3D anisotropic Finite-Difference modelling on the standard staggered grid

Master's thesis of

Daniel Mai

at the Geophysical Institute (GPI)
KIT-Department of Physics
Karlsruhe Institute of Technology (KIT)

Date of submission:
06.09.2024

Supervisor: Prof. Dr. Thomas Bohlen
Co-supervisor: Prof. Dr. Joachim Ritter

Abstract

3D anisotropic modelling is becoming more important to resolve more complex structures. Therefore, I implemented a new algorithms into an already existing program to be able to simulate anisotropy. This implementation models with finite differences on the standard staggered grid. For complex anisotropic structures a wavefield interpolation is necessary. I have implemented to interpolation options and compared them. There are the common sinc-function interpolation and the FD-consistent interpolation which was not used in this context, yet.

I look at the stability of the algorithm and compare different modelling methods. Most results are as expected and indicate the algorithm works. The FD-consistent interpolation seems to be a lot more consistent than the sinc-function interpolation and it requires a lower spatial FD order to do so.

Contents

1 Introduction

In seismic acquisition the recorded datasets can often be large and it is computationally expensive to get a subsurface model to recorded data by inversion. Therefore, seismic forward modelling can be used to simulate wave propagation with known or estimated parameters to get synthetic traces at receivers placed in the model. These synthetic traces can be compared to already recorded traces. Thereby, seismic modelling plays a very important role in the analysis and simulation of seismic data.

The finite-difference (FD) method is a commonly used modelling method because it is relatively simple to implement comparing with other methods and it is still very efficient for modelling seismic wave propagation [Alterman and Karal Jr, 1968; Virieux, 1986; Igel et al., 1995; Graves, 1996].

The FD method was introduced by Yee in 1966, but was used for the first time in seismic and seismology by Alterman and Karal Jr [1968]. Since the first modelling attempts had some instability problems, Virieux [1986] introduced the staggered grid for FD modelling. The computing possibilities have increased a lot for the recent years, thus even more and bigger simulations are possible now. So, FD modelling can simulate complex forward problems where no analytical solution is known, but even for simpler models the analytical solution is often hard to obtain because of the large number of required calculation.

Finite-difference modelling deals with seismic wave propagation in known or estimated models. The model will be divided in small cells and for every one of those there are parameters calculated and stored. All these grid cells are either cubes or cuboids and the parameters are used to update the propagating wavefield over time.

In anisotropic media the seismic velocities depend on small scale geologic structures compared to the wave length. These directional dependence of the velocities can be caused by crystals, cracks or layers, for example. Vertical anisotropic modelling is recommended for horizontally layer anisotropic structures, but if the layers have stronger dipping like folded media or steep dipping beds, it is required to tilt the anisotropy in the model used for simulation, accordingly [Alaei et al., 2022].

For complex anisotropic structures 2D seismic modelling can not be sufficient because the waves in the simulation can only travel in one plane, but in the three-dimensional real structures, waves can arrive at the receivers taking path outside of the modelling plane. These can influence the recorded trace at receivers and they cannot be modeled correctly in the 2D case. The 2D anisotropic case already exists in SOFI2D by Thomas Bohlen.

In this work the first-order wave equation is used for seismic wave propagation and the grid is a three-dimensional standard staggered grid (SSG). The SSG is quite frequently used for FD modelling, amoung others in the already existing IFOS3D code. IFOS3D is based on SOFI3D by Thomas Bohlen. Both can model forward, but the main difference is that IFOS3D can also do the inversion. The current version of IFOS3D can only do acoustic and isotropic elastic forward modelling. In this work the ansiotropic case will be derived, implemented and analyzed. Some main points include the formulation of the stress update and the required interpolation. Tests of the implemented code contain a homogeneous anisotropic model which is used to produce snapshots of the wavefield and seismic traces recorded at receivers. Using these methods a comparison between different modelling types will be made.

Anisotropic simulations in three dimensions were already done by several works like Li et al. [2021] on the standard staggered grid, Ferrer et al. [2015] on the Lebedev grid and even with a anisotropic version of SOFI3D by Kabanov [2019], but this software only contains vertically layered anisotropy.

The completely new part in this work is the use of a FD-consistent interpolation by Koene et al. [2020] for the wavefield interpolation on the SSG and comparing its performance to commonly used sinc-function interpolations [Igel et al., 1995].

2 Theory

This chapter starts with the first-order wave equations for anisotropic media and looks at two specific cases of anisotropy. The parameters to build the geometries are shown and explained and the updates for the FD algorithm are illustrated. Another main part is the interpolation theory with two possibilities of the weight calculation for needed grid points on the staggered grid.

2.1 Wave equations

As a starting point, the first order elastic, anisotropic wave equations in a stress velocity formulation are used. They contain the equation of motion similar to Newton's law and the stress-strain relation for anisotropic media. The equation of motion

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial x_j} + f_i \tag{1}$$

is valid for all types of elastic media. The left side of the equation shows the density of the media times the time derivative of the particle velocity vector. Since the three-dimensional case is relevant for this work, the particle velocity vector contains three elements (v_1, v_2, v_3) or (v_x, v_z, v_y). The particle velocity equals the time derivative of the displacement vector. On the right-hand side there is a sum of the external body forces and the surface forces defined by the spatial derivatives of the stress tensor. The second important relation, namely the stress-strain relation

$$\sigma_{ij} = C_{ijkl} \cdot \epsilon_{kl}, \tag{2}$$

is based on Hooke's law and the stress tensor is defined by a multiplication of the fourth-order elasticity tensor ($1 \leq i, j, k, l \leq 3$) and the deformation tensor. The elasticity tensor has $3^4 = 81$ elements, but only a maximum of 21 of these elements are independent. The elasticity tensor is defined by

$$\epsilon_{kl} = \frac{1}{2}\left(\frac{\partial u_k}{\partial x_l} + \frac{\partial u_l}{\partial x_k}\right) \tag{3}$$

i.e., the spatial derivatives of the displacement.

To simplify the stress-strain relation, the Voigt notation was introduced Voigt [1910]. The Voigt notation transforms a symmetric 3×3 matrix with six independent elements to a vector having six components. The method is shown in figure 1 where the stress tensor with exactly six independent elements is simplified. A similar simplification is possible for the deformation tensor. The only difference is that $\epsilon_4 = 2\epsilon_{23}$, $\epsilon_5 = 2\epsilon_{13}$ and $\epsilon_6 = 2\epsilon_{12}$.

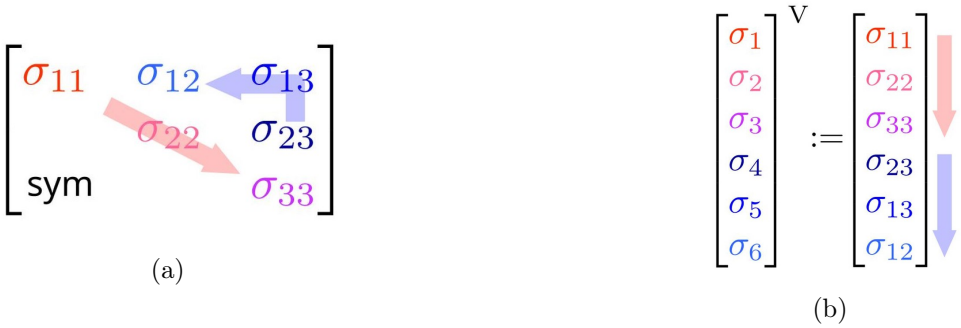


Figure 1: **Voight notation:** Introduces the Voigt notation on the example of the symmetric stress tensor.

Using the Voigt notation the elasticity tensor can be rewritten. The first two indices i and j lead to a new index $n = 1, \ldots, 6$ and the second two indices k and l lead to $m = 1, \ldots, 6$. The new indices n and m are distributed in the same way as shown in figure 1. A combination of these two indices creates a symmetric 6×6 matrix with 21 independent elements for the elasticity tensor C. This results in the matrix-vector product

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{zz} \\ \sigma_{yy} \\ \sigma_{yz} \\ \sigma_{xy} \\ \sigma_{xz} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} & c_{15} & c_{16} \\ c_{21} & c_{22} & c_{23} & c_{24} & c_{25} & c_{26} \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} & c_{36} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} & c_{46} \\ c_{51} & c_{52} & c_{53} & c_{54} & c_{55} & c_{56} \\ c_{61} & c_{62} & c_{63} & c_{64} & c_{65} & c_{66} \end{pmatrix} \cdot \begin{pmatrix} \epsilon_{xx} \\ \epsilon_{zz} \\ \epsilon_{yy} \\ 2\epsilon_{yz} \\ 2\epsilon_{xy} \\ 2\epsilon_{xz} \end{pmatrix} \tag{4}$$

as a new formulation for the stress-strain relation. In this equation the indices $1, 2, 3$ are already replaced by the corresponding coordinates x, z, y. It should be noted that $2 = z$ and $3 = y$, because in IFOS3D the vertical axis is the y-axis and z is the second horizontal axis (crossline).

The equation of motion, eq. 1, has to be changed as well to be consistent. Therefore, a symmetric gradient operator [Auld, 1973].

$$\nabla = \begin{pmatrix} \partial_1 & 0 & 0 & 0 & \partial_3 & \partial_2 \\ 0 & \partial_2 & 0 & \partial_3 & 0 & \partial_1 \\ 0 & 0 & \partial_3 & \partial_2 & \partial_1 & 0 \end{pmatrix} \tag{5}$$

is defined. As a consequence

$$\rho \frac{\partial v}{\partial t} = \nabla \cdot \sigma + f \tag{6}$$

and the time derivative of the strain-displacement relation

$$\dot{\epsilon}_{ij} = \frac{1}{2} \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) \tag{7}$$

turns into

$$\dot{\epsilon} = \nabla^T \cdot v \tag{8}$$

In the next part of this chapter two specific cases for the elasticity tensor in eq. 4 are analyzed.

2.2 VTI and TTI

The elasticity tensor contains the material parameters of the model and has 21 independent elements. It is possible to use all of these independent elements to simulate wave propagation with a new version I created of the IFOS3D code, but in reality it is almost impossible to get so many parameters for an arbitrary material. Sometimes it is already challenging to get enough parameters for the following two specific cases. The first case is called vertical transverse isotropy (VTI). The elasticity tensor

$$C_{VTI} = \begin{pmatrix} c_{11} & c_{11}-2c_{66} & c_{13} & 0 & 0 & 0 \\ c_{11}-2c_{66} & c_{11} & c_{13} & 0 & 0 & 0 \\ c_{13} & c_{13} & c_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & c_{66} \end{pmatrix} \tag{9}$$

contains only five independent elements and overall 24 out of 36 entries are zero. The VTI case describes a horizontally layered medium with a vertical axis of rotational symmetry. To obtain these five parameters five independent equations are necessary:

$$\alpha_0 = \sqrt{\frac{c_{33}}{\rho}} \tag{10}$$

$$\beta_0 = \sqrt{\frac{c_{66}}{\rho}} \tag{11}$$

$$\epsilon = \frac{c_{11}-c_{33}}{2c_{33}} \tag{12}$$

$$\gamma = \frac{c_{66}-c_{44}}{2c_{44}} \tag{13}$$

$$\delta = \frac{1}{2}\frac{(c_{13}+c_{44})^2-(c_{33}-c_{44})^2}{c_{33}(c_{33}-c_{44})} \tag{14}$$

The first two equations, eq. 10 and eq. 11, define the seismic velocities along the vertical axis. These are the P-wave and S-wave velocities denoted by α_0 and β_0. For these waves the angle between the propagating direction and the vertically orientated symmetry axis is 0° and ρ is the density of the material. The other three equations are defined by the Thomsen parameters [Thomsen, 1986]. ϵ describes the P-wave velocity depending on the mentioned angle. For typical rocks ϵ is positive with a value of $0.0-0.26$ [Thomsen, 1986] and this results in P-waves travelling faster in horizontal direction compared to vertical direction. The parameter γ is similar to ϵ and shows the differences in SH-wave velocities depending on the angle. In this case values between 0.0 and 0.5 are typical and SH-waves travel again faster in horizontal direction if γ has positive values. The parameter δ describes the angular dependency (moveout) in the vicinity of the vertical axis and δ is typically between -0.27 and 0.6. To get the five independent parameters

$c_{11}, c_{13}, c_{33}, c_{44}, c_{66}$ of the elasticity tensor eq. 10-14 are solved for these parameters and lead to the following formulas:

$$
\begin{aligned}
c_{11} &= (2\epsilon + 1)c_{33} && (15)\\
c_{13} &= \sqrt{2\delta c_{33}(c_{33} - c_{44}) + (c_{33} - c_{44})^2} - c_{44} && (16)\\
c_{33} &= \rho\alpha_0^2 && (17)\\
c_{44} &= \rho\beta_0^2 && (18)\\
c_{66} &= (2j + 1)c_{44} && (19)
\end{aligned}
$$

The Kelvin-Christoffel equations [Carcione, 2007] can be used to get the velocities of the P-waves, the SV-waves and the SH-waves depending on the ealier described angle. These velocities are defined by

$$
\begin{aligned}
V_P &= \sqrt{\frac{1}{\rho}\left(\frac{a}{2} + \sqrt{\frac{a^2}{4} - b}\right)}, && (20)\\
V_{SV} &= \sqrt{\frac{1}{\rho}\left(\frac{a}{2} - \sqrt{\frac{a^2}{4} - b}\right)}, && (21)\\
V_{SH} &= \sqrt{\rho^{-1}(c_{66}l_1^2 + c_{44}l_3^2)}, && (22)
\end{aligned}
$$

where

$$
\begin{aligned}
a &= c_{33}l_3^2 + c_{11}l_1^2 + c_{55},\\
b &= c_{44}c_{33}l_3^4 + c_{44}^2 + c_{11}c_{33}l_1^2l_3^2 + (c_{11} + c_{44})l_1^2 - (c_{13} + c_{44})^2 l_1^2 l_3^2,\\
l_1 &= \sin(\theta),\\
l_3 &= \cos(\theta).
\end{aligned}
$$

The angle measured against the vertical axis is denoted by θ. So, for vertically propagating waves the angle takes the value $\theta = 0°$ and for horizontally travelling waves $\theta = 90°$. These equations are used to produce figure 2. The values needed for the calculation of the velocities are the same that are used for the homogeneous anisotropic model which is explained later in this work. The parameters δ and ϵ have equal values like those derived for a Greenhorn shale by Jones and Wang [1981]. Using a smaller value for γ in comparison to their work, there is less SH-wave anisotropy in these results. Besides, in order to have easier calculations there are used different values for the P-wave and S-wave velocities.

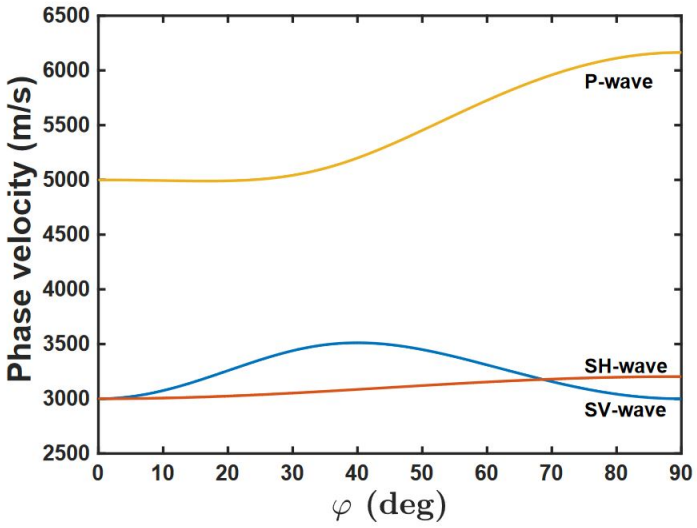


Figure 2: **Angle dependency of seismic velocities**: Gives the velocities of P-, SV-, SH-waves depending on the angle to the vertical symmetry axis. The independent parameters of the elasticity tensor and the density are $c_{11} = 76.0$, $c_{33} = 50.0$, $c_{44} = 18.0$, $c_{66} = 20.5.0$, $c_{13} = 11.4$, $\rho = 2.0$

For getting to the tilted transverse isotropy case abbreviated by TTI, the VTI system has to be rotated by two angles θ and ϕ. Thereby, the anisotropic symmetric system is rotated into the observation system. This thesis follows the steps of Zhu and Dorman [2000] where the so-called Bond transformation is executed. The first rotation corresponds to the dip angle θ. It rotates around the z-axis until the $x-z-$plane coincides with the observation system. The second one turns the system around the vertical y-axis, since the y-axis is already at the same spot in both systems. It is a clockwise rotation around the y-axis with an angle defined by $90° - \phi$. If $\theta = 0°$ and $\phi = 90°$, the two systems are already aligned and no rotation is taking place, thus, the TTI case is equal to VTI. The two rotation matrices are given by

$$A_1 = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{23}$$

$$A_2 = \begin{pmatrix} \sin\phi & cos\phi & 0 \\ -\cos\phi & \sin\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{24}$$

and the multiplication of these results in

$$A = A_2 A_1 = \begin{pmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{pmatrix} \tag{25}$$

where

$$
\begin{aligned}
l_1 &= \cos\theta \sin\phi, & m_1 &= \cos\phi, & n_1 &= \sin\theta \cos\phi, \\
l_2 &= -\cos\theta \cos\phi, & m_2 &= \sin\phi, & n_2 &= -\sin\theta \cos\phi, \\
l_3 &= -\sin\theta, & m_3 &= 0, & n_3 &= \cos\theta.
\end{aligned} \tag{26}
$$

Following Zhu and Dorman [2000] and Bond [1943] the rotation matrix R in Voigt notation is defined by

$$
R = \begin{pmatrix}
l_1^2 & m_1^2 & n_1^2 & 2m_1n_1 & 2n_1l_1 & 2l_1m_1 \\
l_2^2 & m_2^2 & n_2^2 & 2m_2n_2 & 2n_2l_2 & 2l_2m_2 \\
l_3^2 & m_3^2 & n_3^2 & 2m_3n_3 & 2n_3l_3 & 2l_3m_3 \\
l_2l_3 & m_2m_3 & n_2n_3 & m_2n_3 + m_3n_2 & n_2l_3 + n_3l_2 & l_2m_3 + l_3m_2 \\
l_3l_1 & m_3m_1 & n_3n_1 & m_3n_1 + m_1n_3 & n_3l_1 + n_1l_3 & l_3m_1 + l_1m_3 \\
l_1l_2 & m_1m_2 & n_1n_2 & m_1n_2 + m_2n_1 & n_1l_2 + n_2l_1 & l_1m_2 + l_2m_1
\end{pmatrix}. \tag{27}
$$

R was calculated in a way that the whole rotation is defined by

$$
C_{TTI} = R \cdot C_{VTI} \cdot R^T \tag{28}
$$

where R^T denotes the transposed matrix of R. The resulting matrix C_{TTI} is now fully populated - there are some exceptions - and looks like the matrix of eq. 4.

In the following there are some examples for elasticity tensors resulting for various calculations. The values for these examples again equal the ones used for the homogeneous model later on. The VTI-tensor before the Bond transformation is defined by

$$
C' = \begin{pmatrix}
76.00 & 34.96 & 11.39 & 0 & 0 & 0 \\
34.96 & 76.00 & 11.39 & 0 & 0 & 0 \\
11.39 & 11.39 & 50.00 & 0 & 0 & 0 \\
0 & 0 & 0 & 18.00 & 0 & 0 \\
0 & 0 & 0 & 0 & 18.00 & 0 \\
0 & 0 & 0 & 0 & 0 & 20.52
\end{pmatrix} \cdot 10^9 \text{Pa} \tag{29}
$$

using Table 3 for the density, the seismic velocities and the Thomsen parameters. All values in the matrix are given in Gigapascal. In the first example there is only one rotation with the dip angle $\theta = 30°$ taking place. ϕ is set to 90°, so there is no horizontal rotation. This leads to a matrix which is not fully populated and there is a separation between P-SV-waves and SH-waves. Eq. 30 presents the elasticity tensor for this specific case.

$$C = \begin{pmatrix} 63.648 & 29.068 & 17.246 & 0 & -9.008 & 0 \\ 29.068 & 76.00 & 17.285 & 0 & -10.204 & 0 \\ 17.246 & 17.285 & 50.648 & 0 & -2.25 & 0 \\ 0 & 0 & 0 & 18.63 & 0 & -1.091 \\ -9.008 & -10.204 & -2.25 & 0 & 23.853 & 0 \\ 0 & 0 & 0 & -1.091 & 0 & 19.89 \end{pmatrix} \cdot 10^9 \text{Pa} \tag{30}$$

In the second example ϕ is equal to 20° that leads to the fully populated tensor displayed in eq. 31.

$$C = \begin{pmatrix} 74.354 & 29.27 & 17.281 & 9.218 & -4.101 & 2.225 \\ 29.27 & 64.891 & 17.251 & 8.836 & -2.47 & 1.745 \\ 17.281 & 17.251 & 50.648 & 2.115 & -0.77 & 0.013 \\ 9.218 & 8.836 & 2.115 & 23.241 & -1.678 & 0.647 \\ -4.101 & -2.47 & -0.77 & -1.678 & 19.241 & 0.654 \\ 2.225 & 1.745 & 0.013 & 0.647 & 0.654 & 20.092 \end{pmatrix} \cdot 10^9 \text{Pa} \tag{31}$$

Gaining the six one-dimensional equations contained in the matrix formulation of eq. 4 leads to

$$\begin{aligned}
\frac{\partial \sigma_{xx}}{\partial t} &= c_{11} \cdot \frac{\partial v_x}{\partial x} + c_{12} \cdot \frac{\partial v_z}{\partial z} + c_{13} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{14} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{15} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{16} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \\
\frac{\partial \sigma_{zz}}{\partial t} &= c_{21} \cdot \frac{\partial v_x}{\partial x} + c_{22} \cdot \frac{\partial v_z}{\partial z} + c_{23} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{24} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{25} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{26} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \\
\frac{\partial \sigma_{yy}}{\partial t} &= c_{31} \cdot \frac{\partial v_x}{\partial x} + c_{32} \cdot \frac{\partial v_z}{\partial z} + c_{33} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{34} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{35} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{36} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \\
\frac{\partial \sigma_{yz}}{\partial t} &= c_{41} \cdot \frac{\partial v_x}{\partial x} + c_{42} \cdot \frac{\partial v_z}{\partial z} + c_{43} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{44} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{45} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{46} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \\
\frac{\partial \sigma_{xy}}{\partial t} &= c_{51} \cdot \frac{\partial v_x}{\partial x} + c_{52} \cdot \frac{\partial v_z}{\partial z} + c_{53} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{54} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{55} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{56} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right) \\
\frac{\partial \sigma_{xz}}{\partial t} &= c_{61} \cdot \frac{\partial v_x}{\partial x} + c_{62} \cdot \frac{\partial v_z}{\partial z} + c_{63} \cdot \frac{\partial v_y}{\partial y} \\
&+ c_{64} \cdot \left(\frac{\partial v_y}{\partial z} + \frac{\partial v_z}{\partial y} \right) + c_{65} \cdot \left(\frac{\partial v_x}{\partial y} + \frac{\partial v_y}{\partial x} \right) + c_{66} \cdot \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right)
\end{aligned} \tag{32}$$

where on the left-hand side time derivatives of the stress tensor components are shown. The first three lines express the diagonal stress components and the bottom three equations the shear-

stress components. On the right side there are six summands where always an element of the elasticity tensor is multiplied with a spatial derivative of a particle velocity which are identical for all six lines of eq. 32. This formulation is already quite useful for the stress update, since it has only the time derivative of the stress components on the left side. For the stress update of the FD algorithm this equation has to be discretized and, since the second accuracy order in time is used, the time derivative will only be a division by timestep dt. This FD algorithm is described in the next part and stress und velocity updates are shown for the 1D-case.

2.3 FD algorithm

The FD algorithm follows some major steps of the previous section. Again, the parameters, the seismic velocities and the Thomsen parameters are used to calculate an elasticity tensor. This tensor can already be applied to VTI FD modelling or it can be extended to the TTI tensor via Bond transformation. Once the final tensor is calculated, this tensor will not change during the progressing time of the simulation. Only the stress and the velocity components will be updated after every timestep. The following set of equations show a discretized update scheme for the stress and velocity components. They only take into account second FD order in time, but higher orders in space. Furthermore, the equations only apply to the one-dimensional case, but they are still displayed because they can show a simpler update compared to the three-dimensional case.

$$\sigma_j^{n+\frac{1}{2}} = \sigma_j^{n-\frac{1}{2}} - \frac{\lambda_j dt}{dy} \sum_{l=1}^{L} \alpha_l \left(v_{j+l-\frac{1}{2}}^n - v_{j-l+\frac{1}{2}}^n \right) \tag{33}$$

$$v_{j+\frac{1}{2}}^n = v_{j+\frac{1}{2}}^{n-1} - \frac{\rho_{j+\frac{1}{2}}^{-1} dt}{dy} \sum_{l=1}^{L} \alpha_l \left(\sigma_{j+l}^{n-\frac{1}{2}} - \sigma_{j-l+1}^{n-\frac{1}{2}} \right) \tag{34}$$

Eq. 33 and eq. 34 alternate for every time step. At time step 1, then the velocities are updated. At time step 1.5 the stress components are updated and at time step 2 the velocities take turn again. Both updates take the previous timestep into account and add some values which mostly depend on current values of the other update. This means, the stress upate depends on the current values of the velocity components and vice versa. Besides, the sum depends on the spatial FD order, since L is equal to the half of the FD order. Thus, the number of summands equals the spatial FD order. Also, the timestep dt, the grid spacing dy and the FD coefficents α_l affect both updates.

2.4 Standard staggered grid

The grid for finite-difference modelling is a so-called standard staggered grid (SSG). The SSG is a combination of two grids, where the derivative operators are placed halfway between the grid points. This leads to a separation of components like the stress from the displacement (velocities). The main reason for using two grids instead of only one is to improve the accuracy and to allow faster convergence of the algorithm [Virieux, 1986; Igel et al., 1995].

Figure 3 shows a 3D model of one grid cell of the whole three-dimensional grid that is used for IFOS3D. x and z are the horizontal axes and y denotes the vertical axis going from top to bottom in positive direction. Marked in black are the full grid points where all three coordinates are integers. On these full grid points the values for the whole elasticity tensor and the density ρ are stored. Furthermore, the values of the diagonal elements of the stress tensor are stored and updated on these points for every timestep of the FD algorithm. The shear stress components are not at the same locations as the diagonal ones on the SSG. They are all located at different positions, but always two of the three coordinates are at a half grid point value and the last coordinate is an integer. The velocity components only have one half grid point value, so they are always located between two full grid points with a half grid point distance to both of them.

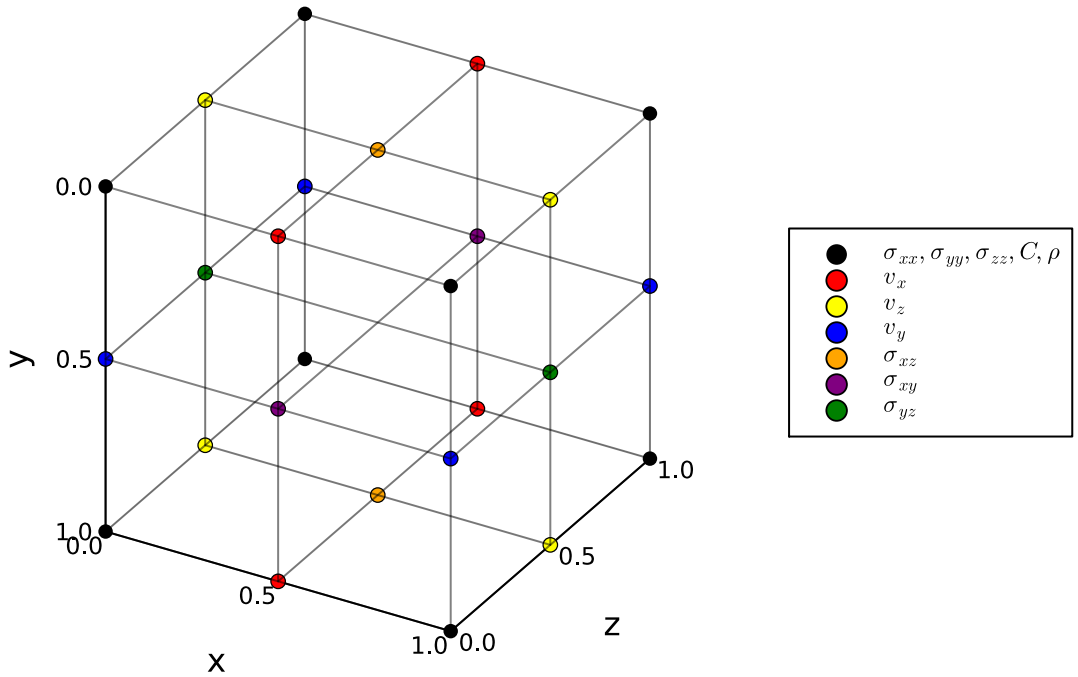


Figure 3: **Standard staggered grid:**3D model of the SSG. Black gives the full grid points and all point with the same color have the same parameters.

Looking at the velocity update, eq. 34, the velocities are updated on a half grid point indicated by $j + 0.5$ on the left hand side of the equation. So now, all the values on the right side are needed at the same grid points as well to make the time update possible. Since dt, dx, dy, dz

and α_l are global variables, they are defined at every spot on both grids. The density and the diagonal stress components are located half a grid point away from the velocities, but only in one dimension. This makes it possible to just perform a linear interpolation for these values. For the density only the direct neighbours are important, but for the stress components the number of used points is equal to the FD order in space. The needed shear stress components are located only a half grid point away as well. This is shown for example for v_y where σ_{xy} and σ_{yz} are only half a grid point away in x and z direction, respectively. Taking all this into account, there is no complex two-dimensional interpolation necessary and there are no changes between the elastic isotropic and the anisotropic case, since the elasticity tensor does not influence this update at all.

For the stress update, eq. 32 and eq. 33, the situation is more complex because the elements of the elasticity tensor are only at the correct spot for the first three lines of eq. 32. For the bottom three lines the values are half a grid point away in two dimensions. The spatial derivatives of the velocities are often not at the correct location as well and if they are in a wrong spot, once again a two-dimensional interpolation is necessary. These interpolations can be quite complex and are explained in more detail in the next section.

2.5 Interpolation

The coordinates shown in figure 3 are referred to as global coordinates of each point in this section, but some of these coordinates contain non-integer numbers causing problems in coding, since loops iterate only over integers. Therefore, so-called local coordinates are introduced for every grid point. Those local coordinates are only integer and they are the same for every grid point belonging to the same grid cell no matter on which grid they actually are. To every cell belong seven points, visually that is one point of each color in figure 3. For example, the seven points with local (x, y, z) coordinates $(0, 0, 0)$ are black $(0, 0, 0)$, yellow $(0, 0, 0.5)$, blue $(0, 0.5, 0)$, red $(0.5, 0, 0)$, green $(0, 0.5, 0.5)$, orange $(0.5, 0, 0.5)$, and purple $(0.5, 0.5, 0)$.

2.5.1 Elasticity tensor averaging

The elasticity tensor has to be interpolated only once because it will not change with time. The values are averaged harmonicly where as an example eq. 35 shows the averaging on the fourth row of the tensor. The target for these values is σ_{yz} (green in Figure 3) with the coordinates $(0, 0.5, 0.5)$. The four closest full grid points (black in Figure 3) are all located in the $y-z-$plane and have the indices $(0, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$ and $(0, 1, 1)$, respectively. Therefore, for example $C_{42}(0, 1, 0)$ is defined by the value C_{42} at the grid point $(0, 1, 0)$.

$$C_{4n}(0, 0.5, 0.5) \quad = \quad \frac{4}{\frac{1}{C_{4n}(0,0,0)} + \frac{1}{C_{4n}(0,1,0)} + \frac{1}{C_{4n}(0,0,1)} + \frac{1}{C_{4n}(0,1,1)}} \quad \text{for } 1 \leq n \leq 6 \qquad (35)$$

It is also possible to use one of the two wavefield interpolation methods that are later described in this section, but it is not really recommended because they are primary used to interpolate wavefields and not to average model parameters.

2.5.2 Spatial derivatives of the velocities

The spatial derivatives of the velocities are not located at the same spot on the grid as the velocity updates because a discretized derivative is just a difference between certain values divided by the grid spacing (dx, dy, dz). As an example taking the derivative of v_x with respect to x, which is denoted by v_{xx}, is defined at the full grid points along the x-axis, so for the diagonal stress components, these values do not need any interpolation. For the shear stress, however, an interpolation is necessary because v_{xx} is not defined at the half-half grid point, but it is defined at the four full grid points with a distance of $(0.5, 0.5)$ to the shear stress component. The sum of v_x differentiated with respect to z and v_z with respect to x having the notation v_{xzzx} for example is defined at half-half grid points where the y-coordinate is an integer (orange in figure 3). Thus, these grid points need to be interpolated for the updates of the diagonal stress tensor elements, but the geometry is the same compared to the interpolation of v_{xx}, since there are four grid points $(0.5, 0.5)$ away from the target.

The most complex interpolation is the interpolation from v_{xzzx} (orange) to σ_{xy} (purple), even though the distance is still the same if an interpolation in the $y-z-$plane is done. But this interpolation is not possible [Igel et al., 1995; Li et al., 2021] because there are only so-called forward and backward interpolations and it would be a mix of both types. It is called a forward interpolation, when the average of all indices of the points that are used for the interpolation is bigger than the current indices. An example for a forward interpolation is the one from the black points in the top layer - y-coordinate equals 0 - of figure 3 to the orange point in the middle of the top layer. The orange point has $x-z-$coordinates of $(0.5, 0.5)$ on the global grid, but on its local grid the coordinates are $(0, 0)$ and, thus, has an average of 0. The four black points have the coordinates $(0, 0)$, $(0, 1)$, $(1, 0)$ and $(1, 1)$, so the average of these coordinates is $0.5 > 0$. A backward interpolation is the opposite and it is used to interpolate four orange points for example in the $x-z-$plane onto the black point with the global and local $x-z-$coordinates $(0, 0)$. The four neighbouring orange points have the local coordinates $(-1, -1)$, $(-1, 0)$, $(0, -1)$, $(0, 0)$ and their average is smaller than 0. The four orange neighbours of the purple point with global coordinates $(0.5, 0.5, 0)$ and local coordinates $(0, 0, 0)$ have the local coordinates of $(0, 0, 0)$, $(0, 1, 0)$, $(0, 0, -1)$ and $(0, 1, -1)$. Thus, both averages are 0 and there is a mixed interpolation, in this case it means there is a forward interpolation in y-direction and a backward interpolation in z-direction which is not possible in one step. To overcome this problem, one backward interpolation in the $x-z-$plane is executed and then a forward interpolation in the $x-y-$plane.

All necessary interpolation steps are displayed in table 1 where plus means forward and minus means backward interpolation for the spatial derivatives of the velocities to the stress components. At the empty spots no interpolation is necessary.

Table 1: **Interpolation directions:** Gives all required interpolations of the spatial derivatives of the velocities for the stress update. The combination of two letters gives the interpolation plane and the sign denotes forward (+) or backward (-) interpolation. For some components a forward and backward interpolation is required.

		spatial derivatives of velocities					
		v_{xx}	v_{zz}	v_{yy}	v_{yzzy}	v_{xyyx}	v_{xzzx}
stress components	σ_{xx}				$yz-$	$xy-$	$xz-$
	σ_{zz}				$yz-$	$xy-$	$xz-$
	σ_{yy}				$yz-$	$xy-$	$xz-$
	σ_{yz}	$yz+$	$yz+$	$yz+$		$xy-$ & $yz+$	$xz-$ & $yz+$
	σ_{xy}	$xy+$	$xy+$	$xy+$	$yz-$ & $xy+$		$xz-$ & $xy+$
	σ_{xz}	$yz+$	$yz+$	$yz+$	$yz-$ & $xz+$	$xy-$ & $xz+$	

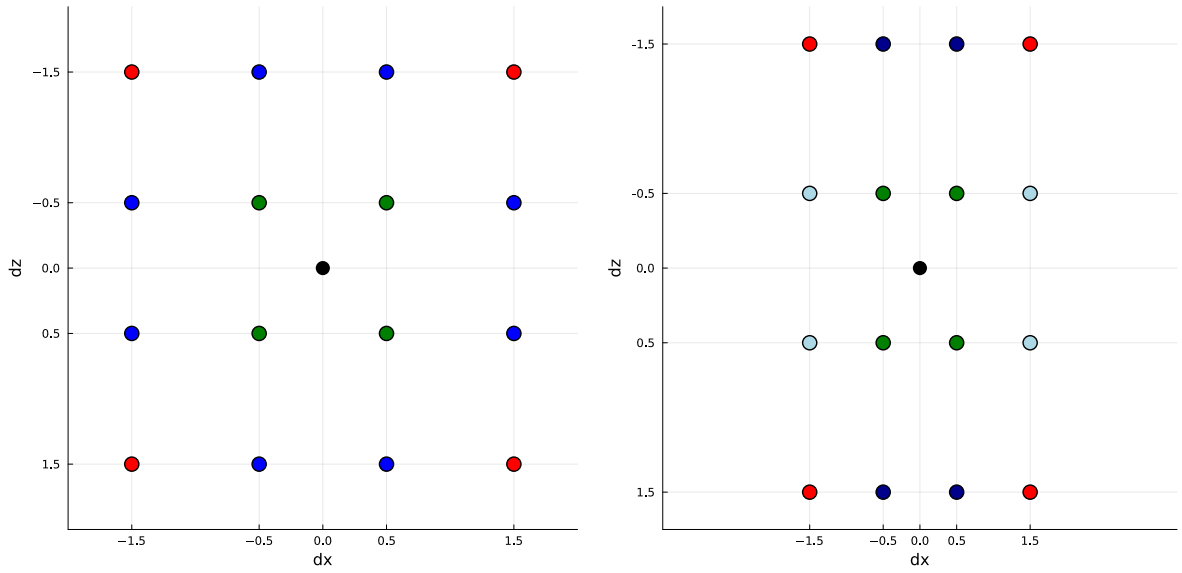
2.5.3 Interpolation weights

The number of points taken into account for each interpolation depends on the spatial FD order and is always equal to the FD order squared. An example for an interpolation for 6^{th} FD order is shown in Table 2 where all indices for 36 points are given. This interpolation is a forward interpolation in the $x-z-$plane and the index i belongs to the x-coordinate and index k to the z-coordinate. The interpolation weights depend on the distance to the target where w_{11} stands for the four closest neighbours. dx is defined by the size of the grid cells in x-direction and dz in z-direction. The grid spacing factors (dxz_{12} and dxz_{21}) are equal to 1 if the grid cells are cubes. If they are cuboids, eq. 36 and eq. 37 are used to calculate a weighting factor according to their distances to the target.

$$dxz_{12} = dxz_{13} = dxz_{23} = \frac{dz + dz}{dx + dz} \tag{36}$$

$$dxz_{21} = dxz_{31} = dxz_{32} = \frac{dx + dx}{dx + dz} \tag{37}$$

These weighting factors are calculated in a way that does not change the sum of the overall weights and is inversely proportional to the distance ratio to the target. Figure 4a shows an example, where $dz = dx$ and Figure 4b where $dz = 2dx$. All green points have the weight w_{11}, the blue points w_{12} and the red points w_{22}, but in figure 4b the distance of the dark blue and the light blue points to the center is not the same and therefore eq. 36 and eq. 37 are needed.



(a) (b)

Figure 4: **Interpolation weights uneven grid:** Points needed for a fourth-order interpolation in the $x-z-$plane. Points in the same color have the same weights contributing to the black point. a) $dz = dx$**:** All blue points have the same distance to the center. b) $dz = 2dx$**:** Light blue points are closer to the center compared to dark blue ones. This leads to different interpolation weights.

The interpolation weights are calculated by different methods. These two methods to interpolate wavefields used in this work are the following:

- sinc-function interpolation [Igel et al., 1995]
- FD-consistent interpolation [Koene et al., 2020]

Table 2: **Sixth order interpolation:** Shows all 36 required indices for a forward interpolation in the $x-z-$plane. The first column gives the interpolation weights and the second column the correction factor if dx is not equal dz. i iterates in x and k in z-direction.

weight	grid spacing	indices			
w_{11}		(i/k)	(i+1/k)	(i/k+1)	(i+1/k+1)
w_{22}		(i-1/k-1)	(i+2/k-1)	(i-1/k+2)	(i+2/k+2)
w_{33}		(i-2/k-2)	(i+3/k-2)	(i-2/k+3)	(i+3/k+3)
w_{12}	dxz_{12}	(i-1/k)	(i+2/k)	(i-1/k+1)	(i+2/k+1)
w_{12}	dxz_{21}	(i/k-1)	(i+1/k-1)	(i/k+2)	(i+1/k+2)
w_{13}	dxz_{13}	(i-2/k)	(i+3/k)	(i-2/k+1)	(i+3/k+1)
w_{13}	dxz_{31}	(i/k-2)	(i+1/k-2)	(i/k+3)	(i+1/k+3)
w_{23}	dxz_{23}	(i-2/k-1)	(i+3/k-1)	(i-2/k+2)	(i+3/k+2)
w_{23}	dxz_{32}	(i-1/k-2)	(i+2/k-2)	(i-1/k+3)	(i+2/k+3)

2.5.4 Sinc-function interpolation

The operator for the sinc-function interpolation [Igel et al., 1995] is defined by

$$d_{\overrightarrow{x}}(x) \quad = \quad \frac{\sin\left(k^{\max}\left(x+\frac{dx}{2}\right)\right)}{\pi\left(x+\frac{dx}{2}\right)} \tag{38}$$

where $k^{\max}$ is the maximum wave number and the interpolation can be discretized by

$$x_{n+\frac{1}{2}} = \left(n+\frac{1}{2}\right)dx \qquad \text{and} \qquad k^{\max} = \frac{\pi}{dx} \tag{39}$$

to obtain the discrete coefficients

$$d_{\overrightarrow{x}}(x_n) \quad = \quad \frac{(-1)^n}{\pi dx\left(n+\frac{1}{2}\right)} \tag{40}$$

Eq. 40 only shows a 1D interpolation, but the extension to 2D is just a multiplication by two of these factors. Figure 5 shows a sinc-function. If the distance is for example half a grid point in two directions, then the interpolation weight for this point can be obtained by the square of the amplitude at a distance of -0.5 or 0.5 grid points for a two-dimensional interpolation. The same accounts for a distance of 1.5 grid points in two directions. Looking at a point that is 0.5 grid points away from the target in one direction and 1.5 grid points away in the other direction, the two obtained amplitudes can again be multiplied to get the needed weight.

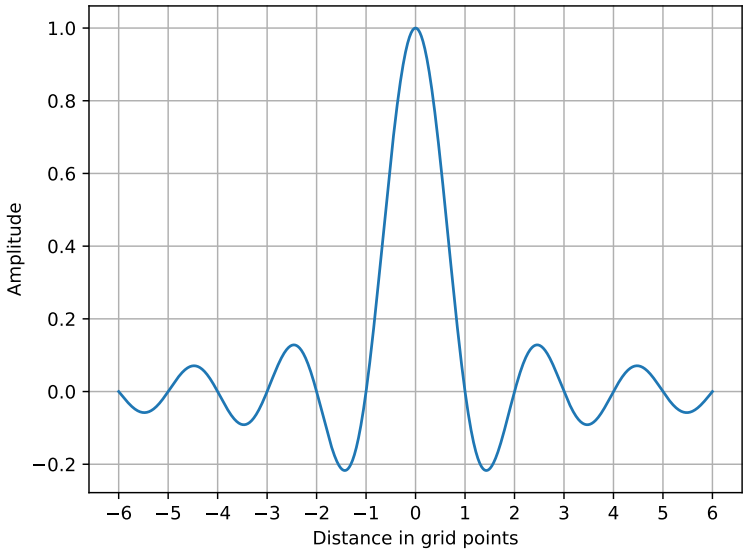


Figure 5: **Sinc-function:** Amplitude plot of a sinc-function with respect to the distance in grid points.

The sum of all interpolation weights are calculated for 2^{nd}, 4^{th} and 6^{th} spatial FD order respectively.

$$\begin{aligned} 2^{nd} \text{ order}: &\quad 4 \cdot \frac{4}{\pi^2} \approx 1.62 \\ 4^{th} \text{ order}: &\quad 4 \cdot \frac{4}{\pi^2} + 8 \cdot \frac{-4}{3\pi^2} + 4 \cdot \frac{4}{9\pi^2} \approx 0.72 \\ 6^{th} \text{ order}: &\quad 4 \cdot \frac{4}{\pi^2} + 8 \cdot \frac{-4}{3\pi^2} + 4 \cdot \frac{4}{9\pi^2} + 8 \cdot \frac{4}{5\pi^2} + 8 \cdot \frac{-4}{15\pi^2} + 4 \cdot \frac{4}{25\pi^2} \approx 1.22 \end{aligned} \tag{41}$$

For second FD order there are only four points with a distance of $(0.5, 0.5)$ grid points each and a weight of $\frac{4}{\pi^2}$. The sum of these weights is about 1.62. This is important to note because if the interpolation is normalized, it will significantly change the amplitudes. The normalization factor is defined by 1 over the sum of the interpolation weight according to the spatial FD order. The second line of eq. 41 shows a sum of the three different weights w_{11}, w_{12} and w_{22}, where each weight is multiplied by the number of points having these weights (similar to figure 4a). Figure 6 shows the sum of all weight depending on the spatial FD order. For high FD orders this function converges to 1. This would make the normalization factor redundant, but it is barely possible to reach FD orders high enough due to computing performance issues.

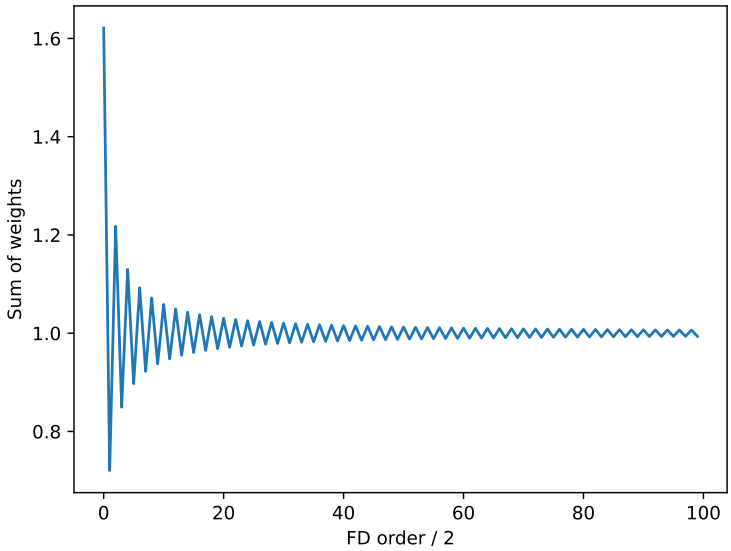


Figure 6: **Sinc-function weights:** Shows the sum of all interpolation weights with respect to the half FD order. Alternates around 1 and converges to 1.

The sinc-function is commonly used for interpolations of wavefields and the FD-consistent interpolation contains sinc-functions as well, but is more complex.

2.5.5 FD-consistent interpolation

The FD-consistent interpolation was developed to interpolate seismic sources located between grid points and depends on the FD-coefficients. This indicates that the interpolation method is applicable for wavefields, but in this work the method is used for the first time to interpolate the stress update and the results will be compared to the sinc-function interpolation. Since the FD-consistent interpolation was designed for such an application, a normalization is not required. The FD-consistent interpolation is defined by

$$\delta^{FDs}(x) = \sum_{l=1}^{L} \frac{\alpha_l (l-0.5)}{dx} \cdot \left(\mathrm{sinc}_\pi \left(\frac{x}{dx} + (l-0.5) \right) + \mathrm{sinc}_\pi \left(\frac{x}{dx} - (l-0.5) \right) \right) \tag{42}$$

and can be simplified to

$$\delta^{FDs}[j - j_s] = \begin{cases} \frac{\alpha_l (l-0.5)}{dx} & \text{with } |j - j_s| = l - 0.5, j_s + 0.5 \in \mathbb{Z} \\ 0 & \text{otherwise} \end{cases} \tag{43}$$

in this specific application [Koene et al., 2020]. The proof for this simplification is in appendix A and figure 7 indicates this already. The amplitude for $l = 1$ at a distance of -0.5 and 0.5 is 1 and all other curves are 0 at these distances. The same applies for $l = 2$ at -1.5 and 1.5 and so on.

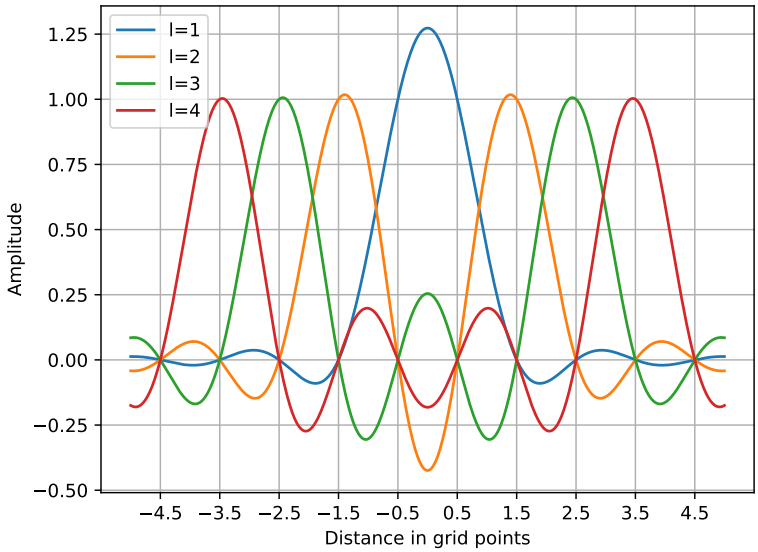


Figure 7: **FD-consistent interpolation function:** Shows the interpolation weights of the FD-consistent interpolation with respect to the distance in grid points. At every line of the vertical grid between -3.5 and 3.5 exactly one curve has an amplitude of 1 and all others are 0. l is the half FD order.

Using eq. 43 and the FD-coefficients the interpolation weights can be calculated. Again, a multiplication is sufficient to perform the interpolation in two dimensions. Similar to eq. 41 the sum of weights is calculated for the different spatial FD orders. The 2^{nd} order has one FD-coefficient $\alpha_1 = 1$ and $L = 1$. The 4^{th} order has two FD-coefficients $\alpha_1 = \frac{9}{8}$ and $\alpha_2 = \frac{-1}{24}$ and $L = 2$. For the 6^{th} FD order, there are three FD-coefficients $\alpha_1 = \frac{75}{64}$, $\alpha_2 = \frac{-25}{384}$ and $\alpha_3 = \frac{3}{640}$ and $L = 3$. This results in the following weights, where the fraction shows the interpolation weights and the factor ahead the number of points with that specific weight:

$$\begin{aligned} 2^{nd}\text{ order}: &\quad 4 \cdot \frac{4}{4} = 1 \\ 4^{th}\text{ order}: &\quad 4 \cdot \frac{81}{256} + 8 \cdot \frac{-9}{256} + 4 \cdot \frac{1}{256} = 1 \\ 6^{th}\text{ order}: &\quad 4 \cdot \frac{5625}{16384} + 8 \cdot \frac{-1875}{32768} + 4 \cdot \frac{625}{65536} + 8 \cdot \frac{225}{32768} + 8 \cdot \frac{-75}{65536} + 4 \cdot \frac{9}{65536} = 1 \end{aligned} \tag{44}$$

It should be noted that the sum of all weights is equal to 1 for every FD order, but the weights of the closest points to the target of the interpolation are changing for every FD order for the FD-consistent interpolation, but not for the sinc-function interpolation. For second FD order the weights for the interpolations are the same for both methods after the normalization of the sinc-function interpolation.

The implementation of all these theoretical concepts is explained and shown in appendix B. There will be some parts of the coding in IFOS3D in the programming language C and some parameters for different modelling methods are explained.

3 Model

Two different models were used to make simulations with the extended IFOS3D code. The first and mainly used model is a homogeneous anisotropic model and the second one a heterogeneous anisotropic model, inspired by a 3D synthetic Asse II model.

3.1 Homogeneous model

The homogeneous model contains five independent parameters and the density for the calculation of the VTI tensor. These values are displayed in table 3 and the calculated VTI elasticity tensor is in eq. 29. The values for the seismic velocities V_P and V_S are $5000\frac{m}{s}$ and $3000\frac{m}{s}$. These values make the calculation easier and the common ratio between V_P and V_S of $\sqrt{3}$ is almost fulfilled. The density in this model is equal to $2000\frac{kg}{m^3}$ which is in the range of common density of rocks. The Thomsen parameters ϵ and δ, table 3, are taken from the measured values for a Greenhorn shale with strong anisotropy [Jones and Wang, 1981]. Here a different γ is used because I wanted to reduce the SH-wave anisotropy to make easier comparisons to 2D simulations where γ is normally not used and in industrial applications SH-wave anisotropy is sometimes neglected. Therefore, I made some simulations with having γ set to 0 as well. For the extension to TTI two more parameters are needed. These are the rotation angles θ and ϕ, but the dip defined by θ is the only one of those existing in 2D simulations. A dip of 30° has been used for some previous 2D simulations, so I took the same value. ϕ was set to 20° for the main tests and to 90° so simulate no horizontal rotation. So, the horizontal rotation is 70° for ϕ=20°.

The size of each grid cell is $1m$ for all three directions resulting in cubes as all cells. There are 600 grid points in each direction, so the total number of grid points for this model is 216 million. The time step interval dt has the value $0.04ms$ and with a total simulation duration of $141ms$ this results in 3525 time steps.

Table 3: **Parameters homogeneous model:** Gives the model parameters for the homogeneous model. If there are two values, the first one is the mainly used.

model parameters	values
dx, dy, dz	$1m$
dt	$0.04ms$
V_P	$5000\frac{m}{s}$
V_S	$3000\frac{m}{s}$
ρ	$2000\frac{kg}{m^3}$
ϵ	0.26
γ	0.07 or 0.00
δ	-0.05
θ	$30°$
ϕ	$20°$ or $90°$
fc	$100HZ$ or $50HZ$
τ	$15ms$ or $30ms$

The source function for this model is a y-directed Ricker wavelet with a center frequency of $100Hz$ and a time delay of $\tau = 15ms$ to make the Ricker causal. The equation for this Ricker wavelet is defined by

$$r(t) \quad = \quad \left(1 - 2\pi^2 f_c^2 (t-\tau)^2\right) \cdot \exp\left(-\pi^2 f_c^2 (t-\tau)^2\right) \tag{45}$$

where $\tau = \frac{1.5}{f_c}$. For a center frequency f_c of $50Hz$ the resulting time delay is $30ms$. The variable t in this equation is the discrete time vector which contains all time values that are used for the simulation. The Ricker wavelet with a center frequency of $100Hz$ is displayed in figure 8 where the peak amplitude is located at $15ms$ (time delay).

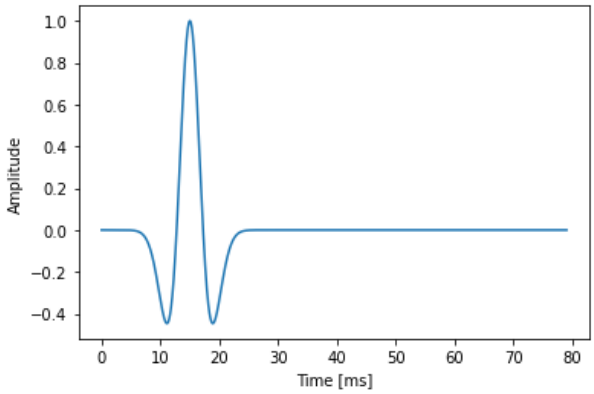


Figure 8: **Ricker wavelet:** Ricker wavelet with a center frequency of $100Hz$ and a time delay of $15ms$.

The source is located in the center of the grid at the grid point with the coordinates $(300, 300, 300)$. There are 26 three-component receivers placed around the source with a distance of at least $200m$ to the source and at least $100m$ to the edge of the model. The receivers cannot be too close to the border to avoid reflection of the edges of the model. To damp the reflections an absorbing boundary layer with 30 grid points width is placed. Noticeable is that the absorbing boundary is part of the model, so the propagating wavefields are influenced at grid points $0 - 30$ and $570 - 600$ of the model. The absorbing boundary uses exponential damping of 8% and is built after Cerjan et al. [1985].

The exact positions of all receivers are marked in figure 9. The receivers are placed in three layers where the middle one contains eight receivers and the source, the top and the bottom layer contain nine receivers each. The numbers for the receivers are for the traces in the results chapter.

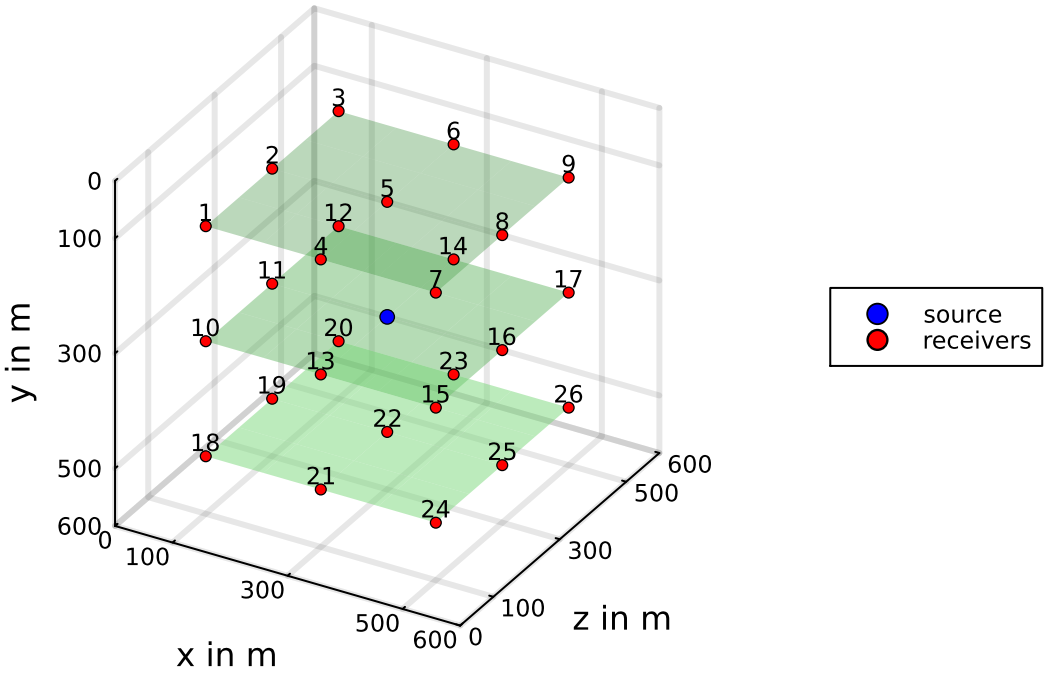


Figure 9: **Source and receivers:** Shows the source and 26 receivers with their specific number for the homogeneous model.

3.2 Heterogeneous model

The heterogeneous model has a grid spacing of $10m$ and a total size of $3800m$ inline (x-direction), $2000m$ crossline (z-direction) and $1300m$ depth (y-direction). This results in $380 \times 130 \times 200$ grid points, thus a total of 9.88 million grid points. The structure visible in the parameter plots in appendix C is similar to the Asse structure, but some values and distances may differ compared to the real structure. So, the results from this model cannot be compared to the reality and the structure is only used to see if the implemented code can deal with structures like this. Besides, the results from different simulation methods can still be compared with each other. The parameters in appendix C are V_P, V_S, ϵ and δ. These parameters vary over the grid and all other parameters are constant over the whole grid. The density was set to $2000\frac{kg}{m^3}$, γ to 0, θ to 65.1° and ϕ to 90°.

The source is again a Ricker wavelet following eq. 45 with a center frequency of $10Hz$ and a resulting time delay of $150ms$. The source is in the center of the model at the grid point with coordinates $(1900, 650, 1000)$ and the 26 receivers are placed again in three layers around the source, but this time the receiver layers are at a depth of $300m$, $650m$ and $1000m$ respectively. In horizontal direction all receivers have a distance of $400m$ to the edge of the model. So, no receiver is closer than $350m$ to the source and 300m to the edge of the model. The boundary has again a damping of 8%, but this time contains only ten grid points, since a boundary of 30 grid points would already interfere with some of the receivers for such a small model. Since the grid cells are larger the time step length can be chosen bigger compared to the homogeneous model. This time $dt = 0.1$ and the simulation duration is $1000ms$ resulting in 10000 time steps.

These two models are used to produce all the results in the next chapter.

4 Results

This chapter shows the results of the modelling and does some quality control and stability analysis. Furthermore, different modelling methods are compared and judged by looking at runtimes, memory usage and differences in amplitudes and arrival times of seismic waves.

All these results are created by serveral modelling methods. The methods for the homogeneous model are elastic isotropic modelling with already existing code in IFOS3D and SOFI3D, VTI anisotropic modelling, TTI ansiotropic modelling with the parameters of table 3 and TTI modelling with $\gamma = 0$ and $\phi = 90°$. Both TTI cases use one time the FD-consistent interpolation and the other time the sinc-function interpolation. This results in a total of six cases and all of these cases are done with 2^{nd}, 4^{th} and 6^{th} spatial FD order, respectively.

The heterogeneous model does not use isotropic modelling and there is only one TTI case. This leads to a total of four cases with three different spatial FD orders each.

4.1 Quality control

4.1.1 Reading the model

The data for the parameters of the homogeneous model is stored in binary format. After reading the model with the IFOS3D code, the parameters are checked if they contain the expected values at a lot of different spots on the grid to make sure every parameter is stored at the correct location. The TTI case needs to do the Bond transformation. This transformation is done with at least ten different combinations of angles θ and ϕ. Two of these combinations are eq. 30 and eq. 31. These elasticity tensors are once calculated by the IFOS3D implementation and separately by a program written by me in python and the results are always matching.

The data of the heterogeneous model is stored in seismic unix (SU) format. The anisotropic model can be read in both formats, but the elastic isotropic case can only be used in binary format. Again, the reading in of the data is checked by printing out some traces and comparing them to the input. For the homogeneous case the TTI elasticity tensor is the same at every grid point, but for the heterogeneous model the elasticity tensor has variation. This fact is not surprizing because the input parameters displayed in appendix C are not constant and this causes changes in the tensor.

The harmonic averaging of the elasticity tensor is controlled by checking the tensor at certain grid points after the bond transformation for both models. Eq. 35 is used to calculate the

expected value after the harmonic averaging. The same values are calculated by the IFOS3D code and the results are matching.

For both models the grid cells are cubic, but to make sure that modelling with non-cubic cells gives the same results, these values are alternated. The homogeneous model for example is tested with $dx = 1m$, $dy = 0.6m$ and $dz = 1.2m$, but there is no visible difference in the resulting traces for all 26 receivers.

4.1.2 Courant instability

The Courant instability [Courant et al., 1928] gives a value for the maximum time step length where the simulation is still stable. For a three-dimensional simulation the time step length fullfills the inequality

$$dt \quad \leq \quad \frac{dh}{\sqrt{3} v_{\max} \sum_{l=1}^{L} |\alpha_l|} \tag{46}$$

where dh is the minimum grid spacing of dx, dy and dz. $v_{\max}$ denotes the maximum velocity in the model and can be calculated by eq. 20 where θ is set to 90°. The sum of the FD-coefficients depends on the spatial FD order. The value of $v_{\max}$ is $6164\frac{m}{s}$ for the homogeneous and $6259\frac{m}{s}$ for the heterogeneous model. The sum of the FD-coefficients are 1, $\frac{7}{6}$, and $\frac{149}{120}$ for 2^{nd}, 4^{th} and 6^{th} spatial FD order, respectively. Since $\frac{149}{120}$ is the biggest value, the right-hand side of the equation is at its smallest for sixth-order. With $dh = 1m$ and $dh = 10m$ dt has to be equal or smaller than $0.073ms$ and $0.72ms$, respectively. This condition is fulfilled for both methods with dt of $0.04ms$ and $0.4ms$.

4.1.3 Energy

After controlling the reading of the models, the simulation over time is considered. Therefore, the energy in the whole grid is checked after every time step. The total energy can be calculated by

$$E(t) \quad = \quad \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \sum_{k=1}^{N_z} v_x^2 + v_y^2 + v_z^2 \tag{47}$$

where the triple sum goes over every grid point in all three dimension and for all times t of the simulation. The energy at one grid point at a certain time is calculated by the sum of the squares of the velocity components. These three velocity components represent time derivative of the ground movement in all three directions. If the algorithm is stable, the total energy should be fairly constant for every time step after the insertion of the source function. So, at the start the energy is 0 and at some point the energy increases until the whole source energy is in the model. If there are any instabilities or artefacts generated by the code for a certain method, the energy of this method should differ from the energy from stable methods. For this test the modelling is performed without an absorbing boundary to avoid damping and the loss of energy in the system.

Figure 10 shows the energy for every time step using the homogeneous model. The simulation runs for 13000 time steps. This is a lot longer than a normal simulation for a model like this ($3000-4000$ time steps) would require for the seismic waves reaching every spot of the grid. The idea behind this long simulation is to make sure the energy stays constant over a longer time. The blue curve shows fourth-order TTI modelling with a FD-consistent interpolation and the orange curve fourth-order VTI modelling. The two graphs are 0 at the start and then increase during the insertion of the source energy. When the whole energy is in the model, both curves stay relatively constant with some small variation, but no major instabilities. Apparently, the VTI case gets more energy from the source compared to the TTI case. A possible explanation is that the y-direction source interferes differently with the VTI and TTI layering.

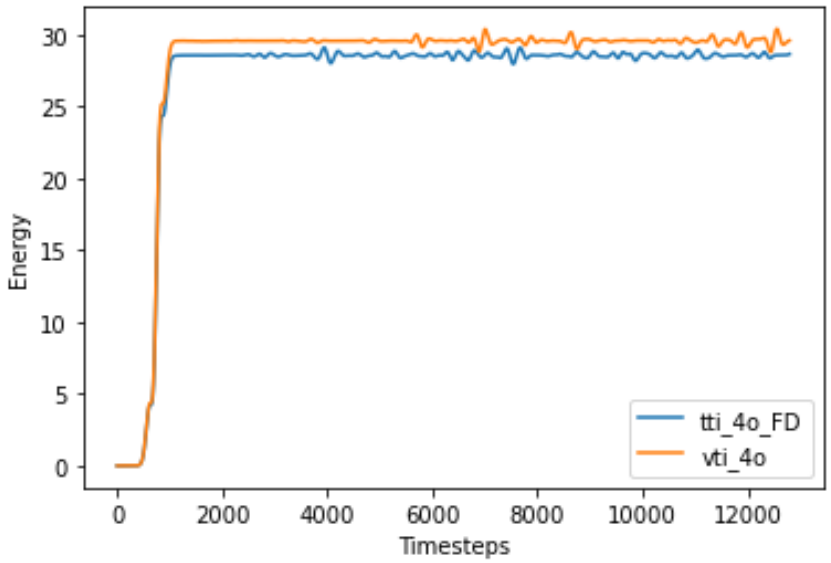


Figure 10: **Energy TTI vs. VTI:** Compares the total energy in the simulation for 13000 time steps of TTI fourth FD order with FD-consistent interpolation to VTI fourth FD order.

Figure 11 displays a similar energy analysis. It compares fourth-order FD-consistent interpolation with sixth-order sinc-function interpolation. The curve in orange is created by removing the whole interpolation. This modelling uses a value of the spatial derivatives from only one neighbouring grid point for the stress update. This leads to wrong results because the values of the spatial derivatives are at the wrong spot, when an interpolation is required. The only reason doing this is to compare the TTI results with a method that cannot have possible instabilities caused by an interpolation in the update of the wavefield. All three curves in Figure 11 are very close to each other and they are almost perfectly overlaying.So, none of the methods shows major instabilities. The energy is calculated for all methods mentioned at the beginning of this chapter, but since they are all very similar only some examples are displayed.

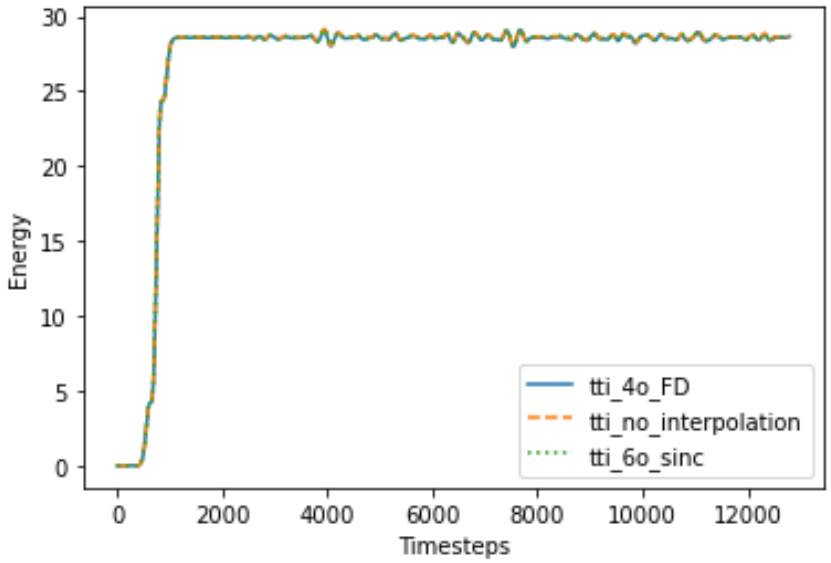


Figure 11: **Energy TTI vs. no interpolation:** Compares the total energy in the simulation for 13000 time steps of TTI fourth FD order with FD-consistent interpolation to sixth-order with sinc-function interpolation and TTI modelling without any interpolation

Figure 12 does an energy comparison between modelling with absorbing boundary and without. This time the heterogeneous model is used and the duration for the simulation is not extended. When the boundary is in place, the energy starts to decrease almost instantly after the source energy is fully inserted. This is caused by the size and the shape of the model since the source is placed at a depth of $650m$ and the boundary starts at $100m$ and $1200m$ depth. In x-direction the source is at $1900m$ and the boundary starts at $100m$ and $3700m$. Therefore, the damping starts at $550m$ wave propagation in y-direction and at $1800m$ in x-direction. The curve with a boundary in place constantly decreases after reaching its maximum like expected.

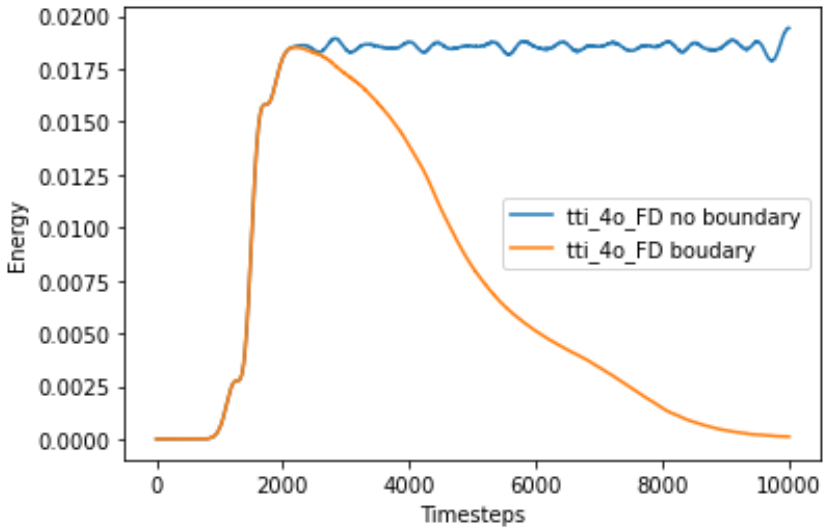


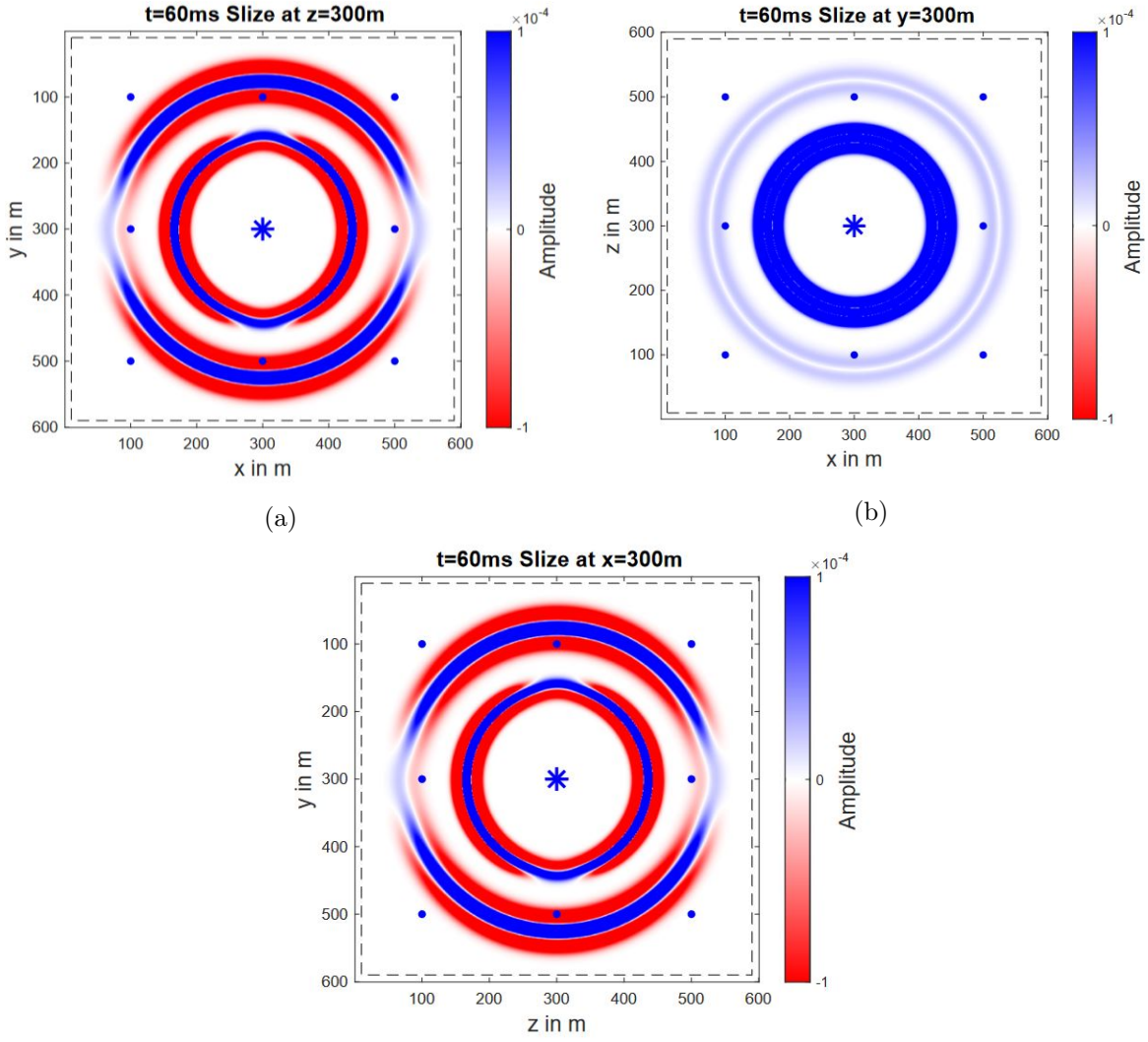
Figure 12: **Energy with boundary:** Shows the total energy in the simulation for 13000 time steps of TTI fourth FD order with FD-consistent interpolation with and without absorbing boundary.

The next section shows the wavefields of different modelling methods.

4.2 Snapshots of the wavefields

The snapshots for the homogeneous model are recorded after a modelling time of $60ms$ (1500 time steps). Since the center frequency of the source is $100Hz$, the peak of the ricker wavelet enters the model at a time of $15ms$ resulting in a travel time of $45ms$ for the peak. The snapshots are two-dimensional and there is a snapshot of the wavefield for every plane (xy, xz, yz). All snapshots contain the center of the model where the source is located which is displayed as a blue star. Besides, they all have eight receivers in every planed being marked as blue points. Thus, all three snapshots have one coordinate set to $300m$ and the waves only propagate in the other two directions. This makes it quite convenient to compare different cases like elastic isotropic, VTI and TTI.

The snapshots for elastic isotropic modelling with 6^{th} spatial FD order are in figure 13. The start of the absorbing boundary is marked by the dashed black lines. Figure 13a shows the slize of the $x-y-$plane at $z = 300m$, figure 13b the $x-z-$plane at $y = 300m$ and figure 13c the $y-z-$plane at $x = 300m$. This is consistent for all other snapshots in this work. For the elastic case the wavefields in the two vertical planes are the same and in the horizontal $x-z-$plane the wavefield differs because of the y-direction of the source. The horizontally propagating wavefield shows circles for the outer, faster P-wave and the S-wave. The other two wavefields in figure 13a and figure 13c have circular shape as well, but the y-direction of the source causes higher amplitudes for the P-wave in vertical than in horizontal direction. The S-wave has the higher amplitudes in horizontal direction. These effects are notable by comparing figure 13a where the amplitudes for P-waves and S-waves are similar to the ones in figure 13b where the S-wave has a lot higher amplitudes than the P-wave.



(a) (b) (c)

Figure 13: **Snapshot isotropic:** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-plane$, b) $x-z-plane$, c) $y-z-plane$. The star marks the source and blue points the receiver in the plane.

Figure 14 shows the wavefield snapshots for anisotropic VTI modelling with 6^{th} FD order. In the horizontal plane the P-wave velocity increases a lot compared to the isotropic case because of the strong anisotropy caused by ϵ. γ is quite small and therefore the SH-wave velocity barely increases and the S-wave wavefront looks similar to the isotropic case. For VTI both wavefields in the vertical plane look the same, but compared to the isotropic case the shape of the P-wave changes because again the P-wave is faster in horizontal direction.

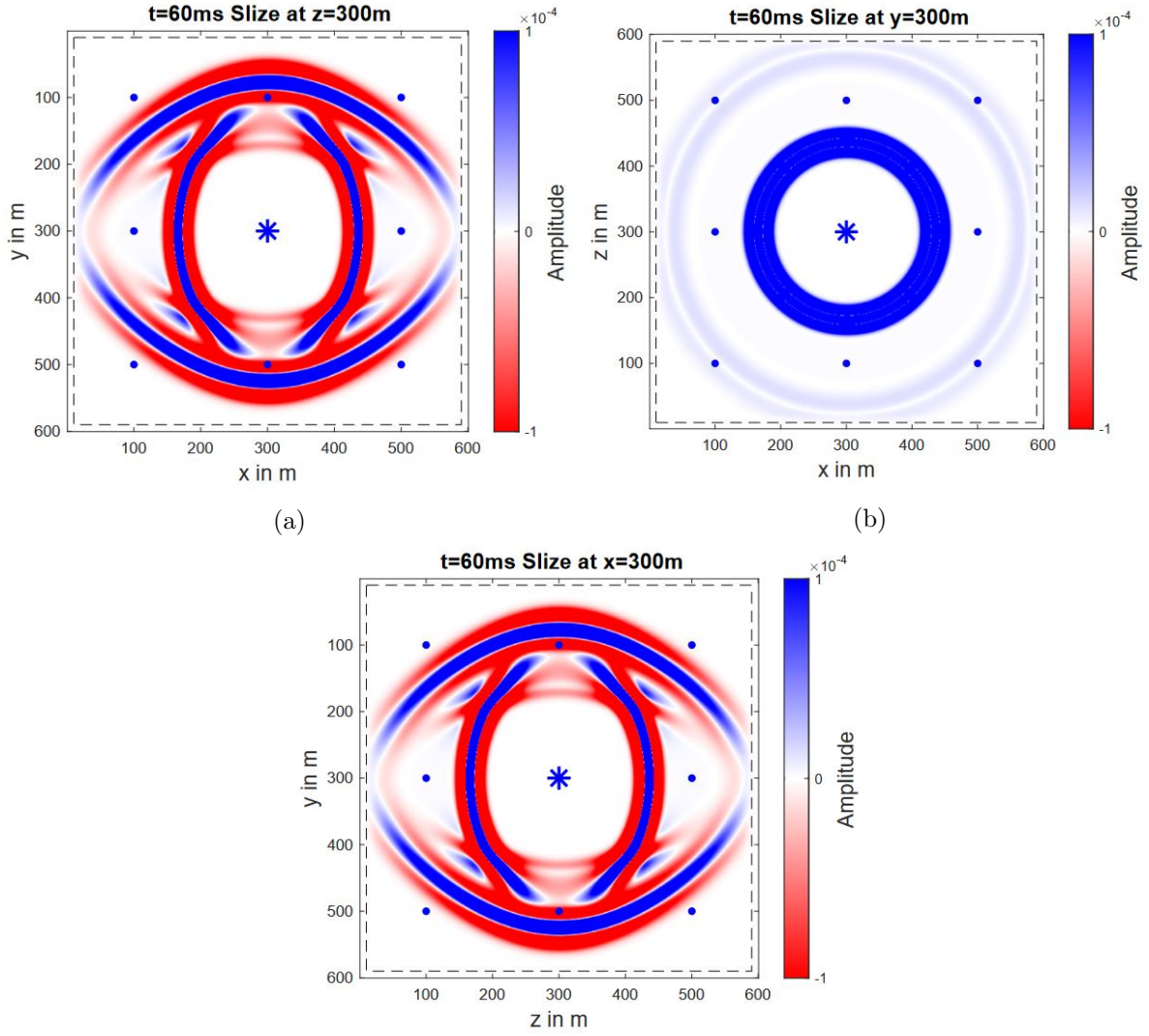


Figure 14: **Snapshot VTI:** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-$plane, b) $x-z-$plane, c) $y-z-$plane. The star marks the source and blue points the receiver in the plane.

The elasticity tensor of eq. 31 is used to model the 6^{th} FD order TTI snapshots in figure 15. Since the tensor is fully populated, there are velocity differences in every plane of the snapshots. Figure 15a and figure 15c are not symmetric in this case and even in horizontal direction the P-wave velocity is not constant. In these snapshots with both rotations it is challenging to estimate if these results are correct or not.

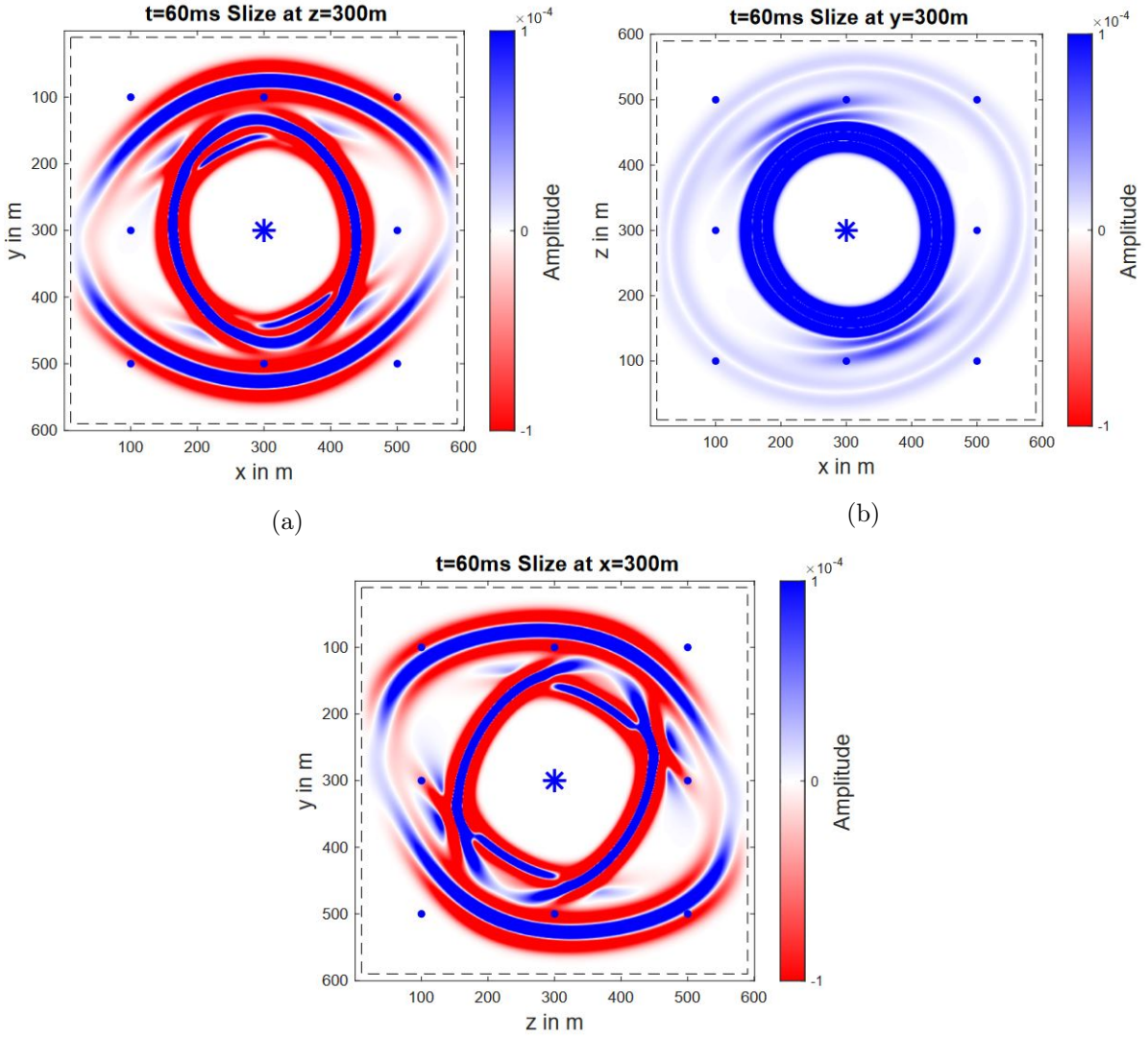


Figure 15: **Snapshot TTI:** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-$plane, b) $x-z-$plane, c) $y-z-$plane. Sixth FD order with FD-consistent interpolation is used The star marks the source and blue points the receiver in the plane.

For making a comparison with two-dimensional modelling possible, ϕ is set to 90° and γ to 0 because these two parameters do not exist in the 2D case. The elasticity tensor for this case is in eq. 30 and it is not fully populated. The results are displayed in figure 16. Figure 16a shows the anisotropy in the same way ($x-y-$plane) like 2D simulations previously done by Thomas Bohlen. The tilt of the structure compared to the VTI case is 30°. Figure 16b indicates faster traveling P-waves in z-direction compared to x-direction. Since there is only a rotation around the z-axis, the P-wave velocity in z-direction is the same velocity as in the VTI case in the horizontal plane. This velocity is the maximum velocity in the model. The P-wave velocity in x-direction becomes slower because of the tilt. This causes that the maximum velocity does not align anymore with the horizontal plane in this direction. Figure 16c shows a symmetric wavefield similar to figure 14c because there is not rotation angle ϕ.

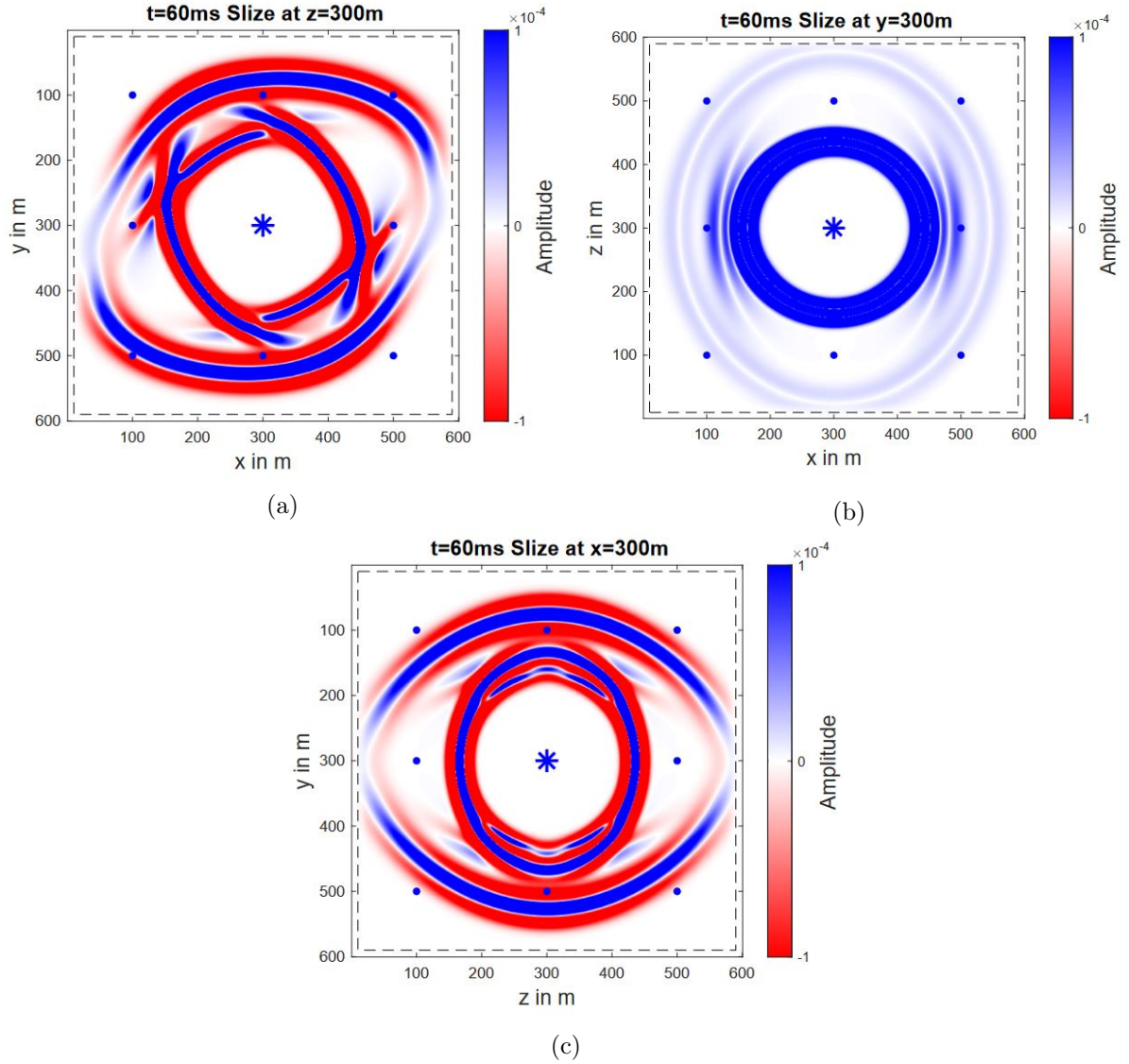


Figure 16: **Snapshot TTI (2D):** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-$plane, b) $x-z-$plane, c) $y-z-$plane. Sixth FD order with FD-consistent interpolation and $\phi = 90°$, $\gamma = 0$ is used. The star marks the source and blue points the receiver in the plane.

Figure 17 shows a TTI simulation without using any interpolation. This result is expected to be wrong, but in a comparison to figure 15 which uses the same elasticity tensor there are no visible differences. So, the snapshots do not have good enough resolution to compare TTI results produced by different interpolations or FD orders.

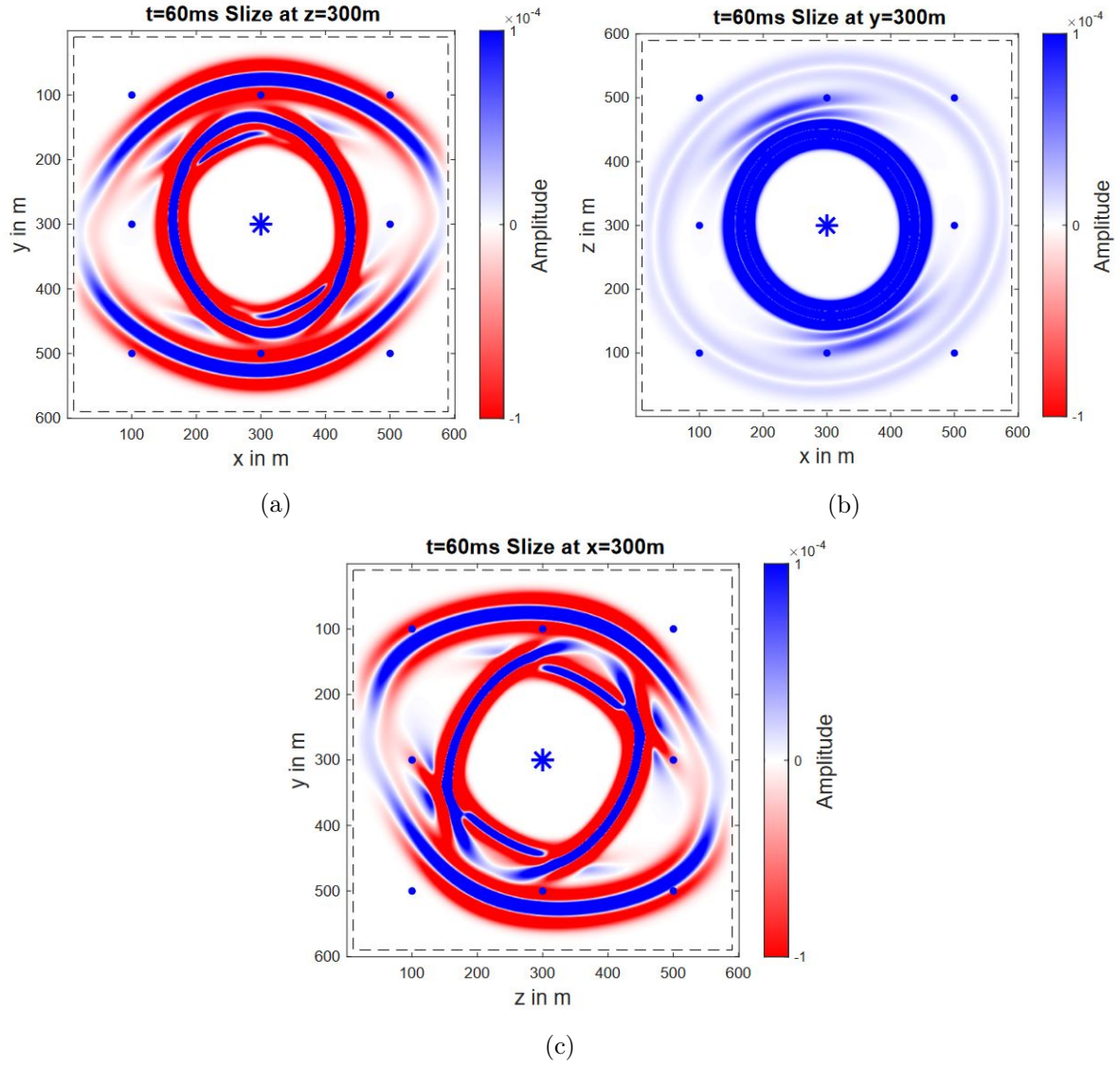


Figure 17: **Snapshot no interpolation:** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-$plane, b) $x-z-$plane, c) $y-z-$plane. The star marks the source and blue points the receiver in the plane.

Looking at snapshots for the heterogeneous model is even more challenging because the seismic velocities are stronger influenced by the heterogeneous velocity structures (appendix C) than by the anisotropy. Figure 18 is an example of 6^{th} order TTI for this model simulated with FD-consistent interpolation. The P-waves and S-waves travel faster downwards than upwards because deeper layers have higher velocities. Since the model is relatively small by the number of grid points and the frequency has to be quite low to be stable, the separation of different wave types is not that strong, but already visible in all three plots of figure 18.

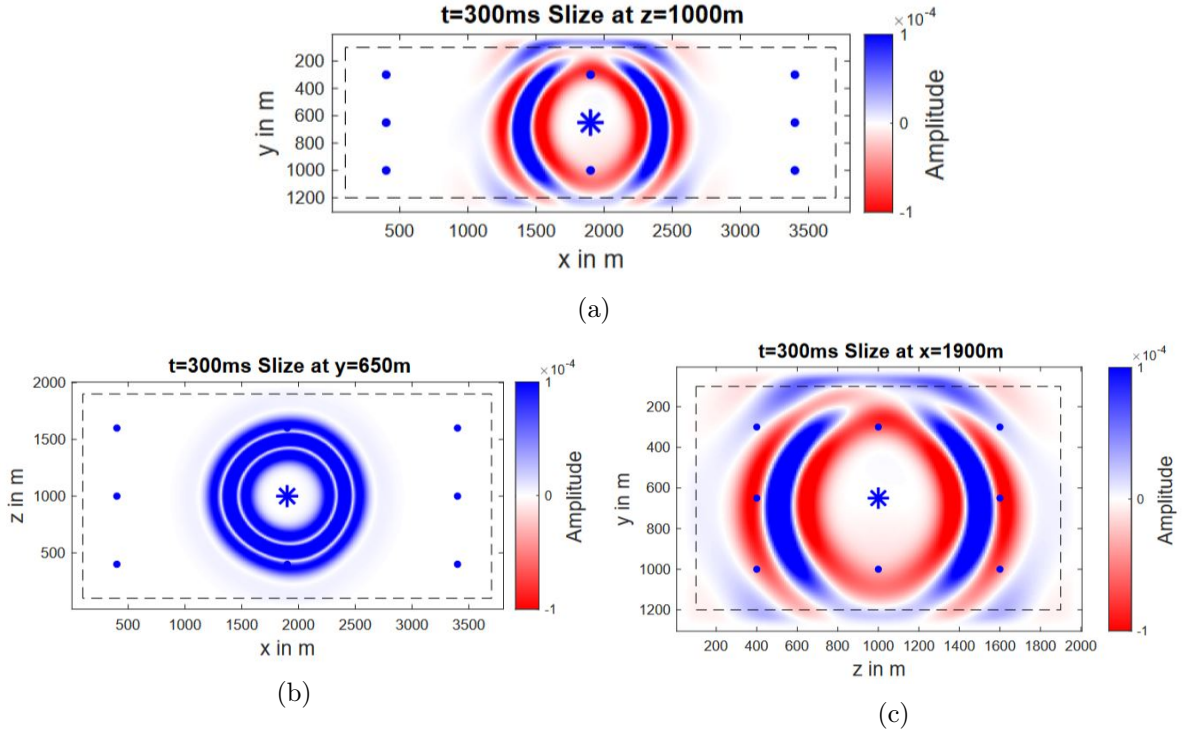


Figure 18: **Snapshot heterogenious TTI:** Shows the wavefield after a simulation time of $60ms$ for a cut through the center of the model in each plane: a) $x-y-$plane, b) $x-z-$plane, c) $y-z-$plane. Sixth FD order with FD-consistent interpolation is used. The star marks the source and blue points the receiver in the plane.

4.3 Seismic traces

The receiver geometries in this section are consistent to the model chapter. The receiver numbers are referring to figure 9 for the homogeneous model and for the heterogeneous model the setup is the same, but the distances are different. In certain figures receiver 9 is shown specifically. Thereby, receiver 9 is a reasonable choice because it has the coordinates $(500m, 100m, 500m)$ and none of the coordinates equals the source $(300m, 300m, 300m)$. Therefore, the waves have to travel in all three dimensions of the model and this lets all wave types contribute to the amplitudes. All following figures show the y-component of the receivers because of the y-directed source.

Figures 19-21 - all being displayed in appendix C - show all 26 receivers for the homogeneous model and all three figures use 6^{th} spatial FD order. Figure 19 is the isotropic elastic case where the first clear visible P-wave onsets are at receivers 5 and 22 because they are directly above and below the source. At the same time P-waves arrive at receivers 11, 13, 14, and 16, but their amplitudes on the y-component are very small compared to the S-wave amplitudes. All these receivers are $200m$ away from the source and with a P-wave velocity of $5000\frac{m}{s}$ and a time delay of $15ms$ of the source, the peaks of the P-wave arrive at $55ms$. The waveforms in the traces all have the shape of a Ricker wavelet because the only energy input into the model is a Ricker wavelet and there is no anisotropy to separate waveforms or cause interference. Furthermore, figure 19 shows a lot of symmetry as expected for isotropic homogeneous modelling since all receivers with the same distance to the source have common arrival times for P- and S-waves.

The VTI case in Figure 20 shows that the peak arrival times for receivers 5 and 22 are still at $55ms$, but all other receivers have earlier arrival times since the P-wave velocity increases from $5000\frac{m}{s}$ to $6164\frac{m}{s}$ (vgl. eq. 20). The only receivers with similar arrivals for the S-waves are located in the same horizontal plane as the source (receivers $10 - 17$). Their SV-wave velocity stays at $3000\frac{m}{s}$ (vgl. eq. 21, Figure 2), but the SH-wave velocity increases slightly.

Figure 21 displays the results of 6^{th} order TTI modelling and the trace recordings look more complex. There is still some symmetry since receiver 1 has the same trace as receiver 26 and receiver 2 as 25 and so on, but the arrivals for all other traces differ depending on the direction from the source to the receiver. Receivers 2, 4, 6 and 8 for example have all the same distance to the source and they are located in the same horizontal plane above the source, but they have varying arrival times for the P-waves caused by the tilt of the fastest plane for P-waves away from all coordinate axes. On some receivers shear wave splitting is present (receivers 2, 3, 9, ...).

A comparison of VTI and TTI is done in figure 22a for all traces and in figure 22b for trace 9 only. For the receivers in the same horizontal plane compared to the source, the S-waves arrive earlier for TTI because the SV-waves are the slowest in horizontal travel direction for VTI. The VTI SV-wave are the fastest if the angle to the vertical symmetry axis is approximately 40° (calculated by eq. 21). The travel path from the source to receivers 2, 4, 6, 8, 19, 21, 23 and 25 is 45°. Since these values are quite close to 40°, the SV-wave velocity is almost maximum in these directions and causes earlier arrivals compared to TTI. The only exception are receivers 2 and 25. After the rotation of the symmetry axis for the TTI case by $\theta = 30°$ and $\phi = 20°$, the angle between this new symmetry axis and the raypath is 42.8°. This value is even closer to the maximum SV-wave velocity and therefore the earlier arrivals for the TTI case are justified. Receiver 9 shows an earlier peak for the P-wave, but a later peak for the S-wave for VTI. For both cases the P-wave shows a Ricker waveform, but the TTI case has stronger interference between the different S-waves.

Figure 23 compares second-order TTI with FD-consistent interpolation to sixth-order with the same interpolation method. When looking at all traces the difference is quite small, but already visible for the large S-wave amplitudes in traces 2, 10, 13 and 14 for example. By zooming into trace 9 in figure 23b differences become more visible that are unnoticed in figure 23a. The P-wave still looks the same, but the S-wave has some differences in the waveform and amplitudes. Thus, the spatial FD order influences the results.

A similar comparison is done for the two interpolation methods, figure 24. Again, there are differences in the results already visible in the figure with all traces. For trace 9 the S-wave amplitudes vary at similar spots compared to figure 23b, but there are differences. Since in both cases, figure 23b and figure 24b, the black curve is the same, but the amplitudes of the red curves overshoot at different spots.

In figure 25b the black curve is again created by 6^{th} order FD modelling with FD-consistent interpolation and it is compared to modelling without using any interpolation. Therefore, it can be assumed that the red curve is wrong at some spots else the whole interpolation and the calculation on the correct grid points would not be necessary. The difference here is even bigger than in previous comparisons, but the result is still similar.

Figure 26 shows the 26 receivers for the heterogeneous model where 6^{th} order TTI modelling with FD-consistent interpolation is used to create these traces. The arrival times vary quite strongly because of the model shape and therefore all eight receivers having the same x-coordinate as the source have earlier onsets. For this model receivers 1 and 26 do not record the same. This is caused by heterogenety of V_P and V_S for this model. Since the velocities are higher for larger depths, the receivers in the lower plane $(18 - 26)$ have earlier arrivals. There are some small amplitudes in the traces after the main arrivals and they are probably caused by a too small boundary. This issue is hard to avoid because the model is so small. For the absorbing 30 points are recommended by Cerjan [1985], but here are only 10 in place because the receivers have only a distance of 30 grid points to the edge of the model.

The differences between FD-consistent and sinc-function interpolation are in figure 27. The plot with all traces has no big differences, but after taking a closer look at trace 9 the differences become more clear. This shows that for both models the two interpolations give slightly different results.

These differences will be quantified in the following section.

4.4 Error analysis

Since it is almost impossible to visually compare the differences in the synthetic seismic traces, I defined an error function for the relative error of a specific trace (receiver 9) and one for all 26 traces combined. For the homogeneous model these two functions are

$$E_{rel} = \frac{\sum_{i=1}^{3526} |M1_{i9} - M2_{i9}|}{\sum_{i=1}^{3526} |M2_{i9}|} \tag{48}$$

$$E_{tot} = \frac{\sum_{j=1}^{26} \sum_{i=1}^{3526} |M1_{ij} - M2_{ij}|}{\sum_{j=1}^{26} \sum_{i=1}^{3526} |M2_{ij}|} \tag{49}$$

where $M1, M2$ are 3526×26 matrices. The first function calculates the absolute difference between every sample (3526) of two traces. These traces both belong to receiver 9, but they are calculated by different modelling methods. Then these absolute differences are summed and they are divided by the absolute sum of the amplitudes of one trace (reference trace). Thus, this method gives a relative error between two traces related to the total amplitudes. The same is done for all traces by the second equation of this section.

Table 4 gives the relative error for the trace of receiver 9 and for all traces. Method 2 is always the reference for the error calculation. There are 15 comparisons between two methods for the homogeneous model and are numbered to make referring easier. $i1$ stands for FD-consistent, $i3$ for sinc-function, and no i for no interpolation.

For elastic isotropic modelling and for VTI modelling the second order differs by a lot more than the fourth-order from the sixth-order. Number 1 and 3 in the table are by a factor of about 100 larger than number 2 and 4. This shows that the 4^{th} and the 6^{th} FD order are very similar because the differences in resulting amplitude are below 0.1%. Number 5 and 6 do the same comparison for TTI with FD-consistent interpolation. Interestingly, the differences are even a bit smaller between different spatial FD orders compared to VTI and isotropic cases. The sinc-function interpolations with number 7 and 8 have larger errors and between 4^{th} and 6^{th} order the differences stay big. That indicates that even for a homogeneous model higher FD orders could be required.

The difference between second order of both interpolation methods is 0 because they are the same, but for higher order produce give different results. Number 12 and 13 compare the no interpolation case to both interpolations and the errors are bigger than for number 11 where the two interpolations are compared. This indicates that neglecting the interpolation is no option to obtain good results.

If different methods like VTI and TTI get compared, the difference is obviously huge because the arrival times already differ and the amplitudes do not overlay. So trying to model a homogeneous medium with strong dipping anisotropy with VTI modelling is not possible.

Table 4: **Error homogeneous:** The first column gives a reference number for every comparison. Method 1 is compared to method 2 where method 2 is the reference. Column 4 gives the differences in trace 9 and column 5 the differences for all 26 traces

number	method 1	method 2	E_{rel} trace 9 in %	E_{rel} total in %
1	elastic 2o	elastic 6o	8.54	11.17
2	elastic 4o	elastic 6o	0.04	0.10
3	VTI 2o	VTI 6o	8.03	9.43
4	VTI 4o	VTI 6o	0.04	0.08
5	TTI 2o i1	TTI 6o i1	5.80	7.22
6	TTI 4o i1	TTI 6o i1	0.03	0.05
7	TTI 2o i3	TTI 6o i3	10.46	13.55
8	TTI 4o i3	TTI 6o i3	10.25	15.26
9	TTI 2o i3	TTI 2o i1	0.00	0.00
10	TTI 4o i3	TTI 4o i1	4.93	7.43
11	TTI 6o i3	TTI 6o i1	5.30	7.88
12	TTI no i	TTI 6o i1	7.11	9.03
13	TTI no i	TTI 6o i3	11.90	15.43
14	elastic 6o	TTI 6o i1	173.14	159.73
15	VTI 6o	TTI 6o i1	170.45	171.39

Similar error calculations are done for the heterogeneous model. The numbers in table 5 are the same as in table 4, but the elastic isotropic case is not considered. The differences between fourth and sixth FD order are again far below 1% for VTI and TTI with FD-consistent interpolation. Again, the sinc-function interpolation is less consistent because the differences are still over 4%.

By looking at the errors of the sinc-function interpolation for both models for different FD orders, the error is quite large. The error between the FD-consistent and the sinc-function interpolation is always smaller in comparison. Since the FD-consistent error between fourth and sixth FD order is almost 0, the traces almost do not change. These facts indicate that the results of the sinc-function interpolation are "jumping" around the more constant FD-consistens results. A reason and explanation for this phenomenon could be figure 6, since the sinc-function is strongly alternating for low FD orders. There is a normalization factor in place but it is maybe not enough to completely avoid this effect.

Table 5: **Error heterogeneous:** The first column gives a reference number for every comparison. Method 1 is compared to method 2 where method 2 is the reference. Column 4 gives the differences in trace 9 and column 5 the differences for all 26 traces

number	method 1	method 2	E_{rel} trace 9 in %	E_{rel} total in %
3	VTI 2o	VTI 6o	10.58	6.85
4	VTI 4o	VTI 6o	0.13	0.16
5	TTI 2o i1	TTI 6o i1	9.02	5.74
6	TTI 4o i1	TTI 6o i1	0.16	0.14
7	TTI 2o i3	TTI 6o i3	5.62	4.67
8	TTI 4o i3	TTI 6o i3	8.45	4.36
9	TTI 2o i3	TTI 2o i1	0.00	0.00
10	TTI 4o i3	TTI 4o i1	4.14	2.14
11	TTI 6o i3	TTI 6o i1	4.42	2.22
12	TTI no i	TTI 6o i1	8.36	5.88
13	TTI no i	TTI 6o i3	5.27	4.92
15	VTI 6o	TTI 6o i1	204.79	103.19

4.5 Runtime and memory usage

To compare runtimes and memory usages the isotropic elastic, the VTI, and the TTI case were all simulated with second, fourth, and sixth FD order. The model was the homogeneous model with 600^3 grid points for 3000 timesteps on 125 CPUs. All these simulations are executed at least ten times and outliers are neglected. The ten runtimes are usually within 2%, but if some other programs run on the same nodes, the performance reduces drastically and therefore these outliers are not considered.

Table 6 shows the runtimes and memory usages for the methods. The relative runtime and the relative memory is always in perspective to elastic second order modelling. The runtime for isotropic and VTI modelling is almost the same but VTI need approximately 30% more memory. TTI needs about twice the memory of VTI and the runtimes are 6 to 8 times as long. The memory for higher FD orders increases only slightly, but the runtime already increases by a factor of almost 2 between second and sixth-order for isotropic and VTI simulations. For TTI this factor is about 2.5. In total TTI modelling drastically increases the runtime since a lot of interpolation are done for every timestep.

Table 6: **Runtime and memory:** Gives the absolute, relative runtime for the homogeneous model with 600^3 grid points on 125 CPUs for 3000 time steps for different modelling methods. The total memory usage for every method is shown as well. Every simulation is done at least 10 times and outliers are neglected.

method	runtime in min	relative runtime	memory in GB	relative memory
elastic 2^{nd} order	5.0	1.0	27.4	1.0
elastic 4^{th} order	6.5	1.3	28.4	1.04
elastic 6^{th} order	9.2	1.84	30.3	1.11
VTI 2^{nd} order	5.1	1.02	35.8	1.31
VTI 4^{th} order	7.0	1.4	37.7	1.38
VTI 6^{th} order	9.4	1.88	39.5	1.44
TTI 2^{nd} order	30.8	6.16	68.3	2.49
TTI 4^{th} order	50.0	10.0	72.5	2.65
TTI 6^{th} order	76.9	15.38	74.7	2.73

5 Conclusions

The new implemented code seems to be numerically stable as the fairly constant energy indicates. These energies are constant for all methods, independent of spatial FD order or the type of modelling like VTI and TTI with both interpolations. Both models fulfill this stability indicator.

The snapshots of the wavefields show expected results, but a detailed analysis is not possible. The seismic traces show a lot more details and make comparisons of traces for different methods possible. The differences between VTI and TTI are consistent and indicate that TTI modelling is needed for tilted anisotropy. By comparing different spatial FD orders for different methods, for both models fourth-order almost gets the same results as sixth-order. This can be convenient since the runtime increases a lot with higher FD orders. The exception is the sinc-function interpolation because for this method fourth-order and sixth-order differ for both models.

The FD-consistent interpolation performs better than the sinc-function interpolation, because the results for different FD orders are almost the same and a normalization is not required. Therefore, I can recommend using the FD-consistent interpolation for this application, but further tests are still required.

Possible next steps can be to run the code with a fully heterogeneous anisotropic model and to compare results. Another point is to proof the implementation and especially the FD-consistent interpolation with an analytical solution. This would show an absolute error and not only an error by comparing different methods with each other.

6 Appendix

6.1 Appendix A

Let's take the formula

$$\delta^{FDs}(x) \quad = \quad \sum_{l=1}^{L} \frac{\alpha_l(l-0.5)}{dx} \cdot \Big(\mathrm{sinc}_\pi\Big(\frac{x}{dx} + (l-0.5)\Big) + \mathrm{sinc}_\pi\Big(\frac{x}{dx} - (l-0.5)\Big)\Big)$$

Let $j \in \mathbb{Z}$ be the number of a node with the value $j \cdot dx$, where dx is the length of the grid. Let $j_s \in 0.5 + \mathbb{Z}$ be the number of a source with the value $j_s \cdot dx$. Now put the values of the "node number difference" $j - j_s$ into the forumula:

$$\begin{aligned}
\delta^{FDs}[j-j_s] \quad &= \quad \sum_{l=1}^{L} \frac{\alpha_l(l-0.5)}{dx} \cdot \Big(\mathrm{sinc}_\pi\Big(\frac{jdx - j_s dx}{dx} + (l-0.5)\Big) + \mathrm{sinc}_\pi\Big(\frac{jdx - j_s dx}{dx} - (l-0.5)\Big)\Big) \\
&= \quad \sum_{l=1}^{L} \frac{\alpha_l(l-0.5)}{dx} \cdot \Big(\mathrm{sinc}_\pi\Big(j - j_s + (l-0.5)\Big) + \mathrm{sinc}_\pi\Big(j - j_s - (l-0.5)\Big)\Big) \\
&= \quad \sum_{l=1}^{L} \frac{\alpha_l(l-0.5)}{dx} \cdot \Big(\mathrm{sinc}_\pi(b) + \mathrm{sinc}_\pi(c)\Big)
\end{aligned}$$

where b, c are defined as

$$b = j - j_s + (l-0.5) \quad \text{and} \quad c = j - j_s - (l-0.5)$$

Then $b, c \in \mathbb{Z}$ $(\star)$ because $j, l \in \mathbb{Z}$ and $j_s, 0.5 \in 0.5 + \mathbb{Z}$. Taking a closer look at the $sinc_\pi$ function:

$$\mathrm{sinc}_\pi(j - j_s + (l-0.5)) \quad = \quad \mathrm{sinc}_\pi(b) \quad = \quad \frac{\sin(\pi \cdot b)}{\pi \cdot b} \quad \overset{(\star)}{=} \quad \frac{0}{\pi \cdot b}$$

This is only well-defined for $b \neq 0$ and then has the value 0. If b is getting very close to 0, the $sinc_\pi$ function has the following value:

$$\lim_{b \to 0} \mathrm{sinc}_\pi(b) \quad = \quad \lim_{b \to 0} \frac{\sin(\pi \cdot b)}{\pi \cdot b} \quad \overset{L'Hospital}{=} \quad \lim_{b \to 0} \frac{\pi \cdot \cos(\pi \cdot b)}{\pi} \quad = \quad 1$$

With the consideration of the limit for $b = 0$ which is equivalent to $j - j_s + (l - 0.5) = 0$ and thus equivalent to $-(j - j_s) = l - 0.5$. So again in consideration of taking the limit

$$\text{sinc}_\pi(b) = \begin{cases} 0 & \text{for } -(j - j_s) \neq l - 0.5 \\ 1 & \text{for } -(j - j_s) = l - 0.5 \end{cases}$$

Taking a look at the second $sinc_\pi$ function of the formula. This is completely analogous with the argument c instead of b. Having the Equivalence $c = 0 \Leftrightarrow j - j_s - (l - 0.5) = 0 \Leftrightarrow j - j_s = l - 0.5$ and taking at $c = 0$ the limit

$$\text{sinc}_\pi(c) = \begin{cases} 0 & \text{for } j - j_s \neq l - 0.5 \\ 1 & \text{for } j - j_s = l - 0.5 \end{cases}$$

In summary there are 3 cases now:
<u>case 1</u>: $j - j_s = l - 0.5$

$$\begin{aligned} \delta^{FDs}[j - j_s] &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(\text{sinc}_\pi(b) + \text{sinc}_\pi(c)\Big) \\ &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(0 + 1\Big) \\ &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \end{aligned}$$

<u>case 2</u>: $-(j - j_s) = l - 0.5$

$$\begin{aligned} \delta^{FDs}[j - j_s] &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(\text{sinc}_\pi(b) + \text{sinc}_\pi(c)\Big) \\ &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(1 + 0\Big) \\ &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \end{aligned}$$

<u>case 3</u>: $j - j_s \neq l - 0.5$ and $-(j - j_s) \neq l - 0.5$

$$\begin{aligned} \delta^{FDs}[j - j_s] &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(\text{sinc}_\pi(b) + \text{sinc}_\pi(c)\Big) \\ &= \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx} \cdot \Big(0 + 0\Big) \\ &= 0 \end{aligned}$$

So for $|j - j_s| = l - 0.5$ - case 1 and case 2 -

$$\delta^{FDs}[j - j_s] \quad = \quad \sum_{l=1}^{L} \frac{\alpha_l (l - 0.5)}{dx}$$

and for $|j - j_s| \neq l - 0.5$ - case 3 -

$$\delta^{FDs}[j - j_s] \quad = \quad 0$$

6.2 Appendix B

All the major changes compared to the elastic case happen in the folder src of the IFOS3D code. There is a new reading method for the anisotropic modelling called readmod_elastic_aniso.c. This file reads the model parameters if ansiotropic modelling is selected by the WEQ-parameter. The options for WEQ are 1 = elastic isotropic, 2 = acoustic, 4 = VTI and 6 = TTI, so for 4 and 6 the new readmod function will be used. This function can read from binary and seismic unix files. For VTI the values of the elasticity tensor are calculated by eq. 15-19 and stored in the model struct. For TTI case the required Bond transformation is done by the readmod function and the fully populated tensor is stored in the model struct. The file av_mat.c averages the model parameters of the VTI and TTI tensor. The averaging is done by eq. 35 for the bottom three rows of the elasticity tensor. For the VTI case this requires only three calculations compared to the 18 of TTI.

In the timeloop the velocity update is not changed, because the isotropic and the anisotropic case require the same calculations. By selecting WEQ = 4 the VTI stress update is selected (update_s_ssg_elastic_aniso_VTI.c) and for WEQ = 6 the TTI stress update (update_s_ssg_elastic_aniso_TTI.c). WEQ = 6 can theoretically be used for anisotropy with 21 independent parameters as well, where the only change is in the reading of the model. The VTI stress update does not require any wavefield interpolations, because all values that need an interpolation are 0 acording to Table 1. Therefore, one set of loops over the whole grid for each timestep is enough. In the TTI case interpolations and even double interpolations are necessary. This makes the stress update a lot more complicated, since some of the spatial derivatives of the velocities need to be interpolated for every timestep. For the double interpolations first the spatial derivatives of the velocities are calculated and stored in the first set of loops over the whole grid and then one interpolation is done over the whole grid and the interpolated values are stored in a second set of loops. After this in another set of loops over the grid the second interpolation is done and then the stress can be updated. Thus, for the TTI stress update a separation in 3 sets of loops over the whole grid for every timestep is necessary.

The file interpolate_4_FD.c contains all the interpolation functions that are used for the stress update. There is an interpolating function for each of the three planes and by setting the parameter m to -1 or 1 a forward or backward interpolation is performed. A different set of interpolation functions is used according the spatial FD order. This file is the same for FD-consistent and sinc-function interpolation, because only the parameters used in this function differ. These pa-

rameters are calculated in interpolate_4_FD_weights.c. The weights ($w11, w12, w22, w23, w33$) are taken from eq. 44 for FD-consistent and eq. 41 for sinc-function interpolation. The weighting factors of eq. 36 and eq. 37 are calculated as well as the sinc-function normalization factor. The parameter INTP_TYPE decides which weight calculation is done. INTP_TYPE=1 (default) means FD-consistent interpolation and INTP_TYPE=3 sinc-function interpolation. The values 2 and 4 exist as well, but they are not recommended, since they use the wavefield interpolations to average the model, so they are mostly made for testing and they will probably be excluded at some point.

Now some parts of the files above are shown for 4^{th} spatial FD order as an example.

av_mat.c:

```
for (int j = 1; j <= gv->NY; j++) {
    for (int i = 1; i <= gv->NX; i++) {
        for (int k = 1; k <= gv->NZ; k++) {
        /* harmonic averaging of elasticity tensor */
            if (gv->WEQ >= 4) {
                mod_av->pc44i[j][i][k] = 4.0 / (1.0 / mod->pc44[j][i][k] + 1.0  mod->pc44[j+1][i][k] + 1.0 / mod-
                >pc44[j][i][k+1] + 1.0 / mod->pc44[j+1][i][k+1]);
                mod_av->pc55i[j][i][k] = 4.0 / (1.0 / mod->pc55[j][i][k] + 1.0  mod->pc55[j+1][i][k] + 1.0 / mod-
                >pc55[j][i+1][k] + 1.0 / mod->pc55[j+1][i+1][k]);
                mod_av->pc66i[j][i][k] = 4.0 / (1.0 / mod->pc66[j][i][k] + 1.0 / mod->pc66[j][i+1][k] + 1.0 / mod-
                >pc66[j][i][k+1] + 1.0 / mod->pc66[j][i+1][k+1]);
                }
            if (gv->WEQ == 6) {
                mod_av->pc41i[j][i][k] = 4.0 / (1.0 / mod->pc14[j][i][k] + 1.0 / mod->pc14[j+1][i][k] + 1.0 / mod-
                >pc14[j][i][k+1] + 1.0 / mod->pc14[j+1][i][k+1]);
                mod_av->pc42i[j][i][k] = 4.0 / (1.0 / mod->pc24[j][i][k] + 1.0 / mod->pc24[j+1][i][k] + 1.0 / mod-
                >pc24[j][i][k+1] + 1.0 / mod->pc24[j+1][i][k+1]);
                mod_av->pc43i[j][i][k] = 4.0 / (1.0 / mod->pc34[j][i][k] + 1.0 / mod->pc34[j+1][i][k] + 1.0 / mod-
                >pc34[j][i][k+1] + 1.0 / mod->pc34[j+1][i][k+1]);
                mod_av->pc45i[j][i][k] = 4.0 / (1.0 / mod->pc45[j][i][k] + 1.0 / mod->pc45[j+1][i][k] + 1.0 / mod-
                >pc45[j][i][k+1] + 1.0 / mod->pc45[j+1][i][k+1]);
                mod_av->pc46i[j][i][k] = 4.0 / (1.0 / mod->pc46[j][i][k] + 1.0 / mod->pc46[j+1][i][k] + 1.0 / mod-
                >pc46[j][i][k+1] + 1.0 / mod->pc46[j+1][i][k+1]);
                mod_av->pc51i[j][i][k] = 4.0 / (1.0 / mod->pc15[j][i][k] + 1.0 / mod->pc15[j+1][i][k] + 1.0 / mod-
                >pc15[j][i+1][k] + 1.0 / mod->pc15[j+1][i+1][k]);
                mod_av->pc52i[j][i][k] = 4.0 / (1.0 / mod->pc25[j][i][k] + 1.0 / mod->pc25[j+1][i][k] + 1.0 / mod-
                >pc25[j][i+1][k] + 1.0 / mod->pc25[j+1][i+1][k]);
                mod_av->pc53i[j][i][k] = 4.0 / (1.0 / mod->pc35[j][i][k] + 1.0 / mod->pc35[j+1][i][k] + 1.0 / mod-
                >pc35[j][i+1][k] + 1.0 / mod->pc35[j+1][i+1][k]);
                mod_av->pc54i[j][i][k] = 4.0 / (1.0 / mod->pc45[j][i][k] + 1.0 / mod->pc45[j+1][i][k] + 1.0 / mod-
                >pc45[j][i+1][k] + 1.0 / mod->pc45[j+1][i+1][k]);
                mod_av->pc56i[j][i][k] = 4.0 / (1.0 / mod->pc56[j][i][k] + 1.0 / mod->pc56[j+1][i][k] + 1.0 / mod-
                >pc56[j][i+1][k] + 1.0 / mod->pc56[j+1][i+1][k]);
                mod_av->pc61i[j][i][k] = 4.0 / (1.0 / mod->pc16[j][i][k] + 1.0 / mod->pc16[j][i+1][k] + 1.0 / mod-
                >pc16[j][i][k+1] + 1.0 / mod->pc16[j][i+1][k+1]);
                mod_av->pc62i[j][i][k] = 4.0 / (1.0 / mod->pc26[j][i][k] + 1.0 / mod->pc26[j][i+1][k] + 1.0 / mod-
                >pc26[j][i][k+1] + 1.0 / mod->pc26[j][i+1][k+1]);
                mod_av->pc63i[j][i][k] = 4.0 / (1.0 / mod->pc36[j][i][k] + 1.0 / mod->pc36[j][i+1][k] + 1.0 / mod-
                >pc36[j][i][k+1] + 1.0 / mod->pc36[j][i+1][k+1]);
                mod_av->pc64i[j][i][k] = 4.0 / (1.0 / mod->pc46[j][i][k] + 1.0 / mod->pc46[j][i+1][k] + 1.0 / mod-
                >pc46[j][i][k+1] + 1.0 / mod->pc46[j][i+1][k+1]);
                mod_av->pc65i[j][i][k] = 4.0 / (1.0 / mod->pc56[j][i][k] + 1.0 / mod->pc56[j][i+1][k] + 1.0 / mod-
                >pc56[j][i][k+1] + 1.0 / mod->pc56[j][i+1][k+1]);
            }
        }
```

```
    }
}
```

update_s_ssg_elastic_aniso_TTI.c:

```
for (int j = nb->ny1; j <= nb->ny2; j++) {
    for (int i = nb->nx1; i <= nb->nx2; i++) {
        for (int k = nb->nz1; k <= nb->nz2; k++) {
            /* spatial derivatives of the components of the velocities * are computed */
            vel->vxx[j][i][k] =
                (b1 * (vel->vx[j][i][k] - vel->vx[j][i - 1][k]) + b2 * (vel->vx[j][i + 1][k] - vel->vx[j][i - 2][k])) / gv->DX;
            vxy =
                (b1 * (vel->vx[j + 1][i][k] - vel->vx[j][i][k]) + b2 * (vel->vx[j + 2][i][k] - vel->vx[j - 1][i][k])) / gv->DY;
            vxz =
                (b1 * (vel->vx[j][i][k + 1] - vel->vx[j][i][k]) + b2 * (vel->vx[j][i][k + 2] - vel->vx[j][i][k - 1])) / gv->DZ;
            vyx =
                (b1 * (vel->vy[j][i + 1][k] - vel->vy[j][i][k]) + b2 * (vel->vy[j][i + 2][k] - vel->vy[j][i - 1][k])) / gv->DX;
            vel->vyy[j][i][k] =
                (b1 * (vel->vy[j][i][k] - vel->vy[j - 1][i][k]) + b2 * (vel->vy[j + 1][i][k] - vel->vy[j - 2][i][k])) / gv->DY;
            vyz =
                (b1 * (vel->vy[j][i][k + 1] - vel->vy[j][i][k]) + b2 * (vel->vy[j][i][k + 2] - vel->vy[j][i][k - 1])) / gv->DZ;
            vzx =
                (b1 * (vel->vz[j][i + 1][k] - vel->vz[j][i][k]) + b2 * (vel->vz[j][i + 2][k] - vel->vz[j][i - 1][k])) / gv->DX;
            vzy =
                (b1 * (vel->vz[j + 1][i][k] - vel->vz[j][i][k]) + b2 * (vel->vz[j + 2][i][k] - vel->vz[j - 1][i][k])) / gv->DY;
            vel->vzz[j][i][k] =
                (b1 * (vel->vz[j][i][k] - vel->vz[j][i][k - 1]) + b2 * (vel->vz[j][i][k + 1] - vel->vz[j][i][k - 2])) / gv->DZ;
            vel->vxyyx[j][i][k] = vxy + vyx;
            vel->vyzzy[j][i][k] = vyz + vzy;
            vel->vxzzx[j][i][k] = vxz + vzx;
        }
    }
}
for (int j = nb->ny1; j <= nb->ny2; j++) {
    for (int i = nb->nx1; i <= nb->nx2; i++) {
        for (int k = nb->nz1; k <= nb->nz2; k++) {
            vel->vyzzy_yz[j][i][k] = interpolate_yz_4(vel->vyzzy, i, j, k, -1, mod_av);
            vel->vxzzx_xz[j][i][k] = interpolate_xz_4(vel->vxzzx, i, j, k, -1, mod_av);
            vel->vxyyx_xy[j][i][k] = interpolate_xy_4(vel->vxyyx, i, j, k, -1, mod_av);
        }
    }
}
for (int j = nb->ny1; j <= nb->ny2; j++) {
    for (int i = nb->nx1; i <= nb->nx2; i++) {
        for (int k = nb->nz1; k <= nb->nz2; k++) {
            vxx_xz = interpolate_xz_4(vel->vxx, i, j, k, 1, mod_av);
            vxx_xy = interpolate_xy_4(vel->vxx, i, j, k, 1, mod_av);
            vxx_yz = interpolate_yz_4(vel->vxx, i, j, k, 1, mod_av);
            vyy_xz = interpolate_xz_4(vel->vyy, i, j, k, 1, mod_av);
            vyy_xy = interpolate_xy_4(vel->vyy, i, j, k, 1, mod_av);
            vyy_yz = interpolate_yz_4(vel->vyy, i, j, k, 1, mod_av);
            vzz_xz = interpolate_xz_4(vel->vzz, i, j, k, 1, mod_av);
            vzz_xy = interpolate_xy_4(vel->vzz, i, j, k, 1, mod_av);
            vzz_yz = interpolate_yz_4(vel->vzz, i, j, k, 1, mod_av);
            vyzzy_xz = interpolate_xz_4(vel->vyzzy_yz, i, j, k, 1, mod_av);
            vyzzy_xy = interpolate_xy_4(vel->vyzzy_yz, i, j, k, 1, mod_av);
            vxzzx_xy = interpolate_xy_4(vel->vxzzx_xz, i, j, k, 1, mod_av);
            vxzzx_yz = interpolate_yz_4(vel->vxzzx_xz, i, j, k, 1, mod_av);
            vxyyx_xz = interpolate_xz_4(vel->vxyyx_xy, i, j, k, 1, mod_av);
            vxyyx_yz = interpolate_yz_4(vel->vxyyx_xy, i, j, k, 1, mod_av);
```

```
            stress->sxz[j][i][k] += (mod_av->pc61i[j][i][k] * vxx_xz + mod_av->pc62i[j][i][k] * vzz_xz + mod_av-
            >pc63i[j][i][k] * vyy_xz + mod_av->pc64i[j][i][k] * vyzzy_xz + mod_av->pc65i[j][i][k] * vxyyx_xz + mod_av-
            >pc66i[j][i][k] * vel->vxzzx[j][i][k]) * gv->DT;
            stress->syz[j][i][k] += (mod_av->pc41i[j][i][k] * vxx_yz + mod_av->pc42i[j][i][k] * vzz_yz + mod_av-
            >pc43i[j][i][k] * vyy_yz + mod_av->pc44i[j][i][k] * vel→vyzzy[j][i] [k] + mod_av->pc45i[j][i][k] * vxyyx_yz
            + mod_av->pc46i[j][i][k] * vxzzx_yz) * gv- >DT;
            stress->sxy[j][i][k] += (mod_av->pc51i[j][i][k] * vxx_xy + mod_av->pc52i[j][i][k] * vzz_xy + mod_av-
            >pc53i[j][i][k] * vyy_xy + mod_av->pc54i[j][i][k] * vyzzy_xy + mod_av->pc55i[j][i][k] * vel->vxyyx[j][i][k]
            + mod_av->pc56i[j][i][k] * vxzzx_xy) * gv->DT;
            stress->sxx[j][i][k] += (mod->pc11[j][i][k] * vel->vxx[j][i][k] + mod->pc12[j][i][k] * vel->vzz[j][i][k] + mod-
            >pc13[j][i][k] * vel->vyy[j][i][k] + mod->pc14[j][i][k] * vel- >vyzzy_yz[j][i][k] + mod->pc15[j][i][k] * vel-
            >vxyyx_xy[j][i][k] + mod->pc16[j][i][k] * vel->vxzzx_xz[j][i][k]) * gv->DT;
            stress->szz[j][i][k] += (mod->pc12[j][i][k] * vel->vxx[j][i][k] + mod->pc22[j][i][k] * vel->vzz[j][i][k] + mod-
            >pc23[j][i][k] * vel->vyy[j][i][k] + mod->pc24[j][i][k] * vel- >vyzzy_yz[j][i][k] + mod->pc25[j][i][k] * vel-
            >vxyyx_xy[j][i][k] + mod->pc26[j][i][k] * vel->vxzzx_xz[j][i][k]) * gv->DT;
            stress->syy[j][i][k] += (mod->pc13[j][i][k] * vel->vxx[j][i][k] + mod->pc23[j][i][k] * vel->vzz[j][i][k] + mod-
            >pc33[j][i][k] * vel->vyy[j][i][k] + mod->pc34[j][i][k] * vel- >vyzzy_yz[j][i][k] + mod->pc35[j][i][k] * vel-
            >vxyyx_xy[j][i][k] + mod->pc36[j][i][k] * vel->vxzzx_xz[j][i][k]) * gv->DT;
        }
    }
}
```

interpolate_4_FD.c:

```
float interpolate_xz_4(float ***a, int i, int j, int k, int m, st_model_av *mod_av) {
    float ai =
        ((a[j][i][k] + a[j][i+1*m][k] + a[j][i][k+1*m] + a[j][i+1*m][k+1*m]) * mod_av->w11 + ((a[j][i-1*m][k] + a[j][i+2*m][k]
        + a[j][i-1*m][k+1*m] + a[j][i+2*m][k+1*m]) * mod_av- >DXZ12 + (a[j][i][k-1*m] + a[j][i+1*m][k-1*m] +
        a[j][i][k+2*m] + a[j][i+1*m][k+2*m]) * mod_av- >DXZ21) * mod_av->w12 + (a[j][i-1*m][k-1*m] + a[j][i+2*m][k-
        1*m] + a[j][i-1*m][k+2*m] + a[j][i+2*m][k+2*m]) * mod_av->w22) * mod_av->sinc_norm;
    return ai;
}
float interpolate_xy_4(float ***a, int i, int j, int k, int m, st_model_av *mod_av) {
    float ai =
        ((a[j][i][k] + a[j][i+1*m][k] + a[j+1*m][i][k] + a[j+1*m][i+1*m][k]) * mod_av->w11 + ((a[j][i-1*m][k] + a[j][i+2*m][k]
        + a[j+1*m][i-1*m][k] + a[j+1*m][i+2*m][k]) * mod_av- >DXY12 + (a[j-1*m][i][k] + a[j-1*m][i+1*m][k] +
        a[j+2*m][i][k] + a[j+2*m][i+1*m][k]) * mod_av- >DXY21) * mod_av->w12 + (a[j-1*m][i-1*m][k] + a[j-
        1*m][i+2*m][k] + a[j+2*m][i-1*m][k] + a[j+2*m][i+2*m][k]) * mod_av->w22) * mod_av->sinc_norm;
    return ai;
}
float interpolate_yz_4(float ***a, int i, int j, int k, int m, st_model_av *mod_av) {
    float ai =
        ((a[j][i][k] + a[j+1*m][i][k] + a[j][i][k+1*m] + a[j+1*m][i][k+1*m]) * mod_av->w11 + ((a[j-1*m][i][k] + a[j+2*m][i][k]
        + a[j-1*m][i][k+1*m] + a[j+2*m][i][k+1*m]) * mod_av- >DYZ12 + (a[j][i][k-1*m] + a[j+1*m][i][k-1*m] +
        a[j][i][k+2*m] + a[j+1*m][i][k+2*m]) * mod_av- >DYZ21) * mod_av->w12 + (a[j-1*m][i][k-1*m] + a[j+2*m][i][k-
        1*m] + a[j-1*m][i][k+2*m] + a[j+2*m][i][k+2*m]) * mod_av->w22) * mod_av->sinc_norm;
    return ai;
}
```

interpolate_4_FD_weights.c:

```
if (gv->FDORDER == 4) {

    mod_av->DXY12 = gv->DY * 2 / (gv->DX + gv->DY);

    mod_av->DXY21 = gv->DX * 2 / (gv->DX + gv->DY);

    mod_av->DXZ12 = gv->DZ * 2 / (gv->DX + gv->DZ);

    mod_av->DXZ21 = gv->DX * 2 / (gv->DX + gv->DZ);

    mod_av->DYZ12 = gv->DZ * 2 / (gv->DY + gv->DZ);

    mod_av->DYZ21 = gv->DY * 2 / (gv->DY + gv->DZ);
```

```
    if (gv->INTP_TYPE <= 2) {
        b1 = 9.0 / 8.0;
        b2 = -1.0 / 24.0;
        mod_av->w11 = b1 * 0.5 * b1 * 0.5;
        mod_av->w12 = b1 * 0.5 * b2 * 1.5;
        mod_av->w22 = b2 * 1.5 * b2 * 1.5;
        mod_av->sinc_norm = 1.0;
    } else {
        mod_av->w11 = 4.0 / PI / PI;
        mod_av->w12 = -4.0 / 3.0 / PI / PI;
        mod_av->w22 = 4.0 / 9.0 / PI / PI;
        mod_av->sinc_norm = 9.0 * PI * PI / 64.0;
    }
}
```

6.3 Appendix C

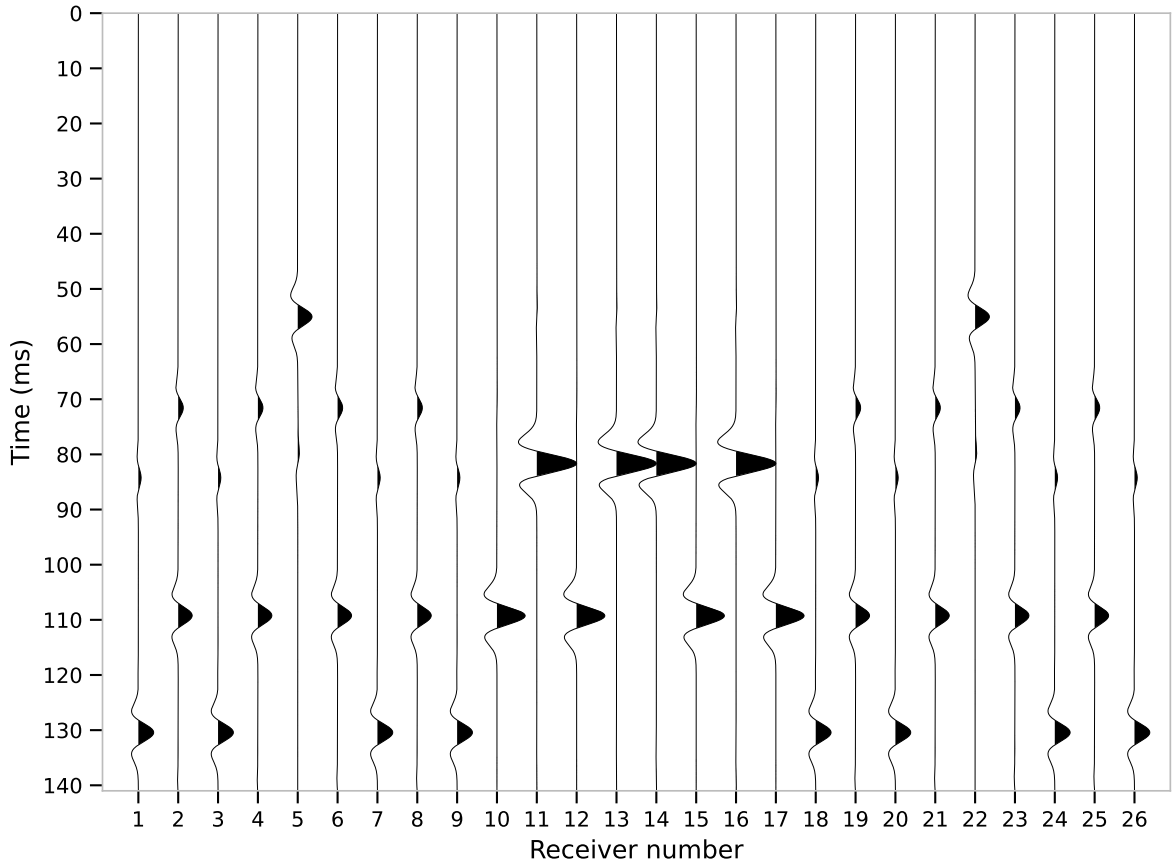


Figure 19: **Traces isotropic:** Seismic traces for all 26 receivers for the homogeneous model, isotropic sixth spatial FD order.

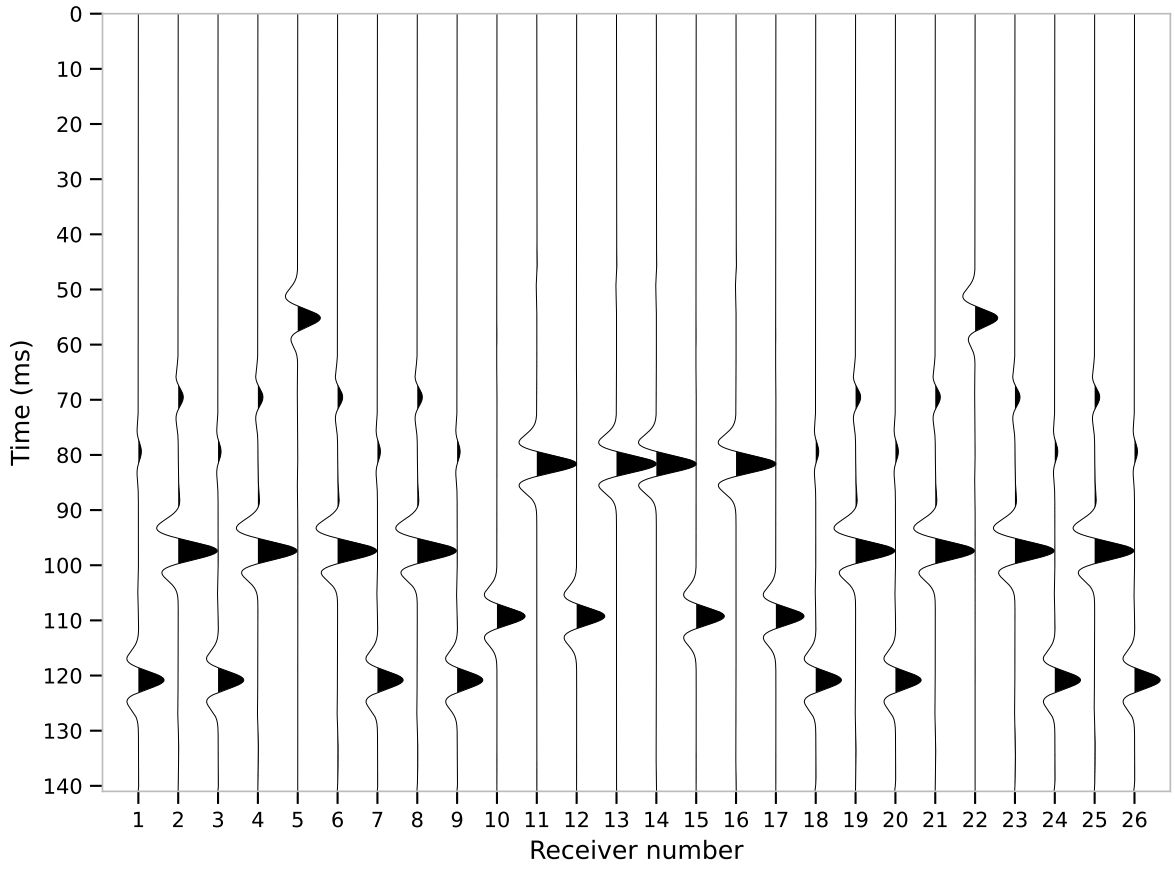


Figure 20: **Traces VTI:** Seismic traces for all 26 receivers for the homogeneous model, VTI sixth spatial FD order.

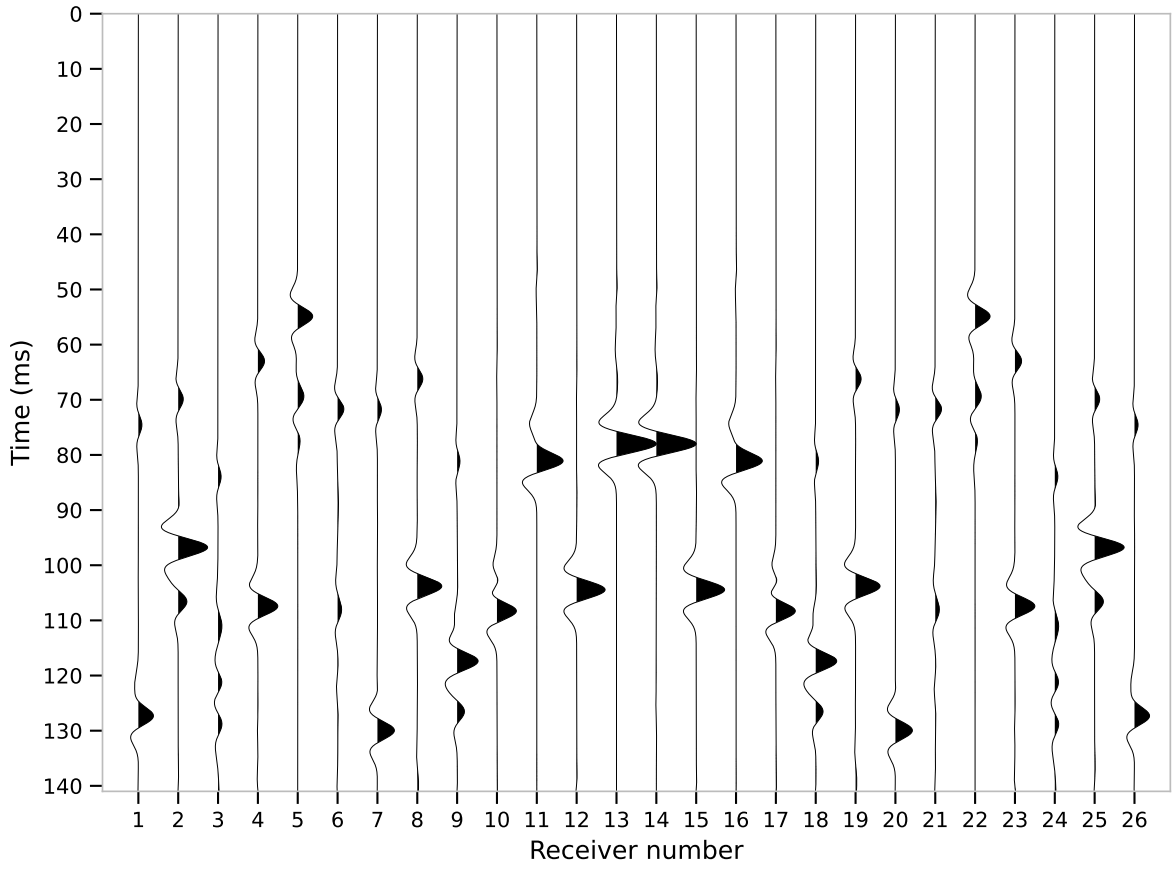
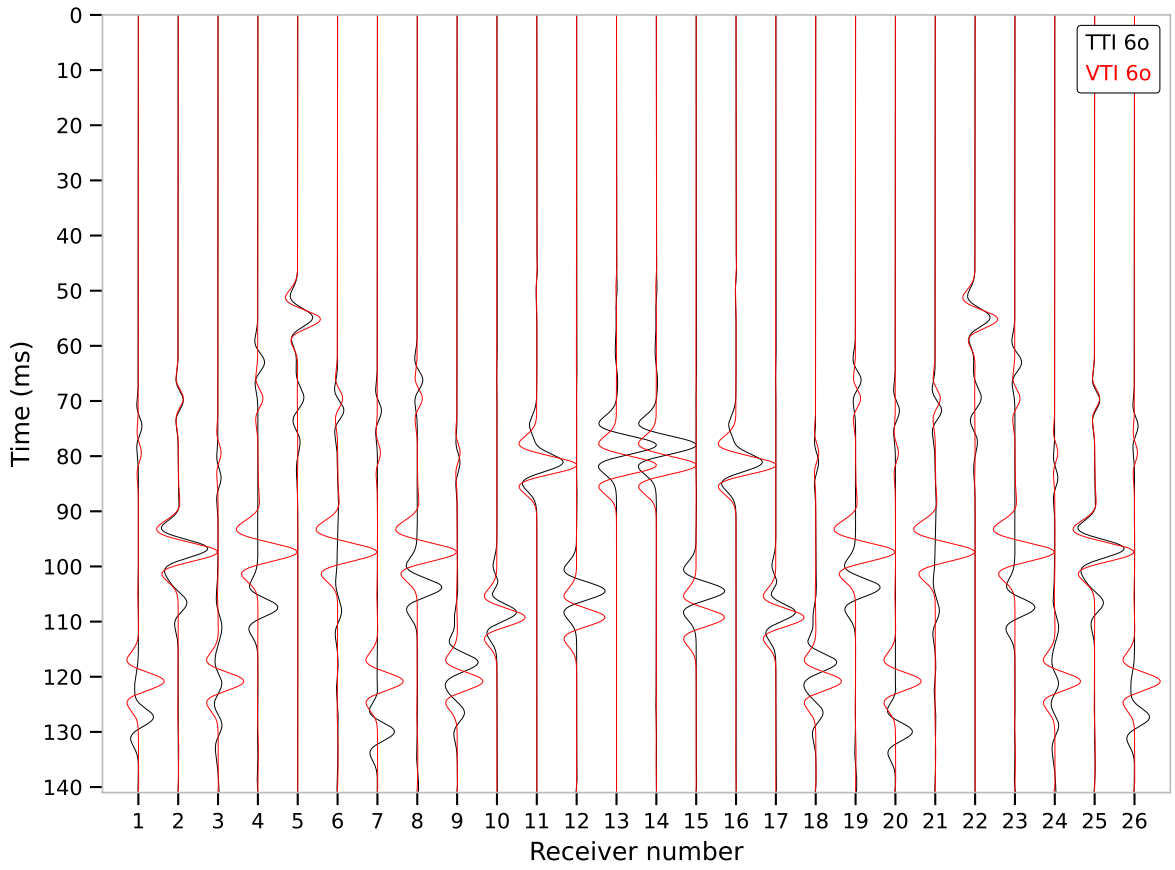
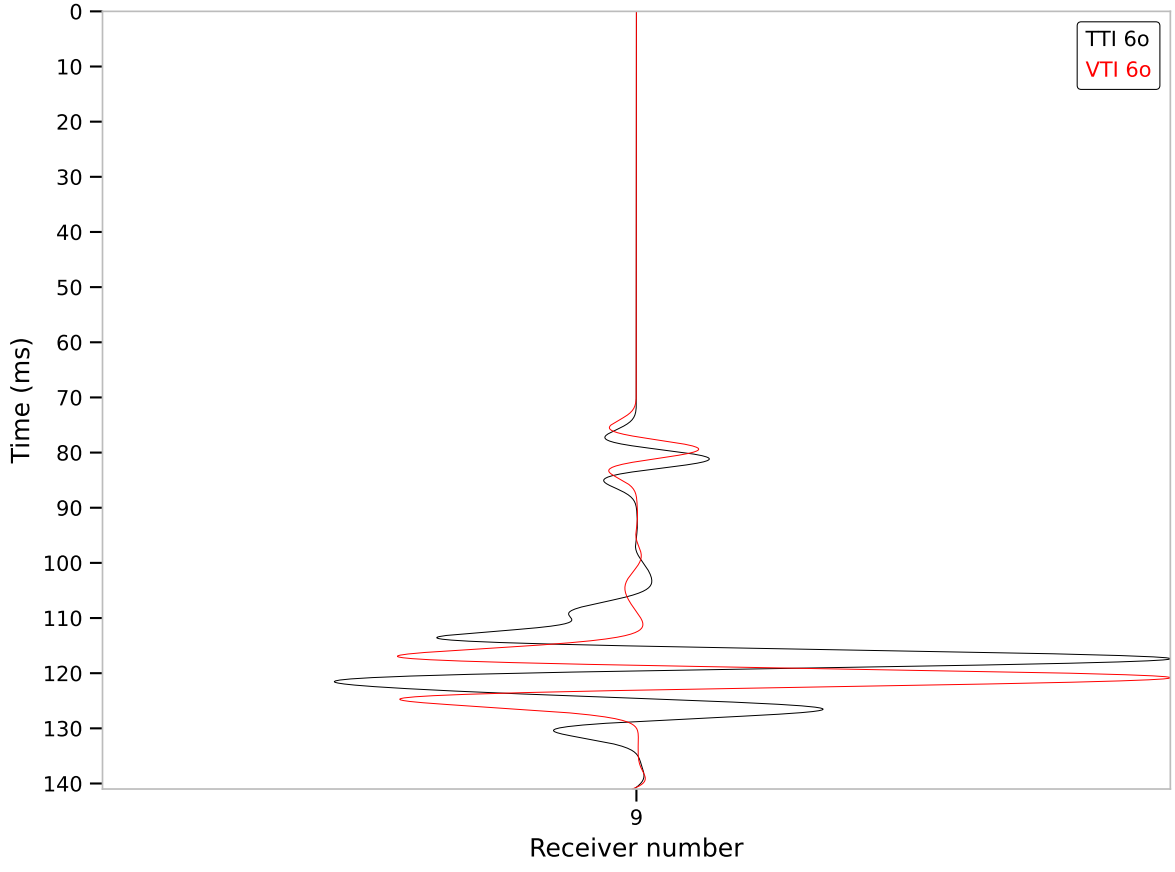


Figure 21: **Traces TTI:** Seismic traces for all 26 receivers for the homogeneous model, TTI sixth spatial FD order with FD-consistent interpolation.

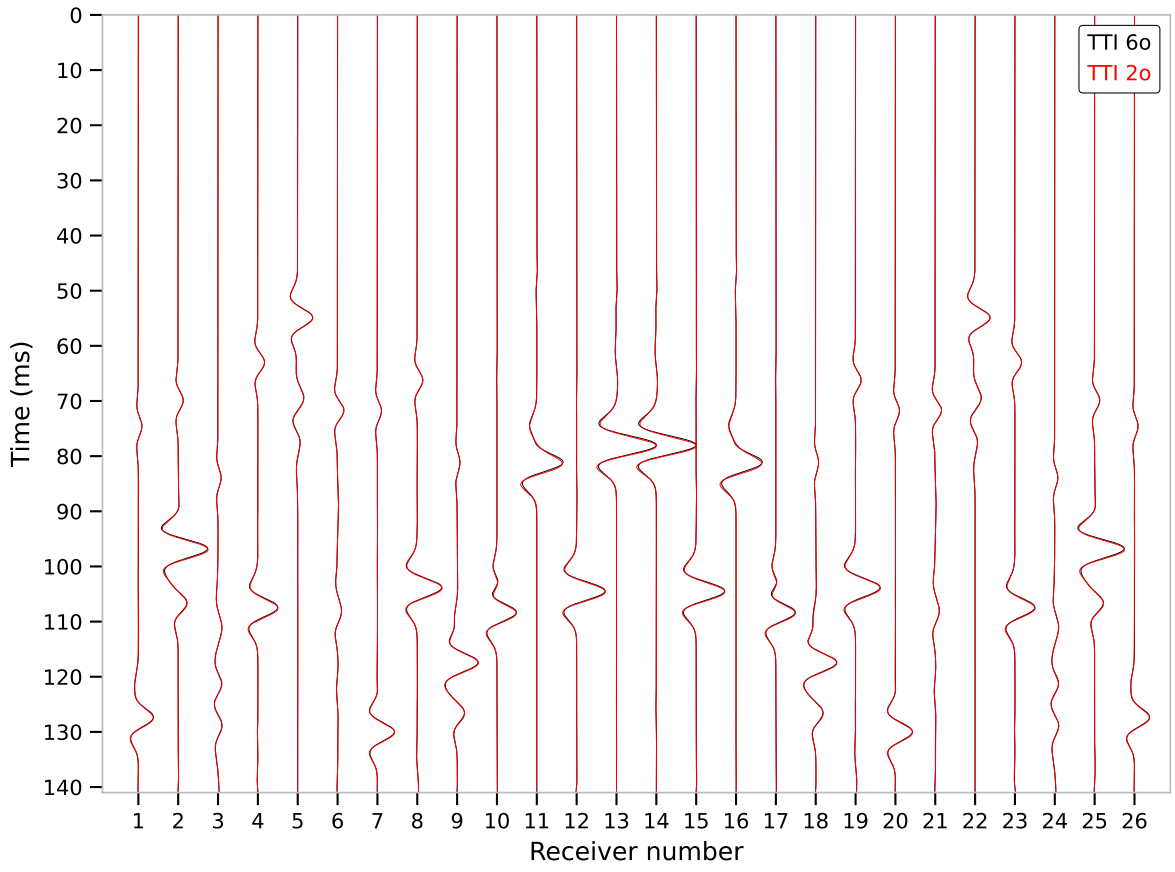


(a)

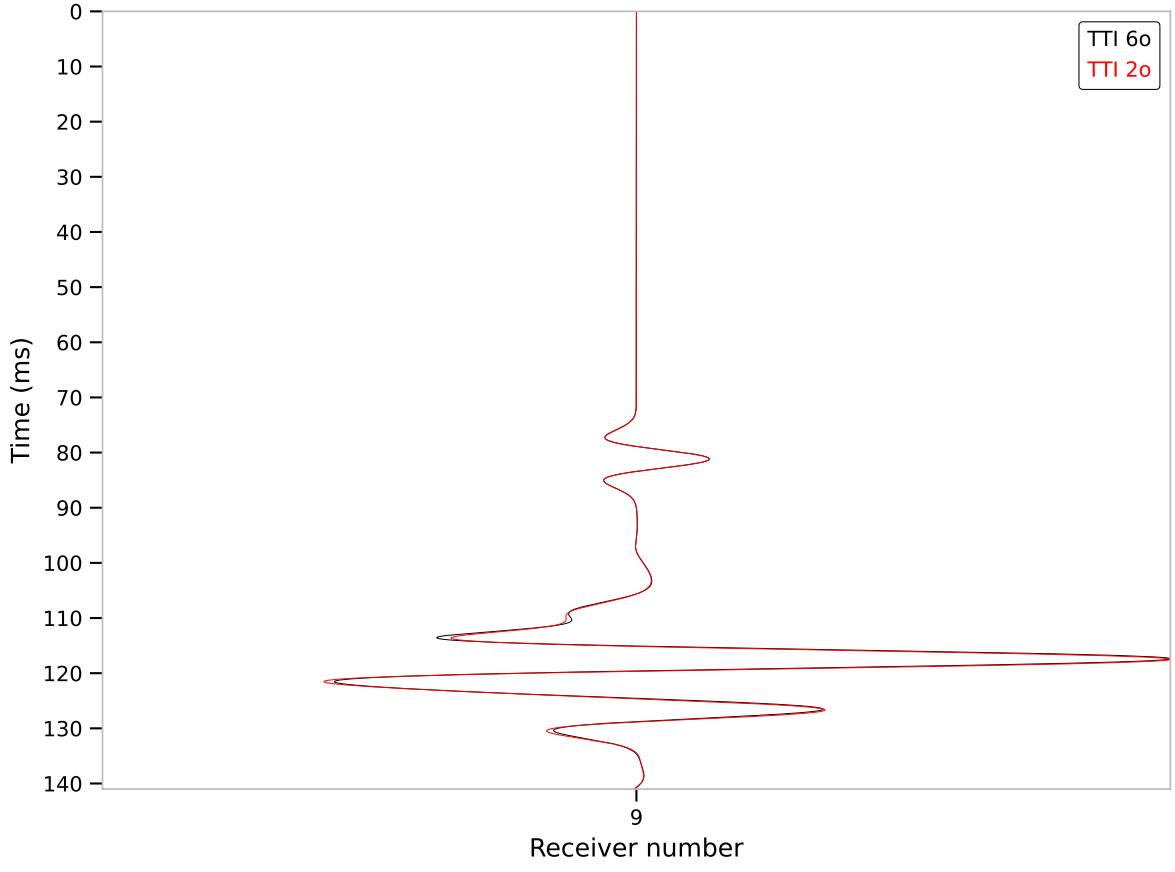


(b)

Figure 22: **Traces TTI vs. VTI:** TTI sixth order with FD-consistent interpolation in black and VTI sixth order in red. a) Seismic traces for all 26 receivers. b) Trace for receiver number 9. 51

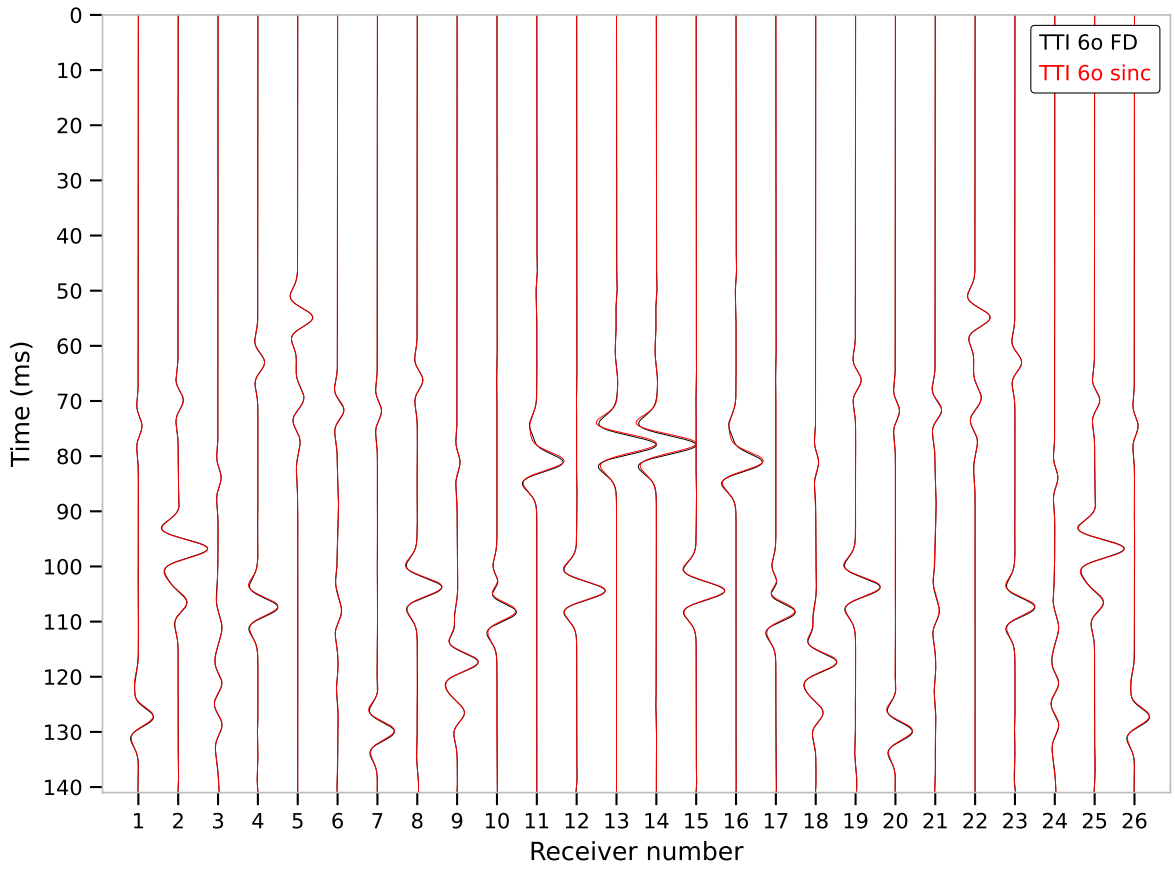


(a)

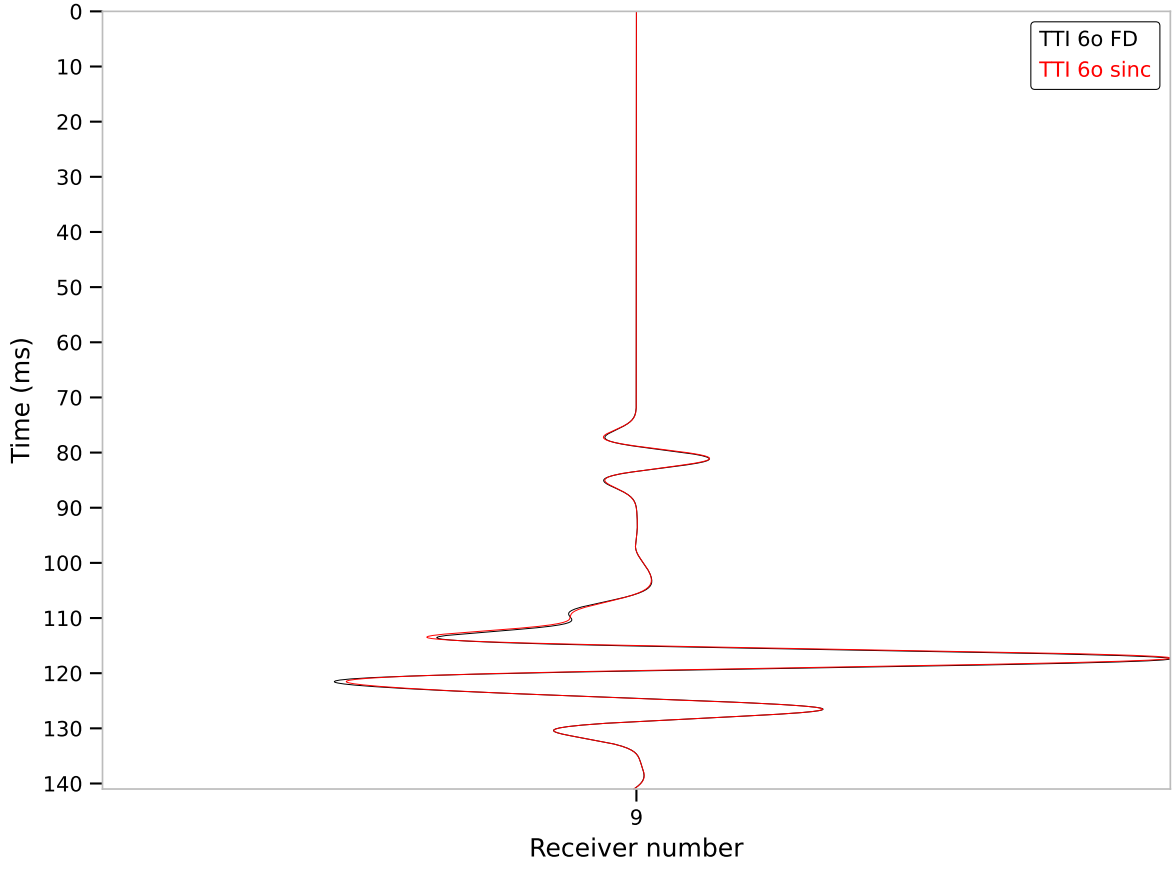


(b)

Figure 23: **Traces TTI FD order:** TTI sixth order with FD-consistent interpolation in black and TTI second order in red. a) Seismic traces for all 26 receivers. b) Trace for receiver number 9.

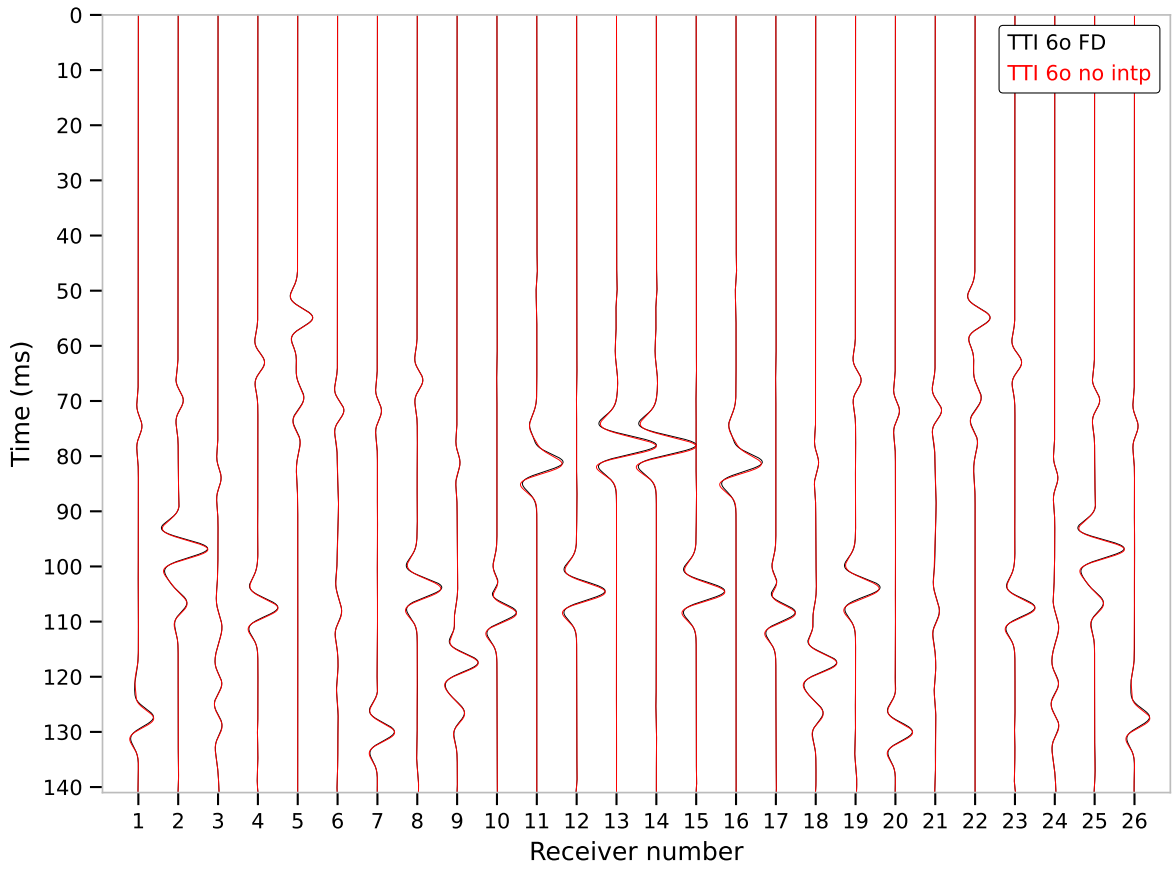


(a)

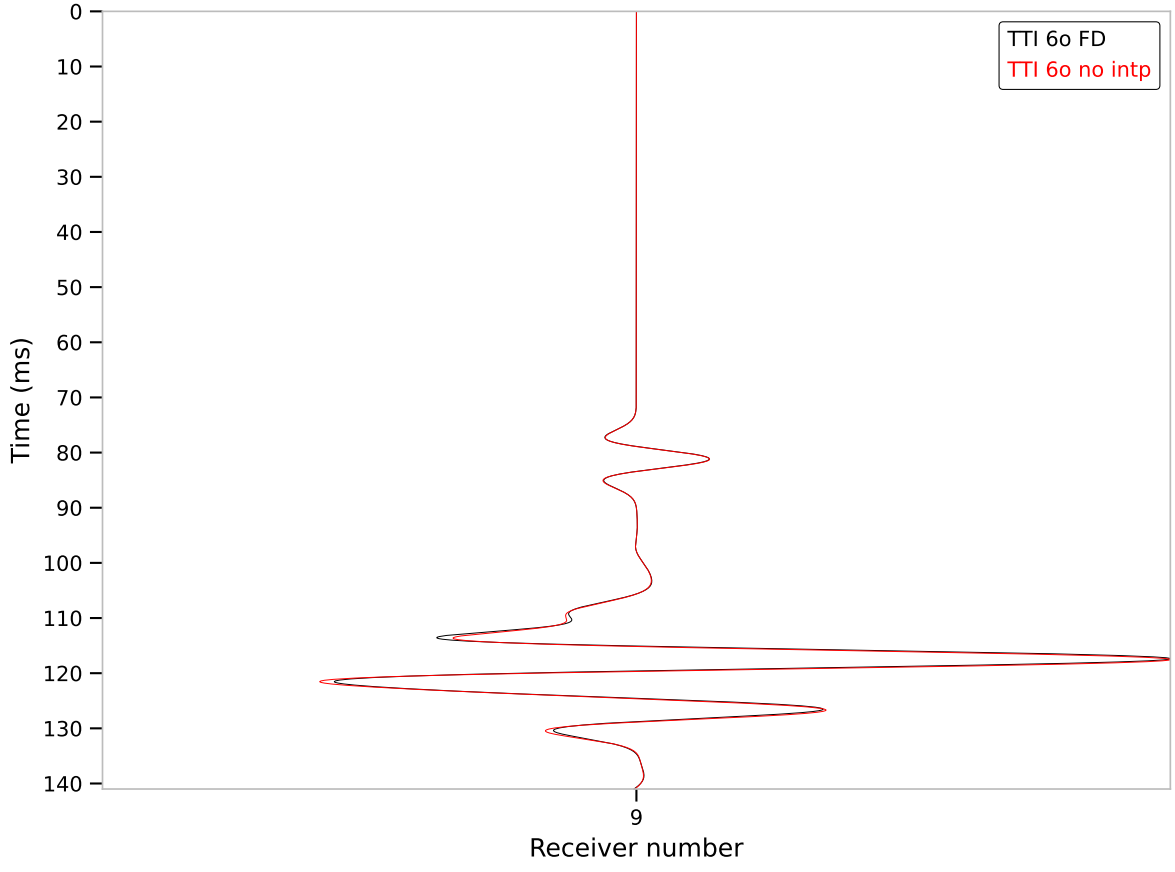


(b)

Figure 24: **Traces TTI FD vs. sinc:** TTI sixth order with FD-consistent interpolation in black and with sinc-function interpolation in red. a) Seismic traces for all 26 receivers. b) Trace for receiver number 9.



(a)



(b)

Figure 25: **Traces TTI FD vs. no interpolation:** TTI sixth order with FD-consistent interpolation in black and TTI with no interpolation in red. a) Seismic traces for all 26 receivers. b) Trace for receiver number 9.

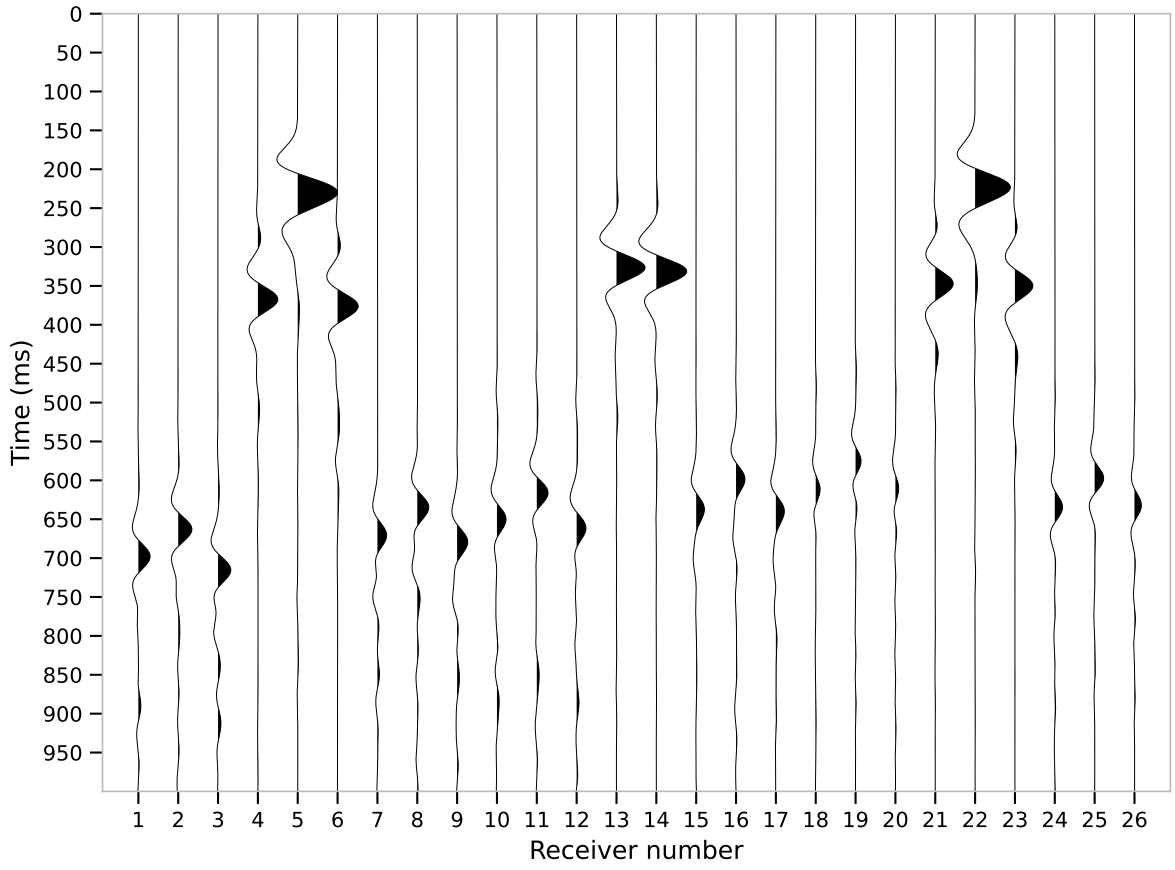
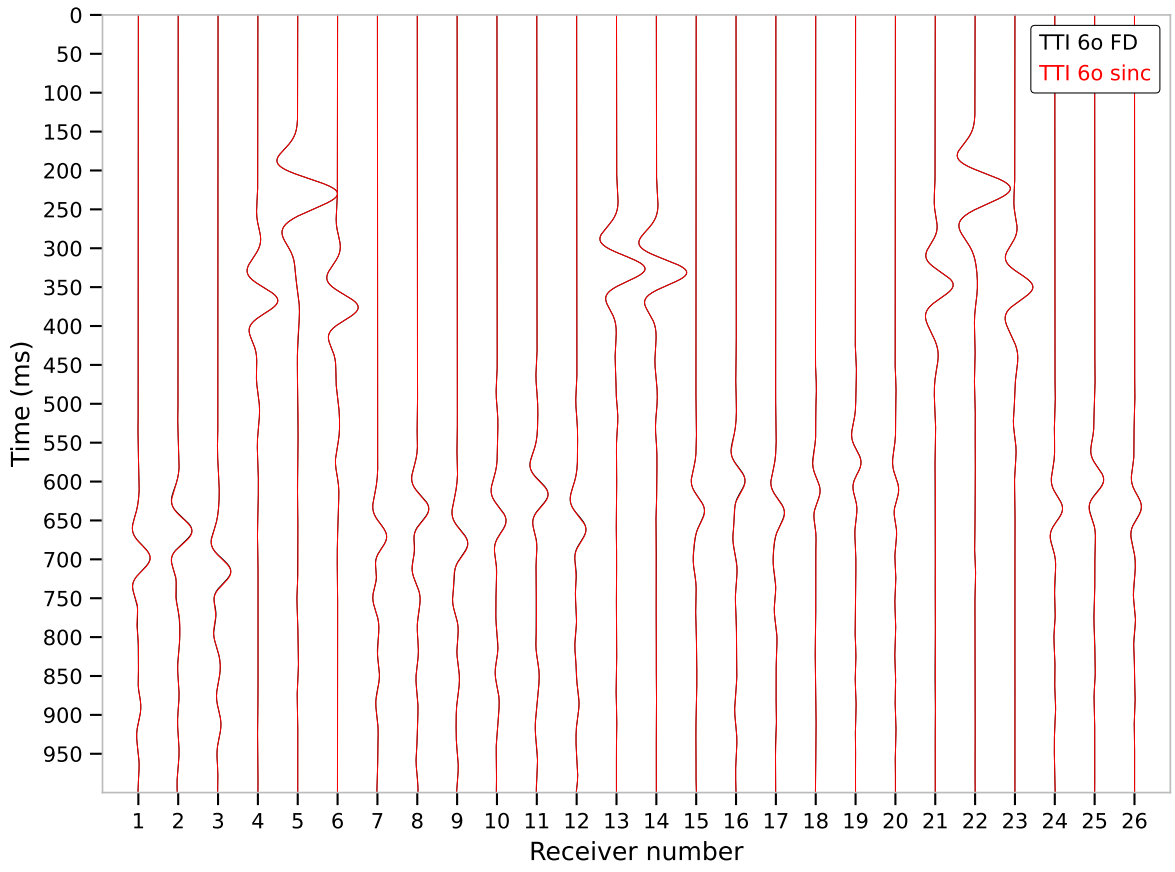
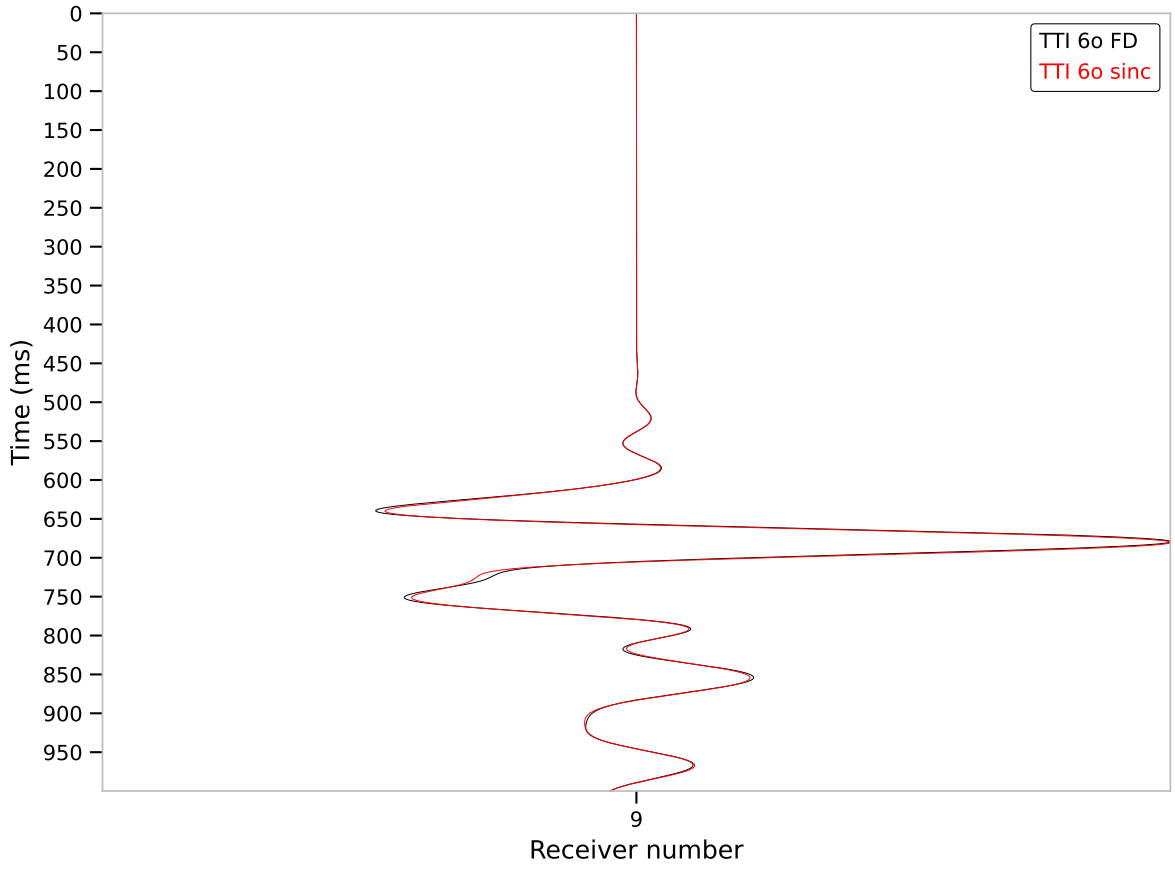


Figure 26: **Traces isotropic heterogenious:** Seismic traces for all 26 receivers for the heterogeneous model, TTI sixth spatial FD order with FD-consistent interpolation.



(a)



(b)

Figure 27: **Traces TTI FD vs. sinc heterogeneous:** TTI sixth order with FD-consistent interpolation in black and with sinc-function interpolation in red for the heterogeneous model. a) Seismic traces for all 26 receivers. b) Trace for receiver number 9.

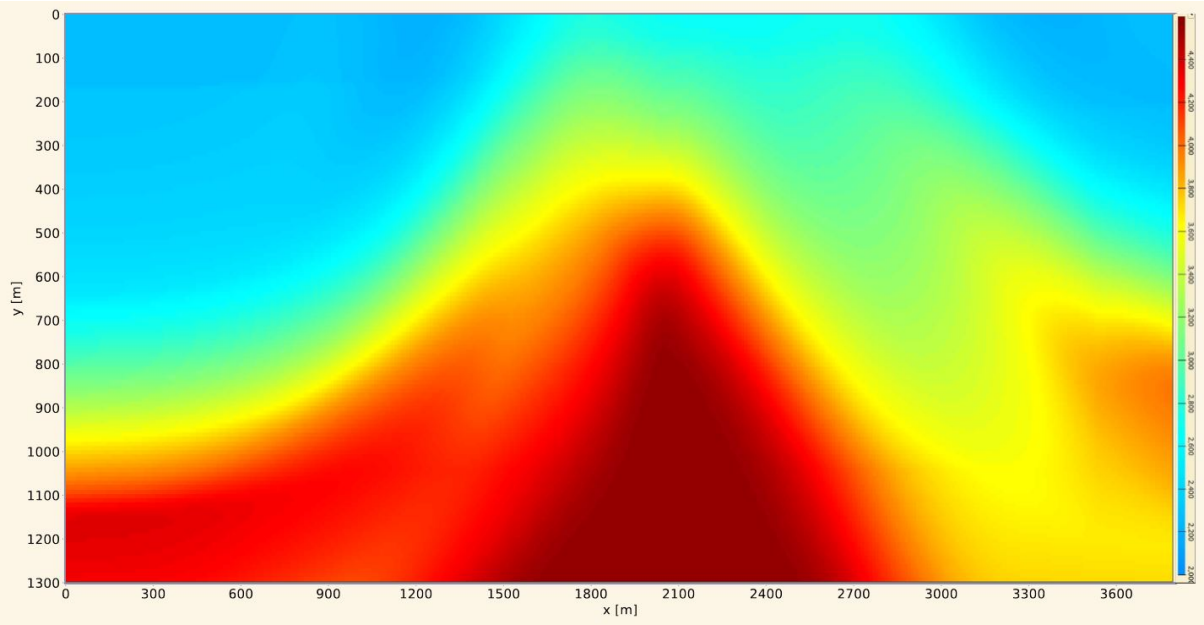


Figure 28: V_P

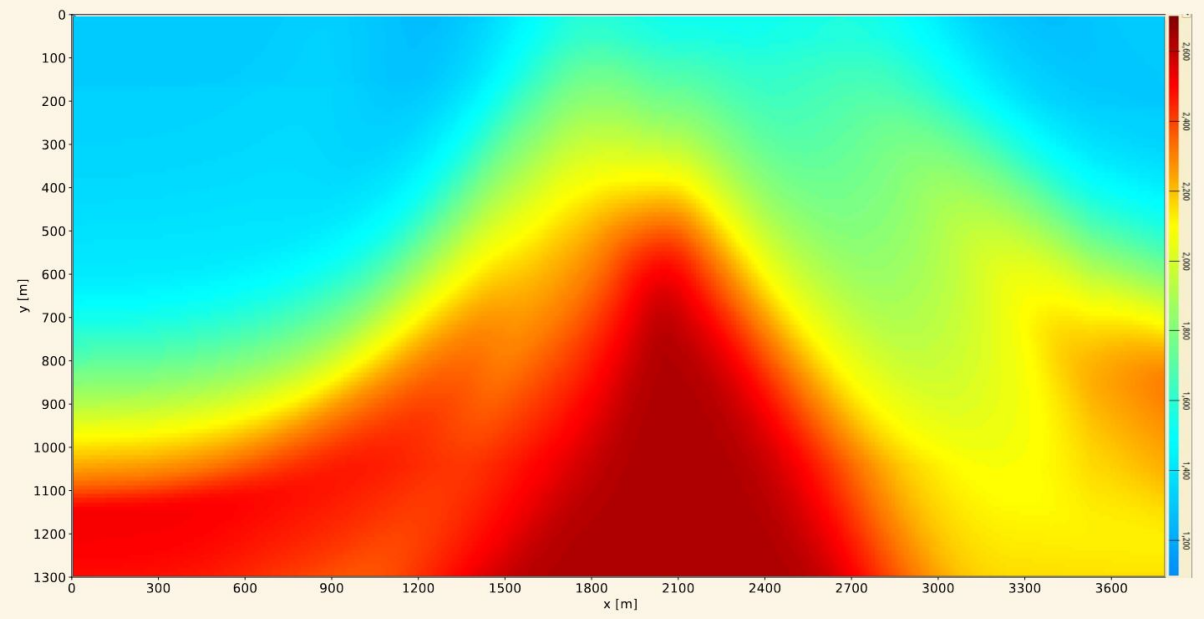


Figure 29: V_S

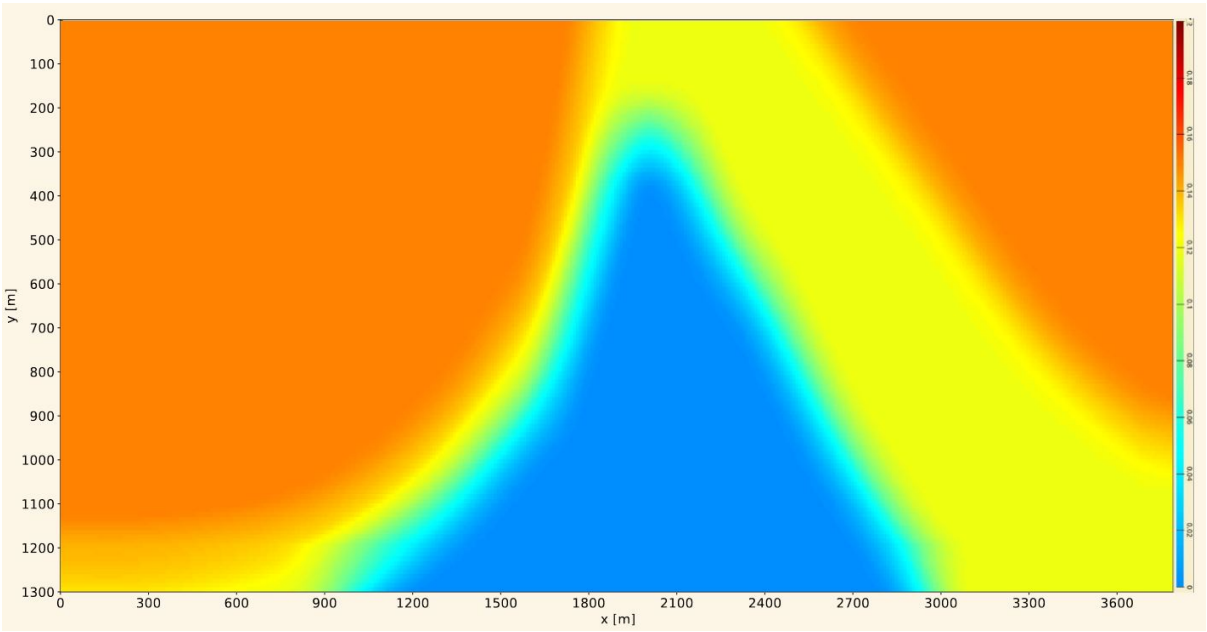


Figure 30: ϵ

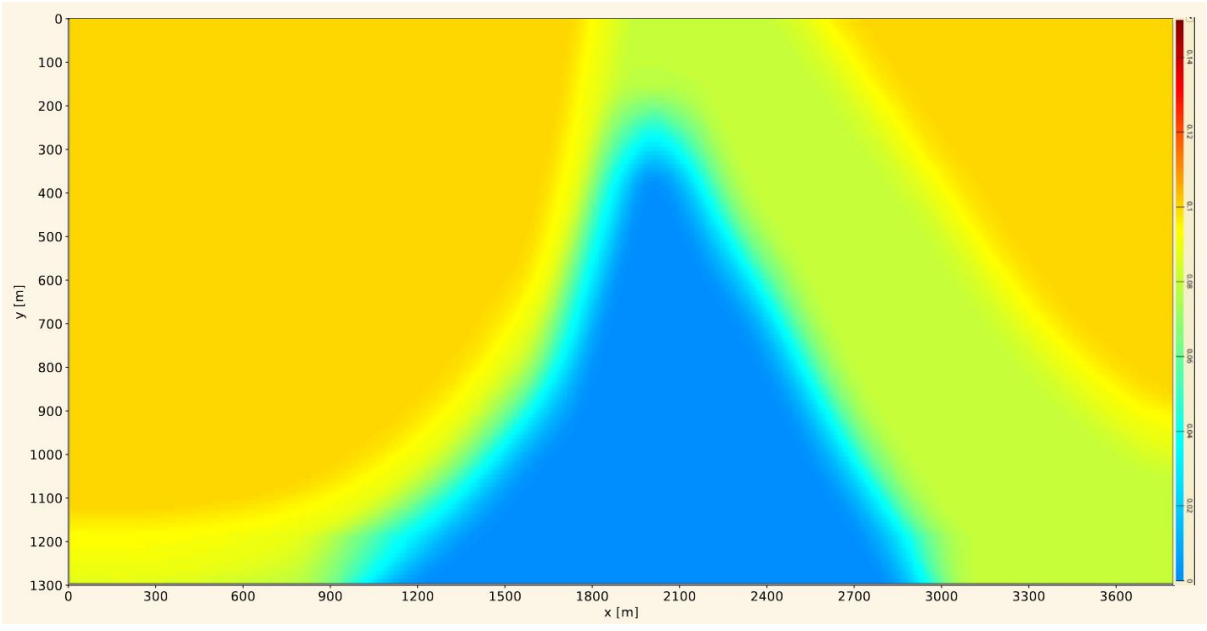


Figure 31: δ

List of Figures

List of Tables

Bibliography

Alaei, N., Soleimani Monfared, M., Roshandel Kahoo, A., and Bohlen, T. (2022). Seismic imaging of complex velocity structures by 2d pseudo-viscoelastic time-domain full-waveform inversion. *Applied Sciences*, 12(15):7741.

Alterman, Z. and Karal Jr, F. (1968). Propagation of elastic waves in layered media by finite difference methods. *Bulletin of the Seismological Society of America*, 58(1):367–398.

Auld, B. A. (1973). *Acoustic fields and waves in solids*.

Bond, W. L. (1943). The mathematics of the physical properties of crystals. *The Bell System Technical Journal*, 22(1):1–72.

Carcione, J. M. (2007). *Wave fields in real media: Wave propagation in anisotropic, anelastic, porous and electromagnetic media*. Elsevier.

Cerjan, C., Kosloff, D., Kosloff, R., and Reshef, M. (1985). A nonreflecting boundary condition for discrete acoustic and elastic wave equations. *Geophysics*, 50(4):705–708.

Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische annalen*, 100(1):32–74.

Ferrer, M., de la Puente, J., Farrés, A., and Castillo, J. E. (2015). 3d viscoelastic anisotropic seismic modeling with high-order mimetic finite differences. In *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2014: Selected papers from the ICOSAHOM conference, June 23-27, 2014, Salt Lake City, Utah, USA*, pages 217–225. Springer.

Graves, R. W. (1996). Simulating seismic wave propagation in 3d elastic media using staggered-grid finite differences. *Bulletin of the seismological society of America*, 86(4):1091–1106.

Igel, H., Mora, P., and Riollet, B. (1995). Anisotropic wave propagation through finite-difference grids. *Geophysics*, 60(4):1203–1216.

Jones, L. E. and Wang, H. F. (1981). Ultrasonic velocities in cretaceous shales from the williston basin. *Geophysics*, 46(3):288–297.

Koene, E. F., Robertsson, J. O., and Andersson, F. (2020). A consistent implementation of point sources on finite-difference grids. *Geophysical Journal International*, 223(2):1144–1161.

Li, L., Tan, J., Zhang, D., Malkoti, A., Abakumov, I., and Xie, Y. (2021). Fdwave3d: a matlab solver for the 3d anisotropic wave equation using the fi *Computational Geosciences*, 25:1565–1578.

Thomsen, L. (1986). Weak elastic anisotropy. *Geophysics*, 51(10):1954–1966.

Virieux, J. (1986). P-sv wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 51(4):889–901.

Voigt, W. (1910). *Lehrbuch der kristallphysik:(mit ausschluss der kristalloptik)*, volume 34. BG Teubner.

Yee, K. (1966). Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE Transactions on antennas and propagation*, 14(3):302–307.

Zhu, J. and Dorman, J. (2000). Two-dimensional, three-component wave propagation in a transversely isotropic medium with arbitrary-orientation—finite-element modeling. *Geophysics*, 65(3):934–942.

Acknowledments

I want to thank the whole Applied Geophysics work group of Prof. Dr. Thomas Bohlen and especially Prof. Dr. Thomas Bohlen himself for helping me with the implementation of this work and for answering my questions. I want to thank Lars Haupt, Thomas Hertweck and Laura Gassner for always being there to help me with solving problems that occurred.

A big thank you goes to my family for always supporting me no matter what and thank you to my sister, Michelle for revising this thesis.

Eidesstattliche Erklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig verfasst, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde sowie die Satzung des KIT zur Sicherung guter wissenschaftlicher Praxis in der momentan gültigen Fassung beachtet zu haben.

I declare truthfully that I have written this thesis all by myself, that I have fully and accurately specified all auxiliary means used, that I have correctly cited (marked) everything that was taken, either unchanged or with modification, from the work of others, and that I have complied with the current version of the KIT statutes for safeguarding good scientific practice.

Karlsruhe, 06.09.2024

Daniel Mai