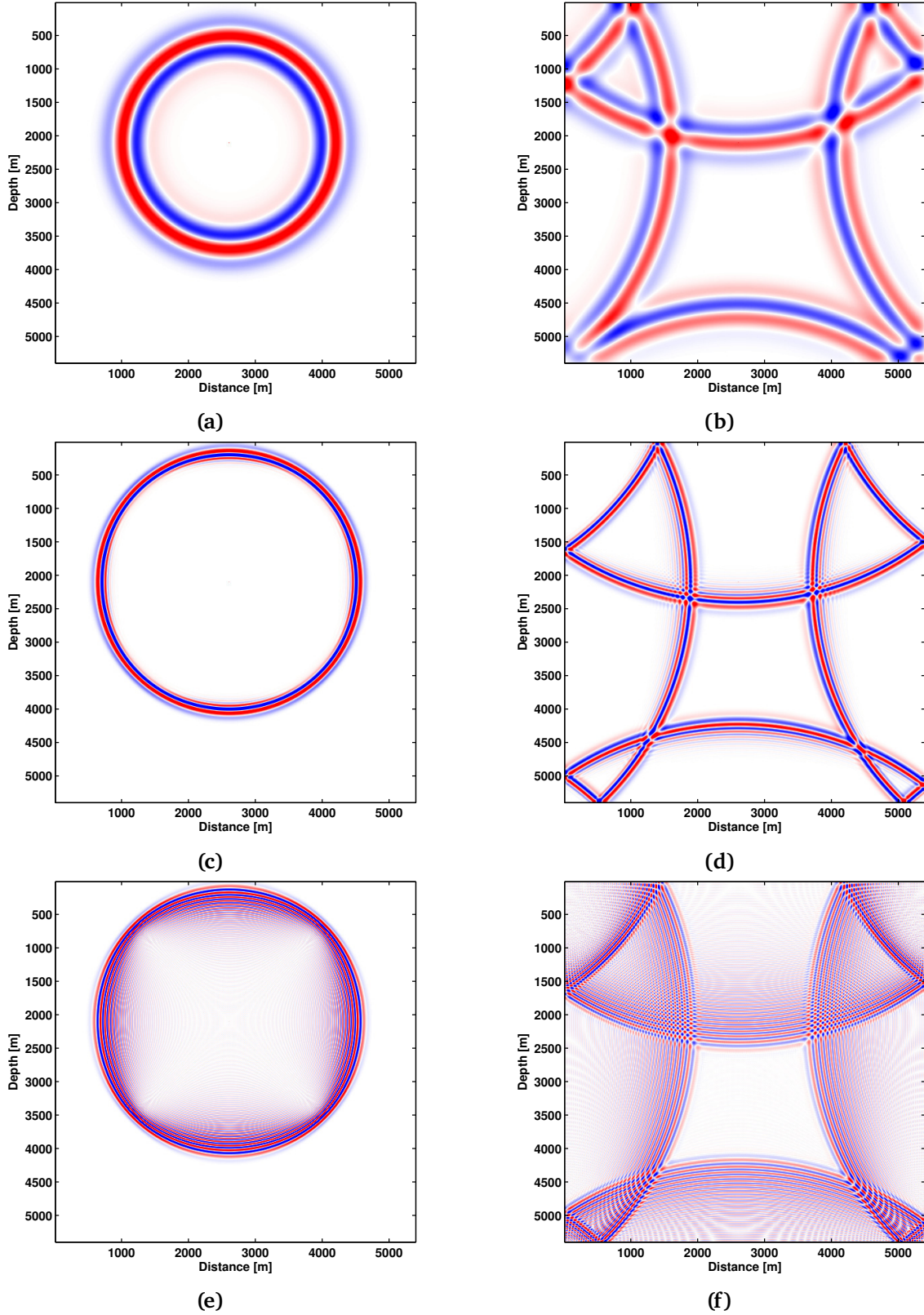
Depending on the accuracy of the used FD operator, the parameter  $n$  is different. In Table 3 the parameter  $n$  is listed for different FD operator lengths and types (Taylor and Holberg operators). The Holberg coefficients are calculated for a minimum dispersion error of 0.1% at  $3f_{max}$ . For short operators  $n$  should be chosen relatively large, so the spatial grid spacing is small, while for longer FD operators  $n$  is smaller and the grid spacing can be larger.

**Table 2:** Comparison of the analytical solution Eq. 65 with the numerical solution Eq. 66 for different grid spacing  $\Delta x = \lambda/n$ .

<b>n</b>	<b><math>\Delta x</math> [m]</b>	<b><math>\left. \frac{dv_x}{dx} \right _{x=0}</math></b>
analytical	-	$2\pi \approx 6.283$
2	$\lambda/2$	4.0
4	$\lambda/4$	5.657
8	$\lambda/8$	6.123
16	$\lambda/16$	6.2429
32	$\lambda/32$	6.2731

### 3.5.2 The Courant instability

Beside the spatial, the temporal discretization has to satisfy a sampling criterion to ensure the stability of the FD code. If a wave is propagating on a discrete grid, then the timestep  $\Delta t$  has to be less than the time for the wave to travel between two adjacent grid points with



**Figure 8:** The influence of grid dispersion in FD modeling: Spatial sampling of the wavefield using  $n = 16$  (top),  $n = 4$  (center) and  $n = 2$  gridpoints (bottom) per minimum wavelength.

grid spacing  $\Delta h$ . For an elastic 2D grid this means mathematically:

$$\Delta t \leq \frac{\Delta h}{h\sqrt{2}V_{max}}, \quad (68)$$

**Table 3:** The number of grid points per minimum wavelength  $n$  for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators. For the Holberg coefficients  $n$  is calculated for a minimum dispersion error of 0.1% at  $3 f_{max}$ .

FD-order	n (Taylor)	n (Holberg)
2nd	12	12
4th	8	8.32
6th	6	4.77
8th	5	3.69
10th	5	3.19
12th	4	2.91

where  $V_{max}$  is the maximum velocity in the model. The factor  $h$  depends on the order of the FD operator and can easily calculated by summing over the weighting coefficients  $\beta_i$ :  $h = \sum_i \beta_i$ . In Table 4 the parameter  $h$  is listed for different FD operator lengths and types (Taylor and Holberg operators). Criterion Eq. 68 is called **Courant-Friedrichs-Lewy criterion** (Courant et al., 1928, Courant et al., 1967). Fig. 9 shows the evolution of the pressure field when the Courant criterion is violated. After a few time steps the amplitudes are growing to infinity and the calculation becomes unstable.

**Table 4:** The factor  $h$  in the Courant criterion for different orders (2nd-12th) and types (Taylor and Holberg) of FD operators.

FD-order	h (Taylor)	h (Holberg)
2nd	1.0	1.0
4th	7/6	1.184614
6th	149/120	1.283482
8th	2161/1680	1.345927
10th	53089/40320	1.387660
12th	1187803/887040	1.417065

## 4 Definition of the modeling parameters

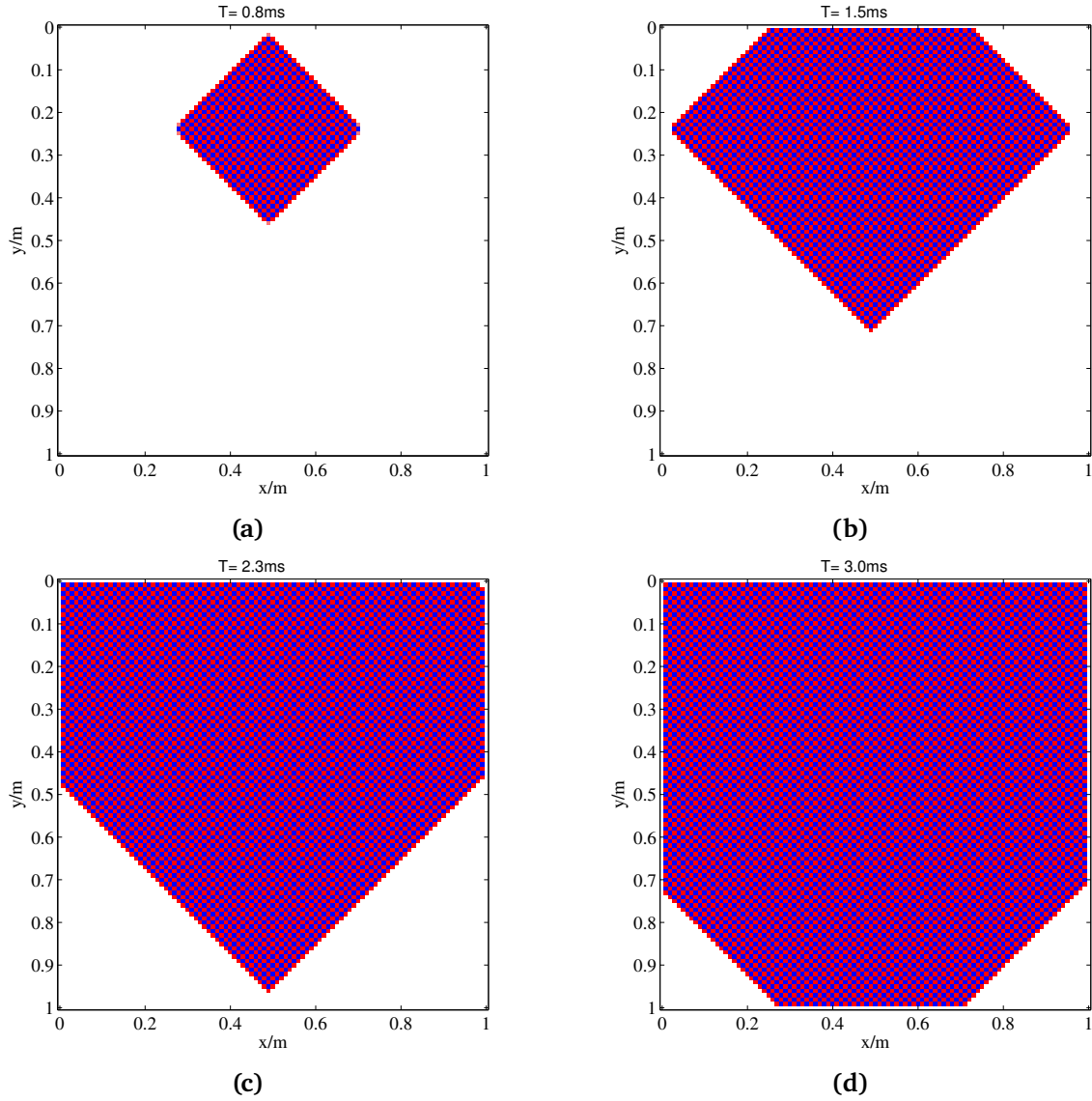
The geometry of the FD grid and all parameters for the wavefield simulation have to be defined in a parameter file (which we name in this case `sofi2D.json`). In the following we will explain all input parameters in detail. All lines in the parameter file are formatted according to the JSON standard (<http://www.json.org>) and organized as

```
"VARNAME" = "Parameter value",
```

where a comment line can look like this:

```
"Comment" = "This is a useful comment",
"3D Grid information" = "comment",
```

Here, VARNAME is a symbolic name for the actual different parameters shown below. Basically, all non-JSON conforming lines will be ignored. The order of the various parameters does not matter, but we typically group parameters that belong together. If critical parameters are missing, the code will stop and an error message will appear. The program outputs the read and all updated/adjusted and possibly calculated parameters unless such output is suppressed.



**Figure 9:** Temporal evolution of the Courant instability. In the colored areas the wave amplitudes are extremely large.

#### 4.1 Domain decomposition

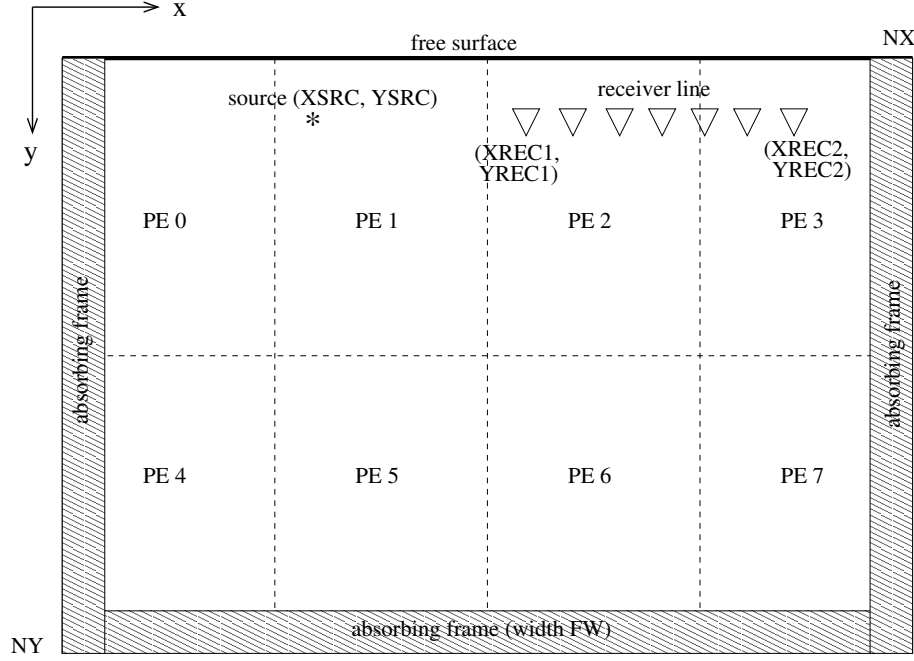
```
"Domain Decomposition" : "comment",
    "NPROCX" : "4",
    "NPROCY" : "2",
```

with

NPROCX : number of (MPI) processors in x-direction

NPROCY : number of (MPI) processors in y-direction

Parallelization is based on domain decomposition (see Fig. 10), i.e., each processing element (PE) updates the wavefield within its portion of the grid. The model is decomposed by the program into subgrids (blocks). After decomposition, each processing elements (PE) saves only its sub-volume of the grid. NPROCX and NPROCY specify the number of (MPI) processors in horizontal and vertical direction, respectively (Fig. 10). Thus, the total number of processors is  $NP = NPROCX * NPROCY$ ; this values must be specified when starting the



**Figure 10:** Geometry of the numerical FD grid using 4 processors in the  $x$ -direction ( $NPROCX=4$ ) and 2 processors in the  $y$ -direction ( $NPROCY=2$ ). At the top of the numerical mesh, the PEs apply a free surface boundary condition if `FREE_SURF=1`, otherwise an absorbing boundary condition is used; the width of the absorbing frame is  $(FW-1)*DH$  in meters. The size of the total grid is  $NX$  grid points in the  $x$ -direction and  $NY$  gridpoints in the  $y$ -direction. The origin of the Cartesian coordinate system is at the top left corner of the grid, i.e., the first upper-left grid point corresponds to coordinate  $x = 0$ ,  $y = 0$ .

program with the following command: `mpirun -np <NP> ../bin/sofi2D sofi2D.json`. If the total number of processors in `sofi2D.json` and the number of MPI processes differ, the program will terminate immediately with a corresponding error message. Obviously, the total number of PEs ( $NPROCX*NPROCY$ ) used to decompose the model should be less or equal the total number of CPU cores available on your parallel processing machines or cluster. You can oversubscribe compute nodes but this usually hurts performance. In order to reduce the amount of data transferred between PEs, one should try to realize more or less cubic subgrids. In our example, we use 4 PEs in horizontal direction ( $NPROCX=4$ ) and 2 PEs in vertical direction ( $NPROCY=2$ ). The total number of PEs used by the program, and hence the number of MPI processes, is  $NP=NPROCX*NPROCY=8$ .

## 4.2 Order of the FD operator

```
"FD order" : "comment",
  "FDORDER" : "4",
  "FDORDER_TIME" : "2",
  "MAXRELError" : "1",
```

with

`FDORDER` : order of the spatial FD operator (2, 4, 6, 8, 10 or 12)

`FDORDER_TIME` : order of the temporal FD operator (2 or 4)

`MAXRELError` : Maximum relative group velocity error (Taylor=0; Holberg=1)

`FDORDER_TIME=2` corresponds to the classical leapfrog scheme and `FDORDER_TIME=4` to a fourth order accurate scheme. By the option `MAXRELError`, you can switch between Taylor

and Holberg FD coefficients but also define a maximum of tolerated group velocity error in percent (Taylor coefficients=0, Holberg coefficients: 1 for  $E = 0.1\%$ , 2 for  $E = 0.5\%$ , 3 for  $E = 1.0\%$ , and 4 for  $E = 3.0\%$ ). The chosen FD operator and FD coefficients have an influence on the numerical stability and grid dispersion (see next section).

### 4.3 Space discretization

```
"2-D Grid" : "comment",
      "NX"  : "400",
      "NY"  : "400",
      "DH"  : "2.0",
```

with

NX : number of grid points in the x-direction

NY : number of grid points in the y-direction

DH : distance between grid points in both spatial directions (grid size in meters)

These lines specify the size of the total grid (Fig. 10) and therefore the size of external models. An equidistant grid is used in the FD software. The size of the total grid in  $x$ -direction is  $(NX-1)*DH$  and in  $y$ -direction  $(NY-1)*DH$ . **Note that the  $y$ -direction always corresponds to the vertical direction** (i.e., to depth).  $NX/NPROCX$  and  $NY/NPROCY$  must be integer values (for the time being, i.e., subdomains must have equal sizes – this will be changed in a future version of the program).

To avoid numerical dispersion, the wavefield must be discretized with a certain number of grid points per wavelength. The number of grid points per wavelength depends on the order of the spatial FD operators used in the simulation. The criterion to avoid numerical dispersion is shown in eq. 67; the program assumes that the maximum frequency of the source signal is approximately two times the center frequency. To avoid numerical dispersion of body waves, 8 grid points per minimum wavelength ( $N = 8$ ) are necessary for a fourth order algorithm and  $N = 12$  grid points for a second order algorithm (Table 3). The dispersion criterion is checked by the FD software. See the SOFI2D log for further information or warnings. Please note that the FD code will not terminate due to grid dispersion.

### 4.4 Time stepping

```
"Time Stepping" : "comment",
      "TIME" : "0.5",
      "DT"  : "1.0e-4",
```

with

TIME : propagation time of seismic waves in the entire model (in seconds)

DT : time stepping interval (in seconds)

The time stepping interval  $DT$  has to fulfill the stability criterion in eq. 68, which is checked for the entire model. If the criterion is violated, the FD program will terminate with a corresponding error message. The maximum value for  $DT$  at the stability limit is output, so you may simply try a certain  $DT$ , start the program, and then check what the program suggests as time step interval.

### 4.5 Wave equations

The parameter `WEQ` specifies the type of wave equation to use for modelling. Example:



```
"Wave Equation" : "comment",
    "WEQ" : "VEL_TTI",
```

This parameter is given as string and can have the following values:

- **AC\_ISO**: acoustic isotropic wave equation
- **VAC\_ISO**: viscoacoustic isotropic wave equation
- **EL\_ISO**: elastic isotropic wave equation
- **EL\_VTI**: elastic VTI wave equation
- **EL\_TTI**: elastic TTI wave equation
- **VEL\_ISO**: viscoelastic isotropic wave equation
- **VEL\_VTI**: viscoelastic VTI wave equation
- **VEL\_TTI**: viscoelastic TTI wave equation

The type of wave equation specified for WEQ determines which model files are read from disk and whether Q-related parameters are required, etc. Note that at the time of writing this manual not all wave equations have been implemented yet.

## 4.6 Sources

```
"Source" : "comment",
    "SOURCE_SHAPE" : "1",
    "SIGNAL_FILE" : "signal_mseis.tz",
    "SIGOUT" : "0",
    "SIGOUT_FILE" : "./sources/signal_out01",
    "SIGOUT_FORMAT" : "1",
    "SOURCE_TYPE" : "3",
    "SOURCE_TOPO" : "0",
    "SRCREC" : "1",
    "SOURCE_FILE" : "./sources/source.dat",
    "RUN_MULTIPLE_SHOTS" : "0",
    "PLANE_WAVE_DEPTH" : "2106.0",
    "PLANE_WAVE_ANGLE" : "0.0",
    "TS" : "0.2",
```

with

SOURCE\_SHAPE : shape of source-signal (Ricker=1; Fuchs-Müller wavelet=2; from SIGNAL\_FILE=3;  $\sin^3$ =4; Berlage=5; Klauder=6)

SIGNAL\_FILE : external signal file name

TS : duration of the source-signal (in seconds)

Five built-in wavelets for the seismic source are available. The corresponding time functions are defined in `src/wavelet.c`. You may modify the time functions in this file and recompile to include your own analytical wavelet or to modify the shape of the built-in wavelets.

**Ricker wavelet:**

$$A_r(\tau) = \left(1 - 2\tau^2\right) \exp(-\tau^2) \quad \text{with} \quad \tau = \frac{\pi(t - 1.5/f_c - t_d)}{1.0/f_c} . \quad (69)$$

**Fuchs-Müller wavelet:**

$$A_{fm}(t) = \begin{cases} \sin(2\pi(t - t_d)f_c) - 0.5 \sin(4\pi(t - t_d)f_c), & t \in [t_d, t_d + 1/f_c] \\ 0, & \text{otherwise} \end{cases} . \quad (70)$$

**$\sin^3$  wavelet:**

$$A_s(t) = \begin{cases} \frac{3}{4}\pi f_c \sin(\pi(t + t_d)f_c)^3, & t \in [t_d, t_d + 1/f_c] \\ 0, & \text{otherwise} \end{cases} . \quad (71)$$

**Berlage wavelet** (after Aldridge, 1990):

$$A_b(t) = (t - t_d)^n e^{-\alpha(t-t_d)} \cos(2\pi f_c(t - t_d) + \phi_0) . \quad (72)$$

**Klauder wavelet** (after Neelima et al., 2018):

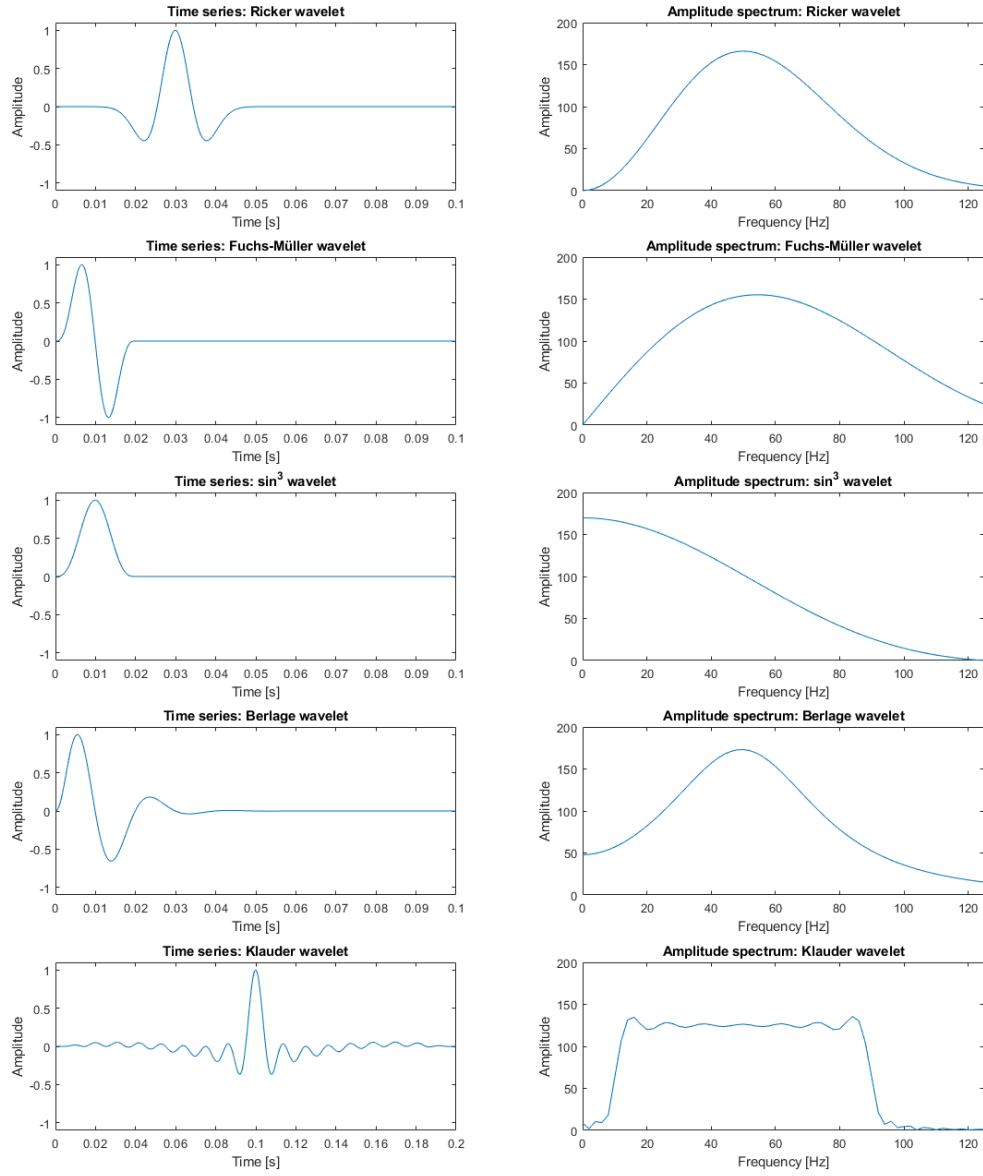
$$A_k(\tau) = \Re \left\{ \frac{\sin(\pi k(\tau - t_d)(t_{\text{sweep}} - \tau + t_d))}{\pi k(\tau - t_d)e^{2\pi i f_c(\tau - t_d)}} \right\} \quad (73)$$

with  $k = \frac{f_{\text{max}} - f_{\text{min}}}{t_{\text{sweep}}}$ ,  $\tau = t - \frac{n_{\text{twidth}}}{f_c}$  and  $f_c = \frac{f_{\text{max}} + f_{\text{min}}}{2}$ .

In these equations  $t$  denotes time and  $f_c$  is the center frequency. It is specified in the SOURCE\_FILE either directly, or for the Klauder wavelet calculated as half of  $f_{\text{max}} + f_{\text{min}}$ . If parameters are not provided through an external SOURCE\_FILE (SRCREC=2), the center frequency is derived through  $f_c = 1/TS$ .  $t_d$  is a time delay which can be defined for each source position in SOURCE\_FILE. Variable time delays for different sources locations allow the simulation of passive acoustic emission. Note that the symmetric (zero-phase) Ricker and Klauder signals are always delayed by  $1.5/f_c$  for Ricker, and a shift of  $n_{\text{twidth}}/f_c$  for Klauder, where  $n_{\text{twidth}}$  is half the width of the wavelet in center periods. The Klauder wavelet is tapered on the first and last 20% with a cosine taper function. This means that for these two cases the maximum amplitude is excited at the source location after one and a half or  $n_{\text{twidth}}$  centre periods, respectively. All source wavelets and the corresponding amplitude spectra for a center frequency of  $f_c = 50$  Hz and a delay of  $t_d = 0$  are plotted in Fig. 11. Note the delay of the Ricker and Klauder signal described above. The Fuchs-Müller wavelet has a slightly higher center frequency and covers a broader frequency range. The Klauder wavelet covers a frequency range of 10 Hz to 90 Hz and uses a sweep duration of 7 s and  $n_{\text{twidth}}$  of 5 center periods. For the Berlage wavelet, the additional parameters were set to  $n = 1.5$ ,  $\alpha = 210$ , and  $\phi_0 = -90^\circ$ .

You may also use your own digitized time function as the source wavelet (for instance the signal of the first arrival recorded by a geophone at near offsets). Specify SOURCE\_SHAPE=3 and save the samples of your source wavelet in ASCII-format in SIGNAL\_FILE; this file should have a file suffix .dat, .txt or .asc. SIGNAL\_FILE should contain one sample (amplitude value) per line and have the following form:

```
# DT=0.0001
0.0
0.01
0.03
...
```



**Figure 11:** Plot of built-in source wavelets for a center frequency of  $f_c = 50$  Hz ( $TS = 1/f_c = 0.02$  s): Ricker signal (1<sup>st</sup> row), Fuchs-Müller signal (2<sup>nd</sup> row),  $\sin^3$ -signal (3<sup>rd</sup> row), Berlage signal (4<sup>th</sup> row), Klauder signal (5<sup>th</sup> row). Time function (left column) and amplitude spectrum (right column).

The time interval between the samples must equal the time step interval (DT) of the FD simulation (see above)! It is therefore often necessary to resample/interpolate a given source time function with a smaller sample rate.

The ASCII file may contain a comment line in the beginning where the sampling interval can be specified as DT=<value> with the value being in seconds. This allows the program to cross-check that the FD time step interval matches the signature time sampling interval and the user does not accidentally modify DT in the parameter file without also updating the sampling interval of the source signature. If the aforementioned comment line is missing in the ASCII file, no cross-check is performed. In case of a mismatch, the program will abort.

The source signature can also be provided as SU-formatted file containing once trace (additional traces will be ignored). In this case, SIGNAL\_FILE should have the file suffix .su. Just like models in SU format, this file needs to use 32-bit native floats. The SU header delrt must be zero (signatures starting at non-zero times are not supported). The SU header dt (in microseconds), if given and greater than 0, is converted to seconds and then cross-checked against the FD time step interval. In case of a mismatch, the program will abort. Note that there is never an automatic interpolation performed by the program (neither when reading ASCII nor when reading SU files).

SIGOUT : output source wavelet (yes=1; no=0)

SIGOUT\_FILE : external signal file name for output

SIGOUT\_FORMAT : supported output formats (SU=1; ASCII=2; BINARY=3)

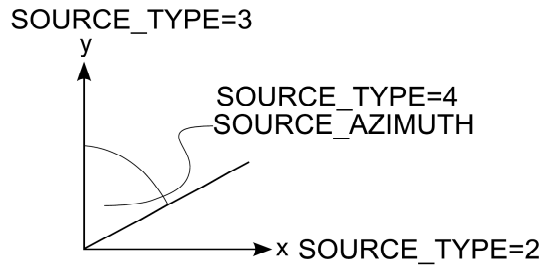
The wavelet can be output for quality control purpose by setting SIGOUT to 1 (default value is 0, i.e., no output). This works for both internally generated wavelets based on the parameters provided in SOURCE\_FILE and for externally provided wavelets read from SIGNAL\_FILE. In case QC output is requested, a basename for the output file has to be provided (SIGOUT\_FILE) and an output format (SIGOUT\_FORMAT).

SOURCE\_TYPE : source type (explosive=1; force in x-direction=2; force in y-direction=3; custom force=4)

SOURCE\_TYPE=1 is equivalent to an explosive source that excites P-waves only radiating with the same amplitude into all directions. SOURCE\_TYPE=2 and SOURCE\_TYPE=3 simulate point forces in the x- and y-directions, respectively. With SOURCE\_TYPE=4, a source in a user-defined direction can be chosen which is defined by the parameter SOURCE\_AZIMUTH specified in the SOURCE\_FILE (see Fig. 12). These sources have specific radiation characteristics and excite both P- and S-waves. In case of an explosive source the diagonal components of the stress tensor are assigned with the source wavelet. In case of a directive force in the x- or y-directions, respectively, the corresponding external body force component ( $f_i$ , see Eq. 11 of (Bohlen, 2002)) is assigned with the source wavelet amplitudes. Please note, that due to the 2D FD scheme, a point force is physically described as a line source. The excited wavefield is therefore not directly comparable to a 3D wave propagation. For 1D media, there is a 2D-to-3D conversion of both body and surface waves.

SOURCE\_TOPO : place sources relative to topography (=1) or at absolute depth (=0)

The SOURCE\_TOPO option is only relevant in case sources are read from a file (SRCREC=1). Under normal circumstances, i.e., when SOURCE\_TOPO=0 which is also the default, sources are placed at the absolute depth specified in the external source file given by SOURCE\_FILE. However, sometimes an explicit air layer is part of the model in case the Earth has topography. When SOURCE\_TOPO=1, then sources are not placed at an absolute depth value but the depth specified in the external source file becomes a depth below topography. The topography is determined by opening the P-wave velocity model and scanning each trace from top



**Figure 12:** Scheme for `SOURCE_TYPE=4`. The parameter `SOURCE_AZIMUTH` denotes the angle between the vertical  $y$ - and horizontal  $x$ -coordinate.

to bottom until a change of the first P-wave velocity value that was read in is found (this usually is the interface between air and the solid Earth). The actual P-wave velocity value of the first (air) layer itself does not matter, we only scan for a change. Example: if you would like to simulate sources in 12 m deep shot holes along a free surface with topographic variations, you need an explicit air layer in the model, you need to set the depth of the source (`YSRC`, see below) to "12.0" in the external source file, and you finally need to switch on the `SOURCE_TOPO` option.

`SRCREC` : read source positions (from external `SOURCE_FILE=1`; from internal `PLANE_WAVE=2`)

`SOURCE_FILE` : external source file name

A source file should have the following format:

<code>XSRC</code>	<code>YSRC</code>	<code>TD</code>	<code>FC</code>	<code>AMP</code>	<code>SOURCE_AZIMUTH</code>	<code>SOURCE_TYPE</code>
-------------------	-------------------	-----------------	-----------------	------------------	-----------------------------	--------------------------

where

`XSRC` :  $x$ -coordinate of a source point (in meters)

`YSRC` :  $y$ -coordinate (depth) of a source point (in meters)

`TD` : time-delay for the source point (in seconds)

`FC` : center frequency of the source signal (in Hz)

`AMP` : maximum amplitude of the source signal

Optional parameters:

`SOURCE_AZIMUTH` : If `SOURCE_TYPE=4`, it represents the angle in degrees between the  $y$ - and  $x$ -directions; a `SOURCE_AZIMUTH=0` corresponds to the case `SOURCE_TYPE=3` and `SOURCE_AZIMUTH=90` corresponds to `SOURCE_TYPE=2` (please note that there is a numerical inaccuracy between `SOURCE_TYPE=2` and its analog version `SOURCE_AZIMUTH=90`)

`SOURCE_TYPE` : If `SOURCE_TYPE` is set here, the value of `SOURCE_TYPE` in the input file is ignored

`RUN_MULTIPLE_SHOTS` : multiple shots (modeled simultaneously=0; modeled individually=1)

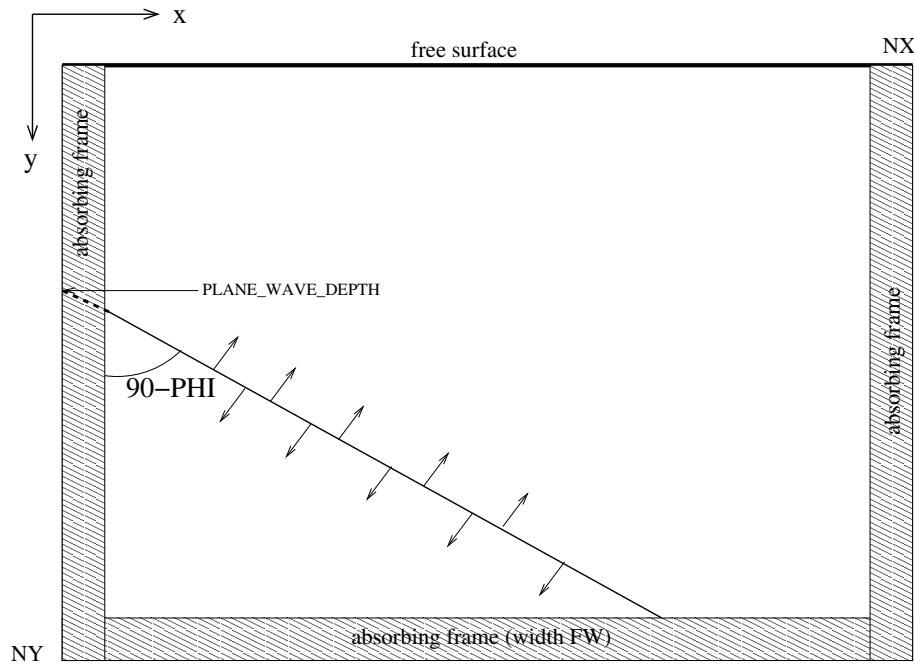
This parameter defines if multiple shots are modeled simultaneously or whether each shot is modeled individually. If the parameter is set to 0, multiple sources are modeled together. If more than one source is defined in `SOURCE_FILE`, each source is triggered at the same time. For `RUN_MULTIPLE_SHOTS=1`, every source specified in the `SOURCE_FILE` is modeled individually. The `SEIS_FILE` name then contains the number of the shot, e.g., `test_vx.su.shot1.0`.

`PLANE_WAVE_DEPTH` : Depth of plane wave excitation (in meters)

`PLANE_WAVE_ANGLE` : Dip of plane wave from vertical (in degrees)

In some applications, e.g., transmission experiments or the simulation of teleseismic events,

the generation of plane waves are required. If you specify a `PLANE_WAVE_DEPTH>0`, a plain wave at a depth of `PLANE_WAVE_DEPTH` is excited. In the case of `PLANE_WAVE_DEPTH<0`, point sources are applied; thus, `SRCREC` should be set to 1, otherwise no sources are applied. The plane wave is simply generated by applying the source time function (`SOURCE_SHAPE`) with the source characteristic (`SOURCE_TYPE`) at each grid point along a straight line (2D modeling). You can also excite plane waves with a certain dip from the vertical direction. See Fig. 13 for a description of the geometry for the generation of dipping plane waves. In case of plane waves, the duration of the source signal, and hence the center frequency of the source ( $1/TS$ ), is specified by  $TS$  in the parameter file. In case of point sources (`PLANE_WAVE_DEPTH=0.0` and `SRCREC=1`) the center frequency is defined in `SOURCE_FILE` together with the location and the time delay. In case of point sources, the parameter  $TS$  is ignored.



**Figure 13:** Plane waves are generated by assigned source points along straight lines (2D simulation) or planar planes (3D simulation) with the source wavelet. The parameter `PLANE_WAVE_DEPTH` defines the depth of this line/plane at  $x = 0$ , and the parameter `PLANE_WAVE_ANGLE` specifies the incidence angle with respect to the vertical direction (`PLANE_WAVE_ANGLE=0` corresponds to vertical incidence).

## 4.7 Generation of models

```
"Model" : "comment",
          "READMOD" : "1",
          "MFILE" : "model/test/TTI_Thomsen_1",
          "WRITE_MODELFILES" : "0",
```

with

`READMOD` : read model parameters from `MFILE` (yes=1; generated “on the fly”=0)

`MFILE` : basename for the model files, used to read from file and/or to write model for validation

`WRITE_MODELFILES` : switch on which models are written to file if generated internally “on the fly” (no model=0; all models=1; only the density model=2).

If READMOD=1, the P-wave, S-wave, and density model grid (if  $L = 0$ , see Chapter 4.8) and additionally the attenuation grids (if  $L > 0$ ) are read from external files. These files can either be in plain binary format, or SU format – if SU-formatted files (suffix .su) are present, they will be preferred. Note that the upper left sample of the model (i.e., the first sample on the first trace) is considered to be at Cartesian coordinate (0, 0). MFILE defines the basic file name that is expanded by the following extensions: **P-wave velocity model** “.vp.su” (or “.vp”), **S-wave velocity model** “.vs.su” (or “.vs”), both in m/s, **density model** “.rho.su” (or “.rho”) in  $\text{kg/m}^3$ , **P-wave attenuation model** “.qp.su” (or “.qp”), **S-wave attenuation model** “.qs.su” (or “.qs”). In case of anisotropy, the program will also read the **Thomsen parameter models** “.epsilon.su” (or “.epsilon”) and “.delta.su” (or “.delta”), and in case of TTI the tilt angle “.theta.su” (or “.theta”). The angle is defined as angle in degree against the vertical axis, with positive angles being measured anti-clockwise. A constant zero-degree tilt angle section for TTI corresponds to the VTI case.

In plain binary files, each material parameter value must be saved as 32-bit (4-byte) native float. The fast dimension is the  $y$ -direction (see, e.g., src/readmod\*.c). The number of samples for the entire model in the  $x$ -direction is  $NX$  and the number of values in the  $y$ -direction is always  $NY$ . The file size of each model file thus must be  $NX * NY * 4$  bytes. You may check the model structure using the SU command **ximage**: `ximage n1=<NY> n2=<NX>` < model/test.vp.

For SU-formatted model files, the set-up is similar. They must use 32-bit native floats. The number of samples per trace must equal  $NY$ , and the number of traces must equal  $NX$ . Note: if there are more than  $NY$  samples per trace in the SU file, the additional samples per trace will be ignored. That means, the code will only ever read  $NY$  samples. The same holds if there are more than  $NX$  traces in the file – the code will only ever read  $NX$  traces (starting from the beginning of the file) as specified in the parameter file. In case the entire file is used, it should have size  $NX * NY * 4 + NX * 240$  bytes (as the SU tracer header has a length of 240 bytes).

If READMOD=0, the model is generated “on the fly” by SOFI2D, i.e., it is generated internally before the time loop starts. Note that if READMOD=0, the function building the model is called from within the software and must therefore be specified in the Makefile; you need to re-compile SOFI2D by changing to the build directory and typing `make all` in case you would like to modify and/or update the internal model generation.

As the variable MFILE is also used to write the internally created model(s), you can specify which models are output (WRITE\_MODELFILES). Be aware that the output of additional models besides density will cause extra (but temporal) memory allocation of the size of the local subgrid times the number of models!

## 4.8 Q-approximation

```
"Q-approximation" : "comment",
    "L" : "1",
    "F_REF" : "50.0",
    "FL1" : "50.0",
    "TAU" : "0.1",
```

with

L : number of relaxation mechanisms (elastic=0; viscoelastic>0; up to 5)

F\_REF : reference frequency

FL1 : relaxation frequencies (one for each relaxation mechanism; FL2, . . . , FL5)

TAU : ratio of strain-retardation and stress-relaxation times (describes the directional variation of attenuation); only used for internal model generation

These lines are omitted in the elastic version (if  $L=0$ ). The frequency dependence of the (intrinsic) quality factor  $Q(\omega)$  is defined by the  $L$  relaxation frequencies ( $FL1 = f_l = 2\pi/\tau_{ol}$ ). In the current model (see `model.c`), these values are assigned to all gridpoints for both P- and S-waves. Thus, intrinsic attenuation is homogeneous and equal for P- and S-waves ( $Q_p(\omega) = Q_s(\omega)$ ). However, it is possible to simulate any spatial distribution of absorption by assigning the gridpoints with different  $\tau$ -values. Note that for a single relaxation mechanism ( $L=1$ ),  $Q \approx 2/\tau$  (Bohlen, 2002;  $Q(\omega)$  for  $L=1$ ,  $FL1 = f_l = 70$  Hz, and  $TAU = \tau = 0.04$  is shown in Fig. 11). **The Matlab script `/mfiles/qplot.m` can be used to plot  $Q(\omega)$  for different values of  $L$ ,  $FL1$ , and  $TAU$ .** The script `qapprox.m` finds optimal values in a way that they fit a desired function  $Q(\omega) = \text{const}$  in a least-squares sense.

Please note, that due to dispersive media properties the viscoelastic velocity model is only defined for the reference frequency only. In SOFI2D, this reference frequency is specified as the center source frequency. At the exact reference frequency, elastic and viscoelastic models are equivalent. As a consequence, slightly smaller and larger minimum and maximum velocity values occur in the viscoelastic model.

#### 4.9 Boundary conditions

```
"Boundary Conditions" : "comment",
    "FREE_SURF" : "0",
    "BOUNDARY" : "0",
    "FW" : "20",
    "ABS_TYPE" : "2",
    "NPOWER" : "4.0",
    "K_MAX_CPML" : "1.0",
    "VPPML" : "3500.0",
    "FPML" : "15.0",
    "DAMPING" : "8.0",
```

with

FREE\_SURF : free surface at the top of the model (no=0; yes=1)

BOUNDARY : periodic boundary condition at edges (no=0; left and right=1)

FW : width of absorbing frame in grid points (no=0; exponential damping applied at the left/right, front/back, and bottom of the grid > 0 - Cerjan et al., 1985)

ABS\_TYPE: type of absorbing boundary (CPML=1; damping=2)

NPOWER : exponent for calculation of damping profile

K\_MAX\_CPML :

VPPML : attenuation velocity within the PML, approximately the propagation velocity of the dominant wave near the model boundaries (in m/s)

FPML : dominant signal frequency (in Hz)

In some cases, it is useful to apply periodic boundary conditions, for example when modeling seismic wave transmission through random media (Bohlen, 2002). If BOUNDARY=1 no absorbing boundaries are installed at the left and right sides of the grid. Instead, wavefield information is copied from left to right and vice versa. The effect is, for example, that a wave which leaves the model at the left side enters the model again at the right side.

The convolutional perfectly matched layer (CPML) boundary condition implementation is based on Komatitsch and Martin (2007) and Martin and Komatitsch (2009). A width of the absorbing frame of FW=10-20 grid points should be sufficient. For the optimal realization



of the PML boundary condition, you have to specify the dominant signal frequency FPML occurring during the wave simulation; this is usually the center source frequency FC specified in the source file.

DAMPING : attenuation at the edges of the grid (in %)

With the DAMPING boundary, attenuation is specified at the edges of the numerical grid, i.e., amplitudes are multiplied by a factor of  $1 - \text{DAMPING}$  at the edges. The width of the absorbing frame should be  $\text{FW} > 20$  gridpoints (Cerjan et al., 1985); a good choice is 8.0%; for larger values, reflections at the onset of the absorbing frame might occur (Cerjan et al., 1985). In order to avoid such an impedance contrast between the model domain and the absorbing boundary layer, FW should increase and DAMPING should decrease. If  $\text{FREE\_SURF} = 0$ , damping is also applied at the top of the model.

#### 4.10 Wavefield snapshots

```
"Snapshots" : "comment",
  "SNAP" : "1",
  "TSNAP1" : "0.01",
  "TSNAP2" : "0.5",
  "TSNAPINC" : "0.01",
  "IDX" : "1",
  "IDY" : "1",
  "SNAP_FORMAT" : "3",
  "SNAP_FILE" : "./snap/hh_e_t",
```

with

SNAP : output of snapshots (no seismograms=0; particle velocities=1; pressure field/hydrophones=2; curl and divergence energy=3; velocities, pressure, and energy=4)

TSNAP1 : first snapshot (in seconds)

TSNAP2 : last snapshot (in seconds)

TSNAPINC : increment (in seconds; should be a multiple of DT)

IDX : increment in  $x$ -direction (grid points)

IDY : increment in  $y$ -direction (grid points)

SNAP\_FORMAT : data format (ASCII=2; binary=3)

SNAP\_FILE : output basename

If  $\text{SNAP} > 0$ , wavefield information (particle velocities, pressure or curl and divergence of particle velocities) for the entire model is saved on disk; therefore, be sure that you have enough free space! Each PE is writing its sub-volume to disk. The filenames have the basename SNAP\_FILE plus an extension that indicates the PE number in the logical processor array (see Fig. 10). Note that the file sizes increase during the simulation. It may therefore be necessary to reduce the amount of snapshot data by increasing IDX and IDY and/or TSNAPINC. In order to merge the separate snapshot of each PE after the completion of the wave modeling, you can use the program snapmerge.

#### 4.11 Receivers

```
"Receiver" : "comment",
  "SEISMO" : "0",
  "READREC" : "1",
  "REC_TOPO" : "0",
  "REC_FILE" : "./receiver/receiver.dat",
  "REFRECX, REFRECY" : "0.0 , 0.0",
```

```
"XREC1,YREC1" : "50.0 , 1.0",
"XREC2,YREC2" : "350.0 , 1.0",
"NGEOPH" : "1",
```

with

SEISMO : output of seismograms (no seismograms=0; particle velocities=1; pressure/hydrophones=2; curl and div=3; everything=4)

If SEISMO>0, seismograms are saved on disk; when SEISMO is 1, x- and y-components of particle velocity will be written according to parameters specified in section 4.13; if SEISMO=2, pressure (sum of the diagonal components of the stress tensor) recorded at the receiver locations is written; if SEISMO=3 the curl and divergence are saved. The curl and divergence of the particle velocities are useful to separate P- and S-waves in the snapshots of the wavefield. SOFI2D calculates the divergence and the magnitude of the curl of the particle velocity field (Dougherty and Stephen, 1988). The motivation for this is as follows: according to Morse and Freshbach (1953), the energy of P- and S-wave particle velocities, respectively, are

$$E_p = (\lambda + 2\mu) [\text{div}(\vec{v})]^2 \quad \text{and} \quad E_s = \mu |\text{rot}(\vec{v})|^2, \quad (74)$$

where  $\lambda$  and  $\mu$  are the Lamè parameters, and  $\vec{v}$  is the particle velocity vector. In order to preserve the divergence and curl sign information while showing relative compressional and shear particle velocity amplitudes, we plot the following quantities:

$$\bar{E}_p = \text{sign}(\text{div} \vec{v}) E_p^{1/2} \quad \text{and} \quad \bar{E}_s = \text{sign}(\text{rot} \vec{v}) E_s^{1/2}. \quad (75)$$

The magnitudes of  $\bar{E}_p$  and  $\bar{E}_s$  are proportional to the magnitudes of the P- and S-particle velocities, respectively. Note that interface waves like Rayleigh waves contain both a P- and S-wave component and therefore show up on both quantities of Eq. 75.

READREC : read receiver positions from file (from the parameter file=0; from an external file=1)

REC\_TOPO : place receivers relative to topography (=1) or at absolute depth (=0)

The REC\_TOPO option is only relevant in case receivers are read from an external file (READREC=1). Under normal circumstances, i.e., when REC\_TOPO=0 which is also the default, receivers are placed at the absolute depth specified in the external receiver file given by REC\_FILE. However, sometimes an explicit air layer is part of the model in case the Earth has topography. When REC\_TOPO=1, then receivers are not placed at an absolute depth value but the depth specified in the external receiver file becomes a depth below topography. For details on how the topography is found, please check the SOURCE\_TOPO option.

REC\_FILE : external receiver file name (ASCII-file)

REFREC : reference point for receiver coordinate system (if 1, the following 3 parameters are ignored)

XREC1,YREC1 : position of the first receiver (in meters)

XREC2,YREC2 : position of the last receiver (in meters)

NGEOPH : distance between two adjacent receivers (in grid points)

The locations of the receivers may either be specified in the parameter file or in a separate file (REC\_FILE). When reading them from the parameter file, it is assumed that the receivers are located along a straight line. The first receiver position is defined by (XREC1, YREC1), and the last receiver position by (XREC2, YREC2). The spacing between receivers is NGEOPH grid points; a vertical seismic profile (VSP) is realized when XREC1=XREC2 and YREC1<YREC2.

When reading from an ASCII-file, each line should contain the receivers coordinates (in meters) as below: the horizontal  $x$ - and then the vertical  $y$ -coordinate of each receiver position, respectively.

```
50.0  2.0
100.0 2.0
150.0 2.0
...
```

These receiver coordinates are possibly shifted by `REFREC[1]` and `REFREC[2]` into the  $x$ - and  $y$ -direction, respectively. This allows for completely moving the receiver spread without modifying `REC_FILE`. This may be useful for the simulation of moving profiles in reflection seismics.

Receivers are always located on full grid indices, i.e., a receiver that is located between two grid points will be shifted by the FD program to the closest next grid point. It is not possible to output seismograms for arbitrary receiver locations since this would require a certain wavefield interpolation. It is important to note that the actual receiver positions defined in the `REC_FILE` or in `sofi2D.json` may vary by  $DH/2$  due to the staggered positions of the particle velocities and stress tensor components.

#### 4.12 Receiver array

```
"Receiver array" : "comment",
    "REC_ARRAY" : "0",
    "REC_ARRAY_DEPTH" : "70.0",
    "REC_ARRAY_DIST" : "40.0",
    "DRX" : "4",
```

with

`REC_ARRAY` : number of receivers in 1D receiver array (no simulation=0)  
`REC_ARRAY_DEPTH` : depth of first plane (in meters)  
`REC_ARRAY_DIST` : increment between receiver planes (in meters)  
`DRX` : increment between receivers in each plane (in gridpoints)

A horizontal 1D array of receivers is simulated if `REC_ARRAY>0`. This option specifies the number of receiver planes horizontal to the surface of the model (parallel to the  $x$ -axis). The distance between receivers within each plane is `DRX*DH` (in meters).

#### 4.13 Seismograms

```
"Seismograms" : "comment",
    "NDT" : "5",
    "SEIS_FORMAT" : "1",
    "SEIS_FILE" : "./su/test_t003",
```

with

`NDT` : sampling rate (in FD time steps)

If `SEISMO>0`, seismograms recorded at the receiver positions are written to the corresponding output files. The sampling rate of the seismograms is `NDT*DT` seconds. In case of a small time step interval and a large number of time steps, it might be useful to choose a relatively large `NDT` to avoid an unnecessary detailed sampling of the seismograms and consequently large files of seismogram data.

SEIS\_FORMAT : data output format (SU=1; ASCII=2; binary=3)

It is recommended to use SU (native 4-byte-floats (IEEE on PC)/little endian on PC) format for saving the seismograms. The main advantage of this format is that the time step interval (NDT\*DT) and the acquisition geometry (shot and receiver locations) are stored in the corresponding SU header words. Moreover, additional header words like offset are set by SOFI2D. This format thus facilitates a further visualization and processing of the synthetic seismograms. Note, however, that SU cannot handle sampling rates smaller than  $1.0 \times 10^{-3}$  ms and the number of samples is limited to about 32 000. In such cases, you should increase the sampling rate by increasing NDT. If this is impossible (for example, because the Nyquist criterion is violated), you must choose a different output format (ASCII or binary).

SU-files can be merged together using the Unix command `cat`.

SEIS\_FILE : basename for output of seismograms (SEISMO)

Separate seismogram files are output, looking like `SEIS_FILE_vz.su` or `SEIS_FILE_div.bin`. Each PE internally stores seismograms which are recorded in its portion of the grid. After finishing the time step loop, the seismogram parts are internally exchanged and the merged seismograms are collectively saved to the output file. A certain suffix is added to the basename `SEIS_FILE` dependent on the chosen `SEISMO` option (pressure, particle velocity, curl, and divergence). To give an example, the program writes the merged seismograms of the x-component of particle velocity to `SEIS_FILE_vx.su`.

### Wavelet shapes – practical issues

When using an explosive source, a pressure sensor, and a Ricker wavelet as source function with the standard elastic wave equation, the resulting seismograms have to be integrated one-and-a-half times (`sufrac power=-1.5`) in order to get back to the original Ricker shape and its spectrum. In addition, there is a polarity reversal due to the definition of pressure; in such a case, the additional parameter `phasefac=1` would revert the polarity. When using a force source, a velocity sensor, and a Ricker wavelet as source function with the standard elastic wave equation, the resulting seismograms have to be half-integrated (`sufrac power=-0.5`) in order to get back to the original Ricker shape and its spectrum. There is no polarity reversal.

## 4.14 Monitoring the simulation

```
"Monitoring the simulation" : "comment",
    "LOG_FILE" : "log/test.log",
    "LOG" : "0",
    "LOG_VERBOSITY" : "INFO",
    "OUT_TIMESTEP_INFO" : "100",
```

with

`LOG_FILE` : log-file for information about progress of program (each PE is printing log-information to `LOG_FILE.MYID`)

`LOG` : output of logging information of node (all ranks output to terminal=0; all ranks output to `LOG_FILE`=1; rank 0 outputs to terminal and all other ranks to `LOG_FILE`=2)

`SOFI2D` can output a lot of useful and debug information about the modeling parameters and the status of the modeling process. The major part of this information is output by the main MPI process. The `LOG` parameter in the `json` file sets the logging output stream. If `LOG=0`, the main MPI process writes its log to `stdout`, i.e., on the terminal; this is generally recommended to monitor the modeling process. You may want to save this screen log to

an output file by adding `2>&1 | tee sofi2D.jout` to your start command (if you use a queuing system, this output is normally captured anyway and returned by the queuing system at the end of the job). `LOG=1` sends all output to log files as specified in the json file. `LOG=2` send the output of the main MPI process to stdout and all other ranks write their output to log files.

`LOG_VERBOSITY` : set how much information is output (DEBUG, INFO, WARN, SILENT)

If the parameter is not given, `LOG_VERBOSITY` is internally set to INFO, which means the program should output information that is useful for the user. On DEBUG level, additional output (including code line numbers) occurs which can help in debugging issues. On level WARN, only warnings and errors are output, while SILENT means only errors are output – they cannot be ignored by the user.

`OUT_TIMESTEP_INFO` : time step increment upon which information is output

As the output of information can both slow down the computation for very small models and produce very large log files, you can choose by `OUT_TIMESTEP_INFO` after how many time steps such intermediate information is logged.

## 4.15 Postprocessing

The wavefield snapshots can be merged using the program `snapmerge`. The program `snapmerge` is not an MPI program. Therefore, it can be executed without MPI and the `mpirun` command. You can run `snapmerge` on any PC since a MPI environment (e.g., LAM) is not required. You may therefore copy the snapshot outputs of the different nodes to another non-MPI computer to merge the files together. `snapmerge` reads the required information from the `sofi2D` parameter file. Simply type

```
../bin/snapmerge /path/to/sofi2D.json
```

Depending on the model size and number of MPI processes, the merge process may take some time. For the simple block model, it only takes a few seconds. At the end of the merge process, `snapmerge` will output SU commands that allow you to run the snapshots as movie.

## References

- Aldridge, D. F. (1990). “The Berlage wavelet”. *GEOPHYSICS* 55.11, pp. 1508–1511. DOI: 10.1190/1.1442799.
- Bai, T. and Tsvankin, I. (2016). “Time-domain finite-difference modeling for attenuative anisotropic media”. *Geophysics* 81.2, pp. C69–C77. DOI: 10.1190/GEO2015-0424.1.
- Bai, T., Zhu, T., and Tsvankin, I. (2019). “Attenuation compensation for time-reversal imaging in VTI media”. *Geophysics* 84.4, pp. C205–C216. DOI: 10.1190/geo2018-0532.1.
- Blanch, J., Robertsson, J., and Symes, W. (1995). “Modeling of a constant Q: Methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique”. *Geophysics* 60.1, pp. 176–184. DOI: 10.1190/1.1443744.
- Bohlen, T. and Wittkamp, F. (2015a). “3-D viscoelastic FDTD seismic modelling using the staggered Adams-Bashforth time integrator”. *Geophysical Journal International*.
- Bohlen, T. and Wittkamp, F. (2015b). “Higher Order FDTD Seismic Modelling Using the Staggered Adams-Bashforth Time Integrator”. In: *77th EAGE Conference and Exhibition 2015*.
- Bohlen, T. (2002). “Parallel 3-D viscoelastic finite-difference seismic modelling”. *Comput. and Geosci.* 28.8, pp. 887–899.

- Bond, W. L. (1943). "The Mathematics of the Physical Properties of Crystals". *Bell System Technical Journal* 22.1, pp. 1–72. DOI: 10.1002/j.1538-7305.1943.tb01304.x.
- Carcione, J. M., Kosloff, D., and Kosloff, R. (1988). "Wave-propagation simulation in an elastic anisotropic (transversely isotropic) solid". *The Quarterly Journal of Mechanics and Applied Mathematics* 41.3, pp. 319–346. DOI: 10.1093/qjmam/41.3.319.
- Carcione, J. M. (2007). *Wave fields in real media: Wave propagation in anisotropic, anelastic, porous and electromagnetic media*. Vol. 38. Elsevier.
- Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M. (1985). "A nonreflecting boundary condition for discrete acoustic and elastic wave equations". *Geophysics* 50, pp. 705–708.
- Collino, F. and Tsogka, C. (2001). "Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media". *Geophysics* 66.1, pp. 294–307.
- Courant, R., Friedrichs, K., and Lewy, H. (1928). "Über die partiellen Differenzengleichungen der mathematischen Physik". *Mathematische Annalen* 100, pp. 32–74.
- Courant, R., Friedrichs, K., and Lewy, H. (1967). "On the partial difference equations of mathematical physics". *IBM Journal*, pp. 215–234.
- Dougherty, M. and Stephen, R. (1988). "Seismic Energy Partitioning and Scattering in Laterally Heterogeneous Ocean Crust". *Pure Appl. Geophys.* 128.1/2, pp. 195–239.
- Emmerich, H. and Korn, M. (1987). "Incorporation of attenuation into time-domain computations of seismic wave fields". *Geophysics* 52.9, pp. 1252–1264.
- Falk, J. (1998). "Efficient Seismic Modeling of Small-Scale Inhomogeneities by the Finite-Difference Method". PhD thesis. University of Hamburg.
- Fellinger, P., Marklein, R., K.J., L., and Klaholz, S. (1995). "Numerical modeling of elastic wave propagation and scattering with EFIT - elastodynamic finite integration technique". *Wave Motion* 21, pp. 47–66.
- Ghrist, M., Fornberg, B., and Driscoll, T. A. (2000). "Staggered time integrators for wave equations". *SIAM Journal on Numerical Analysis* 38.3, pp. 718–741.
- Graves, R. (1996). "Simulating Seismic Wave Propagation in 3D Elastic Media Using Staggered-Grid Finite Differences". *Bull., Seis Soc. Am.* 86.4, pp. 1091–1106.
- Holberg, O. (1987). "Computational aspects of the choice of operator and sampling interval for numerical differentiation in large-scale simulation of wave phenomena". *Geophysical Prospecting* 35, pp. 629–655.
- Jastram, C. (1992). "Seismische Modellierung mit Finiten Differenzen höherer Ordnung auf einem Gitter mit vertikal variierendem Gitterabstand". PhD thesis. Universität Hamburg.
- Knopoff, L. and MacDonald, G. (1958). "Attenuation of small amplitude stress waves in solids". *Reviews of Modern Physics* 30, pp. 1178–1192.
- Komatitsch, D. and Martin, R. (2007). "An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation". *Geophysics* 72.5, pp. 155–167.
- Komatitsch, D. and Tromp, J. (1999). "Introduction to the spectral element method for three-dimensional seismic wave propagation". *Geophysical Journal International* 139.3, pp. 806–822. DOI: 10.1046/j.1365-246x.1999.00967.x.
- Levander, A. (1988). "Fourth-order finite-difference P-SV seismograms". *Geophysics* 53.11, pp. 1425–1436.
- Liu, H.-P., Anderson, D. L., and Kanamori, H. (1976). "Velocity dispersion due to anelasticity; implications for seismology and mantle composition". *Geophysical Journal of the Royal Astronomical Society* 47.1, pp. 41–58. DOI: 10.1111/j.1365-246X.1976.tb01261.x.
- Martin, R. and Komatitsch, D. (2009). "An unsplit convolutional perfectly matched layer technique improved at grazing incidence for the viscoelastic wave equation". *Geophys. Prosp.* 179.1, pp. 333–344.

- Moczo, P., Kristek, J., Vavrycuk, V., Archuleta, R., and Halada, L. (2002). “Heterogeneous staggered-grid finite-difference modeling of seismic motion with volume harmonic and arithmetic averaging of elastic moduli and densities”. *Bull., Seis Soc. Am.* 92.8, pp. 3042–3066.
- Morse, P. and Freshbach, H. (1953). *Methods of theoretical physics*. McGraw-Hill.
- Neelima, S., Saritha, A., Ramesh, K., and Koteswara Rao, S. (2018). “Application of Klauder wavelet for generation of synthetic seismic signals”. *International Journal of Engineering and Technology* 7, pp. 903–905.
- O’Connell, R. J. and Budiansky, B. (1978). “Measures of dissipation in viscoelastic media”. *Geophysical Research Letters* 5, pp. 5–8. doi: 10.1029/GL005i001p00005.
- Oh, J. W., Shin, Y., Alkhalifah, T., and Min, D. J. (2020). “Multistage elastic full-waveform inversion for tilted transverse isotropic media”. *Geophysical Journal International* 223.1, pp. 57–76. doi: 10.1093/gji/ggaa295.
- Robertsson, J., Blanch, J. O., and Symes, W. (1994). “Viscoelastic finite-difference modeling”. *Geophysics* 59, pp. 1444–1456. doi: 10.1190/1.1443701.
- Robertsson, J., Levander, A., Symes, W., and Holliger, K. (1995). “A comparative study of free-surface boundary conditions for finite-difference simulation of elastic/viscoelastic wave propagation”. In: Houston, Texas, pp. 1277–1280.
- TGS (2017). *Anisotropic Parameter Estimation*. URL: <https://www.tgs.com/products-services/processing/imaging/depth-technologies/iso-vti-tti-ort-tort> (visited on 07/12/2022).
- Thomsen, L. (1986). “Weak elastic anisotropy”. *Geophysics* 51.10, pp. 1954–1966. doi: 10.1190/1.1442051.
- Tsvankin, I. (2012). *Seismic Signatures and Analysis of Reflection Data in Anisotropic Media, Third Edition*. Society of Exploration Geophysicists. doi: 10.1190/1.9781560803003.
- Virieux, J. (1986). “P-SV wave propagation in heterogeneous media: velocity-stress finite-difference method”. *Geophysics* 51.4, pp. 889–901.
- Voigt, W. (1910). *Lehrbuch der Kristallphysik*. Leipzig, Berlin: B. G. Teubner.
- Vossen, R., Robertsson, J., and Chapmann, C. (2002). “Finite-Difference modeling of wave propagation in a fluid-solid configuration”. *Geophysics* 67.2, pp. 618–624.
- Zahradník, J., Moczo, P., and Hron, F. (1993). “Testing four elastic finite difference schemes for behaviour at discontinuities”. *Bulletin of the Seismological Society of America* 83, pp. 107–129. doi: 10.1785/BSSA0830010107.
- Zener, C. M. (1948). *Elasticity and Anelasticity of Metals*. Vol. 55. University of Chicago Press, Chicago Illinois, p. 170.
- Zhu, Y. and Tsvankin, I. (2006). “Plane-wave propagation in attenuative transversely isotropic media”. *Geophysics* 71.2, pp. 17–30. doi: 10.1190/1.2187792.

## A Appendix: Parameters used in the code

### MODEL PARAMETERS

#### general parameters

```

READMOD=0 ..... switch to read model parameters from
                                MFILE
MFILE="" ..... model file name
NX=0 ..... number of grid points in x-direction
NY=0 ..... number of grid points in y-direction
                                (depth)

```

SOURCE_TYPE=0	.....	type of source
SOURCE_SHAPE=0	.....	shape of source-signal
SIGNAL_FILE=""	.....	name of external signal file
SRCREC=0	.....	read source parameters from external source file
SOURCE_FILE=""	.....	name of source parameter file
SOURCE_TOPO=0	.....	place sources along topography
RUN_MULTIPLE_SHOTS=0	.....	multiple shots modeled simultaneously or individually
PLANE_WAVE_DEPTH=0.0	.....	depth of plane wave excitation [m]
PLANE_WAVE_ANGLE=0.0	.....	dip of plane wave from vertical [deg]
TS=0.0	.....	duration of source signal [s]

```

READREC=0 ..... switch to read receiver positions from
                  file
REC_FILE="" ..... name of external receiver file
REC_TOPO=0 ..... place receivers along topography
REFREC[4]={0.0, 0.0, 0.0, 0.0} ... reference point for receiver coordinate
                                  system
XREC1=0.0 ..... x-position of first receiver [m]
XREC2=0.0 ..... x-position of last receiver [m]
YREC1=0.0 ..... y-position of first receiver [m]
YREC2=0.0 ..... y-position of last receiver [m]
NGEOPH ..... distance between two adjacent receivers
              [grid points]
REC_ARRAY=0 ..... number of receivers in 1D receiver array
REC_ARRAY_DEPTH=0.0 ..... depth of first plane [m]
REC_ARRAY_DIST=0.0 ..... increment between receiver planes [m]
DRX=0..... increment between receivers in each plane
              [grid points]

```

L=0	.....	number of relaxation parameters
F_REF	.....	reference frequency [Hz]
FL=0.0	.....	frequency of each relaxation parameters [Hz]
TAU=0.0	.....	ratio of retardation and relaxation time

```

SEISMO=0 ..... switch to output components of seismograms
NDT=1 ..... sampling rate of seismograms [time steps]
SEIS_FORMAT=0 ..... data output format for seismograms
SEIS_FILE="" ..... name of output file of seismograms

```



#### snapshot parameters

SNAP=0 ..... switch to output of snapshots  
SNAP\_FORMAT=0 ..... data output format for snapshots  
SNAP\_FILE="" ..... name of output file of snapshots  
TSNAP1=0.0 ..... first snapshot [s]  
TSNAP2=0.0 ..... last snapshot [s]  
TSNAPINC=0.0 ..... increment between snapshots [s]  
IDX=1 ..... increment in x-direction [grid points]  
IDY=1 ..... increment in y-direction [grid points]

#### other parameters

WRITE\_MODELFILES=0 ..... switch to output model files  
SIGOUT=0 ..... switch to output source wavelet  
SIGOUT\_FORMAT=0 ..... data output format for source wavelet  
SIGOUT\_FILE="" ..... name of output file of source wavelet  
LOG=0 ..... switch to output logging information  
LOG\_FILE="" ..... name of output file of logging information  
LOG\_VERBOSITY="INFO" ..... set how much information is output  
OUTNTIMESTEPINFO=1 ..... information on the time step given to  
screen/file

### FD PARAMETERS

#### general parameters

WEQ="" ..... wave equation  
DH=0.0 ..... spacial increment [m]  
FDORDER=0 ..... spatial FD order  
TIME=0.0 ..... modeling time [s]  
DT=0.0 ..... modeling time increment [s]  
FDORDER\_TIME=0 ..... temporal FD order

#### MPI parameters

MAXRELError=0..... maximum relative group velocity error  
NPROCX=1 ..... number of processors in x-direction  
NPROCY=1 ..... number of processors in y-direction

#### boundary parameters

FREE\_SURF=0 ..... switch to apply free surface at the top  
of the model  
BOUNDARY=0..... switch to apply periodic boundary condi-  
tion at edges  
ABS\_TYPE ..... type of the absorbing boundary  
FW=0 ..... width of absorbing frame [grid points]  
DAMPING=0.0 ..... attenuation at the edges of the grid  
[%]

#### PML parameters

NPOWER .....	exponent for calculation of damping profile
K_MAX_CPML .....	
FPML .....	dominant signal frequency (usually FC) [Hz]
VPPML.....	attenuation velocity within the PML boundary [m/s]