Technische Universität München
TUM School of Computation, Information and Technology

A Fully Coupled Model for Petascale Earthquake-Tsunami and Earthquake-Sound Simulations

Lukas Daniel Sidney Krenz

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 20.09.2023 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 29.01.2024 angenommen.

Acknowledgments

First, I want to thank my advisor, Michael, for allowing me to work on this topic, for his support, and for providing an environment full of opportunities.

The “old” generation of our group taught me the basics. Thanks to Leo for introducing me to numerical methods and HPC during my Master’s thesis and for – somehow – convincing me to pursue a PhD. Thanks to Carsten for showing me most of what I know about SeisSol: the good and especially the bad parts. Finally, I want to thank Anne, with whom I had the pleasure of designing and teaching a new lab course, and for showing me how tactical stubbornness can overcome nearly any obstacle.

I am grateful to have shared my office with Sebastian, who had to endure a fair bit of complaining. Thanks to Ravil for hours of technical discussions, from which I’ve learned a lot.

I particularly enjoyed collaborating with other people. For the earthquake-tsunami coupling project, I want to thank Alice, Aniko, and Thomas of LMU and Lauren and Eric from Stanford University. Both groups tremendously impacted this thesis. The LMU group contributed interesting application examples, and the Stanford group introduced me to the fully coupled model. I want to thank Gregor, who taught me a lot about seismology during our earthquake-sound coupling project and invited me to visit Helsinki for multiple weeks.

Thanks to Marc, David, and Vanessa for their feedback on the drafts of this thesis. All remaining errors are solely my fault.

Thanks to my friends and family for their continuous support and for distracting me from work whenever necessary. Finally, I want to thank my wife, Vanessa, for tolerating my PhD-induced bad moods and for her relentless support, without which this thesis may not exist.

Abstract

We introduce an ADER Discontinuous Galerkin discretization for fully coupled elastic-acoustic earthquake simulations. The multiphysics model combines dynamic earthquake rupture, wave propagation in elastic and acoustic media, and tsunami propagation modeled by a linearized gravitational free surface condition. We show an ADER-DG discretization of this model that uses an exact Riemann solver. The gravitational boundary condition results in an element-local ODE, which we integrate with an ADER-based Taylor series approach. We prove that the resulting scheme is stable.

We introduce an actor model for clustered local time-stepping (LTS), which groups elements with similar time step sizes in clusters and updates them together. Our scheduling algorithm combines state machines, which track the local status, with explicit message passing, which informs other clusters of updates. This model includes shared and distributed memory parallelization, resulting in an elegant abstraction that manages computations and communication. Furthermore, we introduce two features that can automatically fine-tune the clustering: a wiggle factor, which moves the cluster boundaries, and the capability to automatically merge clusters with large time step sizes.

We discuss earthquake-tsunami coupling strategies and explain where the fully coupled model fits in. We demonstrate with carefully selected scenarios that our discretization reaches a high-order convergence rate. We introduce three applications. The first shows that the fully coupled model gives reasonable results for an earthquake-tsunami benchmark scenario. The second applies the fully coupled model to the Palu, Sulawesi, 2018 earthquake-tsunami event. We introduce large-scale models (89 and 518 million elements), which capture this event with great detail. A comparison with one-way linking reveals that the overall tsunami is captured correctly; however, the fully coupled model results in a more complex wavefield. The third scenario is an elastic-acoustic scenario for an earthquake induced by the stimulation of an enhanced geothermal system in the metropolitan region of Helsinki. We model the earthquake and the sound it generates. Our results match the observations to first order. We generate maps highlighting areas of high velocity or high sound pressure levels. LTS led to a speedup of 30 for the largest Palu setup and 65 for the fully coupled Helsinki setup.

We discuss single-node performance, non-uniform memory access, and mesh partitioning for the fully coupled model. Finally, we show strong scaling results for the Palu scenario. The largest setup achieves a parallel efficiency of 83 % when scaling from 500 to 6000 nodes of the cluster Frontera. When using the wiggle factor, the time-to-solution at the largest scale is 74.7 % of the simulation without.

These results show that the resulting solver is stable, reaches high-order convergence, scales to large machines, and can be used for multiple applications.

Contents

Chapter 1.

Introduction

A devastating M_W 7.5 earthquake and a resulting tsunami hit Palu, Sulawesi, in September 2018, causing thousands of deaths and displacing hundreds of thousands of people [117]. This is an example of the devastating effect of large earthquakes. Most earthquakes are smaller events; however, they can still affect people.

Consider the Otaniemi project, an enhanced geothermal system (EGS) in the Helsinki metropolitan region. In 2018 and 2020, the operator of this project, St1 Deep Heat Oy, pumped millions of liters of water down the EGS's injection well. Such stimulations are necessary to increase the reservoir flow rate, but, as a side-effect, they lead to seismicity. For example, the 2018 stimulation induced thousands of small earthquakes [90]. These earthquakes were not dangerous, but they annoyed the public. The macroseismic questionnaire of the Institute of Seismology, University of Helsinki, collected over 300 reports of felt or heard effects from these earthquakes [4, 64, 91, 128].

Even though both events were of vastly different sizes, both impacted human lives. They have in common that they illustrate that effects from the coupling between solids (earthquakes) and fluids (ocean and air) can endanger and discomfort the public. We can model the solid as an elastic medium and the fluids as an acoustic medium. This thesis introduces a fully coupled elastic-acoustic model and its realization as high-performance software that runs on large supercomputers and can simulate both events mentioned above.

But what does the term “fully coupled” mean? A fully coupled model simulates the entire physical process that drives an event. For earthquake-tsunami coupling, this includes earthquake rupture, wave propagation in elastic and acoustic media and tsunami propagation. The earthquake rupture process should be modeled by a dynamic rupture model, a physics-based description that couples the fracture mechanics with the wave propagation [123]. Furthermore, the coupling should be bi-directional. For example, elastic waves should be able to influence the acoustic medium and vice versa. These properties allow us to capture wave types that other methods cannot, which is essential when comparing with measurements. The available data is continuously growing as more related sensors are being deployed. For example, the DONET in Nakai Trough [74] or the S-net in the Japan trench [184] are arrays of seafloor sensors. An increasingly popular class of sensors is distributed acoustic sensing (DAS), which uses fiber-glass cables and can measure the strain over a long distance with high spatiotemporal accuracy [45, 96, 116]. They can use pre-existing fiber networks, which makes it easy to deploy and surveil a large area. Hence, DAS can provide economical sensors on the seafloor [129, 143]. They can record ocean waves with an increasingly high resolution. Acoustic waves, which move much faster than tsunami waves and thus arrive earlier, could be used for tsunami early warning. Another type of wave, T Waves, propagate for long distances because they are trapped in the SOFAR channel, a low-velocity ocean layer. They can be used to detect small earthquakes [115]. Another type of interesting acoustic signal emitted by earthquakes is infrasound [42, 60, 112, 140].

We extend a two-dimensional model, first published by [97]. The critical insight of this approach is that tsunami propagation can be included by an elegant linearization of the free surface condition on the moving seafloor, which is computationally efficient. In [97], the model was discretized with a summation-by-parts finite difference method. It was used to investigate the validity of initial conditions used for tsunami simulations [99] and to simulate megathrust earthquakes and tsunamis in the Japan trench, the Nakai Trough and the Cascadia subduction zone [98]. The same physical model was also discretized by a finite element method in [177] and was used to run fully coupled simulations in the Cascadia subduction zone. Another fully coupled model, which uses a body force instead of a boundary condition and a finite difference discretization, was used to provide three-dimensional fully coupled simulations [103].

Our implementation [79] was the first to simulate a real-world event in 3D. We use the ADER-DG discretization, which consists of two parts. The first part is the Discontinuous Galerkin (DG) space discretization. Initially developed for the neutron transport equation [124], it was later extended to hyperbolic partial differential equations (PDEs) [16]. It combines capabilities for complex geometries, high-order accuracy, and an explicit semi-discrete form. As the DG method ensures local conservation, it is well suited for the simulation of conservation laws [61]. These properties make it very attractive for the simulation of earthquakes. Because of these advantages, the DG method is now widely used for elastic [35] and coupled acoustic-elastic simulations [7, 43, 175], and many other applications [20, 61]. Still, other numerical methods such as the finite difference method [97, 98], the finite volume method [36], and the continuous finite element method [10, 76, 177] can also be used for earthquake simulations. The second ingredient of the ADER-DG method is the Arbitrary Derivative (ADER) time discretization [156, 157, 159]. It uses a local expansion of the numerical solution to predict the time evolution and later corrects for neighboring elements. This results in a one-step numerical method of arbitrary order in both time and space. It was originally developed for linear problems but was later extended to non-linear equations and used for a large class of hyperbolic equations [33, 125].

We use and extend the software SeisSol in this thesis. SeisSol is a high-performance computing (HPC) earthquake dynamics and wave propagation solver that uses an ADER-DG discretization. It was developed in two phases. In the first phase, it was a research software for numerical methods, which included many physical models but was not tuned for performance. In the second phase, SeisSol's core was rewritten as an optimized HPC software with less functionality. In fact, in 2020, [164] observed that there has never been a SeisSol code base that combines the functionality of the first phase with the performance of the second phase. However, as of now, this statement is increasingly less true. SeisSol currently supports isotropic [35] and anisotropic [30, 180] elastic materials. Further materials are poroelasticity [29, 181], viscoelasticity [70, 165] and off-fault plasticity [182]. It also supports elastic-acoustic coupled materials [3, 69, 79, 82], which is the main focus of this thesis. As earthquake sources, it supports kinematic [72] and dynamic earthquake rupture [27, 118, 119]. Finally, it has been tuned for CPU [14, 15, 57, 58, 79, 166, 168] and GPU [32] architectures. SeisSol makes use of asynchronous output [126]. We integrated all features discussed in this thesis into version 1.0.1 [167] of SeisSol.

SeisSol is heavily optimized for modern hardware architectures. But how long can we expect the performance of our software to increase? Moore's law is the name of the observation that the number of transistors in an integrated circuit doubles every few years [107, 111]. Initially postulated in 1965, the exact rate of doubling has been revised many times. At some point, the doubling has to stop due to physical limits. While other factors contribute to the increase in the performance of hardware, such as microarchitectural improvements, the future of Moore's law is increasingly uncertain. However, there is another way of improving the performance of HPC applications: improve the algorithms. We can state an "algorithmic Moore's law", as named by [24]: Algorithmic advances can provide exponential speedup! This has been shown empirically for certain applications [107].

This law also applies to earthquake simulations. Especially in elastic-acoustic simulations, we observe strong differences in the wave speeds between materials. As we use explicit time-integration and thus must adhere to the CFL condition [22], this leads to an equally strong heterogeneity of the legal time step sizes. Using local time-stepping (LTS) can lead to significant speedups. The clustered LTS approach of [13] fuses HPC performance with the time-to-solution speedup of LTS. However, during the work on the fully coupled simulations, we noticed that this implementation of LTS was not robust: Some simulations did not finish. Hence, we present a novel LTS formalism based on an actor approach that combines state machines to keep track of the local state, with explicit message passing, which propagates the updates.

This thesis presents an efficient and robust solver for a fully coupled earthquake-tsunami model. In detail, we make the following contributions:

- In chapter 2, we derive models for wave propagation in the Earth and ocean. We show how both models can be elegantly coupled to achieve a fully coupled model that encompasses earthquake rupture, elastic (Earth), and acoustic wave propagation (ocean), and tsunami propagation.

- Chapter 3 introduces Riemann problems and their solution for coupled and elastic media. This includes discussing how we can include boundary conditions in our model.

- We present an ADER-DG discretization of our fully coupled model in chapter 4. This extends the state-of-the-art with a discussion on efficiently computing the gravitational boundary condition.

- In chapter 5, we prove the semi-discrete stability of the discretization of the ocean part of our fully coupled model, including the gravitational boundary condition.

- In chapter 6, we present a novel formulation of local-time-stepping that uses the actor model to schedule updates.
- Chapter 7 briefly summarizes how we can combine earthquake and tsunami simulations. We introduce common approximations used in tsunami modeling and compare standard one-way linking strategies with our fully coupled approach.
- In chapter 8, we verify the correctness of our implementation. To do this, we discuss analytical solutions targeting specific parts of our model. We start by investigating planar waves in acoustic and elastic media and evaluate which order of convergence our model achieves. We use two scenarios for wave propagation in coupled media (Snell's law and Scholte wave) and one model for tsunami generation in acoustics.
- We verify the performance of our model with multiple scenarios in chapter 9. We start with a simple benchmark scenario for tsunami generation. We apply our fully coupled model to the Palu, Sulawesi, 2018 earthquake-tsunami event. Furthermore, we show how we can use our method to simulate elastic and sound waves emitted from an earthquake induced by the stimulation of an enhanced geothermal system (EGS) in the metropolitan area of Helsinki.
- In chapter 10, we discuss high-performance computing aspects such as single node performance for AMD Rome and Fujitsu A64FX CPUs, how to pin threads to accommodate non-uniform memory access (NUMA), and modifications to the mesh partitioning scheme necessitated by the fully coupled scheme. We evaluate the parallel efficiency of our Palu scenario by running a strong scaling test.
- Chapter 11 summarizes the thesis and mentions further research avenues.

Several contributions of this thesis were published in peer-reviewed journals and conference proceedings. They are the result of two significant collaborations: The first one considered earthquake-tsunami coupling. It resulted in two papers. The first, Krenz, Uphoff, Ulrich, Gabriel, Abrahams, Dunham, and Bader "3D Acoustic-Elastic Coupling with Gravity: The Dynamics of the 2018 Palu, Sulawesi Earthquake and Tsunami" *SC' 21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2021) [79], introduced our implementation of the fully-coupled model (chapters 2 to 4) and used it to simulate an earthquake-tsunami benchmark (section 9.1) and the Palu, Sulawesi, 2018 earthquake and tsunami (section 9.2). Furthermore, it investigated HPC aspects, which I explain in sections 10.1 to 10.3. The second paper, Abrahams, Krenz, Dunham, Gabriel, and Saito "Comparison of Methods for Coupled Earthquake and Tsunami Modelling" *Geophysical Journal International* (2023) [3], discussed geophysical aspects of earthquake-tsunami coupling. Chapter 7 summarizes this paper, which also introduces the test case that I describe in section 8.4

The second collaboration investigated earthquake-sound coupling. It resulted in the paper Krenz, Wolf, Hillers, Gabriel, and Bader "Numerical Simulations of Seismoacoustic Nuisance Patterns from an Induced M 1.8 Earthquake in the Helsinki, Southern Finland, Metropolitan Area" *Bulletin of the Seismological Society of America* (2023) [82], which applied the fully-coupled model to simulate an EGS-induced earthquake in the Helsinki metropolitan area (section 9.3). Furthermore, it used but did not describe the LTS scheduling scheme introduced in chapter 6.

In this thesis, I present multiple as of now unpublished results. This includes a new discretization of the gravitational boundary condition (section 4.3) and a proof of the semi-stability of the discretization (chapter 5). Furthermore, as an algorithmic contribution, I describe a new LTS scheduling scheme in chapter 6. The convergence studies in chapter 8 use established scenarios. The results for the acoustic and acoustic-elastic test cases are new. Additionally, I simulated both earthquake-tsunami scenarios with the new implementation of the gravitational boundary and LTS to verify their correctness and performance (sections 9.1 and 9.2). Finally, I present single node performance results for the A64FX architecture in section 10.1 and show new strong-scaling results in section 10.4.

Chapter 2.

Equations

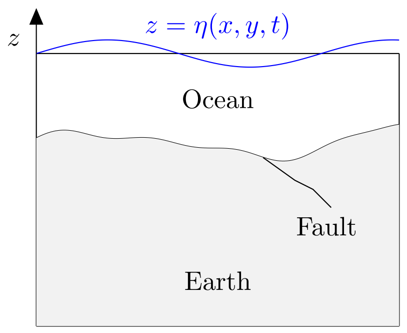


Figure 2.1.: Two-dimensional slice of our three-dimensional setup. It uses two coupled models: In the Earth (shaded gray), we use the elastic wave equation to describe the propagation of seismic waves and a separate model for earthquake rupture at the fault. We use the acoustic wave equation to model acoustic waves in the ocean and include tsunami propagation by modeling a moving sea surface (blue) with height $\eta(x, y, t)$. The black line on top of the ocean illustrates the unperturbed ocean at $z = 0$.

This chapter describes the physical laws governing wave propagation in coupled elastic-acoustic media with gravity. Figure 2.1 depicts the resulting setup. In section 2.1, we describe the equations that govern the solid part of our setup, including earthquake rupture along a fault. Section 2.2 explains how we model the fluid. In section 2.2.1, we detail the Euler equations and the physical boundary conditions required for tsunami propagation. We linearize the Euler equations around a hydrostatic background stress in section 2.2.2 to create an efficient model. Finally, section 2.3 shows how we can combine both models.

Throughout this thesis, we work in a Cartesian coordinate system with three-dimensional space coordinates $\boldsymbol{x} = (x, y, z)^T$ and the time coordinate t. The sea surface is initially at $z = 0$ and is later perturbed to $z = \eta(x, y, t)$. Similarly, the seafloor is located initially at $z = -H(x, y)$ and is perturbed by an uplift $b(x, y, t)$ to $z = -h(x, y, t)$ with $h(x, y, t) = H(x, y) + b(x, y, t)$. We use the convention that z points upwards.

2.1. Earthquakes

In this section, we describe the linear elastic wave equations, which govern wave propagation in elastic solids for small perturbations. They are a system of linear PDEs. Furthermore, we briefly introduce how earthquakes are sourced. This gives us the theoretical background to describe the entire dynamics of an earthquake from rupture to wave propagation.

2.1.1. Linear Elasticity

This part of the description follows [141] and [95, Cha. 22]. We consider a Lagrangian approach. A particle initially at $\boldsymbol{p_0}$ that moved to $\boldsymbol{p}$ at time t has a displacement of

$$\boldsymbol{u}(\boldsymbol{p_0}, t) = \begin{pmatrix} u_1(\boldsymbol{p_0}, t) \\ u_2(\boldsymbol{p_0}, t) \\ u_3(\boldsymbol{p_0}, t) \end{pmatrix} = \boldsymbol{p} - \boldsymbol{p_0}. \tag{2.1}$$

From the displacement, we define the velocity and acceleration vectors

$$\boldsymbol{v}(\boldsymbol{x}, t) = \begin{pmatrix} v_1(\boldsymbol{x}, t) \\ v_2(\boldsymbol{x}, t) \\ v_3(\boldsymbol{x}, t) \end{pmatrix} = \frac{\partial \boldsymbol{u}}{\partial t}, \quad \boldsymbol{a}(\boldsymbol{x}, t) = \begin{pmatrix} a_1(\boldsymbol{x}, t) \\ a_2(\boldsymbol{x}, t) \\ a_3(\boldsymbol{x}, t) \end{pmatrix} = \frac{\partial \boldsymbol{v}}{\partial t}. \tag{2.2}$$

Now, we consider a particle that moved from a reference position $\boldsymbol{x_0}$ to a nearby new position $\boldsymbol{x}$. We linearize equation (2.1) around $\boldsymbol{x_0}$, which results in

$$\boldsymbol{u}(\boldsymbol{x_0} + \boldsymbol{x'}, t) \approx \boldsymbol{u}(\boldsymbol{x_0}, t) + \boldsymbol{J}\boldsymbol{x'}, \tag{2.3}$$

where we introduced the Jacobian of the displacement

$$J_{ij} = \frac{\partial u_i}{\partial j}. \tag{2.4}$$

This linearization assumes that the space derivatives of the displacement are small, which is typically true in seismology [141].

Following [95], we split the Jacobian into the symmetric strain matrix $\boldsymbol{\varepsilon}$ and the skew-symmetric rotation matrix $\boldsymbol{\Omega}$, leading to

$$J_{ij} = \varepsilon_{ij} + \Omega_{ij}, \tag{2.5}$$

with

$$\varepsilon_{ij} = \frac{1}{2}\left(J_{ij} + J_{ji}\right) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{2.6}$$

$$\Omega_{ij} = \frac{1}{2}\left(J_{ij} - J_{ji}\right) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right). \tag{2.7}$$

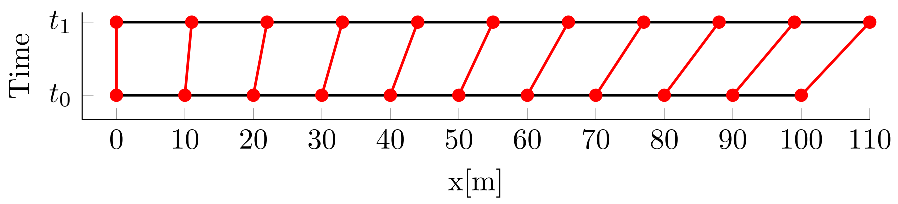


Figure 2.2.: This figure gives an example of extensional strain, which is the length change relative to the length. It is inspired by an example in [141]. Assume that a string had an initial length of 100 m at $t = t_0$. It is fixed at $x = 0$. At $t = t_1$, it was stretched to 110 m. The red dots indicate the position of particles on this string. We can see that the displacement of a particle depends on its position on the string. At $x = 0$, the displacement is zero and at $x = 100$ m, the particle is displaced 10 m to the new position at 110 m. However, the strain is 0.1 at every position.

In the following, we ignore solid-body rotations and only consider the strain matrix. Off-diagonal entries of $\boldsymbol{\varepsilon}$ lead to shear strain, while the trace

$$\Theta = \operatorname{tr}(\boldsymbol{\varepsilon}) = \varepsilon_{11} + \varepsilon_{22} + \varepsilon_{33} = \nabla \cdot \boldsymbol{u} \tag{2.8}$$

measures the volume change and is called the dilation. As Figure 2.2 shows, the displacement measures the absolute change of particle positions, while the strain is a measure of the relative deformation.

The strain results in internal forces called stress. Our description follows [141]. Assume that we have an infinitesimal plane defined by its normal vector $\boldsymbol{n} = (n_x, n_y, n_z)^T$. The traction $\boldsymbol{T}(\boldsymbol{n}) = (t_1(\boldsymbol{n}), t_2(\boldsymbol{n}), t_3(\boldsymbol{n}))^T$ gives the force per unit area in the direction of $\boldsymbol{n}$. We have the point symmetry $\boldsymbol{T}(-\boldsymbol{n}) = -\boldsymbol{T}(\boldsymbol{n})$. The traction normal to the plane is called normal stress, and the traction parallel to the plane is called shear stress. Figure 2.3 visualizes how traction acts on the surfaces of an infinitesimal cube. The stress tensor $\boldsymbol{\sigma}$ maps the normal vector to tractions. It is defined as

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{pmatrix} = \begin{pmatrix} t_1(\hat{\boldsymbol{x}}) & t_1(\hat{\boldsymbol{y}}) & t_1(\hat{\boldsymbol{z}}) \\ t_2(\hat{\boldsymbol{x}}) & t_2(\hat{\boldsymbol{y}}) & t_2(\hat{\boldsymbol{z}}) \\ t_3(\hat{\boldsymbol{x}}) & t_3(\hat{\boldsymbol{y}}) & t_3(\hat{\boldsymbol{z}}) \end{pmatrix}, \tag{2.9}$$

which uses the tractions acting on each face of our cube that have the normal vectors $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}$ and $\hat{\boldsymbol{z}}$.

We assume that our solid is in static equilibrium. Hence, there can be no net rotation, and thus, the tensor $\boldsymbol{\sigma} = \boldsymbol{\sigma}^T$ is symmetric. Finally, the traction across an arbitrary plane with normal $\boldsymbol{n}$ is given by

$$T_i(\boldsymbol{n}) = \sum_j \sigma_{ji} n_j = \sigma_{ji} n_j, \tag{2.10}$$

where we used the Einstein summation notation, meaning repeated free indices indicate an implied summation. We will use this notation throughout this work.

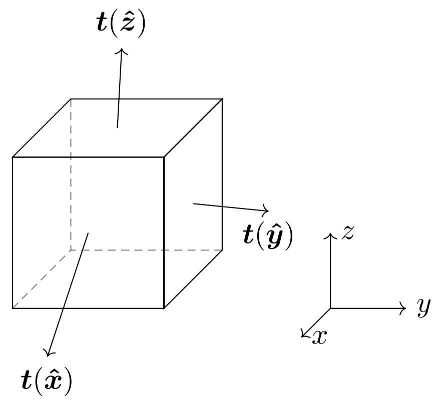


Figure 2.3.: An infinitesimal cube. Its yz, xz and xy planes have the normal vectors $\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$, respectively. The forces acting on the faces of this cube are given by the tractions $\boldsymbol{t}(\hat{\boldsymbol{x}}), \boldsymbol{t}(\hat{\boldsymbol{y}})$ and $\boldsymbol{t}(\hat{\boldsymbol{z}})$. Figure reproduced from [141].

Next, we discuss the relationship between stress and strain. We assume a linear stress-strain relation

$$\sigma_{ij} = C_{ijkl}\varepsilon_{kl} \tag{2.11}$$

with the tensor $\boldsymbol{C}$. Equation (2.11) is also called generalized Hooke's law [141]. It is a good assumption for small perturbations, which are typical in seismology. Note, however, that the relation discussed here does not apply to large background stresses [141], as the linearity assumption is not valid in this regime.

Out of 81 entries of $\boldsymbol{C}$, only 21 are independent due to symmetry, which follows directly from the symmetry of the stress $\boldsymbol{\sigma}$ and strain tensor $\boldsymbol{\varepsilon}$, and thermodynamical considerations [5, 92]. For further details, we refer the interested reader to the discussion in [92, §4]. We make the additional assumption of an isotropic material, for which $\boldsymbol{C}$ does not depend on rotation. This assumes that the speed of a wave does not depend on its direction. By introducing the Kronecker delta

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{else,} \end{cases} \tag{2.12}$$

we can express the isotropic stress-strain relation as

$$c_{ijkl} = \lambda\delta_{ij}\delta_{kl} + \mu\left(\delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}\right). \tag{2.13}$$

Then, the stress tensor further simplifies to

$$\sigma_{ij} = \lambda\delta_{ij}\varepsilon_{kk} + 2\mu\varepsilon_{ij} \tag{2.14}$$

which we can express in terms of the dilation (equation (2.8)) as

$$\boldsymbol{\sigma} = \lambda\Theta\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}. \tag{2.15}$$

The quantities λ and μ are called Lamé parameters and have, like the stress, dimensions of force per unit area, which in SI units corresponds to Pascal (Pa), defined as $1\,\mathrm{Pa} = 1\,\mathrm{N\,m^{-2}}$. The shear modulus μ measures the resistance of a material to shearing, as the relation

$$\mu = \frac{\sigma_{12}}{2\varepsilon_{12}} = \frac{\sigma_{13}}{2\varepsilon_{13}} = \frac{\sigma_{23}}{2\varepsilon_{23}}, \tag{2.16}$$

which is a direct consequence of equation (2.14), demonstrates. In a fluid, there are no shear stresses; thus, $\mu = 0$ vanishes.

As λ does not have a convenient physical interpretation, the bulk modulus $K = \lambda + 2/3\mu$ is often used instead. It relates the mean normal stress, defined as one-third of the trace of $\boldsymbol{\sigma}$, to the dilation (equation (2.8))

$$\frac{1}{3}\operatorname{tr}(\boldsymbol{\sigma}) = K\Theta, \tag{2.17}$$

which follows from equation (2.14) [95, Sec. 22.1.2]. In a fluid, as a direct consequence of equation (2.15), the stress tensor is given by $\boldsymbol{\sigma} = K\Theta\boldsymbol{I}$. Defining the pressure as

$$p = -K\Theta, \tag{2.18}$$

the stress tensor in a fluid can be written as $\boldsymbol{\sigma} = -Kp\boldsymbol{I}$.[1] The bulk modulus K relates the pressure to the volume change. Hence, it measures the incompressibility of the material [141]. Furthermore, the density of our material is called ρ and has dimensions of mass per length cubed and thus SI units of $\mathrm{kg\,m^{-3}}$.

We can now give the wave speeds of the two body wave types in seismology. The faster wave is called primary or pressure wave (P wave) and has a speed of

$$c_p = \sqrt{\frac{\lambda + 2\mu}{\rho}}, \tag{2.19}$$

and the secondary or shear wave (S wave) has a speed of

$$c_s = \sqrt{\frac{\mu}{\rho}}. \tag{2.20}$$

Shear waves do not propagate in fluids and thus $c_s = 0$ there.

Next, we derive the momentum equation for elastic solids. Our derivation follows [5]. We consider an arbitrary three-dimensional domain Ω with boundary $\partial\Omega$. The momentum equation is simply Newton's second law, often stated as $\boldsymbol{F} = m\boldsymbol{a}$: The force $\boldsymbol{F}$ is equal to mass m times the acceleration $\boldsymbol{a}$. In our case, the mass $m = \int_\Omega \rho \,\mathrm{d}\boldsymbol{x}$ can be written as the volume integral of the density, and the forces are given in terms of the traction and a vector $\boldsymbol{f}$ that summarizes the body forces. As total force has units of Newton N, it is clear that the traction has units of force per unit area (Pa), while $\boldsymbol{f}$ has units of force per volume ($\mathrm{Nm^{-3}}$).

[1] Note this leads to a different sign convention: pressure is positive in compression; stress is negative.

The rate of change of the momentum of particles is balanced with the forces acting on this particle, resulting in

$$\frac{\partial}{\partial t}\int_\Omega \rho \frac{\partial \boldsymbol{u}}{\partial t}\,\mathrm{d}\boldsymbol{x} = \int_{\partial\Omega} \boldsymbol{T}(\boldsymbol{n})\,\mathrm{d}S + \int_\Omega \boldsymbol{f}\,\mathrm{d}\boldsymbol{x}. \tag{2.21}$$

After using equation (2.10) and the divergence theorem, we arrive at

$$\int_{\partial\Omega} T_i\,\mathrm{d}S = \int_{\partial\Omega} \sigma_{ji} n_j\,\mathrm{d}S = \int_\Omega \frac{\partial \sigma_{ji}}{\partial j}\,\mathrm{d}\boldsymbol{x}, \tag{2.22}$$

for each component i, thus turning the surface integral into a volume integral. From this, the relation

$$\frac{\partial}{\partial t}\int_\Omega \rho \frac{\partial u_i}{\partial t}\,\mathrm{d}\boldsymbol{x} - \int_\Omega \frac{\partial \sigma_{ji}}{\partial j}\,\mathrm{d}\boldsymbol{x} = \int_\Omega f_i\,\mathrm{d}\boldsymbol{x} \tag{2.23}$$

follows. As ρ stays constant, we are allowed to write the integral as

$$\int_\Omega \rho \frac{\partial u_i}{\partial t^2} - \frac{\partial \sigma_{ji}}{\partial j}\,\mathrm{d}\boldsymbol{x} = \int_\Omega f_i\,\mathrm{d}\boldsymbol{x}. \tag{2.24}$$

Because we follow a Lagrangian description, the domain and its boundary move with the particles. We, as usually done in seismology, ignore this and equate the total derivative of the displacement $\boldsymbol{u}$ with the partial derivative [5, 141].

Finally, this is equivalent to

$$\rho \frac{\partial^2 u_i}{\partial t^2} = \frac{\partial \sigma_{ik}}{\partial x_k} + f_i, \tag{2.25}$$

as the domain Ω is arbitrary.[2] Equation (2.25) can be written for the velocity $\boldsymbol{v} = \frac{\partial \boldsymbol{u}}{\partial t}$ as

$$\rho \frac{\partial v_i}{\partial t} = \frac{\partial \sigma_{ik}}{\partial x_k} + f_i. \tag{2.26}$$

Inserting the strain-stress relation (equation (2.11)) and the definition of the strain tensor (equation (2.6)) into equation (2.26) would result in the second order formulation of the elastic wave equation. In this work, however, we consider the velocity-stress formulation. We can derive it by differentiating the strain-stress relation (equation (2.11)) in time and combining it with equation (2.26). Assuming an isotropic material, we arrive at

$$\frac{\partial \sigma_{ij}}{\partial t} - \lambda \delta_{ij} \frac{\partial v_k}{\partial x_k} - \mu \left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i} \right) = 0, \tag{2.27}$$

$$\rho \frac{\partial v_i}{\partial t} - \frac{\partial \sigma_{ij}}{\partial x_j} = f_i. \tag{2.28}$$

[2]This follows from the vanishing theorem: Let $\int_a^b f(x)\,\mathrm{d}x = 0$ for a continuous function $f(x)$ in the closed interval $[a, b]$. Then, $f(x)$ is identically zero. For a proof, see for example [149, Sec. A.1].

The material parameters μ, λ, ρ may depend on space. We collect the unknowns in the vector $\boldsymbol{q} = (\sigma_{11}, \sigma_{22}, \sigma_{33}, \sigma_{12}, \sigma_{23}, \sigma_{13}, v_1, v_2, v_3)$ [35]. We summarize equations (2.27) and (2.28) in the form

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{A}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial x} - \boldsymbol{B}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial y} - \boldsymbol{C}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial z}, \tag{2.29}$$

with the flux matrices[3] given by

$$\boldsymbol{A} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -\lambda - 2\,\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu \\ -\frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 \end{pmatrix},$$

$$\boldsymbol{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda - 2\,\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{2.30}$$

$$\boldsymbol{C} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\lambda - 2\,\mu \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

As a direct consequence of the isotropic material, all matrices share the same set of eigenvalues, $(-c_p, -c_s, -c_s, 0, 0, 0, c_s, c_s, c_p)$. Further, note that the eigenvalues $\pm c_s$ have a multiplicity of 2 while the eigenvalues $\pm c_p$ have a multiplicity of 1. Hence, the wave structure consists of two S waves and one P wave in each direction. Equation (2.29) is the general form of a non-conservative linear hyperbolic PDE. A PDE of this form is

[3]These are often called Jacobians, which is incorrect because the equations are not conservative [169].

hyperbolic if the plane-wave operator

$$\hat{\boldsymbol{A}}(\boldsymbol{x},\boldsymbol{n}) = n_x\boldsymbol{A}(\boldsymbol{x}) + n_y\boldsymbol{B}(\boldsymbol{x}) + n_z\boldsymbol{C}(\boldsymbol{x}) \tag{2.31}$$

is diagonalizable with real eigenvalues for every unit vector $\boldsymbol{n}$ and for all $\boldsymbol{x}$ [95, Def. 18.7.1]. In addition, as the material can be inhomogeneous, leading to spatially varying flux matrices, we call equation (2.29) a variable coefficient equation.

We have described thus far how we can set up the elastic wave equations to describe wave propagation in the interior of our domain. To close the problem, we need to introduce boundary conditions. In the elastic part, we can prescribe the traction $\boldsymbol{T}\cdot\boldsymbol{n}$ or the velocity at the boundary [95]. In practice, we are only interested in two special cases.

The first is called the free surface boundary condition. It models the interface between the Earth and a fluid with negligible density. Physically, we enforce the condition

$$\boldsymbol{T}\cdot\boldsymbol{n} = 0, \tag{2.32}$$

i.e., the traction should be zero.

The second is the absorbing boundary condition. It allows us to use a simulated region that is smaller than the physical region. We set it up such that waves can exit the domain but cannot enter it. We are using a naive but imperfect approach: While it allows waves to exit the domain, it leads to reflections coming from the boundary for waves that arrive non-planarly, which is a known problem [95]. It can be fixed by expensive and complicated boundary conditions such as the perfectly matched layer. As a discussion of these and alternatives is out of scope for this thesis, we refer the interested reader to the discussion in [46].

2.1.2. Earthquake Sourcing

Earthquakes can be triggered by different sourcing mechanisms. We assume that we have a displacement discontinuity—called slip—along a fault, which is embedded in Earth. This discontinuity leads to a problem: On the fault, our PDEs do not hold anymore. Hence, we need to define separate models for this.

In this section, we briefly discuss two different earthquake sourcing models. The first one, kinematic point sources, collapses the entirety of the source into single points. This assumption results in a body force that summarizes the displacement, which is valid only under certain assumptions [5]. The second one, dynamic rupture (DR), describes earthquake ruptures based on the physics of frictional contact. This results in an interior boundary condition along the fault. It is a physics-based model that requires more information about the source. Furthermore, it complicates the mesh generation, as the geometry of the fault now needs to be integrated into the mesh. Finally, it necessitates more complicated numerical solvers. For both methods, including multiple sources in the model is possible.

Kinematic sources The idea behind point sources is to collapse the entire earthquake sourcing to one point. This assumes that the wavelength of the observed waves is longer than the fault surface. Our description of the source follows [72]. A point source adds a source term of the form

$$\boldsymbol{S}(\boldsymbol{x}, t) = s(t)\delta(\boldsymbol{x} - \boldsymbol{x_c}), \tag{2.33}$$

where $s(t)$ is some source time function. Due to the delta function, equation (2.33) acts on a single point $\boldsymbol{x_c}$.

We use the symmetric moment tensor

$$M_{pq}(t) = AS_i(t)c_{ijpq}n_j \tag{2.34}$$

which is a collection of the force couples M_{pq}, i.e., pairs of opposing forces separated in direction q acting towards the direction p [141]. Here, $\boldsymbol{n}$ is the normal of the fault surface. The moment tensor depends on the area of the fault A and a slip rate function $S(t)$ that summarizes the time-dependent behavior of the source function. For a detailed explanation of the theory of point sources, we refer the interested reader to [5]. This force is typically applied to Newton's second law, i.e., it modifies the velocities. SeisSol's implementation instead modifies the stress tensor directly, resulting in

$$\hat{\sigma}_{ij} = \sigma_{ij}(\boldsymbol{x}, t) - M_{ji}(t)\delta(\boldsymbol{x} - \boldsymbol{x_c}), \tag{2.35}$$

which is numerically more convenient [72, 164]. We can plug this term into equation (2.27), which results in a body force for the stresses. It is possible, and for large events necessary, to split the faults into sub-faults and use a separate point source for each subfault [72].

Dynamic rupture Our description follows [26] and the notation follows [164]. Across the fault, we assume that the traction is continuous, but the displacement is not. The jump in velocity across the fault is called the slip rate. It is defined as $[\![\boldsymbol{v}]\!] = \boldsymbol{v^R} - \boldsymbol{v^L}$. Here, the superscripts R and L indicate the right and left sides of the fault. We denote the tangential part of $[\![\boldsymbol{v}]\!]$ as $\boldsymbol{s}$. The shear traction $\boldsymbol{\tau}$ is defined as the tangential part of the traction $\boldsymbol{t}$. Finally, we call the normal stress at the fault $\boldsymbol{\sigma_n}$.

The conditions

$$\begin{aligned} \|\boldsymbol{\tau}\| &\leq \tau_s, \\ \tau_s \boldsymbol{s} - \boldsymbol{\tau}\|\boldsymbol{s}\| &= 0, \end{aligned} \tag{2.36}$$

with frictional strength τ_s have to hold at the fault. The frictional strength is evolved by

$$\begin{aligned} \tau_s &= \max(0, -\sigma_n f(\|\boldsymbol{s}\|, \psi)), \\ \frac{\mathrm{d}\psi}{\mathrm{d}t} &= g(\|\boldsymbol{s}\|, \psi), \end{aligned} \tag{2.37}$$

where f and g define a possible non-linear friction law.

Hence, dynamic rupture models the frictional contact at a fault. As a consequence of equation (2.36), as long as the fault strength is larger than the shear traction, the fault remains locked and slip is not initiated. The fault breaks after the shear traction grows larger than the fault strength. The friction law (equation (2.37)) interacts with the wavefield, which can lead to complex, non-linear patterns. We refer the interested reader to [26, 38], which explains the interface conditions in more detail.

2.2. Fluid Mechanics

In this section, we describe the equations governing fluid flow. We start by introducing the non-linear Euler equations in section 2.2.1 together with boundary conditions that govern tsunami propagation. We then linearize the resulting system in section 2.2.2.

2.2.1. Euler Equations

The description and derivation of the Euler equations follows [149, Sec. 13.2] We consider a fluid in a region Ω with boundary $\partial\Omega$. The fluid has a density ρ and a velocity $\boldsymbol{v}$.

We can describe the mass in our region by the integral

$$m(t) = \int_\Omega \rho \,\mathrm{d}\boldsymbol{x}. \tag{2.38}$$

It is a conserved quantity that can only change due to the flux at the boundaries of the domain

$$\int_\Omega \frac{\partial \rho}{\partial t} \,\mathrm{d}\boldsymbol{x} = -\int_{\partial\Omega} \rho \boldsymbol{v} \cdot n \,\mathrm{d}S. \tag{2.39}$$

We use the divergence theorem to turn the surface integral into a volume integral, arriving at the conservation law

$$\int_\Omega \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) \,\mathrm{d}\boldsymbol{x} = 0. \tag{2.40}$$

As the domain Ω is arbitrary, we have that

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{v}) = 0, \tag{2.41}$$

which is called the continuity equation of fluid mechanics.

For a fluid, the momentum is another conserved quantity. Here, similar to the discussion for elastic bodies, we consider forces on the fluid, given by the pressure p and external forces. We summarize them in the vector $\boldsymbol{f}$. Again, a change of momentum can only come in from the boundaries, leading us to

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_\Omega \rho v_i \,\mathrm{d}\boldsymbol{x} + \underbrace{\int_{\partial\Omega} \rho v_i \boldsymbol{v} \cdot \boldsymbol{n} \,\mathrm{d}S}_{\text{momentum flux}} + \underbrace{\int_{\partial\Omega} p n_i \,\mathrm{d}S}_{\text{net pressure}} = \int_\Omega f_i \,\mathrm{d}\boldsymbol{x}, \tag{2.42}$$

for the ith component. Once again, we apply the divergence theorem, resulting in

$$\int_\Omega \frac{\partial (\rho v_i)}{\partial t} + \nabla \cdot (\rho v_i \boldsymbol{v}) + \frac{\partial p}{\partial x_i} - f_i \,\mathrm{d}\boldsymbol{x} = 0. \tag{2.43}$$

Using the fact that the domain Ω is arbitrary and after collecting terms in vectors, we arrive at

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\boldsymbol{v} \otimes \rho \boldsymbol{v} + \boldsymbol{I} p) = \boldsymbol{f}. \tag{2.44}$$

Here, $\boldsymbol{I}$ is an identity matrix of appropriate size and

$$(\boldsymbol{a} \otimes \boldsymbol{b})_{ij} = a_i b_j \tag{2.45}$$

denotes the Kronecker product. It takes two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ and creates a dyadic tensor.

In our case, we use the external force $\boldsymbol{f} = (0, 0, -\rho g)$, where $g = 9.81\,\mathrm{m\,s^{-2}}$ is the gravitational acceleration on Earth. This results in the conservation of momentum

$$\frac{\partial \rho \boldsymbol{v}}{\partial t} + \nabla \cdot (\boldsymbol{v} \otimes \rho \boldsymbol{v} + \boldsymbol{I} p) = \begin{pmatrix} 0 \\ 0 \\ -\rho g \end{pmatrix}. \tag{2.46}$$

The units are consistent with section 2.1: Pressure has units of force per area (Pa), and the external force $-\rho g$ has units of force per volume ($\mathrm{N\,m^{-3}}$).

Finally, the equation of state relates the pressure to the density. In our case, we assume a relation of the form

$$p = p(\rho), \tag{2.47}$$

i.e., the pressure is a function of only the density, which assumes isentropic flows [86]. More general equations of state can also depend on other variables, such as temperature. For isentropic flows, we do not need to consider the conservation of energy equation [95]. Hence, we neglect it.

We now consider the steady state, consisting of the hydrostatic pressure $p_0(z)$, density $\rho_0(z)$, and velocity $\boldsymbol{v_0} = 0$. Because the velocity is zero, the pressure gradient exactly balances the gravitational force. Hence, equation (2.46) reduces to the ordinary differential equation (ODE)

$$\frac{\mathrm{d}p_0}{\mathrm{d}z} = -\rho_0 g, \tag{2.48}$$

with initial condition

$$p_0(z = 0) = p_a, \tag{2.49}$$

where p_a is the atmospheric pressure. Using the equation of state (equation (2.47)) and the chain rule, we can write equation (2.48) as

$$\frac{\mathrm{d}p_0}{\mathrm{d}z} = \frac{\mathrm{d}p_0}{\mathrm{d}\rho_0}(\rho_0(z)) \frac{\mathrm{d}\rho_0}{\mathrm{d}z}. \tag{2.50}$$

Defining the bulk modulus in the equilibrium as

$$K_0(z) = \rho_0 \frac{\mathrm{d}p_0}{\mathrm{d}\rho_0}(\rho_0(z)), \tag{2.51}$$

we can give the pressure gradient as

$$\frac{\mathrm{d}p_0}{\mathrm{d}z} = \frac{K_0}{\rho_0} \frac{\mathrm{d}\rho_0}{\mathrm{d}z}(z). \tag{2.52}$$

Solving this for the density, we arrive at

$$\frac{\mathrm{d}\rho_0}{\mathrm{d}z}(z) = \frac{\rho_0}{K_0}\frac{\mathrm{d}p_0}{\mathrm{d}z}. \tag{2.53}$$

We have introduced the Euler equations for isentropic flow, which describe the fluid flow in the interior of our domain. To close the system, we need to introduce boundary conditions. For tsunami propagation, we enforce boundary conditions on the sea surface. First, we have the free surface boundary condition. It consists of two parts that are enforced on the moving sea surface with height $\eta(x, y, t)$. The first part is the dynamic boundary condition

$$p = p_a \quad \text{at } z = \eta(x, y, t). \tag{2.54}$$

The second part is the kinematic boundary condition

$$v_3 = \frac{\partial \eta(x, y, t)}{\partial t} + v_1 \frac{\partial \eta(x, y, t)}{\partial x} + v_2 \frac{\partial \eta(x, y, t)}{\partial y} \quad \text{at } z = \eta(x, y, t), \tag{2.55}$$

which relates the motion to the shape of the surface [131].[4] This boundary condition assumes that the ocean surface is continuous and thus that no wave breaking can occur.

On the seafloor, which is at $-h(x, y, t)$, we assume that the normal velocity

$$\boldsymbol{v} \cdot \boldsymbol{n} = 0 \quad \text{at } z = -h(x, y, t) \tag{2.56}$$

vanishes. We call this a rigid boundary. Following [131], we also have the kinetic boundary condition at the sea bottom

$$v_3 = -v_1 \frac{\partial h(x, y)}{\partial x} - v_2 \frac{\partial h(x, y)}{\partial y} \quad \text{at } z = -h(x, y, t). \tag{2.57}$$

2.2.2. Linear Acoustics with Gravity

This section derives the acoustic wave equation from the Euler equations. The resulting equations will accompany us for the rest of this thesis. We follow the derivation in [97] but provide more detail. We split the variables

$$\begin{aligned}
\boldsymbol{v}(x, y, z, t) &= \boldsymbol{v_0}(x, y, z) + \boldsymbol{v'}(x, y, z, t), \\
\rho(x, y, z, t) &= \rho_0(x, y, z) + \rho'(x, y, z, t), \\
p(x, y, z, t) &= p_0(x, y, z) + p'(x, y, z, t),
\end{aligned} \tag{2.58}$$

into a background state $(\boldsymbol{v_0}, p_0, \rho_0)$ and perturbations $(\boldsymbol{v'}, p', \rho')$ The background state stays constant over time. Furthermore, we assume that the background is in hydrostatic equilibrium, implying a zero background velocity $\boldsymbol{v_0} = 0$. This also means that the background pressure $p_0(z)$ and density $\rho_0(z)$ only vary vertically but not horizontally. Finally, we assume that we only have small perturbations. The following derivation is

[4]In other words, the level set $f(x, y, z, t) = z - \eta(x, y, t) = 0$ defines the shape of the free surface.

straightforward and consists of linearizing the equation of state, the conservation of both mass and momentum, and the boundary conditions.

We begin by linearizing the equation of state (equation (2.47)) around the background density ρ_0. For a small density perturbation, we have

$$\begin{aligned} p(\rho) &\approx p(\rho_0) + (\rho - \rho_0)\frac{\mathrm{d}p}{\mathrm{d}\rho}(\rho_0) \\ &= p(\rho_0) + \frac{\rho - \rho_0}{\rho_0}\rho_0\frac{\mathrm{d}p}{\mathrm{d}\rho}(\rho_0). \end{aligned} \tag{2.59}$$

Introducing the bulk modulus[5]

$$K(\rho_0) = \rho_0\frac{\mathrm{d}p}{\mathrm{d}\rho}(\rho_0), \tag{2.60}$$

and noticing that $\rho - \rho_0 = \rho'$, we arrive at the linearized equation of state

$$p(\rho) \approx p_0 + \frac{\rho'}{\rho_0}K(\rho_0). \tag{2.61}$$

Finally, after splitting the pressure with equation (2.58), we can solve for the density perturbation

$$\rho'(p') = p'\frac{\rho_0}{K}. \tag{2.62}$$

Hence, the concrete choice of the non-linearized equation of state is only reflected in K. However, the equation of state must have the form of equation (2.47), i.e., that the density only depends on the pressure.

Next, we look at the linearization of the mass conservation. Inserting our split variables (equation (2.58)) into equation (2.41) and realizing that the background state does not depend on time,

$$\underbrace{\frac{\partial \rho_0}{\partial t}}_{=0} + \frac{\partial \rho'}{\partial t} + \nabla \cdot \left((\rho_0 + \rho')(\underbrace{\boldsymbol{v_0}}_{=0} + \boldsymbol{v'}) \right) = 0, \tag{2.63}$$

we arrive at

$$\frac{\partial \rho'}{\partial t} + \nabla \cdot \big((\rho_0 + \rho')(\boldsymbol{v'})\big) = 0. \tag{2.64}$$

After inserting our linearized equation of state (equation (2.62)), we get

$$\frac{\rho_0}{K}\frac{\partial p'}{\partial t} + \nabla \cdot (\rho_0 \boldsymbol{v'}) + \nabla \cdot (\frac{\rho_0}{K}p'\boldsymbol{v'}) = 0. \tag{2.65}$$

[5] In general, it can be different than the one from the hydrostatic equilibrium (equation (2.51)), but, as in [97], we assume in the following that $K_0 = K$.

The second term only contains products of perturbations and is zero after linearization. Hence, we focus on the first term and expand it into its components

$$\nabla \cdot \left(\rho_0 \boldsymbol{v'}\right) = \frac{\partial \rho_0 v_1'}{\partial x} + \frac{\partial \rho_0 v_2'}{\partial y} + \frac{\partial \rho_0 v_3'}{\partial z} \tag{2.66}$$

$$= \rho_0 \frac{\partial v_1'}{\partial x} + \underbrace{v_1' \frac{\partial \rho_0}{\partial x}}_{=0} + \rho_0 \frac{\partial v_2'}{\partial y} + \underbrace{v_2' \frac{\partial \rho_0}{\partial y}}_{=0} + \rho_0 \frac{\partial v_3'}{\partial z} + v_3' \frac{\partial \rho_0}{\partial z} \tag{2.67}$$

$$= \rho_0 \nabla \cdot \boldsymbol{v'} - \frac{\rho_0}{K_0} \rho_0 g v_3'. \tag{2.68}$$

Equation (2.67) follows from the chain rule and equation (2.68) from the definition of the hydrostatic equilibrium (equations (2.48) and (2.53)). Inserting equation (2.67) into equation (2.65) results in

$$\frac{\rho_0}{K} \frac{\partial p'}{\partial t} + \rho_0 \nabla \cdot (\boldsymbol{v'}) = \frac{\rho_0}{K} \rho_0 g v_3', \tag{2.69}$$

which, after dividing by ρ_0, leads us to

$$\frac{1}{K} \frac{\partial p'}{\partial t} + \nabla \cdot (\boldsymbol{v'}) = \frac{\rho_0}{K} g v_3'. \tag{2.70}$$

Finally, we look at the conservation of momentum. For this, it is convenient to introduce the unit vectors

$$\boldsymbol{e_x} = (1,0,0)^T\,, \quad \boldsymbol{e_y} = (0,1,0)^T\,, \quad \boldsymbol{e_z} = (0,0,1)^T\,. \tag{2.71}$$

By inserting our perturbations equation (2.58) into equation (2.46), we get

$$\begin{aligned} \frac{\partial (\rho_0 + \rho')(\boldsymbol{v_0} + \boldsymbol{v'})}{\partial t} &+ \nabla \cdot \Big(\big((\boldsymbol{v_0} + \boldsymbol{v'}) \otimes \big((\rho_0 + \rho')(\boldsymbol{v_0} + \boldsymbol{v'}) \big) \big) + I(p_0 + p') \Big) \\ &= -(\rho_0 + \rho') g \boldsymbol{e_z}. \end{aligned} \tag{2.72}$$

We now look at this term by term. The time derivative is simplified to

$$\begin{aligned} \frac{\partial (\rho_0 + \rho')(\boldsymbol{v_0} + \boldsymbol{v'})}{\partial t} &= \frac{\partial \rho_0 \boldsymbol{v_0}}{\partial t} + \frac{\partial \rho_0 \boldsymbol{v'}}{\partial t} + \frac{\partial \rho' \boldsymbol{v_0}}{\partial t} + \frac{\partial \rho' \boldsymbol{v'}}{\partial t} \\ &= \frac{\partial \rho_0 \boldsymbol{v'}}{\partial t} + \frac{\partial \rho' \boldsymbol{v'}}{\partial t} \\ &\approx \rho_0 \frac{\partial \boldsymbol{v'}}{\partial t} \end{aligned} \tag{2.73}$$

because the background is in a steady state and hence, $\boldsymbol{v_0} = 0$ and $\frac{\partial \rho_0}{\partial t} = 0$. In the final step, we have made the approximation that we can ignore products of perturbations. For the same reasons, the term

$$\begin{aligned} \big((\boldsymbol{v_0} + \boldsymbol{v'}) \otimes \big((\rho_0 + \rho')(\boldsymbol{v_0} + \boldsymbol{v'}) \big) \big) &= \nabla \cdot \big(\boldsymbol{v'} \otimes ((\rho_0 + \rho') \boldsymbol{v'}) \big) \\ &\approx 0 \end{aligned} \tag{2.74}$$

vanishes completely, as every term of the product contains two multiplied velocity perturbations. We simplify the pressure term by inserting the hydrostatic background (equation (2.48))

$$\begin{aligned}\nabla \cdot \big(\boldsymbol{I}(p_0 + p')\big) &= \nabla \cdot \big(\boldsymbol{I}p'\big) + \underbrace{\frac{\partial p_0}{\partial x}\boldsymbol{e_x} + \frac{\partial p_0}{\partial y}\boldsymbol{e_y}}_{=0} + \frac{\partial p_0}{\partial z}\boldsymbol{e_z} \\ &= \nabla \cdot \big(\boldsymbol{I}p'\big) - \rho_0 g\boldsymbol{e_z},\end{aligned} \tag{2.75}$$

which used the fact that the background pressure does not vary horizontally. We insert the equation of state (equation (2.62)) into the gravitational source term

$$-(\rho_0 + \rho')g\boldsymbol{e_z} = -\rho_0 g\boldsymbol{e_z} - \frac{\rho_0}{K}p' g\boldsymbol{e_z}. \tag{2.76}$$

Finally, we insert equations (2.73) to (2.76) into equation (2.72) and arrive at

$$\rho_0 \frac{\partial \boldsymbol{v'}}{\partial t} + \nabla \cdot \big(\boldsymbol{I}p'\big) - \rho_0 g\boldsymbol{e_z} = -\rho_0 g\boldsymbol{e_z} - \frac{\rho_0}{K}p' g\boldsymbol{e_z}. \tag{2.77}$$

After simplifying, we get

$$\rho_0 \frac{\partial \boldsymbol{v'}}{\partial t} + \nabla \cdot \big(\boldsymbol{I}p'\big) = -\frac{\rho_0}{K}p' g\boldsymbol{e_z}. \tag{2.78}$$

Combining equations (2.70) and (2.78) leads to the acoustic wave equation in velocity-pressure formulation, given by

$$\begin{aligned}\frac{\partial p}{\partial t} + K\frac{\partial v_k}{\partial x_k} &= \rho g v_3, \\ \rho\frac{\partial v_i}{\partial t} + \frac{\partial p}{\partial x_i} &= -\delta_{i2}\frac{\rho g}{K}p.\end{aligned} \tag{2.79}$$

For simplicity, we dropped the prime that marked variables as perturbations and wrote ρ instead of ρ_0. Equation (2.79) is essentially equal to the elastic wave equations (equations (2.27) and (2.28)) with $\mu = 0$: With this, the shear stresses vanish. Thus, the stress tensor collapses into a scalar, which is the negative of the pressure (equation (2.18)). As in [3, 97], we neglect the source terms, leading to errors of size $\mathcal{O}(gH/c^2)$, where H is the depth of the ocean and

$$c = \sqrt{\frac{K}{\rho}} \tag{2.80}$$

is the acoustic wave speed. This type of error only has a negligible effect on the solution, assuming material parameters typical for Earth's oceans.

Again, equation (2.79) is a linear hyperbolic PDE of the same form as equation (2.29). Hence, in contrast to the Euler equations, the acoustic wave equation is non-conservative for spatially varying materials. We can thus rewrite equation (2.79) as

$$\frac{\partial \boldsymbol{q}^{\text{ac}}}{\partial t} + \boldsymbol{A}^{\text{ac}}\frac{\partial \boldsymbol{q}^{\text{ac}}}{\partial x} + \boldsymbol{B}^{\text{ac}}\frac{\partial \boldsymbol{q}^{\text{ac}}}{\partial y} + \boldsymbol{C}^{\text{ac}}\frac{\partial \boldsymbol{q}^{\text{ac}}}{\partial z} = 0, \tag{2.81}$$

with the vector of quantities $\boldsymbol{q}^{\text{ac}} = (p, v_1, v_2, v_3)^T$ and the flux matrices

$$\boldsymbol{A}^{\text{ac}} = \begin{pmatrix} 0 & K & 0 & 0 \\ \frac{1}{\rho} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{B}^{\text{ac}} = \begin{pmatrix} 0 & 0 & K & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{\rho} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \boldsymbol{C}^{\text{ac}} = \begin{pmatrix} 0 & 0 & 0 & K \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{\rho} & 0 & 0 & 0 \end{pmatrix}. \tag{2.82}$$

Their eigenvalues are given by $(-c, 0, c)$. Hence, the acoustic equation does not allow the propagation of shear waves in contrast to the elastic wave equation. Equation (2.81) has the same form as equation (2.29) but different flux matrices.

It is common to write the acoustic wave equation (equations (2.81) and (2.82)) in second-order formulation, which we can do either for the pressure or the velocity field [95]. To get the second-order form for the pressure, we have to differentiate the conservation of momentum in space, resulting in

$$\frac{\partial^2 v_i}{\partial t \, \partial x_i} = -\frac{1}{\rho} \frac{\partial^2 p}{\partial (x_i)^2} \tag{2.83}$$

for the ith component. Next, we differentiate the mass conservation

$$\frac{\partial^2 p}{\partial t^2} = -K \frac{\partial^2 v_k}{\partial t \, \partial x_k} \tag{2.84}$$

in time. By inserting equation (2.83) into equation (2.84), we arrive at

$$\frac{\partial^2 p}{\partial t^2} = -c^2 \sum_k \frac{\partial^2 p}{\partial (x_k)^2} = -c^2 \Delta p. \tag{2.85}$$

We have seen how to linearize the Euler equations around a hydrostatic background state. However, we have not yet discussed the linearization of the boundary conditions, which is a crucial step to including tsunami propagation in our model. Tsunamis are caused by gravity [131], but we decided to neglect the gravitational source term. Hence, we need to choose our boundary conditions carefully. In the acoustic region, we can prescribe either the pressure or the normal velocity [95]. We are interested in four cases: The first is the rigid boundary condition, which sets the normal velocity to zero. The second is the free surface condition (without gravity) that sets the pressure to zero. The third is the absorbing boundary condition, where waves can only exit but not enter the domain. The fourth is the gravitational free surface boundary condition. We want to enforce the moving free surface boundary condition equation (2.54), which is problematic: The surface level η is time-dependent, so we must impose this boundary condition on a moving mesh! We do not want to do this and instead follow the approach from [97]. Figure 2.4 shows a two-dimensional view of our three-dimensional setup and visualizes how we treat the boundary condition in our linear model.

It is important to note that the pressure at some point z is not, as we may naively think, equal to $p_0(z) + p'(x, y, z, t)$. The reason lies in the difference between the Lagrangian and Eulerian descriptions. Consider a particle at position $z = z_0 + u_3(x, y, z_0, t)$. The

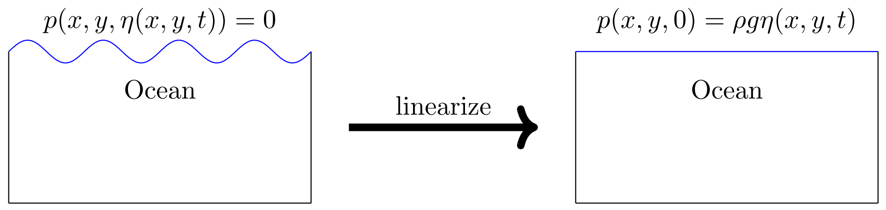


Figure 2.4.: Two-dimensional slice of our gravitational boundary condition. On the left, we see the non-linearized boundary condition equation (2.54). After linearizing, we enforce the boundary condition at the unperturbed sea surface $z = 0$. The blue lines in both figures indicate where we enforce the boundary condition.

particle has moved from its original position at z_0 due to the displacement. Then, we linearize the pressure at position z around the unperturbed position z_0, arriving at the first-order approximation

$$\begin{aligned}
p(x,y,z=z_0+u_3,t) &\approx p(x,y,z_0,t) + u_3(x,y,z_0,t)\left.\frac{\partial p}{\partial z}\right|_{z=z_0} \\
&= p_0(z_0) + p'(x,y,z_0,t) + u_3(x,y,z_0,t)\left(\left.\frac{\partial p_0}{\partial z}\right|_{z=z_0} + \left.\frac{\partial p'}{\partial z}\right|_{z=z_0}\right) \\
&\approx p_0(z_0) + p'(x,y,z_0,t) + u_3(x,y,0,t)\left.\frac{\partial p_0}{\partial z}\right|_{z=z_0} \\
&= p_a + p'(x,y,0,t) - \rho_0(z_0)\, g\,(z_0 + u_3(x,y,z_0,t)). \qquad (2.86)
\end{aligned}$$

Here, we neglected the higher-order term $u_3 \left.\frac{\partial p'}{\partial z}\right|_{z=z_0}$, which is usual for this kind of problem [97, 131].

But how does this help us with the free surface condition? Consider a particle on the free surface that moved from its unperturbed location at $z = 0$ to $z = \eta$. We linearize the dynamic pressure boundary condition (equation (2.54)) around this position using equation (2.86), leading us to the boundary condition for the total pressure of

$$\begin{aligned}
p_a &= p(x,y,z=0+\eta(x,y,t),t) \\
&\approx p(x,y,z_0,t) + \eta(x,y,t)\left.\frac{\partial p}{\partial z}\right|_{z=0} \qquad (2.87) \\
&\approx p_a - \rho_0(0) g \eta(x,y,t) + p'(x,y,0,t).
\end{aligned}$$

This linearization assumes that η is small, typical for tsunami propagation sufficiently far from the coast [131]. Canceling out the common term p_a, we arrive at the first-order approximation for the boundary condition on the pressure perturbation

$$p'(x,y,z,t) = \rho_0 g \eta(x,y,t) \qquad \text{at } z = 0. \qquad (2.88)$$

This is a remarkable trick: We have turned a computationally costly boundary condition on a moving mesh into a relatively cheap static pressure boundary condition! To close the boundary condition, we need to relate the surface displacement η to the velocities at the boundary. For this, following [97, 131], we linearize the kinematic boundary condition (equation (2.55)) to

$$\frac{\partial \eta}{\partial t} = v_3(x, y, z, t) \qquad \text{at } z = 0. \tag{2.89}$$

Note that neglecting gravitational effects at the boundary (i.e., setting $g = 0$) results in the standard free surface boundary condition.

When we specify a displacement b on the seafloor, following [3], we require the linearized version of the kinematic boundary condition on the seafloor (equation (2.57)) given by

$$\frac{\partial b}{\partial t} = v_1 \frac{\partial H}{\partial x} + v_2 \frac{\partial H}{\partial y} + v_3 \qquad \text{at } z = -H(x, y). \tag{2.90}$$

2.3. Fully Coupled Model

We discussed the elastic and acoustic wave equations in the previous sections. This section explains how we can combine both into a fully coupled model.

To do this, we use the fact that, as mentioned in section 2.2.2, the acoustic wave equations are a special case for the elastic wave equations for $\mu = 0$. Hence, we set the stress tensor for the acoustic part to

$$\begin{aligned} \sigma_{11} = \sigma_{22} = \sigma_{33} &= -p, \\ \sigma_{12} = \sigma_{23} = \sigma_{13} &= 0. \end{aligned} \tag{2.91}$$

The definition of the stress tensor follows directly from equation (2.14), the equivalence to the pressure from equation (2.18). With this, both equations can use the same vector of quantities. In the following, we will call this set of quantities the embedded set. The PDEs then only differ in the choice of flux matrices and boundary conditions. We can obtain the acoustic ones by setting $\mu = 0$ in the elastic matrices (equation (2.30)). For the rest of this thesis, we will use both sets of variables in our descriptions, as each is advantageous for some situations. However, we always use the extended variable set in our implementation. This introduces significant overhead but allows us to use the same computational kernels for both.

2.3.1. Interface Conditions

We have seen how to model wave propagation in both elastic and acoustic media. However, we have not yet discussed how we handle the material discontinuities and the interface between solids and fluids. We assume that our material is defined as a piece-wise constant function. This section discusses how to deal with the discontinuities and couple elastic and acoustic media. We do this by introducing conditions that have to hold at material interfaces. Our discussion follows [144, 164, 175], which describe the interface conditions for elastic-elastic, acoustic-acoustic, acoustic-elastic, and elastic-acoustic interfaces.

Along an interface, parameterized by its outer normal $\boldsymbol{n}$, we denote values on the left with an L superscript and values on the right with an R. For an elastic-elastic interface, the velocity $\boldsymbol{v}$ and traction $\boldsymbol{\sigma} \cdot \boldsymbol{n}$ must be continuous

$$\begin{aligned} \boldsymbol{v^L} &= \boldsymbol{v^R}, \\ \boldsymbol{\sigma^L} \cdot \boldsymbol{n} &= \boldsymbol{\sigma^R} \cdot \boldsymbol{n}. \end{aligned} \tag{2.92}$$

At an acoustic-acoustic interface, we must ensure only the continuity of the normal velocity $\boldsymbol{v} \cdot \boldsymbol{n}$ and of the pressure

$$\begin{aligned} \boldsymbol{v^L} \cdot \boldsymbol{n} &= \boldsymbol{v^R} \cdot \boldsymbol{n}, \\ p^L &= p^R. \end{aligned} \tag{2.93}$$

Finally, the normal velocity and traction are continuous for an elastic-acoustic or acoustic-elastic interface. Using the embedding (equation (2.91)) for these coupled interfaces is beneficial. Hence, we have

$$\begin{aligned} \boldsymbol{v^L} \cdot \boldsymbol{n} &= \boldsymbol{v^R} \cdot \boldsymbol{n}, \\ \boldsymbol{\sigma^L} \cdot \boldsymbol{n} &= \boldsymbol{\sigma^R} \cdot \boldsymbol{n} = \begin{pmatrix} -p^R & & \\ & -p^R & \\ & & -p^R \end{pmatrix} \cdot \boldsymbol{n}, \end{aligned} \tag{2.94}$$

for an elastic-acoustic interface and

$$\begin{aligned} \boldsymbol{v^L} \cdot \boldsymbol{n} &= \boldsymbol{v^R} \cdot \boldsymbol{n}, \\ \begin{pmatrix} -p^L & & \\ & -p^L & \\ & & -p^L \end{pmatrix} \cdot \boldsymbol{n} &= \boldsymbol{\sigma^L} \cdot \boldsymbol{n} = \boldsymbol{\sigma^R} \cdot \boldsymbol{n}, \end{aligned} \tag{2.95}$$

for an elastic-acoustic interface.

Observe that this means that the traction depends on the shear stress in the elastic part.

2.3.2. Energy

Finally, we can describe the energy of our complete system. The definition of energy follows [164, 175] for the elastic and acoustic region and [97] for the gravitational boundary condition. We define the energy as

$$E(\boldsymbol{q}) = \underbrace{\int_\Omega \frac{1}{2} \rho v_i v_i \, \mathrm{d}\boldsymbol{x}}_{\text{kinetic energy}} + \underbrace{\int_E \frac{1}{2} \sigma_{ij} \varepsilon_{ij} \, \mathrm{d}\boldsymbol{x}}_{\text{strain energy}} + \underbrace{\int_A \frac{1}{2K} p^2 \, \mathrm{d}\boldsymbol{x}}_{\text{acoustic energy}} + \underbrace{\int_{S_O} \frac{1}{2} \rho g \eta^2 \, \mathrm{d}S}_{\text{gravitational energy}}, \tag{2.96}$$

where the region $\Omega = E \cup A$ is divided into the elastic $E \subseteq \Omega$ and the acoustic region $A = \Omega \setminus E$. The acoustic energy is a special case of the strain energy, which follows from applying equation (2.18). The sea surface is denoted by S_O. It is the only contributor to the gravitational energy, which corresponds to the energy of the tsunami [131]. For more details, we refer the interested reader to [97, 131], which contains a derivation of this energy. The kinetic energy has contributions from both elastic and acoustic parts.

It is important to note that we only discuss energy perturbations in this thesis. Let $\boldsymbol{q} = \boldsymbol{q_0} + \boldsymbol{q'}$ be the state of the simulation, which is split into the background state $\boldsymbol{q_0}$ and the perturbation $\boldsymbol{q'}$, which our PDEs model. Then, the total energy $E(\boldsymbol{q})$ can be bounded by

$$E(\boldsymbol{q}) \leq E(\boldsymbol{q_0}) + E(\boldsymbol{q'}), \tag{2.97}$$

which is the triangle inequality.[6] However, this only affects the strain and acoustic energies, as the background velocity and sea surface height are assumed to be zero.

2.3.3. Discussion

In this chapter, we derived and discussed the partial differential equations that form the basis for the model we use throughout this thesis. We obtained a fully coupled model covering the entire dynamics from earthquake rupture to wave propagation in both elastic and acoustic media and tsunami propagation at the free surface. Earthquake rupture (section 2.1.2) can be included by kinematic point sources or dynamic rupture. The wave propagation uses the PDE (equation (2.29)) with flux matrices (equation (2.30)) for the elastic part. The acoustic region uses the flux matrices (equation (2.82)), which are implemented as elastic flux matrices through the embedding given by equation (2.91). Finally, equations (2.88) and (2.89) include tsunami propagation efficiently as a linearized boundary condition. To our knowledge, our model, initially published in [79], is the first model capable of simulating large-scale, three-dimensional, fully coupled earthquake-tsunami simulations.

[103] introduced another way of modeling fully coupled simulations. They derived a global source term that includes tsunami propagation in the wavefield. Here, we summarize this method and compare it with our strategy. The equations in this section differ slightly from those in [103], which uses a different sign convention. Briefly, they decompose the normal stresses

$$\sigma_{ii} = \sigma_{ii}^D + p, \tag{2.98}$$

into a dynamic part and a pressure that is in hydrostatic equilibrium (equation (2.48)). The sea surface height is

$$\eta(x, y, t) = \int_0^t v_3(x, y, z = 0, \tau) \,\mathrm{d}\tau. \tag{2.99}$$

The pressure at some position z is given by integrating equation (2.48)

$$p(x, y, z, t) = p_a - \int_{\eta(x,y,t)}^{z} \rho(x, y, z') g \,\mathrm{d}z'. \tag{2.100}$$

[6] We can interpret equation (2.96) as a norm for the extended variable set obtained by concatenating $\boldsymbol{q}$ with η.

Assuming that variations in ρ are negligible, we have for the x-derivative that

$$\begin{aligned}\frac{\partial p}{\partial x} &= \rho\left(x, \eta(x,y,t), z\right) g \frac{\partial \eta}{\partial x}(x,y,t) - g \int_{\eta(x,y,t}^{z} \frac{\partial \rho}{\partial x}(x,y,z)\,\mathrm{d}z' \\ &\approx \underbrace{\rho\left(x, \eta(x,y,t), z\right)}_{=\rho_w} g \frac{\partial \eta}{\partial x}(x,y,t).\end{aligned} \tag{2.101}$$

Splitting the stress tensor in the equations of motion (equation (2.28)) with equation (2.98) and evaluating the pressure derivatives (equation (2.101)) leads us to

$$\begin{aligned}\rho\frac{\partial v_1}{\partial t} - \frac{\partial \sigma_{11}^D}{\partial x} - \frac{\partial \sigma_{12}}{\partial y} - \frac{\partial \sigma_{13}}{\partial z} &= \rho_w g \frac{\partial \eta}{\partial x}, \\ \rho\frac{\partial v_2}{\partial t} - \frac{\partial \sigma_{12}}{\partial x} - \frac{\partial \sigma_{22}^D}{\partial y} - \frac{\partial \sigma_{23}}{\partial z} &= \rho_w g \frac{\partial \eta}{\partial y}, \\ \rho\frac{\partial v_3}{\partial t} - \frac{\partial \sigma_{13}}{\partial x} - \frac{\partial \sigma_{23}}{\partial y} - \frac{\partial \sigma_{33}^D}{\partial z} &= 0.\end{aligned} \tag{2.102}$$

There is no source term for v_3 because the z-derivative of the hydrostatic pressure p cancels out the gravitational source term of $-\rho g$.

Many of the concepts used to derive this model are similar to ours. Both approaches lead to a fully coupled model that allows the simulation of elastic and acoustic wave propagation in the ocean together with tsunami propagation. But how they include gravitational effects differs. The source terms $\frac{\partial \eta}{\partial x}$ and $\frac{\partial \eta}{\partial y}$ depend only on the derivative of the sea surface height, which is inexpensive to compute. However, because this derivative is used in the entire domain, it must be communicated globally for every time step, which is—especially in a distributed memory environment—prohibitively expensive. In contrast, our model's computational costs are constrained to a two-dimensional boundary condition. Hence, it requires no additional communication, which renders it computationally far more economical.

Chapter 3.

Riemann Problems & Boundary Conditions

In this chapter, we describe how we can solve Riemann problems for the PDEs introduced in chapter 2. A Riemann problem is composed of a hyperbolic PDE with a piece-wise discontinuous initial condition [160]. Riemann solvers provide a solution for this problem. They are fundamental for analyzing and interpreting hyperbolic PDEs because they provide insight into the wave structure of our model. Furthermore, we will use them as a critical component of our numerical scheme.

Our introduction to Riemann problems follows [61, 95]. We are interested in solving the Riemann problem along an interface. The interface is parametrized with a local coordinate system, given by the orthonormal basis

$$\boldsymbol{n} = \begin{pmatrix} n_x & n_y & n_z \end{pmatrix}^T, \quad \boldsymbol{s} = \begin{pmatrix} s_x & s_y & s_z \end{pmatrix}^T, \quad \boldsymbol{t} = \begin{pmatrix} t_x & t_y & t_z \end{pmatrix}^T. \tag{3.1}$$

In this set of vectors, $\boldsymbol{n}$ is the unit vector pointing outward from a face, and $\boldsymbol{s}$ and $\boldsymbol{t}$ are tangential vectors. Figure 3.1 shows a two-dimensional visualization of the setup. We define the scalar

$$d(\boldsymbol{x}, \boldsymbol{n}) = \boldsymbol{x} \cdot \boldsymbol{n}, \tag{3.2}$$

which is the signed distance of a point with coordinates $\boldsymbol{x}$ along the normal to the interface. The flux matrix in normal direction is given by

$$\hat{\boldsymbol{A}}(\boldsymbol{x}, \boldsymbol{n}) = n_x \boldsymbol{A}(\boldsymbol{x}) + n_y \boldsymbol{B}(\boldsymbol{x}) + n_z \boldsymbol{C}(\boldsymbol{x}). \tag{2.31 revisited}$$

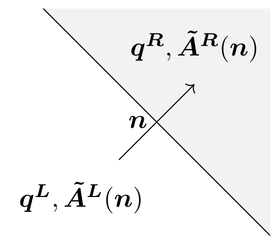

$\boldsymbol{q^R}, \tilde{\boldsymbol{A}}^{\boldsymbol{R}}(\boldsymbol{n})$
$\boldsymbol{n}$
$\boldsymbol{q^L}, \tilde{\boldsymbol{A}}^{\boldsymbol{L}}(\boldsymbol{n})$

Figure 3.1.: Two-dimensional visualization of the setup of a Riemann problem. The interface is parametrized by its outer normal vector $\boldsymbol{n}$. In the unshaded area, the state is given by $\boldsymbol{q^L}$, and the wave propagation is governed by the flux matrix in normal direction $\hat{\boldsymbol{A}}^L$. In the gray area, the state and the flux matrix are given by $\boldsymbol{q^R}$ and $\tilde{\boldsymbol{A}}^R$.

Definition 1. *For an arbitrary hyperbolic PDE, the problem of computing the solution of*

$$\begin{aligned} \frac{\partial \boldsymbol{q}}{\partial t} + \tilde{\boldsymbol{A}}(d)\frac{\partial \boldsymbol{q}}{\partial d} &= 0, \\ \boldsymbol{q}(d, t=0) &= \begin{cases} \boldsymbol{q^L} & \textit{if } d < 0, \\ \boldsymbol{q^R} & \textit{otherwise}, \end{cases} \\ \tilde{\boldsymbol{A}}(d) &= \begin{cases} \tilde{\boldsymbol{A}}^{\boldsymbol{L}} & \textit{if } d < 0, \\ \tilde{\boldsymbol{A}}^{\boldsymbol{R}} & \textit{otherwise}, \end{cases} \end{aligned} \tag{3.3}$$

at $d = 0$ is called the Riemann problem. The initial condition and the material may be discontinuous along an interface. We assume a piece-wise constant initial condition and material. Because the flux matrix $\tilde{\boldsymbol{A}}$ depends on the material, it can also be discontinuous.

We are interested in solving the problem given by definition 1 at $t \to 0$, i.e., at a moment in time directly after the initial discontinuity. This chapter explains how we can achieve this. The resulting Riemann solver itself is not novel as it has been published before, for example, in [164, 175]. However, we will take a more detailed look into how this solver is constructed, focusing on the elastic-acoustic coupling. Furthermore, we investigate all boundary conditions required for our fully coupled model, which is not done in the literature.

We begin by summarizing the rotational invariance properties in section 3.1, which we use to reduce the three-dimensional problem given by definition 1 into an equivalent one-dimensional form. Next, we introduce theoretical aspects: The technique of characteristic variables (section 3.2) allows us to transform our PDEs from the set of variables $(\boldsymbol{v}, \boldsymbol{\sigma})$ into a new set of variables that decompose the solution into waves. Section 3.3 introduces the Rankine-Hugoniot jump conditions, which reduce the Riemann problem into an eigenproblem. Both techniques only work directly for homogeneous material. We, however, are interested in the more general variable coefficient case: The flux matrices $\boldsymbol{A}(\boldsymbol{x}), \boldsymbol{B}(\boldsymbol{x})$ and $\boldsymbol{C}(\boldsymbol{x})$ may vary across element interfaces. Section 3.4 shows how we can combine physical interface conditions, initially introduced in section 2.3.1, with the techniques mentioned above. This not only provides a solution for the acoustic-acoustic Riemann problem but also gives us a strategy that we can use as a blueprint to devise the solution for the elastic-elastic (section 3.5), and elastic-acoustic (section 3.6) problems. We additionally describe how we can include boundary conditions for elastic and acoustic materials. Finally, section 3.7 explains how we can compute the solution of our Riemann problems efficiently.

3.1. Rotational Invariance

Definition 1 depends on the orientation of the interface, making it harder to find a general solution. We can use the rotational invariance of the elastic wave equation to simplify the problem [35].

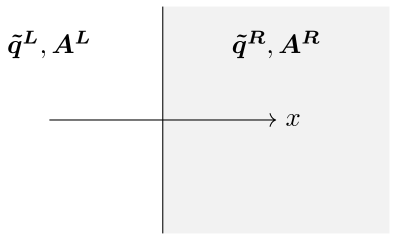


Figure 3.2.: The one-dimensional Riemann problem, obtained after applying lemma 2. Similar to figure 3.1, we have an interface parametrized by an outer normal vector $\boldsymbol{n}$. However, this time, we are considering a one-dimensional problem. In the unshaded area, the state is $\tilde{\boldsymbol{q}}^{\boldsymbol{L}}$ and the one-dimensional flux matrix governs the wave propagation in x-direction $\boldsymbol{A}^L$. In the gray area, the state and flux matrix are given by $\tilde{\boldsymbol{q}}^{\boldsymbol{R}}$ and $\boldsymbol{A}^R$. Now, the meaning of the indices L and R becomes clear: They mean left or right from the interface.

Lemma 1. *The plane wave operator (equation* (2.31)*) is rotationally invariant for the elastic wave equation. Let $\boldsymbol{\mathcal{T}}$ be a (specifically defined) rotation matrix, which depends on the orientation of the face and thus on the face-aligned basis vectors $\boldsymbol{n}, \boldsymbol{s}$ and $\boldsymbol{t}$. Then,*

$$\tilde{\boldsymbol{A}} = n_x\boldsymbol{A} + n_y\boldsymbol{B} + n_z\boldsymbol{C} = \boldsymbol{\mathcal{T}}\boldsymbol{A}\boldsymbol{\mathcal{T}}^{-1}. \tag{3.4}$$

The rotational invariance of the acoustic wave equation follows directly from our embedding (equation (2.91)) and lemma 1. For the sake of brevity, we do not include a proof of lemma 1 here but we refer the interested reader to [164, Lemma 1] for an explicit proof of the invariance of the elastic wave equation and to [175] for a discussion of the rotational invariance for the elastic and acoustic wave equations.

The rotation matrix is defined in [35, 164]. We derive the rotation matrix $\mathcal{T}$ here as neither source explicitly constructs it. We define the three-dimensional rotation matrix

$$T = \begin{pmatrix} | & | & | \\ \boldsymbol{n} & \boldsymbol{s} & \boldsymbol{t} \\ | & | & | \end{pmatrix}. \tag{3.5}$$

and use $\tilde{\boldsymbol{v}}$ and $\tilde{\boldsymbol{\sigma}}$ for the rotated velocities and stresses. As the velocity is a column vector, it transforms as

$$\boldsymbol{v} = \boldsymbol{T}\tilde{\boldsymbol{v}}. \tag{3.6}$$

The stress tensor transforms as

$$\boldsymbol{\sigma} = \boldsymbol{T}\tilde{\boldsymbol{\sigma}}\boldsymbol{T}^T, \tag{3.7}$$

which is a similarity transform [141]. Now, we apply the well-known identity

$$(\boldsymbol{B} \otimes \boldsymbol{A}) \operatorname{vec}(\boldsymbol{X}) = \operatorname{vec}\left(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}^T\right), \tag{3.8}$$

which allows us to write the change of basis as a product with a single matrix [120, Sec. 10.2]. In this equation, $\operatorname{vec}(\boldsymbol{C})$ denotes the vectorization of $\boldsymbol{C}$, i.e., it is the operator that returns a vector with the entries given by stacking the columns of $\boldsymbol{C}$. We can then use it to rewrite equation (3.7) as

$$\operatorname{vec}(\boldsymbol{\sigma}) = (\boldsymbol{T} \otimes \boldsymbol{T}) \operatorname{vec}(\boldsymbol{\sigma}) = \operatorname{vec}\left(\boldsymbol{T}\tilde{\boldsymbol{\sigma}}\boldsymbol{T}^T\right), \tag{3.9}$$

which describes the linear transformation as a matrix. As the Kronecker product of two orthogonal matrices results in an orthogonal matrix, the rotation operator for the entries of the stress tensor

$$\boldsymbol{T} \otimes \boldsymbol{T} = \left(\begin{array}{ccc|ccc|ccc}
n_x^2 & n_x s_x & n_x t_x & n_x s_x & s_x^2 & s_x t_x & n_x t_x & s_x t_x & t_x^2 \\
n_x n_y & n_x s_y & n_x t_y & n_y s_x & s_x s_y & s_x t_y & n_y t_x & s_y t_x & t_x t_y \\
n_x n_z & n_x s_z & n_x t_z & n_z s_x & s_x s_z & s_x t_z & n_z t_x & s_z t_x & t_x t_z \\
\hline
n_x n_y & n_y s_x & n_y t_x & n_x s_y & s_x s_y & s_y t_x & n_x t_y & s_x t_y & t_x t_y \\
n_y^2 & n_y s_y & n_y t_y & n_y s_y & s_y^2 & s_y t_y & n_y t_y & s_y t_y & t_y^2 \\
n_y n_z & n_y s_z & n_y t_z & n_z s_y & s_y s_z & s_y t_z & n_z t_y & s_z t_y & t_y t_z \\
\hline
n_x n_z & n_z s_x & n_z t_x & n_x s_z & s_x s_z & s_z t_x & n_x t_z & s_x t_z & t_x t_z \\
n_y n_z & n_z s_y & n_z t_y & n_y s_z & s_y s_z & s_z t_y & n_y t_z & s_y t_z & t_y t_z \\
n_z^2 & n_z s_z & n_z t_z & n_z s_z & s_z^2 & s_z t_z & n_z t_z & s_z t_z & t_z^2
\end{array}\right) \tag{3.10}$$

is also orthogonal. However, our coefficient vector $\boldsymbol{q}$ only contains the six independent entries of $\boldsymbol{\sigma}$. Hence, we need to combine rows and columns of equation (3.10), leading to the rotation matrix

$$\boldsymbol{T}_{\boldsymbol{\sigma}} = \begin{pmatrix}
n_x^2 & s_x^2 & t_x^2 & 2n_x s_x & 2s_x t_x & 2n_x t_x \\
n_y^2 & s_y^2 & t_y^2 & 2n_y s_y & 2s_y t_y & 2n_y t_y \\
n_z^2 & s_z^2 & t_z^2 & 2n_z s_z & 2s_z t_z & 2n_z t_z \\
n_x n_y & s_x s_y & t_x t_y & n_x s_y + n_y s_x & s_x t_y + s_y t_x & n_x t_y + n_y t_x \\
n_y n_z & s_y s_z & t_y t_z & n_y s_z + n_z s_y & s_y t_z + s_z t_y & n_y t_z + n_z t_y \\
n_x n_z & s_x s_z & t_x t_z & n_x s_z + n_z s_x & s_x t_z + s_z t_x & n_x t_z + n_z t_x
\end{pmatrix}. \tag{3.11}$$

Finally, we can define the rotated coefficient vector $\tilde{\boldsymbol{q}}$ as

$$\boldsymbol{q} = \underbrace{\begin{pmatrix} \boldsymbol{T}_{\boldsymbol{\sigma}} & 0 \\ 0 & \boldsymbol{T} \end{pmatrix}}_{=\boldsymbol{\mathcal{T}}} \tilde{\boldsymbol{q}}, \tag{3.12}$$

where the combined rotation matrix $\boldsymbol{\mathcal{T}}$ agrees with the rotation matrix presented in [35, 164]. For the acoustic wave equation, this reduces to

$$\boldsymbol{q}^{\text{ac}} = \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & \boldsymbol{T} \end{pmatrix}}_{=\boldsymbol{\mathcal{T}}^{\text{ac}}} \tilde{\boldsymbol{q}}^{\text{ac}}, \tag{3.13}$$

as the pressure is scalar and thus rotation-invariant. Due to their block-orthogonal structure, $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{T}}^{\text{ac}}$ are orthogonal.

Lemma 2. *After applying the rotational invariance property (lemma 1), the three-dimensional Riemann problem (definition 1) reduces to the one-dimensional Riemann problem*

$$\begin{aligned}
\frac{\partial \tilde{\boldsymbol{q}}}{\partial t} + \boldsymbol{A}\frac{\partial \tilde{\boldsymbol{q}}}{\partial x} &= 0,\\
\tilde{\boldsymbol{q}}(x, t=0) &= \begin{cases} \tilde{\boldsymbol{q}}^{\boldsymbol{L}} & \text{if } x < 0,\\ \tilde{\boldsymbol{q}}^{\boldsymbol{R}} & \text{otherwise,} \end{cases}\\
\boldsymbol{A}(x) &= \begin{cases} \boldsymbol{A}^{\boldsymbol{L}} & \text{if } x < 0,\\ \boldsymbol{A}^{\boldsymbol{R}} & \text{otherwise.} \end{cases}
\end{aligned} \tag{3.14}$$

Proof. This derivation follows directly from applying lemma 1. Inserting equation (3.4) into the Riemann problem (equation (3.3)), leads us to

$$\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{\mathcal{T}}\boldsymbol{A}\boldsymbol{\mathcal{T}}^{-1}\frac{\partial \boldsymbol{q}}{\partial d} = 0. \tag{3.15}$$

Next, we left multiply with $\boldsymbol{\mathcal{T}}^{-1}$, leading us to

$$\boldsymbol{\mathcal{T}}^{-1}\frac{\partial \boldsymbol{q}}{\partial t} + \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{\mathcal{T}}\boldsymbol{A}\boldsymbol{\mathcal{T}}^{-1}\frac{\partial \boldsymbol{q}}{\partial d} = 0. \tag{3.16}$$

Inserting the rotated quantities (equation (3.12)) and simplifying leads us directly to equation (3.14). □

Lemma 2 allows us to split the solution of the Riemann problem into a rotation, which depends on the orientation of the interface, and the solution of a one-dimensional Riemann problem, which, crucially, does not depend on the orientation. The resulting one-dimensional Riemann problem is depicted by figure 3.2. In the following, we thus only consider solving one-dimensional Riemann problems. To simplify the notation, we will always assume that the quantities have already been rotated. Hence, we will drop the tilde.

3.2. Characteristic Variables

In this section, we explain how we can cast our PDE into a basis in which the wave structure becomes directly apparent. This is a standard technique called characteristic variables. Our discussion follows [95]. This strategy gives us a direct way of directly solving Riemann problems for homogeneous material. Furthermore, it shines a light on the difficulties arising when considering variable coefficient equations

In the following, we assume that we have a one-dimensional hyperbolic system of PDEs with a flux matrix $\boldsymbol{A}$. By definition of (strong) hyperbolicity, we can diagonalize $\boldsymbol{A}$ and thus write it as

$$\boldsymbol{A} = \boldsymbol{R}\boldsymbol{\Lambda}\boldsymbol{R}^{-1}, \tag{3.17}$$

where $\boldsymbol{R}$ has the eigenvectors of $\boldsymbol{A}$ as columns and $\boldsymbol{\Lambda}$ has the corresponding eigenvalues on its diagonal:

$$\begin{aligned} \boldsymbol{R} &= \begin{pmatrix} | & | & | & | \\ \boldsymbol{r_1} & \boldsymbol{r_2} & \dots & \boldsymbol{r_m} \\ | & | & | & | \end{pmatrix}, \\ \boldsymbol{\Lambda} &= \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_m \end{pmatrix}. \end{aligned} \tag{3.18}$$

Both $\boldsymbol{\Lambda}$ and $\boldsymbol{R}$ depend on the material and can thus in general vary in space. For now, we assume that we have a homogeneous material.

Using equation (3.18), we define the characteristic variables as

$$\boldsymbol{w} = \boldsymbol{R}^{-1}\boldsymbol{q}. \tag{3.19}$$

We can use this to turn our hyperbolic system into a system in terms of the characteristic variables. We start by inserting equations (3.17) and (3.19) into our one-dimensional PDE (equation (3.14)), which results in

$$\boldsymbol{R}\frac{\partial \boldsymbol{w}}{\partial t} + \boldsymbol{R}\boldsymbol{\Lambda}\boldsymbol{R}^{-1}\boldsymbol{R}\frac{\partial \boldsymbol{w}}{\partial x} = 0. \tag{3.20}$$

Next, we left-multiply by $\boldsymbol{R}^{-1}$ and simplify, arriving at a system of uncoupled advection equations

$$\frac{\partial \boldsymbol{w}}{\partial t} + \boldsymbol{\Lambda}\frac{\partial \boldsymbol{w}}{\partial x} = 0, \tag{3.21}$$

which describes the time evolution of the characteristic variables. In other words, equation (3.21) describes the propagation of the waves directly.

Let $w_i^0(x)$ denote the initial condition for the ith component of $\boldsymbol{w}$. Then, equation (3.21) is solved exactly by

$$w_i(x,t) = w_i^0(x - \lambda_i t), \tag{3.22}$$

which is the direct solution of the advection equation [95]. From this, a solution strategy for the rotated Riemann problem (equation (3.14)), assuming homogeneous material, directly follows: First, convert the initial condition to the characteristic variables using equation (3.19). Second, write the solution in terms of the characteristic variables using equation (3.22). Third, convert back to the original set of variables using the inverse mapping (equation (3.19)).

However, this approach does not work for variable coefficient equations. Following [95, Sec. 9.8], we see that in this case $\boldsymbol{R}$ varies in space and thus, by the product rule

$$\begin{aligned} \boldsymbol{R}^{-1}(x)\frac{\partial \boldsymbol{q}}{\partial x} &= \frac{\partial \boldsymbol{R}^{-1}(x)\boldsymbol{q}}{\partial x} - \frac{\partial \boldsymbol{R}^{-1}}{\partial x}(x)\boldsymbol{q} \\ &= \frac{\partial \boldsymbol{w}}{\partial x} - \frac{\partial \boldsymbol{R}^{-1}}{\partial x}(x)\boldsymbol{q}. \end{aligned} \tag{3.23}$$

This happens because $\frac{\partial \boldsymbol{R}^{-1}}{\partial x}(x)$ is no longer zero. Now, when we follow the same derivation as before and apply equation (3.23), we arrive at

$$\frac{\partial}{\partial t}\boldsymbol{w} + \boldsymbol{\Lambda}\frac{\partial \boldsymbol{w}}{\partial x} = \boldsymbol{\Lambda}\frac{\partial \boldsymbol{R}^{-1}}{\partial x}(x)\boldsymbol{R}(x)\boldsymbol{w}. \tag{3.24}$$

The structure of the PDE is similar to equation (3.21), as we still have advection equations for the characteristic variables. However, they are now coupled by a source term. Hence, equation (3.23) does not admit a solution like equation (3.22) in the general case.[1] Thus, we need to follow a different solution strategy.

3.3. Rankine-Hugoniot Jump Condition

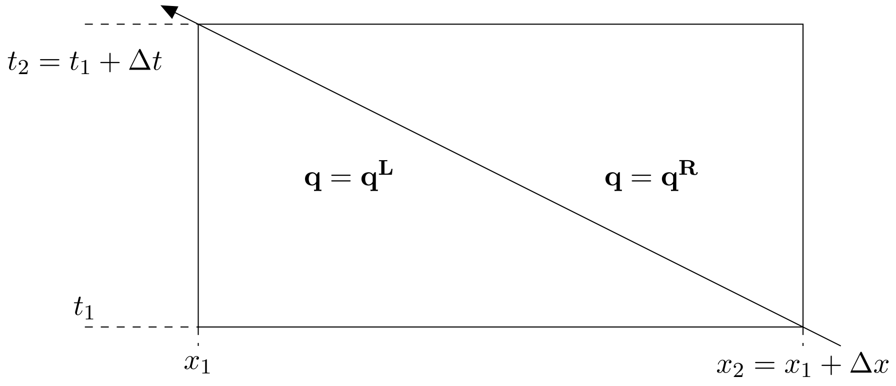


Figure 3.3.: The control volume $[x_1, x_2] \times [t_1, t_2]$ is divided by a wave moving with a speed $\lambda < 0$. The x-axis describes space and the y-axis time. The wave divides the rectangle into two halves. We assume that the solution inside each half is constant. Figure recreated from [95].

In this section, we derive and explain the Rankine-Hugoniot jump conditions, which we use to inspect shock waves resulting, in our case, from a discontinuous initial condition. They are powerful as they allow us to separate a jump into waves, and hence, they directly reveal the wave structure. The derivation of them is simple and follows directly from a conservation statement.

First, following [160, Sec. 2.4], we use the integral form of our one-dimensional PDE (equation (3.14)) in the control volume $[x_1, x_2] \times [t_1, t_2]$.

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{x_1}^{x_2} \boldsymbol{q}(x,t)\,\mathrm{d}x = \boldsymbol{A}\boldsymbol{q}(x_1,t) - \boldsymbol{A}\boldsymbol{q}(x_2,t). \tag{3.25}$$

[1]An interesting special case results from the fact that the source term can become uncoupled. This is the case for homogeneous material and, for example, constant-impedance acoustic material [95]. In this case, the eigenvectors are identical, even though the material varies.

We derive this by integrating the differential form, given by equation (3.14), from x_1 to x_2 and assuming that $\boldsymbol{A}$ is constant in our control volume. This integral form has the advantage that it does not require $\boldsymbol{q}$ to be smooth in contrast to the differential form. We integrate equation (3.25) from $t_1 \leq t_2$ to t_2

$$\int_{t_1}^{t_2} \frac{\mathrm{d}}{\mathrm{d}t} \left(\int_{x_1}^{x_2} \boldsymbol{q}(x,t)\,\mathrm{d}x \right) \mathrm{d}t = \int_{t_1}^{t_2} \boldsymbol{A}\boldsymbol{q}(x_1,t)\,\mathrm{d}t - \int_{t_1}^{t_2} \boldsymbol{A}\boldsymbol{q}(x_2,t)\,\mathrm{d}t. \tag{3.26}$$

After applying the fundamental theorem of calculus to the left side, we get

$$\begin{aligned} \int_{t_1}^{t_2} \left(\frac{\mathrm{d}}{\mathrm{d}t} \int_{x_1}^{x_2} \boldsymbol{q}(x,t)\,\mathrm{d}x \right) \mathrm{d}t &= \\ \int_{x_1}^{x_2} \boldsymbol{q}(x,t_2)\,\mathrm{d}x - \int_{x_1}^{x_2} \boldsymbol{q}(x,t_1)\,\mathrm{d}x&. \end{aligned} \tag{3.27}$$

Inserting this into equation (3.26), leads us to the conservation statement

$$\int_{x_1}^{x_2} \boldsymbol{q}(x,t_2)\,\mathrm{d}x - \int_{x_1}^{x_2} \boldsymbol{q}(x,t_1)\,\mathrm{d}x = \int_{t_1}^{t_2} \boldsymbol{A}\boldsymbol{q}(x_1,t)\,\mathrm{d}t - \int_{t_1}^{t_2} \boldsymbol{A}\boldsymbol{q}(x_2,t)\,\mathrm{d}t. \tag{3.28}$$

Next, we use this form of our PDE to investigate the behavior of our PDE in the situation of a shock. The derivation follows [95, Sec. 11.9]. Figure 3.3 depicts the setup. We consider a wave, moving with a speed of λ, that divides our control rectangle into two triangles, inside which the solution is approximately constant. In our case of a Riemann problem for a linear PDE, the shock speed does not depend on time.

We consider an infinitesimal control region $[x_1, x_2] \times [t_1, t_2]$ where $x_2 = x_1 + \Delta x$ and $t_2 = t_1 + \Delta t$. Here, $\Delta x > 0$ is a space increment, and $\Delta t > 0$ is a small ($\Delta t \ll 1$) time increment. Without loss of generality, we assume that the wave moves leftward, i.e., $\lambda < 0$. We call the state in the left triangle $\boldsymbol{q^L}$ and in the right triangle $\boldsymbol{q^R}$. Because $\boldsymbol{q}$ is constant in each triangle, equation (3.28) simplifies to

$$\Delta x\, \boldsymbol{q^R} - \Delta x\, \boldsymbol{q^L} = \Delta t\, \boldsymbol{A}\boldsymbol{q^L} - \Delta t\, \boldsymbol{q^R} + \mathcal{O}(\Delta t^2), \tag{3.29}$$

where the term $\mathcal{O}(\Delta t^2)$ takes the variation of $\boldsymbol{q}$ with respect to time into account. Dividing by Δt, results in the relation

$$\frac{\Delta x}{\Delta t}\boldsymbol{q^R} - \frac{\Delta x}{\Delta t}\boldsymbol{q^L} = \boldsymbol{A}\boldsymbol{q^L} - \boldsymbol{A}\boldsymbol{q^R} + \mathcal{O}(\Delta t^2) \tag{3.30}$$

If the shock propagates with speed λ, $\Delta x = -\lambda\,\Delta t$. Inserting this and taking the limit $\Delta t \to 0$, we arrive at the Rankine-Hugoniot conditions

$$\lambda(\boldsymbol{q^R} - \boldsymbol{q^L}) = \boldsymbol{A}(\boldsymbol{q^R} - \boldsymbol{q^L}), \tag{3.31}$$

which state that the shock speed λ is an eigenvalue corresponding to the eigenvector $(\boldsymbol{q^R} - \boldsymbol{q^L})$. This is often written in the short form

$$\lambda[\![\boldsymbol{q}]\!] = \boldsymbol{A}[\![\boldsymbol{q}]\!], \tag{3.32}$$

with $[\![\boldsymbol{q}]\!] = \boldsymbol{q^R} - \boldsymbol{q^L}$. In other words, they decompose the jump into waves.

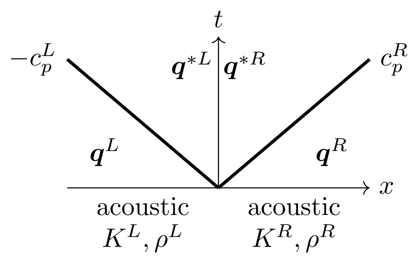


Figure 3.4.: This figure shows the acoustic-acoustic Riemann problem. The black lines correspond to characteristic waves that separate the constant states. We use the convention that the states left and right from the interface are $\boldsymbol{q^L}$ and $\boldsymbol{q^R}$. The states in the middle, also called star states, are the solution of the Riemann problem. They are called $\boldsymbol{q^{*L}}$ and $\boldsymbol{q^{*R}}$. They are not separated by a wave but rather by the discontinuous material.

3.4. The Acoustic Riemann Problem

In the previous sections, we have seen two different ways of analyzing the wave structure of the homogeneous version of our problem. We combine them with the interface conditions defined in section 2.3.1 to construct a Riemann solver for the variable coefficient acoustic wave equation.

Equation (2.82) defines the acoustic flux matrix in x-direction, which we can diagonalize as

$$\boldsymbol{A}^{\text{ac}}(x) = \boldsymbol{R}(x)\boldsymbol{\Lambda}(x)\boldsymbol{R}^{-1}(x), \tag{3.33}$$

with

$$\boldsymbol{\Lambda}(x) = \operatorname{diag}([-c(x), c(x), 0, 0]), \tag{3.34}$$

$$\boldsymbol{R}(x) = \begin{pmatrix} -Z(x) & Z(x) & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.35}$$

Here $Z = \rho\sqrt{K}\rho = c\rho$ is the impedance. The eigenvectors can be arbitrarily scaled. Here, we use the scaling from [95], which allows us to describe the Riemann problem in terms of the impedances.

Figure 3.4 shows the wave structure of the Riemann problem. We have a left-moving wave with speed c^L and a right-moving wave with speed c^R. Due to the inhomogeneous material, we have two intermediate states $\boldsymbol{q^{*L}}$ and $\boldsymbol{q^{*R}}$. These states are called "star states". We only need to consider the pressure and the velocity in x-direction, as the other quantities are in the nullspace of $\boldsymbol{A}^{\text{ac}}$. Therefore, they do not contribute to the waves.

Now, we can apply our methods to derive a solution of a Riemann problem. We define the matrix

$$\boldsymbol{R}^{LR} = \begin{pmatrix} \overbrace{Z^L}^{\boldsymbol{r^L}} & \overbrace{Z^R}^{\boldsymbol{r^R}} & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{3.36}$$

which uses material from the left side for right-going waves and from the right side for left-going waves.

The jumps between waves are governed by the Rankine-Hugoniot conditions (section 3.3)

$$\begin{aligned} \boldsymbol{A}^{\text{ac}} \left(\boldsymbol{q^{*L}} - \boldsymbol{q^L}\right) &= -c^L \left(\boldsymbol{q^{*L}} - \boldsymbol{q^L}\right), \\ \boldsymbol{A}^{\text{ac}} \left(\boldsymbol{q^R} - \boldsymbol{q^{*R}}\right) &= c^R \left(\boldsymbol{q^R} - \boldsymbol{q^{*R}}\right), \end{aligned} \tag{3.37}$$

which are valid because in each region the matrix $\boldsymbol{A}^{\text{ac}}$ is constant. Hence, similar to the homogeneous case discussed in section 3.2, we can describe the jumps in the characteristic variables, i.e., in the eigenvector basis, using the respective flux matrix. From this, we obtain the additional relations

$$\boldsymbol{q^{*L}} - \boldsymbol{q^L} = \alpha^L \boldsymbol{r^L}, \tag{3.38}$$

$$\boldsymbol{q^R} - \boldsymbol{q^{*R}} = \alpha^R \boldsymbol{r^R}. \tag{3.39}$$

The quantities α^L and α^R are the wave strengths. Note that they depend on the normalization of the eigenvectors. However, the Rankine-Hugoniot conditions do not hold over the jump in the material. Hence, we need to use the physical interface conditions (equation (2.93)) to connect the states $\boldsymbol{q^{*L}}$ and $\boldsymbol{q^{*R}}$.

$$\begin{pmatrix} p^{*L} \\ v_1^{*L} \end{pmatrix} = \begin{pmatrix} p^{*R} \\ v_1^{*R}. \end{pmatrix} \tag{3.40}$$

This does not give us any information about the velocities v_2 and v_3 in the star states, as they can be discontinuous. Adding equations (3.38) and (3.39) and using equation (3.40), we arrive at

$$\boldsymbol{q^{*L}} - \boldsymbol{q^L} + \boldsymbol{q^R} - \boldsymbol{q^{*R}} = \boldsymbol{q^R} - \boldsymbol{q^L} = \alpha^L \boldsymbol{r^L} + \alpha^R \boldsymbol{r^R}. \tag{3.41}$$

Here, all vectors should be understood as being restricted to pressure and velocity. This is an abuse of notation; however, as mentioned before, the other values are irrelevant. We can convert equation (3.41) to the linear system

$$\begin{pmatrix} -Z^L & Z^R \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha^L \\ \alpha^R \end{pmatrix} = \begin{pmatrix} p^R - p^L \\ v_1^R - v_1^L \end{pmatrix}. \tag{3.42}$$

Solving this results in

$$\alpha^L = \frac{1}{Z^L + Z^R} \left(p^L - p^R + Z^R(v_1^R - v_1^L)\right), \tag{3.43}$$

$$\alpha^R = \frac{1}{Z^L + Z^R} \left(p^R - p^L + Z^L(v_1^R - v_1^L)\right). \tag{3.44}$$

We can compute the left star state by inserting equation (3.43) into equation (3.38), and the right one by inserting equation (3.43) into equation (3.39). This leads us to

$$\begin{aligned}\begin{pmatrix} p^{*L} \\ v_1^{*L} \end{pmatrix} &= \begin{pmatrix} p^L \\ v_1^L \end{pmatrix} + \alpha^L \begin{pmatrix} Z^L \\ -1 \end{pmatrix}, \\ \begin{pmatrix} p^{*R} \\ v_1^{*R} \end{pmatrix} &= \begin{pmatrix} p^R \\ v_1^R \end{pmatrix} - \alpha^R \begin{pmatrix} Z^R \\ 1 \end{pmatrix}.\end{aligned} \tag{3.45}$$

Due to the interface condition (equation (3.40)), both states are identical and are given by

$$\boldsymbol{q}^* = \begin{pmatrix} p^{*L} \\ v_1^{*L} \end{pmatrix} = \begin{pmatrix} p^{*R} \\ v_1^{*R} \end{pmatrix} = \frac{1}{Z^L + Z^R} \begin{pmatrix} Z^R p^L + Z^L p^R + Z^L Z^R \left(v_1^L - v_1^R\right) \\ Z^L u^L + Z^R u^R + p^L - p^R \end{pmatrix}. \tag{3.46}$$

This provides an exact solution to our Riemann problem. In the constant impedance case, i.e., $Z = Z^L = Z^R$, equation (3.46) simplifies to

$$\boldsymbol{q}^* = \frac{1}{2} \begin{pmatrix} \left(p^L + p^R\right) + Z\left(v_1^L - v_1^R\right) \\ \left(v_1^L + v_1^R\right) + \left(p^L - p^R\right)/Z \end{pmatrix}. \tag{3.47}$$

Hence, the star state is not simply the average of the quantities on both sides but also includes a correction term for the difference in the other quantity.

3.4.1. Boundary Conditions

We have not yet discussed what happens at the boundaries of our domain where we do not have a neighbor state because no physical neighbor exists. We adapt the Riemann problem to this situation by adding a fictitious neighbor, commonly called the ghost cell. Then, we compute the state of this neighbor by solving an “inverse Riemann problem” which is constrained such that the solution of the Riemann problem with the fictitious neighbor on the right ($\boldsymbol{q}^{\boldsymbol{R}}$) and the solution of our local cell ($\boldsymbol{q}^{\boldsymbol{L}}$), which is on the left, results in the boundary state ($\boldsymbol{q}^*$). At first glance, this seems to be a useless exercise. However, in practice, we typically only want to prescribe a part of the solution, for example, the velocity, on the boundary. This approach then gives us the values of the other quantities like the pressure. Finally, it provides us with a direct avenue to implement boundary conditions in a numerical code.

In the following, we assume that the material is continuous across the boundary, i.e., that the material just outside of the domain is extrapolated from the inside. Hence, we drop the indices and, e.g., use $Z = Z^L = Z^R$ in this section.

Velocity inlet

The first boundary condition type is the velocity inlet. We prescribe a velocity given by some function $v_1^* = f(x)$ and assume that the pressure is continuous, i.e., $p^L = p^R$. Equating the star state (equation (3.47)) for the velocity to $f(x)$ leads to the equation

$$\frac{1}{2}(v_1^L + v_1^R) + \underbrace{\frac{1}{2Z}(p^L - p^R)}_{=0} = f(x), \tag{3.48}$$

which we can solve for the velocity v_1^R. This results in the ghost state

$$\begin{pmatrix} p^R \\ v_1^R \end{pmatrix} = \begin{pmatrix} p^R \\ 2f(x) - v_1^L \end{pmatrix}. \tag{3.49}$$

Inserting equation (3.49) into equation (3.47) leads us to the state at the boundary

$$\begin{pmatrix} p^* \\ v_1^* \end{pmatrix} = \begin{pmatrix} p^L + Z(v_1^L - f(x)) \\ f(x) \end{pmatrix}. \tag{3.50}$$

Hence, the velocity at the boundary is exactly equal to the prescribed velocity, and the pressure is equal to the pressure next to the boundary with a correction term that counteracts the jump in velocity. An important special case of this boundary condition is the rigid boundary, given by equation (2.56), where $f(x) = 0$.

Pressure inlet

The second type of boundary condition is the pressure inlet. Here, we prescribe a pressure given by some function $p^* = f(x)$ and assume that the velocity is continuous, i.e., $v_1^L = v_1^R$. We can compute it in the same way as the velocity inlet. We equate the star state (equation (3.47)) for the pressure to $f(x)$, resulting in the equation

$$\frac{1}{2}(p^L + p^R) + \underbrace{\frac{Z}{2}(v_1^L - v_1^R)}_{=0} = f(x), \tag{3.51}$$

which we solve for the pressure p^R. Hence, we have the ghost state

$$\begin{pmatrix} p^R \\ u^R \end{pmatrix} = \begin{pmatrix} 2f(x) - p^L \\ v_1^L \end{pmatrix}. \tag{3.52}$$

Finally, we insert equation (3.52) into equation (3.47) to compute the star state

$$\begin{pmatrix} p^* \\ u^* \end{pmatrix} = \begin{pmatrix} f(x) \\ v_1^L + \frac{1}{Z}(p^L - f(x)) \end{pmatrix}. \tag{3.53}$$

The pressure at the boundary is prescribed exactly, and the velocity is modified by a term that corrects for the jump in the pressure.

The free surface boundary condition is a special case for $p^* = 0$. This results in the states

$$\begin{aligned} q^R &= \begin{pmatrix} -p^L \\ v_1^L \end{pmatrix}, \\ q^* &= \begin{pmatrix} 0 \\ v_1^L + \frac{1}{Z}p^L \end{pmatrix}. \end{aligned} \tag{3.54}$$

Gravitational free surface

Another important special case of the pressure inlet boundary condition is the gravitational free surface boundary condition, given by equation (2.88). For this, we impose a pressure at the boundary of $p^* = \rho\eta g$, which we insert into equations (3.52) and (3.53), resulting in the ghost state

$$\begin{pmatrix} p^R \\ v_1^R \end{pmatrix} = \begin{pmatrix} 2\rho g\eta - p^L \\ v_1^L \end{pmatrix} \tag{3.55}$$

and the boundary state

$$\begin{pmatrix} p^* \\ v_1^* \end{pmatrix} = \begin{pmatrix} \rho g\eta \\ v_1^L + \frac{1}{Z}\left(p^L - \rho g\eta\right) \end{pmatrix}. \tag{3.56}$$

This concludes the dynamic part of this boundary condition. To close the boundary condition, we require the kinematic part, defined by equation (2.89). Hence, we define the displacement by the ODE

$$\frac{\partial\eta}{\partial t} = v_1^* = v_1^L + \frac{1}{Z}\left(p^L - \rho g\eta\right), \tag{3.57}$$

where it is important to note that v_1^* is the state at the boundary. Thus, we must solve an ODE to compute η. This boundary condition reduces to the classical free surface boundary condition (equation (3.54)) in the case of vanishing gravity $g = 0$.

3.4.2. Discussion

We have seen in this section how we can solve an inhomogeneous Riemann problem using the example of acoustic-acoustic interfaces. The solution strategy consisted of using the Rankine-Hugoniot conditions and the technique of characteristic variables to describe the jumps inside each material. We then used the physical interface conditions to close the system. We solved the resulting linear systems for the wave strengths, which we inserted into the jump conditions.

Furthermore, we discussed how we can investigate boundary conditions by computing an inverse Riemann problem. This results in the exact state at the boundary for the quantity we want to prescribe and a penalty term for the other. A further aspect that becomes clear from the inverse Riemann approach is that we can only prescribe either the pressure or the velocity at the interface. We cannot prescribe both, as it would be impossible to derive a boundary state that fulfills equation (3.47).

We have not only learned how to solve one specific type of Riemann problem, but we can apply the solution strategy to other linear Riemann problems and boundary conditions.

3.5. The Elastic Riemann Problem

In this section, we solve the Riemann problem for elastic-elastic interfaces by applying techniques from section 3.4. The flux matrix in x-direction (equation (2.30)) has the

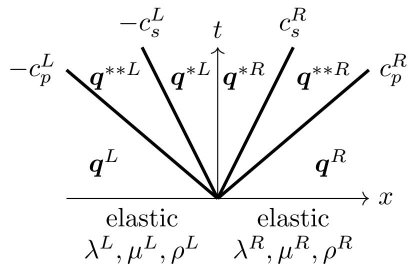


Figure 3.5.: The elastic-elastic Riemann problem. The black lines correspond to characteristic waves that separate the constant states. The left ($\boldsymbol{q^L}$) and right state ($\boldsymbol{q^L}$) are the initial condition. We have four star states, which are called, from left to right, $\boldsymbol{q^{**L}}, \boldsymbol{q^{*L}}, \boldsymbol{q^{*R}}$ and $\boldsymbol{q^{**R}}$.

eigenvalues $(-c_p(x), -c_s(x), -c_s(x), 0, 0, 0, c_s(x), c_s(x), c_p(x))$. Hence, we have two wave types and three non-propagating modes. We define the matrix

$$\boldsymbol{R}^{LR} = \begin{array}{c} \begin{array}{ccccccccc} \underbrace{\boldsymbol{r_p^L}} & \underbrace{\boldsymbol{r_{s1}^L}} & \underbrace{\boldsymbol{r_{s2}^L}} & & & & \underbrace{\boldsymbol{r_{s1}^R}} & \underbrace{\boldsymbol{r_{s2}^R}} & \underbrace{\boldsymbol{r_p^R}} \end{array} \\ \begin{pmatrix} \frac{Z^{L^2}}{\rho} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{Z^{R^2}}{\rho} \\ \lambda^L & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \lambda^R \\ \lambda^L & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \lambda^R \\ 0 & \mu^L & 0 & 0 & 0 & 0 & \mu^R & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \mu^L & 0 & 0 & 0 & 0 & \mu^R & 0 \\ c_p^L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_p^R \\ 0 & c_s^L & 0 & 0 & 0 & 0 & -c_s^R & 0 & 0 \\ 0 & 0 & c_s^L & 0 & 0 & 0 & 0 & -c_s^R & 0 \end{pmatrix} \end{array}, \tag{3.58}$$

which collects the eigenvectors for the left and right material. Here, we used the P wave impedance $Z_p = \rho c_p$ and S wave impedance $Z_s = \rho c_s$. We have one eigenvector per S wave per direction, resulting in the wave structure shown in figure 3.5. We only consider the quantities $(\sigma_{11}, \sigma_{12}, \sigma_{13}, v_1, v_2, v_3)$, as all others lie in the null-space of $\boldsymbol{A}$ and hence do not contribute to propagating wave modes.

We apply the Rankine-Hugoniot conditions to the jump between waves, arriving at

$$\begin{aligned} \boldsymbol{A}\left(\boldsymbol{q^{**L}} - \boldsymbol{q^L}\right) &= -c_p^L \left(\boldsymbol{q^{**L}} - \boldsymbol{q^L}\right), \\ \boldsymbol{A}\left(\boldsymbol{q^{*L}} - \boldsymbol{q^{**L}}\right) &= -c_s^L \left(\boldsymbol{q^{*L}} - \boldsymbol{q^{**L}}\right), \\ \boldsymbol{A}\left(\boldsymbol{q^{**R}} - \boldsymbol{q^{*R}}\right) &= c_s^R \left(\boldsymbol{q^{**R}} - \boldsymbol{q^{*R}}\right), \\ \boldsymbol{A}\left(\boldsymbol{q^R} - \boldsymbol{q^{**R}}\right) &= c_p^R \left(\boldsymbol{q^R} - \boldsymbol{q^{**R}}\right). \end{aligned} \tag{3.59}$$

Again, this is an eigenproblem. Hence, we can expand the jumps in the eigenbasis which results in

$$\begin{aligned}
\boldsymbol{q^{**L}} - \boldsymbol{q^L} &= \alpha_p^L \boldsymbol{r_p^L}, && (3.60)\\
\boldsymbol{q^{*L}} - \boldsymbol{q^{**L}} &= \alpha_{s1}^L \boldsymbol{r_{s1}^L} + \alpha_{s2}^L \boldsymbol{r_{s2}^L}, && (3.61)\\
\boldsymbol{q^{**R}} - \boldsymbol{q^{*R}} &= \alpha_{s1}^R \boldsymbol{r_{s1}^R} + \alpha_{s2}^R \boldsymbol{r_{s2}^R}, && (3.62)\\
\boldsymbol{q^R} - \boldsymbol{q^{**R}} &= \alpha_p^R \boldsymbol{r_p^R}. && (3.63)
\end{aligned}$$

We combine this with the interface conditions (c.f. equation (2.92))

$$\left(\sigma_{11}^L, \sigma_{12}^L, \sigma_{13}^L, v_1^L, v_2^L, v_3^L\right)^T = \left(\sigma_{11}^R, \sigma_{12}^R, \sigma_{13}^R, v_1^R, v_2^R, v_3^R\right)^T \tag{3.64}$$

which closes the system.

Adding equations (3.60) to (3.63) together and applying equation (3.64) results in

$$\alpha_p^L \boldsymbol{r_p^L} + \alpha_{s1}^L \boldsymbol{r_{s1}^L} + \alpha_{s2}^L \boldsymbol{r_{s2}^L} + \alpha_{s1}^R \boldsymbol{r_{s1}^R} + \alpha_{s2}^R \boldsymbol{r_{s2}^R} + \alpha_p^R \boldsymbol{r_p^R} = \boldsymbol{q^R} - \boldsymbol{q^L} \tag{3.65}$$

where, again, all vectors are assumed to be restricted to the relevant quantities. Equation (3.65) is equivalent to the linear system

$$\begin{pmatrix}
\frac{(Z_p^L)^2}{\rho^L} & 0 & 0 & 0 & 0 & \frac{(Z_p^R)^2}{\rho_R} \\
0 & \mu^L & 0 & \mu^R & 0 & 0 \\
0 & 0 & \mu^L & 0 & \mu^R & 0 \\
c_p^L & 0 & 0 & 0 & 0 & -c_p^R \\
0 & c_s^L & 0 & -c_s^R & 0 & 0 \\
0 & 0 & c_s^L & 0 & -c_s^R & 0
\end{pmatrix}
\begin{pmatrix} \alpha_p^L \\ \alpha_{s1}^L \\ \alpha_{s2}^L \\ \alpha_p^R \\ \alpha_{s1}^R \\ \alpha_{s2}^R \end{pmatrix}
=
\begin{pmatrix} \sigma_{11}^R - \sigma_{11}^L \\ \sigma_{12}^R - \sigma_{12}^L \\ \sigma_{13}^R - \sigma_{13}^L \\ v_1^R - v_1^L \\ v_2^R - v_2^L \\ v_3^R - v_3^L \end{pmatrix}. \tag{3.66}$$

While this system looks quite complicated, we can reorder it [164]. By introducing the permutation matrices $\boldsymbol{P_\alpha}$ and $\boldsymbol{P_q}$, we can write the resulting system as

$$\begin{aligned}
\boldsymbol{P_q}^{-1}\ \boldsymbol{P_q R P_\alpha}\ \boldsymbol{P_\alpha}^{-1}\ \boldsymbol{\alpha} &= \boldsymbol{q}^L - \boldsymbol{q}^R,\\
(\boldsymbol{P_q R P_\alpha}\ )\,(\boldsymbol{P_\alpha}^{-1}\ \boldsymbol{\alpha}) &= \boldsymbol{P_q}\ (\boldsymbol{q^L} - \boldsymbol{q^R}).
\end{aligned} \tag{3.67}$$

In our case, this results in the system

$$\begin{pmatrix}
\frac{(Z_p^L)^2}{\rho^L} & \frac{(Z_p)^{R2}}{\rho^R} & 0 & 0 & 0 & 0 \\
c_p^L & -c_p^R & 0 & 0 & 0 & 0 \\
0 & 0 & \mu^L & \mu^R & 0 & 0 \\
0 & 0 & c_s^L & -c_s^R & 0 & 0 \\
0 & 0 & 0 & 0 & \mu^L & \mu^R \\
0 & 0 & 0 & 0 & c_s^L & -c_s^R
\end{pmatrix}
\begin{pmatrix} \alpha_p^L \\ \alpha_p^R \\ \alpha_{s1}^L \\ \alpha_{s1}^R \\ \alpha_{s2}^L \\ \alpha_{s2}^R \end{pmatrix}
=
\begin{pmatrix} \sigma_{11}^R - \sigma_{11}^L \\ v_1^R - v_1^L \\ \sigma_{12}^R - \sigma_{12}^L \\ v_2^R - v_2^L \\ \sigma_{13}^R - \sigma_{13}^L \\ v_3^R - v_3^L \end{pmatrix}, \tag{3.68}$$

which has a block structure. Therefore, only two variables are coupled together.

We can compute the star states by adding equations (3.60) and (3.61) and by adding equations (3.62) and (3.63), which leads to the equations

$$\begin{aligned}
\boldsymbol{q^{*L}} &= \boldsymbol{q^L} + \alpha_p^L \boldsymbol{r_p^L} + \alpha_{s1}^L \boldsymbol{r_{s1}^L} + \alpha_{s2}^L \boldsymbol{r_{s2}^L},\\
\boldsymbol{q^{*R}} &= \boldsymbol{q^R} - \alpha_p^R \boldsymbol{r_p^R} - \alpha_{s1}^R \boldsymbol{r_{s1}^R} - \alpha_{s2}^R \boldsymbol{r_{s2}^R}.
\end{aligned} \tag{3.69}$$

Due to the interface conditions, both star states are identical.

Solving equation (3.68) for the wave strengths and inserting them into the jump condition (equation (3.69)) leads to the star state of

$$\begin{aligned}
\begin{pmatrix}\sigma_{11}^* \\ v_1^*\end{pmatrix} &= \frac{1}{Z_p^L + Z_p^R}\begin{pmatrix} Z_p^R \sigma_{11}^L + Z_p^L \sigma_{11}^R + Z_p^L Z_p^R (v_1^R - v_1^L) \\ Z_p^L v_1^L + Z_p^R v_1^R + \sigma_{11}^R - \sigma_{11}^L \end{pmatrix}, \\
\begin{pmatrix}\sigma_{12}^* \\ v_2^*\end{pmatrix} &= \frac{1}{c_s^R \mu^L + c_s^L \mu^R}\begin{pmatrix} c_s^L \mu^R \sigma_{12}^L + c_s^R \mu^L \sigma_{12}^R + \mu^L \mu^R \left(v_2^R - v_2^L\right) \\ c_s^R \mu^L v_2^L + c_s^L \mu^R v_2^R + c_s^L c_s^R \left(\sigma_{12}^R - \sigma_{12}^L\right) \end{pmatrix}, \\
\begin{pmatrix}\sigma_{13}^* \\ v_3^*\end{pmatrix} &= \frac{1}{c_s^R \mu^L + c_s^L \mu^R}\begin{pmatrix} c_s^L \mu^R \sigma_{13}^L + c_s^R \mu^L \sigma_{13}^R + \mu^L \mu^R \left(v_3^R + v_3^L\right) \\ c_s^R \mu^L v_3^L + c_s^L \mu^R v_3^R + c_s^L c_s^R \left(\sigma_{13}^R - \sigma_{13}^L\right) \end{pmatrix},
\end{aligned} \tag{3.70}$$

where we ignored variables that lie in the nullspace of $\boldsymbol{A}$. We grouped the star states to make it clear which quantities are coupled. Here, it is interesting that the star state for the stress σ_{11}^* is similar to the one we derived earlier for the pressure in the acoustic wave equation (equation (3.46)).

Similar to the boundary conditions for the acoustic case, we assume that the material is continuous over the boundary. The same solution strategy works: We compute the left state by solving the inverse Riemann problem and then the ghost state with equation (3.70).

Free surface

We want to set the traction at the boundary to zero for the free surface condition. The velocity is continuous. Formally, we have the following conditions

$$\begin{aligned}
\sigma_{11}^* = \sigma_{12}^* = \sigma_{13}^* &= 0, \\
\boldsymbol{v}^L &= \boldsymbol{v}^R.
\end{aligned} \tag{3.71}$$

We equate these values for the stress tensor with the star state defined by equation (3.70). Solving for the stress tensor, we arrive at the ghost state

$$\boldsymbol{q^R} = \begin{pmatrix} -\sigma_{11}^L & -\sigma_{12}^L & -\sigma_{13}^L & v_1^L & v_2^L & v_3^L \end{pmatrix}^T . \tag{3.72}$$

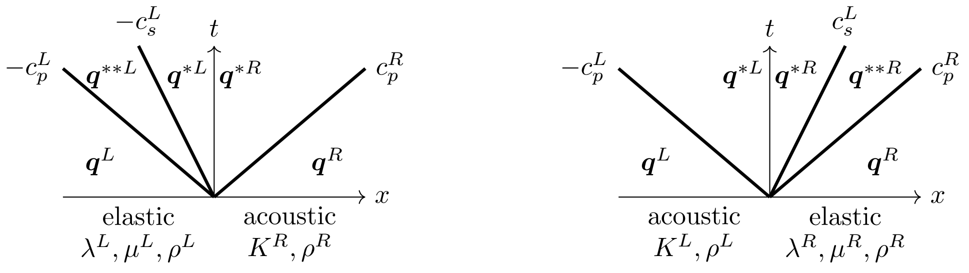
Inserting this state back into equation (3.70) leads us to the boundary state

$$\begin{pmatrix} \sigma_{11}^* \\ \sigma_{12}^* \\ \sigma_{13}^* \\ v_1^* \\ v_2^* \\ v_3^* \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ v_1^L - Z_p^{-1} \sigma_{11}^L \\ v_2^L - Z_s^{-1} \sigma_{12}^L \\ v_3^L - Z_s^{-1} \sigma_{13}^L \end{pmatrix}. \tag{3.73}$$

The traction is zero, as expected, and the velocities have acquired a penalty term that counteracts the jump in pressure. This is, as expected, similar to the acoustic free surface boundary condition (equation (3.54)).

We discussed in this section how we can solve the elastic-elastic Riemann problem. This was done using the same techniques we derived for the acoustic wave equation. Furthermore, we derived the star state for the free surface boundary condition.

3.6. Elastic-Acoustic & Acoustic-Elastic Riemann Problems



(a) The elastic-acoustic Riemann problem. (b) The elastic-acoustic Riemann problem.

Figure 3.6.: Riemann problems for coupled elastic and acoustic media. In each figure, the black lines correspond to characteristic waves that separate the constant states. The wave structure directly represents the underlying material. The left ($\boldsymbol{q^L}$) and right state ($\boldsymbol{q^R}$) are the initial condition. Two states are separated by an acoustic wave in each acoustic region, and in each elastic region, three states are separated by the P and S waves. Figures adapted from [79].

In this section, we tackle the elastic-acoustic and acoustic-elastic Riemann problems. By now, our tool belt is well-equipped to handle this. However, in contrast to the acoustic-acoustic and elastic-elastic Riemann problems, coupling acoustic and elastic regions combines different materials and separate, albeit related, PDEs. This brings additional challenges. It may seem obvious that we could treat the acoustic wave equation as a special case of the elastic wave equation and then reduce the Riemann problem to the elastic-elastic one. However, this leads to a loss of strong hyperbolicity as the flux matrix is no longer diagonalizable [95, 175]. Hence, this approach does not work. However, we can use the same strategy as before: We use the Rankine-Hugoniot jump conditions together with the interface condition. The jump conditions govern the flux in the elastic and acoustic regions, and the interface conditions are used to combine both. Here, we use the embedding of the acoustic wave equation into the elastic wave equation introduced by equation (2.91). Additionally, we also use the flux matrix embedding.

Figure 3.6 shows the wave structure of these problems. We invite the reader to compare this with the structure of the acoustic-acoustic problem (figure 3.4) and of the elastic-elastic problem (figure 3.5). It becomes immediately clear that the wave structure of the coupled problem is simply the combination of both. We discuss both elastic-acoustic and acoustic-elastic problems here for completeness.

Elastic-acoustic First, we discuss the elastic-acoustic Riemann problem, as shown in figure 3.6a. We have the matrix of eigenvalues

$$\boldsymbol{R^{EA}} = \begin{pmatrix} \overbrace{\frac{(Z_p^L)^2}{\rho^L}}^{\boldsymbol{r_p^L}} & \overbrace{0}^{\boldsymbol{r_{s1}^L}} & \overbrace{0}^{\boldsymbol{r_{s2}^L}} & 0 & 0 & 0 & 0 & 0 & \overbrace{1}^{\boldsymbol{r_p^R}} \\ \lambda^L & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \lambda^L & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & \mu^L & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \mu^L & 0 & 0 & 0 & 0 & 0 & 0 \\ c_p^L & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{Z_a^R} \\ 0 & c_s^L & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & c_s^L & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{3.74}$$

which combines the elastic eigenvectors on the left and the acoustic eigenvector on the right. The acoustic eigenvectors are extended using the embedding, given by equation (2.91). We must be careful due to the difference in sign convention between stress and pressure.

The Rankine-Hugoniot conditions are given by

$$\begin{aligned} \boldsymbol{A}\left(\boldsymbol{q^{**L}} - \boldsymbol{q^L}\right) &= -c_p^L\left(\boldsymbol{q^{**L}} - \boldsymbol{q^L}\right), \\ \boldsymbol{A}\left(\boldsymbol{q^{*L}} - \boldsymbol{q^{**L}}\right) &= -c_s^L\left(\boldsymbol{q^{*L}} - \boldsymbol{q^{**L}}\right), \\ \boldsymbol{A}\left(\boldsymbol{q^R} - \boldsymbol{q^{*R}}\right) &= c_p^R\left(\boldsymbol{q^R} - \boldsymbol{q^{*R}}\right). \end{aligned} \tag{3.75}$$

We expand the jumps again in the basis of eigenvectors

$$\begin{aligned} \boldsymbol{q^{**L}} - \boldsymbol{q^L} &= \alpha_p^L \boldsymbol{r_p^L}, \\ \boldsymbol{q^{*L}} - \boldsymbol{q^{**L}} &= \alpha_{s1}^L \boldsymbol{r_{s1}^L} + \alpha_{s2}^L \boldsymbol{r_{s2}^L}, \\ \boldsymbol{q^R} - \boldsymbol{q^{*R}} &= \alpha_p^R \boldsymbol{r_p^R}. \end{aligned} \tag{3.76}$$

We combine them with the physical interface conditions (equation (2.94)), which require

$$\begin{pmatrix} v_1^{*L} \\ \sigma_{11}^{*L} \\ \sigma_{12}^{*L} \\ \sigma_{13}^{*L} \end{pmatrix} = \begin{pmatrix} v_1^{*R} \\ \sigma_{11}^{*R} \\ \sigma_{12}^{*R} \\ \sigma_{13}^{*R} \end{pmatrix}. \tag{3.77}$$

Hence, at this type of interface, the tangential velocities are allowed to be discontinuous! Interestingly, the shear stresses, however, are continuous. Because they do not exist in the acoustic medium, we can expect that they also vanish at the elastic part of the interface. We can check this assumption by straightforward calculations. Here, it is again convenient to use a reduced and reordered matrix, similar to equation (3.67), which we used for the elastic wave equation [164]. We arrive at the linear system for the wave strengths

$$\begin{pmatrix} \frac{(Z_p^L)^2}{\rho^L} & 1 & 0 & 0 \\ c_p^L & -\frac{1}{Z_a^R} & 0 & 0 \\ 0 & 0 & \mu^L & 0 \\ 0 & 0 & 0 & \mu^L \end{pmatrix} \begin{pmatrix} \alpha_p^L \\ \alpha_p^R \\ \alpha_{s1}^L \\ \alpha_{s2}^L \end{pmatrix} = \begin{pmatrix} \sigma_{11}^R - \sigma_{11}^L \\ v_1^R - v_1^L \\ \sigma_{12}^R - \sigma_{12}^L \\ \sigma_{13}^R - \sigma_{13}^L \end{pmatrix}. \tag{3.78}$$

The P waves of both media are coupled; however, the S waves of the elastic part are uncoupled. Because the shear stresses are zero in the fluid, we can directly read the strength of the S waves

$$\alpha_{s1}^L = -\frac{\sigma_{12}^L}{\mu}, \qquad \alpha_{s2}^L = -\frac{\sigma_{13}^L}{\mu}. \tag{3.79}$$

In contrast to the previously discussed numerical fluxes, the elastic-acoustic Riemann problem admits two different star states separated by the material discontinuity. This is because the tangential velocities are allowed to be discontinuous. Hence, for the elastic medium, we have the state

$$\begin{aligned} \begin{pmatrix} \sigma_{11}^{*L} \\ \sigma_{12}^{*L} \\ \sigma_{23}^{*L} \\ v_1^{*L} \\ v_2^{*L} \\ v_3^{*L} \end{pmatrix} &= \begin{pmatrix} \sigma_{11}^{L} \\ \sigma_{12}^{L} \\ \sigma_{23}^{L} \\ v_1^{L} \\ v_2^{L} \\ v_3^{L} \end{pmatrix} + \alpha_p^L \boldsymbol{r_p^L} + \alpha_{s1}^L \boldsymbol{r_{s1}^L} + \alpha_{s2}^L \boldsymbol{r_{s2}^L} \\ &= \begin{pmatrix} \left(Z_p^R Z_p^L + \left(Z_p^L\right)^2\right)^{-1} \left(Z_p^R Z_p^L \sigma_{11}^L + \left(Z_p^L\right)^2 \sigma_{11}^R + Z_p^R \left(Z_p^L\right)^2 \left(v_1^R - v_1^L\right)\right) \\ 0 \\ 0 \\ \left(Z_p^R Z_p^L + \left(Z_p^L\right)^2\right)^{-1} \left(\left(Z_p^L\right)^2 v_1^L + Z_p^R Z_p^L v_1^R + Z_p^L \left(\sigma_{11}^R - \sigma_{11}^L\right)\right) \\ v_2^L - \left(Z_s^L\right)^{-1} \sigma_{12}^L \\ v_3^L - \left(Z_s^L\right)^{-1} \sigma_{13}^L \end{pmatrix}, \end{aligned} \tag{3.80}$$

where we can see that the tangential velocities include a term that penalizes existing shear traction. The shear traction is, as we expected, zero at the interface. On the acoustic side, we have the state

$$\begin{aligned} \begin{pmatrix} \sigma_{11}^{*R} \\ v_1^{*R} \end{pmatrix} &= \begin{pmatrix} \sigma_{11}^{R} \\ v_1^{R} \end{pmatrix} - \alpha_p^L \boldsymbol{r_p^R} \\ &= \begin{pmatrix} \left(Z_p^R Z_p^L + \left(Z_p^L\right)^2\right)^{-1} \left(Z_p^R Z_p^L \sigma_{11}^L + \left(Z_p^L\right)^2 \sigma_{11}^R + Z_p^R \left(Z_p^L\right)^2 \left(v_1^R - v_1^L\right)\right) \\ \left(Z_p^R Z_p^L + \left(Z_p^L\right)^2\right)^{-1} \left(\left(Z_p^L\right)^2 v_1^L + Z_p^R Z_p^L v_1^R + Z_p^L \left(\sigma_{11}^R - \sigma_{11}^L\right)\right) \end{pmatrix} \end{aligned} \tag{3.81}$$

where we have ignored the tangential velocities and shear stresses as they lie in the null space of the flux matrix.

Acoustic-elastic This case is very similar to the elastic-acoustic case. We build the combined eigenvector matrix as

$$
\begin{array}{c}
\begin{array}{ccccccccc}
\underbrace{\boldsymbol{r}_{\boldsymbol{p}}^{\boldsymbol{L}}} & & & & & & \underbrace{\boldsymbol{r}_{\boldsymbol{s1}}^{\boldsymbol{R}}} & \underbrace{\boldsymbol{r}_{\boldsymbol{s2}}^{\boldsymbol{R}}} & \underbrace{\boldsymbol{r}_{\boldsymbol{p}}^{\boldsymbol{R}}}
\end{array} \\
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{(Z_p^R)^2}{\rho^R} \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \lambda^R \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \lambda^R \\
0 & 1 & 0 & 0 & 0 & 0 & \mu^R & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & \mu_R & 0 \\
\frac{1}{Z^L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_p^R \\
0 & 0 & 0 & 0 & 0 & 0 & -c_s^R & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -c_s^R & 0
\end{pmatrix}
\end{array}
. \tag{3.82}
$$

The jumps are depicted in figure 3.6b. For the sake of brevity, we skip directly to the linear system for the wave speeds

$$
\begin{pmatrix}
1 & \frac{Z_R^2}{\rho^R} & 0 & 0 \\
\frac{1}{Z_L} & -c_p^R & 0 & 0 \\
0 & 0 & \mu^R & 0 \\
0 & 0 & 0 & \mu^R
\end{pmatrix}
\begin{pmatrix}
\alpha_p^L \\ \alpha_p^R \\ \alpha_{s1}^R \\ \alpha_{s2}^R
\end{pmatrix}
=
\begin{pmatrix}
\sigma_{11}^R - \sigma_{11}^L \\
v_1^R - v_1^L \\
\sigma_{12}^R - \sigma_{12}^L \\
\sigma_{13}^R - \sigma_{13}^L
\end{pmatrix}
. \tag{3.83}
$$

It has the same structure as the system for the elastic-acoustic case, given by equation (3.78). Again, we can directly read the strengths of the S waves propagating rightward into the elastic medium. From this, it is trivial—but laborsome—to compute the star states. Hence, we skip this here.

In this section, we discussed how we can solve coupled Riemann problems. We were able to apply the same solution strategies as before. The resulting wave structure and the star states followed our intuition that the shear stresses at the boundary are also zero at the elastic side of the boundary. A jump in the shear stresses leads to a velocity that counteracts it. This has also been observed in [164].

3.7. Computational Aspects

In this section, we discuss the computational aspects of the exact Riemann solver. For this, we return to the notation used in section 3.1 and mark vectors that are in the face-oriented coordinate system with a tilde, such as $\tilde{\boldsymbol{q}}$.

We can write the computation in a more concise form by collecting operations in matrices. This is similar to the notation in [164]. First, we define the characteristic matrices

$$\begin{aligned} \chi_{ij}^{L} &= \begin{cases} 1 & \text{if } \lambda_i < 0, \\ 0 & \text{else,} \end{cases} \\ \chi_{ij}^{R} &= \begin{cases} 1 & \text{if } \lambda_i > 0, \\ 0 & \text{else,} \end{cases} \end{aligned} \tag{3.84}$$

which select the left and right-going waves, respectively. In the following, we use $\boldsymbol{R}$ to denote the matrix of eigenvectors that considers the material of both sides. With this, we can write the star state as

$$\tilde{\boldsymbol{q}}^{*} = \tilde{\boldsymbol{q}}^{\boldsymbol{L}} + \boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{L}}\boldsymbol{\alpha}, \tag{3.85}$$

or equivalently as

$$\tilde{\boldsymbol{q}}^{*} = \tilde{\boldsymbol{q}}^{\boldsymbol{R}} - \boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{R}}\boldsymbol{\alpha}. \tag{3.86}$$

Both definitions lead to the same result.[2] This works for all considered Riemann problems, which can be shown by inspecting the corresponding jump conditions given by equations (3.45), (3.69), (3.80), and (3.81). The wave strengths can be computed by

$$\boldsymbol{\alpha} = \boldsymbol{R}^{-1}(\tilde{\boldsymbol{q}}^{\boldsymbol{R}} - \tilde{\boldsymbol{q}}^{\boldsymbol{L}}). \tag{3.87}$$

In the following, we substitute $\boldsymbol{\chi}^{\boldsymbol{R}} = \boldsymbol{I} - \boldsymbol{\chi}^{\boldsymbol{L}}$, which works because while we select additional waves, these waves are not propagating. With this, we can split the computation of the star state with the identity

$$\begin{aligned} \tilde{\boldsymbol{q}}^{*} &= \tilde{\boldsymbol{q}}^{\boldsymbol{L}} + \boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{L}}\boldsymbol{R}^{-1}(\tilde{\boldsymbol{q}}^{\boldsymbol{R}} - \tilde{\boldsymbol{q}}^{\boldsymbol{L}}) \\ &= \boldsymbol{R}(\boldsymbol{I} - \boldsymbol{\chi}^{\boldsymbol{L}})\boldsymbol{R}^{-1}\tilde{\boldsymbol{q}}^{\boldsymbol{L}} + \boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{L}}\boldsymbol{R}^{-1}\tilde{\boldsymbol{q}}^{\boldsymbol{R}} \\ &= \underbrace{\boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{R}}\boldsymbol{R}^{-1}}_{=\boldsymbol{G}^{\boldsymbol{L}}}\tilde{\boldsymbol{q}}^{\boldsymbol{L}} + \underbrace{\boldsymbol{R}\boldsymbol{\chi}^{\boldsymbol{L}}\boldsymbol{R}^{-1}}_{=\boldsymbol{G}^{\boldsymbol{R}}}\tilde{\boldsymbol{q}}^{\boldsymbol{R}} \end{aligned} \tag{3.88}$$

This is convenient because the first and second summands only depend on the left and right state, respectively. We can precompute the matrices $\boldsymbol{G}^{\boldsymbol{L}}$ and $\boldsymbol{G}^{\boldsymbol{R}}$ as they do not depend on the state.

Finally, our original three-dimensional Riemann problem, stated in definition 1, can be solved by combining equation (3.88) with lemma 1. This leads us to

$$\boldsymbol{q}^{*} = \boldsymbol{\mathcal{T}}\boldsymbol{G}^{\boldsymbol{L}}\boldsymbol{\mathcal{T}}^{-1}\boldsymbol{q}^{\boldsymbol{L}} + \boldsymbol{\mathcal{T}}\boldsymbol{G}^{\boldsymbol{R}}\boldsymbol{\mathcal{T}}^{-1}\boldsymbol{q}^{\boldsymbol{R}}. \tag{3.89}$$

Hence, the solution of the plane-wave Riemann is given by a sum of matrix-vector products. It uses two matrices that depend on the orientation of the interface and the material but not on the states $\boldsymbol{q}^{\boldsymbol{L}}$ or $\boldsymbol{q}^{\boldsymbol{R}}$.

[2]Caveat: The values for quantities that lie in the nullspace of the flux matrix can differ.

3.8. Discussion

In this chapter, we derived a method to solve the acoustic-acoustic, elastic-elastic, acoustic-elastic, and elastic-acoustic Riemann problems for inhomogeneous materials. Our Riemann solver considers the materials from both sides of the interface. It is essentially identical to the one presented in [175]. Only taking the material from one side into account, as done, for example, in [69], leads to an inconsistent scheme [175].

We used the rotational invariance of our PDEs to reduce the three-dimensional Riemann problem to a one-dimensional one (section 3.1). Then, we discussed two solution strategies for this reduced Riemann problem: First, section 3.2 introduced the technique of characteristic variables, which allows us to directly solve the Riemann problem for homogeneous materials. Second, we derived the Rankine-Hugoniot jump conditions in section 3.3. They enable us to consider the jump over individual waves as an eigenproblem. Together with the physical interface conditions, we successfully solved Riemann problems that cover all possible combinations of acoustic and elastic media (sections 3.4 to 3.6). Our discussion included a derivation of all necessary boundary conditions. Finally, section 3.7 showed us that we can compute the solution of these problems efficiently.

Chapter 4.

Discretization

In this section, we discuss the Arbitrary DERivatives Discontinuous Galerkin (ADER-DG) discretization for our fully coupled model that we presented in chapter 2. We first introduce the DG space discretization (section 4.1) followed by the ADER time-stepping method (section 4.2). Both sections summarize the work from [35, 164]. Our main contribution is discussed in section 4.3: an efficient arbitrary high-order discretization of the gravitational boundary condition, which combines a nodal basis with ADER integration. Finally, section 4.4 discusses computational aspects of our ADER-DG scheme.

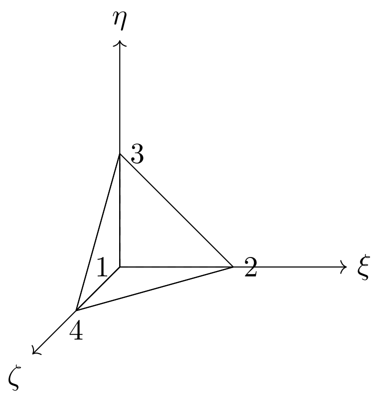
4.1. Discontinuous Galerkin

The Discontinuous Galerkin method was introduced in [124] for the neutron transport equation and was later extended to systems of hyperbolic conservation laws [16]. For a brief history of the method, we refer the interested reader to [20]. In this section, following the description of the DG discretization of the elastic wave equation in [69], we derive a DG space discretization of our fully coupled model on an unstructured tetrahedral mesh. We assume in the following that we use the coupling strategy discussed in section 2.3. Briefly, we consider a variable-coefficient hyperbolic PDE of the form given by equation (2.29). We embed the acoustic wave equation in the elastic wave equation with equation (2.91). Thus, the flux matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and the vector of unknowns $\boldsymbol{q}$ have the same shape in the entire domain. Section 4.1.1 discusses how the tetrahedral mesh is built and introduces the concept of reference elements. In section 4.1.2, we introduce high-order polynomial bases defined on these reference elements. We use these tools to derive and discretize the weak form of our PDE (section 4.1.3). This results in an entirely local scheme. Section 4.1.4 discusses how elements are coupled to each other using surface integrals and a numerical flux. Finally, section 4.1.5 presents a semi-discrete discretization of our PDE.

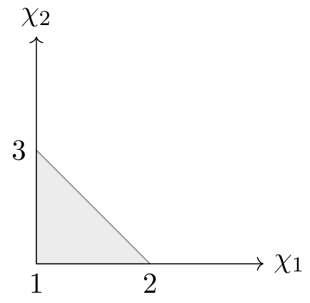
4.1.1. Mesh

In this section, we describe the construction of our computational mesh, which follows [35]. We assume that our computational geometry was tessellated into a conforming mesh of non-overlapping tetrahedra. Formally, we have the domain $\Omega = \bigcup_m \mathcal{T}_m$, where $\mathcal{T}_m$ denotes a tetrahedron. Furthermore, we use the function $\mathcal{N}(\mathcal{T})$ that takes a tetrahedron

$\mathcal{T}$ and returns the set of its neighbors. For each tetrahedron, the cardinality of $\mathcal{N}$ is equal to the number of neighbors not at the boundary of the domain.



(a) The reference tetrahedron $\hat{\mathcal{T}}$. It is defined in the local coordinate system $\boldsymbol{\xi} = (\xi, \eta, \zeta)$. The numbers correspond to vertices.



(b) The reference triangle ε_2. It is defined in the local coordinate system $\boldsymbol{\chi} = (\chi_1, \chi_2)$. The numbers correspond to vertices.

Figure 4.1.: Our reference elements.

To simplify our computations, we use the reference tetrahedron $\hat{\mathcal{T}}$, as shown in figure 4.1a. First, we introduce the connectivity function $c(\mathcal{T}, i)$ that returns the ith vertex of the tetrahedron $\mathcal{T}$. The reference element is defined by

$$c(\hat{\mathcal{T}}, 1) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \quad c(\hat{\mathcal{T}}, 2) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad c(\hat{\mathcal{T}}, 3) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad c(\hat{\mathcal{T}}, 4) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{4.1}$$

We can map from an arbitrary tetrahedron $\mathcal{T}$ to the reference element by using a linear-affine mapping. First, following [164] we define the mapping matrix

$$\mathbf{\Gamma}(\mathcal{T}) = \begin{pmatrix} | & | & | \\ c(\mathcal{T}, 2) - c(\mathcal{T}, 1) & c(\mathcal{T}, 3) - c(\mathcal{T}, 2) & c(\mathcal{T}, 4) - c(\mathcal{T}, 3) \\ | & | & | \end{pmatrix}. \tag{4.2}$$

Next, we use this in the mappings

$$\begin{aligned} \Xi^{-1}(\mathcal{T}, \boldsymbol{\xi}) &= \mathbf{\Gamma}(\mathcal{T})\boldsymbol{\xi} + c(\mathcal{T}, 1), \\ \Xi(\mathcal{T}, \boldsymbol{x}) &= (\mathbf{\Gamma}(\mathcal{T}))^{-1} (\boldsymbol{x} - c(\mathcal{T}, 1)), \end{aligned} \tag{4.3}$$

where $\Xi(\mathcal{T}, \boldsymbol{x})$ maps the coordinates of a point $\boldsymbol{x}$ that is in a tetrahedron $\mathcal{T}$ to the reference coordinate $\boldsymbol{\xi} = (\xi, \eta, \zeta)$.[1] The mapping $\Xi^{-1}(\mathcal{T}, \boldsymbol{\xi})$ is the inverse mapping, i.e., it takes a reference coordinate and transforms it into the global coordinates.

[1]It should be clear from the context whether η refers to the displacement or the reference coordinates.

Using the transformations and their Jacobians for our mapping, we can transform integrals over the physical cells to integrals over a reference cell

$$\int_T f(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_{\hat{T}} f(\Xi^{-1}(\boldsymbol{\xi})) |J| \, \mathrm{d}\xi, \tag{4.4}$$

which uses the determinant of the Jacobian of the mapping $J = \det(\boldsymbol{\Gamma})$, which is constant throughout the element. J is equal to six times the volume of the tetrahedron $\mathcal{T}$.

The derivatives transform as

$$\begin{aligned}
\frac{\partial}{\partial x} &= \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial x}\frac{\partial}{\partial \zeta}, \\
\frac{\partial}{\partial y} &= \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial y}\frac{\partial}{\partial \zeta}, \\
\frac{\partial}{\partial z} &= \frac{\partial \xi}{\partial z}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial z}\frac{\partial}{\partial \eta} + \frac{\partial \zeta}{\partial z}\frac{\partial}{\partial \zeta},
\end{aligned} \tag{4.5}$$

which is a direct result of the chain rule and the definition of our mapping (equation (4.3)) [35].

Furthermore, as shown in 4.1b, we define the reference triangle ε_2 with local coordinates $\boldsymbol{\chi} = (\chi_1, \chi_2)$. It has the vertices $\{(0,0),(1,0),(0,1)\}$. We define the function Ψ^f that maps coordinates on the reference triangle to coordinates on the fth face of a tetrahedron $\mathcal{T}$. With this, we can express integrals over the surface $\partial\mathcal{T}$ of a tetrahedron by

$$\int_{\partial\mathcal{T}} f(\boldsymbol{x}) \, \mathrm{d}S = \sum_f \int_{\varepsilon_2} f(\Psi^f(\boldsymbol{\chi})) \underbrace{\left\| \frac{\partial \Psi^f}{\partial \chi_1} \times \frac{\partial \Psi^f}{\partial \chi_2} \right\|}_{=:|S_f|} \mathrm{d}\boldsymbol{\chi} \tag{4.6}$$

where $|S_f|$ is twice the area of the surface [164]. In addition, we need the mapping $\xi^f = \Xi(\Psi^f(\boldsymbol{\chi}))$ that maps coordinates on the reference triangle to coordinates on the fth face of our reference tetrahedron $\hat{\mathcal{T}}$. Following [35, 164], this function does not depend on the geometry of the element and is given by table 4.1.

When integrating over the boundary of a tetrahedron, we often need to integrate over the face from the perspective of the neighboring tetrahedron. For this, we need to consider the different orientations of this face relative to the tetrahedra. As discussed in [35, 164], this can be summarized by the function $\tilde{\chi}^h$, where h is an index that takes the three possible rotations into account. Table 4.2 defines this function.

This section discussed how to map from physical triangles and tetrahedra, out of which our mesh is composed, to reference triangles and tetrahedra. With these, we can transform integrals over our element and its surface to integrals over the reference elements.

4.1.2. Basis Functions

In this subsection, we discuss the three types of basis functions that we need for our numerical scheme. We consider modal bases for tetrahedra and triangles. Additionally,

Table 4.1.: Mapping from the reference triangle coordinates to the reference tetrahedral coordinate system. Modified from [164].

	ξ	η	ζ
$\xi^0(\boldsymbol{\chi})$	χ_2	χ_1	0
$\xi^1(\boldsymbol{\chi})$	χ_1	0	χ_2
$\xi^2(\boldsymbol{\chi})$	0	χ_2	χ_1
$\xi^3(\boldsymbol{\chi})$	$1-\chi_1-\chi_2$	χ_1	χ_2

Table 4.2.: Mapping from the coordinates of a triangle to the coordinates from the triangle in the other face. This deals with possible rotations. Table taken from [35, 164].

$\boldsymbol{\chi}$	$\tilde{\chi}^0(\boldsymbol{\chi})$	$\tilde{\chi}^1(\boldsymbol{\chi})$	$\tilde{\chi}^2(\boldsymbol{\chi})$
$\tilde{\chi_1}$	χ_2	$1-\chi_1-\chi_2$	χ_1
$\tilde{\chi_2}$	χ_1	χ_2	$1-\chi_1-\chi_2$

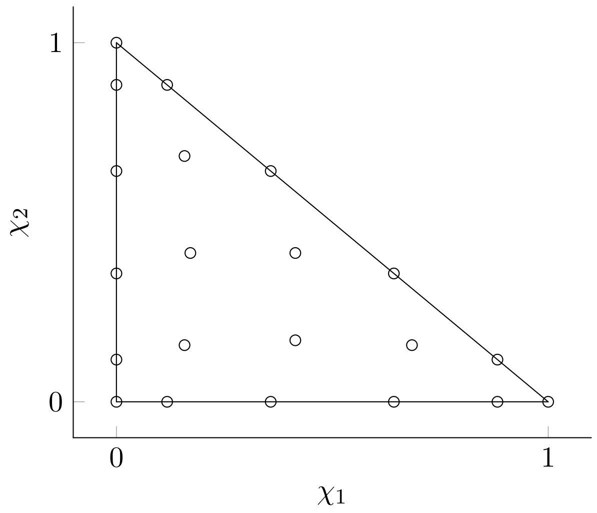


Figure 4.2.: Warp-and-blend nodes on our reference triangle ε_2, as described in [61]. Here, we show nodes for a nodal basis of polynomial order of five.

we introduce a nodal basis for triangles. All of these bases have in common that they are not continuous across elements, i.e., they are only defined within one element. This is the reason why the Discontinuous Galerkin method is called discontinuous. For convenience, we define all bases on the reference elements.

First, we discuss the modal bases. For both triangles and tetrahedra, we use the modal Dubiner basis. It is based on Jacobi polynomials and consists of orthogonal polynomials. Their construction is explained in [20]. The version that we are using is tabled in [28, Appendix A]. We call the basis ϕ_k for our reference tetrahedron and ψ_k for the reference triangle. For a polynomial order of N we have $1/2 N(N+1)$ and $1/6 N(N+1)(N+2)$ basis functions for triangles and tetrahedra respectively.

For example, we can expand the solution $\boldsymbol{q}$ on a tetrahedron $\mathcal{T}_m$ as

$$q_p^m(\boldsymbol{x}, t) = \underline{\mathrm{q}}_{lp}^m(t)\phi_l(\Xi(\mathcal{T}_m, \boldsymbol{x})) \tag{4.7}$$

using the mapping from a physical element to the reference element, which is required because the basis ϕ_l is defined only on the reference element. This introduced the index l for the basis function and the index p for the quantity. The vector of coefficients $\underline{\mathrm{q}}^m(t)$ is defined for each element and depends on time but not on space. We mark all coefficients with an underbar. In the following, we drop the index for the element for convenience. As we always integrate over one single element, its index should be clear from the context. In addition to the expansion on tetrahedra (equation (4.7)), we can also expand quantities that are defined on the reference triangle in the two-dimensional basis

$$f_p^m(\boldsymbol{\chi}, t) = \underline{\mathrm{f}}_{lp}(t)\psi_l(\boldsymbol{\chi}). \tag{4.8}$$

As is well known (see e.g. [122, Cha. 10]), we can approximate any function $f(x)$ by expanding it in a basis of orthogonal polynomials. This is called the generalized Fourier series. We typically truncate the series after $N+1$ terms. For example, for our orthogonal modal tetrahedral basis, we can write

$$f(\boldsymbol{\xi}) \approx \underline{\mathrm{f}}_l \phi_l(\boldsymbol{\xi}), \tag{4.9}$$

where we compute the coefficients

$$\underline{\mathrm{f}}_l = \frac{\int_{\hat{\mathcal{T}}} f(\boldsymbol{\xi})\phi_l(\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi}}{\int_{\hat{\mathcal{T}}} \phi_l(\boldsymbol{\xi})\phi_l(\boldsymbol{\xi}) \,\mathrm{d}\boldsymbol{\xi}} \tag{4.10}$$

by a least-squares projection. This gives us the best (on average in the Euclidean norm) approximation with our chosen basis. The same is also possible for functions defined on our reference triangle, leading us to the expansion

$$f(\boldsymbol{\chi}) \approx \underline{\mathrm{f}}_l \psi_l(\boldsymbol{\chi}), \quad \underline{\mathrm{f}}_l = \frac{\int_{\varepsilon_2} f(\boldsymbol{\chi})\phi_l(\boldsymbol{\chi}) \,\mathrm{d}\boldsymbol{\chi}}{\int_{\varepsilon_2} \phi_l(\boldsymbol{\chi})\phi_l(\boldsymbol{\chi}) \,\mathrm{d}\boldsymbol{\chi}}. \tag{4.11}$$

Some boundary conditions presented in this thesis make it necessary to prescribe a space-dependent state at the boundary, i.e., on a triangle. As the coefficients in the modal basis do not correspond to specific locations in space, we need to project a function to this basis. An alternative and more convenient perspective is describing this function directly in a nodal basis. For this, we expand a function as

$$f_p^m(\boldsymbol{\chi}, t) = \underline{\hat{\mathrm{f}}}_{lp}(\boldsymbol{n_l}, t) L_l(\boldsymbol{\chi}), \tag{4.12}$$

where $\boldsymbol{n}$ is a set of interpolation nodes and L_l is the lth Lagrange polynomial on the triangle. We have the same number of nodes as we have modal basis functions. This setting is very convenient, as the coefficients $\underline{\hat{\mathrm{f}}}$ depend on the space coordinates. Hence, to interpolate a function, we can use the point-wise value of our function at the interpolation nodes directly as coefficients. However, this type of basis opens two questions: Which nodes do we choose, and how do we define the functions L_l?

Choosing a suitable set of nodes is essential because equidistant points lead to the famous Runge phenomenon, manifesting in strong oscillations and a correspondingly large interpolation error [122]. Furthermore, this choice would lead to ill-conditioned operators for higher orders. Hence. we choose the warp-and-blend nodes described in [61, Sec. 6.1]. They can be easily pre-computed and lead to well-behaved interpolation. Figure 4.2 shows the resulting nodes for $N = 5$.

Finally, we need to define the Lagrange basis on our triangle. However, there is no known analytical expression for it on triangles. We need to take a detour and use the modal basis to evaluate the nodal basis. The nodal and modal bases span the same function space. Hence, we can equate both expansions (equations (4.8) and (4.12)). Then, we can compute the nodal coefficients from the modal coefficients by

$$\underline{\hat{\mathrm{f}}}_{lp} = V_{ln} \underline{\mathrm{f}}_{np}, \tag{4.13}$$

where the Vandermonde matrix

$$V_{ln} = \psi_n(\boldsymbol{n_l}) \tag{4.14}$$

evaluates the modal basis at the nodes defined by the nodal basis. As this matrix is invertible, we can use

$$\underline{\mathrm{f}}_{ln} = V_{ln}^{-1} \underline{\hat{\mathrm{f}}}_{np} \tag{4.15}$$

to compute the modal coefficients from the nodal representation [61] on triangles. This gives us a way of evaluating the two-dimensional Lagrange polynomials: We use equation (4.15) to compute a modal representation and then evaluate the resulting expansion (equation (4.8)) at new nodes.

We defined a modal basis for the reference tetrahedron and the reference triangle in this section. Additionally, we introduced a nodal basis on triangles and demonstrated how to convert between the modal and nodal. Furthermore, we showed how we can approximate functions in all bases.

4.1.3. Weak Form

In this section, we use the results from the previous sections to take the first steps to discretize our PDEs. We begin with a general linear variable coefficient PDE

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{A}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial x} - \boldsymbol{B}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial y} - \boldsymbol{C}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial z}. \tag{2.29 revisited}$$

In this section, we only consider a single tetrahedron. We assume that the material and therefore the flux matrices do not vary in this element.

We multiply equation (2.29) by a test function ϕ_k and integrate over a tetrahedron $\mathcal{T}$, leading to

$$\int_{\mathcal{T}} \phi_k \frac{\partial \boldsymbol{q}}{\partial t}\,\mathrm{d}\boldsymbol{x} = -\int_{\mathcal{T}} \phi_k \left(\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x} + \boldsymbol{B}\frac{\partial \boldsymbol{q}}{\partial y} + \boldsymbol{C}\frac{\partial \boldsymbol{q}}{\partial z}\right)\mathrm{d}\boldsymbol{x}. \tag{4.16}$$

Note that we have as many test functions as we have basis functions. In our case, we use the same function space that we used to expand the numerical solution (equation (4.7)). This is called the Galerkin approach. For the sake of brevity, here and in the following, we neglect the space and time dependency of the solution and the basis functions if it is clear from the context. As the basis functions depend on the reference element, we must use equation (4.3) to map from the tetrahedron to the reference tetrahedron. Then, we apply integration by parts to transfer the space derivatives to the test function, resulting in

$$\int_{\mathcal{T}} \phi_k \frac{\partial \boldsymbol{q}}{\partial t}\,\mathrm{d}\boldsymbol{x} = \left(\int_{\mathcal{T}} \frac{\partial \phi_k}{\partial x}\boldsymbol{A}\boldsymbol{q} + \frac{\partial \phi_k}{\partial y}\boldsymbol{B}\boldsymbol{q} + \frac{\partial \phi_k}{\partial z}\boldsymbol{C}\boldsymbol{q}\,\mathrm{d}\boldsymbol{x}\right) - \int_{\partial\Omega} \phi_k \boldsymbol{f}^{*}\,\mathrm{d}S \tag{4.17}$$

Equation (4.17) is called the weak form, as it relaxes the smoothness requirements on $\boldsymbol{q}$. It now allows solutions that are not differentiable, i.e., that only fulfill the PDE in the weak sense. We refer the interested reader to the discussion in [95]. The weak form includes an integral over the surface $\partial\Omega$ of our element. Our basis is only piece-wise continuous, so the solution on the faces shared with other elements is not uniquely defined. Hence, we introduce the numerical flux

$$\boldsymbol{f}^{*} = n_x \boldsymbol{A}\boldsymbol{q} + n_y \boldsymbol{B}\boldsymbol{q} + n_z \boldsymbol{C}\boldsymbol{q}, \tag{4.18}$$

where $\boldsymbol{n}^T = (n_x, n_y, n_z)$ is the outer normal of the face.

We first deal with the volume integrals in equation (4.17). For this, moving our computations to the reference basis is convenient. We start by defining the flux matrices on the reference elements. They are given by

$$\begin{aligned}
A^{*}_{pq} &= A_{pq}\frac{\partial \xi}{\partial x} + B_{pq}\frac{\partial \xi}{\partial y} + C_{pq}\frac{\partial \xi}{\partial z},\\
B^{*}_{pq} &= A_{pq}\frac{\partial \eta}{\partial x} + B_{pq}\frac{\partial \eta}{\partial y} + C_{pq}\frac{\partial \eta}{\partial z},\\
C^{*}_{pq} &= A_{pq}\frac{\partial \zeta}{\partial x} + B_{pq}\frac{\partial \zeta}{\partial y} + C_{pq}\frac{\partial \zeta}{\partial z},
\end{aligned} \tag{4.19}$$

i.e., they are linear combinations of the flux matrices in the x, y, z coordinate system, which follows directly from equation (4.5) [35]. These matrices are defined for each element, as the derivatives of the reference coordinate with respect to the global coordinates depend on the tetrahedron.

We expand the numerical solution with the expansion given by equation (4.7) in our modal basis. We transform all volume integrals to the reference tetrahedron with equations (4.4) and (4.5). After moving constant terms out of the integrals, this leads us to

$$\begin{aligned} |J| \frac{\partial \underline{q}_{lp}}{\partial t} \int_{\hat{\mathcal{T}}} \phi_k \phi_l \, \mathrm{d}\boldsymbol{\xi} = &- \int_{\partial \mathcal{T}} \phi_k \boldsymbol{f}^* \, \mathrm{d}S \\ &+ A^*_{pq} \, \underline{q}_{lq} |J| \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \xi} \phi_l \, \mathrm{d}\boldsymbol{\xi} \\ &+ B^*_{pq} \, \underline{q}_{lq} |J| \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \eta} \phi_l \, \mathrm{d}\boldsymbol{\xi} \\ &+ C^*_{pq} \, \underline{q}_{lq} |J| \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \zeta} \phi_l \, \mathrm{d}\boldsymbol{\xi}. \end{aligned} \tag{4.20}$$

Here, we can directly identify the mass matrix

$$M_{kl} = \int_{\hat{\mathcal{T}}} \phi_k \phi_l \, \mathrm{d}\boldsymbol{\xi} \tag{4.21}$$

and the stiffness matrices

$$\begin{aligned} K^{\xi}_{kl} &= \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \xi} \phi_l \, \mathrm{d}\boldsymbol{\xi}, \\ K^{\eta}_{kl} &= \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \eta} \phi_l \, \mathrm{d}\boldsymbol{\xi}, \\ K^{\zeta}_{kl} &= \int_{\hat{\mathcal{T}}} \frac{\partial \phi_k}{\partial \zeta} \phi_l \, \mathrm{d}\boldsymbol{\xi}, \end{aligned} \tag{4.22}$$

which can be precomputed. The mass matrix is diagonal due to the orthogonality of the Dubiner basis and hence it can be trivially inverted. Both mass matrix and stiffness matrices are defined on the reference element and thus can be pre-computed.

We have now covered the efficient computation of all volume integrals. For this, we used our careful definition of the reference elements and modal basis.

4.1.4. Surface Terms

In this section, we discuss how to compute the surface integral

$$\int_{\partial \mathcal{T}} \phi_k \boldsymbol{f}^* \, \mathrm{d}S \tag{4.23}$$

in equation (4.17). We expand it as a sum over all faces of our tetrahedron $\mathcal{T}$, denoted by the set $\mathcal{F}$, and apply equation (4.6) to describe the resulting face integrals as integrals over the reference triangle ε_2, leading us to

$$\int_{\partial \mathcal{T}} \phi_k \boldsymbol{f}^* \,\mathrm{d}S = \sum_{f \in \mathcal{F}} \int_{\varepsilon_2} \phi_k(\xi^f(\boldsymbol{\chi})) \boldsymbol{f}^*(\Psi^f(\boldsymbol{\chi}))\, |S_f| \,\mathrm{d}\boldsymbol{\chi}. \tag{4.24}$$

Equation (4.18) introduced the numerical flux $\boldsymbol{f}^*$, which is the only mechanism that connects neighboring elements. Hence, it weakly enforces the continuity between elements and is thus a crucial part of the DG method. The numerical flux depends on the value of the flux matrices and the state $\boldsymbol{q}$ at the boundary. However, neither is uniquely defined, as we allow the material and basis functions to be discontinuous between elements. In this work, we use the Godunov flux, which we compute by multiplying the star state, i.e., the solution of the Riemann problem (definition 1), with by the flux matrix in normal direction.[2] We use the Riemann solver discussed in chapter 3. Equation (3.89) gives us a way of computing the solution of the plane-wave Riemann problem (definition 1) by matrix-vector multiplications. We can combine this with lemma 1, which states the rotational invariance of the flux matrix, leading us to

$$\boldsymbol{f}^* = \underbrace{\boldsymbol{\mathcal{T} A G^L \mathcal{T}}^{-1}}_{=:\boldsymbol{A^{*Lf}}} \boldsymbol{q^L} + \underbrace{\boldsymbol{\mathcal{T} A G^R \mathcal{T}}^{-1}}_{=:\boldsymbol{A^{*Rf}}} \boldsymbol{q^R}, \tag{4.25}$$

where the index f corresponds to the face, which is necessary because the rotation matrices depend on its orientation. Equation (4.25), similar to equation (3.89), reduces to the sum of two matrix-vector products. We use the convention in the following that the current element is left with state $\boldsymbol{q^L}$, and the neighboring element is right with state $\boldsymbol{q^R}$. Conveniently, the numerical flux matrices $\boldsymbol{A^{*Lf}}$ and $\boldsymbol{A^{*Rf}}$ can be pre-computed and depend only on the orientation of the interface and the material properties.

Furthermore, equation (4.25) allows us to split the flux computation into two parts: the local integration, which requires contributions only from the current element, and the neighboring integration, which requires contributions only from the neighboring element. Additionally, we split the neighboring part into a sum over interior ($\mathcal{F}^{\text{int}}$) and exterior ($\mathcal{F}^{\text{ext}}$) faces, i.e., faces at the boundary. We apply this to equation (4.24) and insert our numerical flux (equation (4.25)). After expanding the numerical solution in our basis, we arrive at

$$\sum_{f \in \mathcal{F}} \left(|S_f|\, A^{*Lf}_{pq} \underline{q}_{lq} \underbrace{\int_{\varepsilon_2} \phi_k(\xi^f(\boldsymbol{\chi})) \phi_l(\xi^f(\boldsymbol{\chi}))\,\mathrm{d}\boldsymbol{\chi}}_{=:F^{Lf}_{kl}} \right) + \tag{4.26}$$

$$\sum_{f \in \mathcal{F}^{\text{int}}} \left(|S_f|\, A^{*Rf}_{pq} \underline{q}^f_{lq} \underbrace{\int_{\varepsilon_2} \phi_k(\xi^f(\boldsymbol{\chi})) \phi_l(\xi^g(\tilde{\chi}^h(\boldsymbol{\chi})))\,\mathrm{d}\boldsymbol{\chi}}_{=:F^{Rfgh}_{kl}} \right) + \tag{4.27}$$

[2]This is why the solution of the Riemann problem is often also called the Godunov state.

$$\sum_{f \in \mathcal{F}^{\text{ext}}} \left(|S_f| \, A^{*Rf}_{pq} \hat{\underline{q}}_{lq} \underbrace{\int_{\mathcal{E}_2} \phi_k(\xi^f(\boldsymbol{\chi})) L_l(\boldsymbol{\chi}) \, \mathrm{d}\boldsymbol{\chi}}_{=:\hat{F}^{Rf}_{kl}} \right). \tag{4.28}$$

Here, the indices g and h consider the face-local coordinates of the face in the neighboring element with orientation h. We denote the numerical solution in the neighboring element that shares the face f with our element by $\underline{\boldsymbol{q}}^f$. We can pre-compute the integrals, as we did for the volume terms.

Following [168], we use the change of basis

$$\phi_k(\xi^f(\boldsymbol{\chi})) = R^f_{kl} \psi_l(\boldsymbol{\chi}), \tag{4.29}$$

which changes from the tetrahedral modal basis evaluated at the face to the face modal basis (equation (4.8)).

We can define the matrix $\boldsymbol{R^f}$ that converts the coefficients from the volume basis to the face basis. Using a least-squares projection, as defined by equation (4.11), leads to

$$R^f_{kl} = \frac{\int \phi_l(\xi^f(\boldsymbol{\chi})) \psi_k(\boldsymbol{\chi}) \, \mathrm{d}\boldsymbol{\chi}}{\int \psi_k(\boldsymbol{\chi}) \psi_k(\boldsymbol{\chi}) \, \mathrm{d}\boldsymbol{\chi}}. \tag{4.30}$$

In our cases, as the face basis is of the same degree as the volume basis, this projection is exact [164, Lemma 4].

Next, we use this to compute our surface integral matrices. We follow the notation from [164]. We start with the integral for the local contributions used in equation (4.26). It can be computed as

$$F^{Lf}_{kl} = \int_{\mathcal{E}_2} \phi_k(\xi^f(\boldsymbol{\chi})) \phi_l(\xi^f(\boldsymbol{\chi})) \, \mathrm{d}\boldsymbol{\chi} = R^f_{kn} R^f_{lm} \underbrace{\int_{\mathcal{E}_2} \psi_n(\boldsymbol{\chi}) \psi_m(\boldsymbol{\chi}) \, \mathrm{d}\boldsymbol{\chi}}_{=:M^2_{nm}} \tag{4.31}$$

For the neighboring flux, given by equation (4.27), we apply the same principle and arrive at

$$F^{Rfgh}_{kl} = \int_{\mathcal{E}_2} \phi_k(\xi^f(\boldsymbol{\chi})) \phi_l(\xi^g(\tilde{\chi}^h(\boldsymbol{\chi}))) \, \mathrm{d}\boldsymbol{\chi} = R^f_{kn} R^g_{lm} \int_{\mathcal{E}_2} \psi_n(\boldsymbol{\chi}) \psi_m(\tilde{\chi}^h(\boldsymbol{\chi})) \, \mathrm{d}\boldsymbol{\chi} \tag{4.32}$$

For high orders, it is more efficient to not pre-multiply the matrices [168]. Furthermore, we only need to consider four versions of the face projection matrices (one for each face), one face mass matrix and three face configuration matrices. This replaces up to $(3 \times 4 \times 4 = 48)$ different versions of F^{Rfgh} and thus leads to better cache behavior [168]. We believe this is also a more natural approach, as quantities defined on a face are expanded in a basis that is defined on this face.

In addition, we need to incorporate boundary conditions through surface integrals. We treat them as local contributions. As discussed before, it is often easier to prescribe these boundary values in a nodal basis. As we have learned in the discussion of the modal face basis, it suffices to only consider a face basis for the surface terms. This allows us to avoid defining a nodal basis for volume data. Assuming that we have boundary values that are defined on a face in the nodal basis, we can expand the integral in equation (4.28) as

$$\hat{F}^{Rf}_{kl} = \int_{\varepsilon_2} \phi_k(\xi(\boldsymbol{\chi})) L_l(\boldsymbol{\chi}) \, \mathrm{d}\boldsymbol{\chi} = R^f_{kn} M^2_{nm} V_{ml}, \tag{4.33}$$

which is essentially the same as the local flux matrix (equation (4.31)) but with an additional Vandermonde matrix that converts the nodal into modal representation, as defined in equation (4.15). Hence, we only need to project the test function to the triangle basis. In contrast to the neighboring flux matrix used for interior neighbors (equation (4.32)), we do not need to consider the face parametrization, simplifying the implementation. We set $\underline{\hat{\boldsymbol{q}}}$ to correspond to the state of a ghost element such that the solution of the Riemann problem results in the correct state at the boundary. In section 3.4 and section 3.5, we derived these states for boundary conditions for acoustic and elastic materials. As the boundary conditions typically depend on the numerical solution of our element, we convert it to a nodal basis using the Vandermonde matrix given by equation (4.13). Furthermore, boundary conditions are typically stated directly in the face-aligned coordinate system. Hence, we use the rotation matrix defined by equation (3.12) to rotate the degrees of freedom from the global coordinate system to the face-aligned coordinate system and vice versa.

In previous works, the boundary conditions were imposed in the modal basis or by modifying the numerical flux directly [35, 168]. For example, the absorbing boundary condition is obtained by setting the incoming numerical flux from the neighboring elements to zero [35]. The dynamic rupture source is treated as an internal boundary condition, which uses the state of two neighboring elements. A discussion of the treatment of dynamic rupture boundary conditions is out of the scope of this thesis. We refer the interested reader to the detailed discussion in [27, 118, 164].

We derived a discretization of the surface integral in this section. We used the exact Riemann solver (chapter 3) to couple elements and include boundary conditions using a nodal numerical flux.

4.1.5. Summary

Now we can put everything together by combining the volume discretization (equation (4.20)) with the surface discretization (equations (4.26) to (4.28)). For a single element, we arrive at the space-discretization of

$$\begin{aligned}
|J| \frac{\partial \underline{\mathrm{q}}_{lp}}{\partial t} M_{kl} = {} & A^*_{pq}\, \underline{q}_{lq} |J| K^{\xi}_{kl} + B^*_{pq}\, \underline{q}_{lq} |J| K^{\eta}_{kl} + C^*_{pq}\, \underline{q}_{lq} |J| K^{\zeta}_{kl} \\
& - \sum_{f \in \mathcal{F}} \left(|S_f| \, A^{*Lf}_{pq} \underline{\mathrm{q}}^f_{lq} F^{Lf}_{kl} \right) - \sum_{f \in \mathcal{F}^{\text{int}}} \left(|S_f| \, A^{*Rf}_{pq} \underline{\mathrm{q}}_{lq} F^{Rfgh}_{kl} \right) \\
& - \sum_{f \in \mathcal{F}^{\text{ext}}} \left(|S_f| \, A^{*Rf}_{pq} \underline{\hat{\mathrm{q}}}_{lq} \hat{F}^{Rf}_{kl} \right),
\end{aligned} \tag{4.34}$$

which allows us to pre-compute most matrices. These are defined by equations (4.21), (4.22), and (4.31) to (4.33). This results in a high-order discretization in space, where a numerical flux couples elements. Boundary conditions are weakly enforced. Hence, the DG discretization combines the high-polynomial order of finite element methods with the numerical flux of finite volume methods [61].

4.2. ADER

In this section, we discuss the ADER time-stepping method, which is a predictor-corrector scheme [156, 157, 159]. In the first step, the predictor evolves the PDE locally in one element by expanding the numerical solution as a Taylor series in time. In the second step, the corrector, we time-integrate our space discretization, which couples elements through surface integrals, and insert the Taylor expansion. This results in a high-order method in both space and time.

4.2.1. Cauchy-Kowalevski Procedure

The predictor gives us a solution of our PDE "in the small" [55], i.e., only considering one element without its neighbors. [47] gives an overview of possible local predictors. For linear problems without source terms, the most popular predictors lead to the same result. This section follows the approach taken by [35], which develops the solution locally as a Taylor series and computes time derivatives by manipulating the PDE directly. It is called the Cauchy-Kowalevski procedure.

We begin by reminding ourselves of the order N Taylor-expansion of a vector-valued function $\boldsymbol{q}$ in time, expanded around t_0, which is given by

$$\boldsymbol{q}(t) \approx \sum_{i=0}^{N} \frac{(t-t_0)^i}{i!} \frac{\partial^i \boldsymbol{q}}{\partial t^i}(t_0). \tag{4.35}$$

In our case, $\boldsymbol{q}$ is the solution of our PDE (equation (2.29)). Hence, it is not trivial to compute its time derivatives. Let us first look at the continuous case in one dimension. We start with the PDE

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x}. \tag{4.36}$$

Differentiating by t leads us to

$$\begin{aligned}
\frac{\partial^2 \boldsymbol{q}}{\partial t^2} &= \frac{\partial}{\partial t}(-\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x}) \\
&= \frac{\partial}{\partial x}(-\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial t}) \\
&= \frac{\partial}{\partial x}(-\boldsymbol{A}(-\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x})) \\
&= \boldsymbol{A}^2 \frac{\partial^2 \boldsymbol{q}}{\partial x^2}
\end{aligned}, \tag{4.37}$$

where we used the definition of the PDE (equation (4.36)) to replace the time derivative of $\boldsymbol{q}$ with a space derivative. For derivatives of order i, we arrive at

$$\frac{\partial^i \boldsymbol{q}}{\partial t^i} = (-1)^i \boldsymbol{A}^i \frac{\partial^i \boldsymbol{q}}{\partial x^i}. \tag{4.38}$$

We can do the same thing for the discrete solution. As a detailed derivation can be found in [35, 164], we only present a quick derivation. We follow the approach of [35]. The idea is the same that we demonstrated for the continuous case. First, we approximate the ith time derivative of the discrete solution in our basis

$$\frac{\partial^i q_p}{\partial t^i} \approx D^i_{lp} \phi_l \tag{4.39}$$

where we use the numerical solution as the zeroth derivative

$$D^0_{lp} = \underline{\mathrm{q}}_{lp}. \tag{4.40}$$

We use this to expand our degrees of freedom in time around $t = t_0$,

$$\underline{\mathrm{q}}_{lp}(t) \approx \sum_{i=0}^{N} \frac{(t-t_0)^i}{i!} D^i_{lp}, \tag{4.41}$$

similar to equation (4.35). This leaves us with the question: How can we compute the coefficients of the time derivatives?

The first step is to consider our PDE in the reference coordinate frame. This can be done by using the chain rule (equation (4.5)) and the element-local flux matrices (equation (4.19)), leading us to

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{A}^* \frac{\partial \boldsymbol{q}}{\partial \xi} - \boldsymbol{B}^* \frac{\partial \boldsymbol{q}}{\partial \eta} - \boldsymbol{C}^* \frac{\partial \boldsymbol{q}}{\partial \zeta}. \tag{4.42}$$

Next, we differentiate in time recursively, arriving at

$$\frac{\partial^i \boldsymbol{q}}{\partial t^i} = \left(-\boldsymbol{A}^* \frac{\partial}{\partial \xi} - \boldsymbol{B}^* \frac{\partial}{\partial \eta} - \boldsymbol{C}^* \frac{\partial}{\partial \zeta} \right) \frac{\partial^{i-1} \boldsymbol{q}}{\partial t^{i-1}}, \tag{4.43}$$

which gives us the solution for the ith order derivative in time. The second step is expanding the derivative $\frac{\partial^{i-1}\boldsymbol{q}}{\partial t^{i-1}}$ in our basis

$$\frac{\partial^i q_p}{\partial t^i} = \left(-A^*_{pq} \frac{\partial}{\partial \xi} - B^*_{pq} \frac{\partial}{\partial \eta} - C^*_{pq} \frac{\partial}{\partial \zeta} \right) D^i_{ql} \phi_l \tag{4.44}$$

Finally, in the third step, we compute the coefficient with a least-squares projection (equation (4.10)), resulting in

$$\left(\int_{\hat{\mathcal{T}}} \phi_l \phi_k \,\mathrm{d}\boldsymbol{\xi} \right) D^i_{lp} \approx \int_{\hat{\mathcal{T}}} \phi_k \left(-A^*_{pq} \frac{\partial}{\partial \xi} - B^*_{pq} \frac{\partial}{\partial \eta} - C^*_{pq} \frac{\partial}{\partial \zeta} \right) D^{i-1}_{ql} \phi_l \,\mathrm{d}\boldsymbol{\xi}. \tag{4.45}$$

Here, we recognize the mass and stiffness matrices. Simplifying, we arrive at the recursive scheme

$$M_{kl} D^{i}_{lp} = K^{\xi}_{lk} D^{i-1}_{lq} A^{*}_{pq} + K^{\eta}_{lk} D^{i-1}_{lq} B^{*}_{pq} + K^{\zeta}_{lk} D^{i-1}_{lq} C^{*}_{pq}, \tag{4.46}$$

which can be computed efficiently because the matrices can be pre-computed. Due to the sparsity of the involved matrices, the coefficient vectors D^i also become increasingly sparse for higher order derivatives [15], which is unsurprising, as differentiating a polynomial leads to a lower-order polynomial. SeisSol's implementation automatically tunes the kernels for the resulting sparsity pattern [166]. Finally, equations (4.41) and (4.46) allow us to expand the numerical solution in time without considering neighboring elements.

4.2.2. One-Step Update

Now, we combine the Taylor expansion of the numerical solution with our space-discretization (section 4.1.3) to create a fully discrete high-order method. We consider a time step between $t = t_0$ and $t = t_1$. As we use an explicit time-stepping method, we must adhere to the Courant-Friedrichs-Lewy (CFL) condition [22]. In our case, each element has a time step size of

$$\Delta t \leq C(N) h (|\lambda^{\max}|)^{-1}, \tag{4.47}$$

where h is the insphere of the tetrahedron and $\lambda^{\max}$ is the maximum eigenvalue of the flux matrix (i.e., the fastest wave speed) [35]. The constant $C(N) \leq (2N+1)^{-1}$ depends on the polynomial order N. A necessary but not sufficient condition for it is given by the von Neumann stability analysis conducted in [34]. Note that this stability analysis assumes periodic boundary conditions and a simpler homogeneous advection PDE without any sources. Hence, the time step size can be even more restricted in practice.

Integrating a Taylor series (equation (4.35)), expanded at $t = t_e$, between t_0 and t_1 leads to

$$\int_{t_0}^{t_1} \boldsymbol{q}(t) \,\mathrm{d}t \approx \sum_{i=0}^{N} \frac{(t_1 - t_e)^{i+1} - (t_0 - t_e)^{i+1}}{(i+1)!} \frac{\partial^i \boldsymbol{q}}{\partial t^i}(t_e). \tag{4.48}$$

The time step size (equation (4.47)) can differ between elements due to their size or material. Fortunately, the ADER scheme can handle this elegantly. First, we define the function

$$\mathcal{I}(t_0, t_1, t_e) = \sum_{i=0}^{N} \frac{(t_1 - t_e)^{i+1} - (t_0 - t_e)^{i+1}}{(i+1)!} D^i(t_e), \tag{4.49}$$

which integrates the coefficient of our discrete solution (equation (4.41)), which was expanded in a Taylor series around the expansion point t_e, from time $t = t_1$ to $t = t_2$ in one element. This is the discrete equivalent of equation (4.48). The expansion point $t_e \leq t_1$ does not have to be identical to the start of the interval; however, we assume that

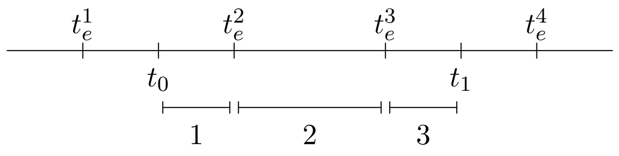


Figure 4.3.: Sub-intervals when integrating between times t_0 and t_1. We assume that we have the expansion points t_e^i. In this case, we need to consider three intervals. The first is the integration of the series with expansion point t_1^e from t_0 to t_e^2, given by the function $I(t_0, t_e^2, t_1^e)$. The second integral is computed by $I(t_e^2, t_e^3, t_e^2)$ and the third by $I(t_e^3, t_1, t_e^3)$.

the expansion point is before the start of the interval for reasons of causality. Due to the aforementioned element-dependent time step size, we need to consider the case that we have multiple Taylor series with different expansion points. For example, this can happen when we compute the surface integral (equation (4.27)), which requires information from neighboring elements with possibly different time step sizes.

We assume that the expansion points t_e^i are ordered such that $t_e^i < t_e^{i+1}$. The clamp function

$$\operatorname{clamp}(x, a, b) = \max(a, \min(x, b)), \tag{4.50}$$

gives the closest point to its input x that is in the interval $[a, b]$. We define a family of intervals indexed by i

$$(t_0^i, t_1^i) := (\operatorname{clamp}(t_e^i, t_0, t_1), \operatorname{clamp}(t_e^{i+1}, t_0, t_1)) \tag{4.51}$$

which gives us all partitions of our integration intervals with expansion points. The function

$$\mathcal{J}(t_0, t_1) = \int_{t_0}^{t_1} \underline{\boldsymbol{q}}(\boldsymbol{x}, t) \,\mathrm{d}t = \sum_i I(t_0^i, t_1^i, t_e^i) \tag{4.52}$$

integrates the coefficient of our numerical solution between t_0 and t_1 using all available expansion points. In practice, the integrals of intervals of length zero are not computed, as they have a value of zero. Figure 4.3 shows an example of the integration of an integral that contains multiple expansion points. Analogously, we use $\mathcal{J}^f(t_0, t_1)$ to denote the integral of the solution of the neighboring element connected by the face f.

Next, we integrate equation (4.34) from t_0 to t_1, leading to

$$\begin{aligned} |J| M_{kl} \mathrm{q}_{lp}(t_1) = {} & |J| M_{kl} \mathrm{q}_{lp}(t_0) \\ & + A^*_{pq}\, \mathcal{J}_{lq}(t_0, t_1) |J| K^\xi_{kl} + B^*_{pq}\, \mathcal{J}_{lq}(t_0, t_1) |J| K^\eta_{kl} + C^*_{pq}\, \mathcal{J}_{lq}(t_0, t_1) |J| K^\zeta_{kl} \\ & - \sum_{f \in \mathcal{F}} \left(|S_f|\, A^{*Lf}_{pq} \mathcal{J}_{lq}(t_0, t_1) F^{Lf}_{kl} \right) - \sum_{f \in \mathcal{F}^{\text{int}}} \left(|S_f|\, A^{*Rf}_{pq} \mathcal{J}^f_{lq}(t_0, t_1) F^{Rfgh}_{kl} \right) \\ & - \sum_{f \in \mathcal{F}^{\text{ext}}} \left(|S_f|\, A^{*Rf}_{pq} \left(\int_{t_0}^{t_1} \hat{\mathrm{q}}_{lq}(\tau) \,\mathrm{d}\tau \right) \hat{F}^{Rf}_{kl} \right). \end{aligned} \tag{4.53}$$

In this way, the time derivative of our coefficients vanishes, and we have achieved a fully discrete one-step update that can compute the solution coefficients at time t_1. This requires the inversion of the mass matrix, but, as mentioned before, this is trivial because it is diagonal.

We need to be careful with the nodal boundary condition in this framework. The coefficients of the nodal boundary condition can depend on the time evolution of the interior degrees of freedom. Additionally, some boundary conditions, for example, the velocity inlet, prescribe time-dependent functions. Hence, in our ADER scheme, they take the coefficients of the numerical solution, the coefficient of its derivative, and t as arguments. After the boundary condition has been computed in the time interval, we have to integrate it in time between t_0 and t_1. In the case of simple boundary conditions that linearly depend on the interior values, we can directly define the integral of the boundary values solely in terms of the time integral of the interior values.

Furthermore, dynamic earthquake rupture, as described in section 2.1.2, is implemented as an interior boundary condition. This adds complexity, as the numerical flux is no longer linear. Formally, the neighboring integral in equation (4.53) requires the solution of a Generalized Riemann Problem, i.e., solving a Riemann problem with piece-wise polynomial initial data. For linear numerical fluxes, we can replace it with a standard Riemann problem as the time integration and the Riemann solver commute in this case. As non-linear fluxes are out of the scope of this thesis, we refer the interested reader to [164, Sec. 3.3]. We can add a point source by using the formulation given by equation (2.35), integrating it in time, and adding it to equation (4.53)[72].[3]

To summarize, combining the Cauchy-Kowalevski expansion with our DG discretization leads to equation (4.53), a one-step update scheme with a high convergence order in time and space.

4.3. Gravitational Free Surface

We introduced the gravitational boundary condition in section 3.4.1. However, we have not yet discussed how we can compute it efficiently. Our fully discrete scheme, as stated by equation (4.53), requires the integral of the boundary values. This section presents a novel and efficient numerical scheme to integrate this boundary condition into our numerical scheme. The scheme is inspired by the ADER framework outlined in section 4.2. We consider a time step from t_0 to t_1.

We remind ourselves of the pressure at the boundary

$$p'(x, y, z, t) = \rho_0 g \eta(x, y, t) \qquad \text{at } z = 0 \tag{2.88 revisited}$$

and the displacement

$$\frac{\partial \eta}{\partial t} = v_3(x, y, z, t) \qquad \text{at } z = 0 \tag{2.89 revisited}$$

[3]It is not added to the predictor. This implementation style uses Godunov splitting, i.e., treating the PDE and source term separately. It can lead to a lower convergence order in time. Nevertheless, schemes of this type tend to work well in practice [95].

Together with the boundary state

$$\begin{pmatrix} p^* \\ v_1^* \end{pmatrix} = \begin{pmatrix} \rho g \eta \\ v_1^L + \frac{1}{Z}\left(p^L - \rho g \eta\right) \end{pmatrix} \tag{3.56 revisited}$$

this defines an ordinary differential equation (ODE) for η at the boundary, given by

$$\frac{\partial \eta}{\partial t} = v_1^* = v_1^L + \frac{1}{Z}\left(p^L - \rho g \eta\right) \tag{3.57 revisited}$$

As this is a linear ODE, we could solve it semi-analytically. This requires a time integral of the numerical velocity v_1^L and pressure p^L. Furthermore, as we need the time integral of η, we would need to use a nested quadrature rule, which is computationally expensive.

Another approach is integrating equation (3.57) with a numerical ODE solver, for example, a Runge-Kutta solver [97]. We can compute the integral of η (called H in the following) by the ODE

$$\begin{aligned} \frac{\partial H}{\partial t} &= \eta, \\ H &= 0 \quad \text{at } t = t_0, \\ \eta &= \eta_0 \quad \text{at } t = t_0, \end{aligned} \tag{4.54}$$

at the same time as computing η itself! We used this approach in our first implementation of this boundary condition [79]. However, it is very expensive, as we need to compute multiple Runge-Kutta stages for each time step. Each Runge-Kutta stage requires the computation of the values of u^L and p^L at multiple times and the projection of these variables at the boundary. A further downside is that this requires the implementation (and verification) of a different ODE solver for each polynomial order to achieve an economical scheme. Especially for high-orders (≥ 5), after the Butcher barrier has been breached, the number of required stages grows faster than the achieved order [19].

In this work, we take an alternative approach and compute the boundary condition with the ADER method. As we will see, this is not only truly of arbitrary order but also much more efficient than the Runge-Kutta approach. We expand the solution of our ODE as a Taylor-Series,

$$\eta(t) \approx \sum_{i=0}^{N} \frac{(t - t_0)^i}{i!} \frac{\partial^i \eta}{\partial t^i}(t_0), \tag{4.55}$$

similar to equation (4.35). We obtain the coefficients as in equation (4.38). Differentiating equation (3.57) in time, we arrive at

$$\frac{\partial^2 \eta}{\partial t^2} = \frac{\partial v^L}{\partial t} + \frac{1}{Z}\left(\frac{\partial p^L}{\partial t} - \rho g \frac{\partial \eta}{\partial t}\right). \tag{4.56}$$

Iteratively differentiating results in

$$
\begin{aligned}
\frac{\partial^0 \eta}{\partial t^0} &= \eta, \\
\frac{\partial^i \eta}{\partial t^i} &= \frac{\partial^i v^L}{\partial t^i} + \frac{1}{Z}\left(\frac{\partial^i p^L}{\partial t^i} - \rho g \frac{\partial^{i-1}\eta}{\partial t^{i-1}}\right),
\end{aligned}
\tag{4.57}
$$

which is a simple recursive computation.

This algorithm requires the time derivatives of v_1^L and p^L evaluated at the beginning of the time interval. They can be computed by the ADER scheme discussed earlier (equation (4.46)). As we use them for the time-stepping of our PDE, this requires no additional effort. However, we need to project them from the three-dimensional modal basis to the two-dimensional nodal face matrix, using the least-squares projection defined by equation (4.11) followed by the Vandermonde matrix given by equation (4.13). Note that this is cheaper than projecting the numerical solution, as we require fewer coefficients to store derivatives in our modal basis as they are of lower polynomial degree.

The expansion as a Taylor series has the advantage that it is trivial to compute the integral of η over the time step, as we can use equation (4.48) to integrate our series. We also need to keep track of the new displacement at the end of the time step, which we use as the initial displacement for the next time step. We can compute it by evaluating the Taylor series (equation (4.55)) at $t = t_1$. The recursion (equation (4.57)) for the ith derivative only depends on the coefficients of the $(i-1)$th derivative. Hence, we do not need to store all coefficients. Algorithm 1 shows a pseudo-code implementation of this method, which computes the displacement and its integral simultaneously. For simplicity, we assume that the current displacement is already given in the face-aligned basis. In practice, most degrees of freedom are stored in the global coordinate system but can be easily rotated using equation (3.12).

In this section, we described a numerical scheme that computes both η and its integral efficiently by expanding it as a Taylor series. This results in a high-order ODE solver that reuses the derivatives of the numerical solution from the ADER time-stepping, as described in section 4.2.

4.4. Summary & Computational Aspects

We derived a fully discrete high-order ADER-DG method, resulting in the one-step update given by equation (4.53). We split this into a two-step scheme involving the predictor $\boldsymbol{p}$ and corrector $\boldsymbol{c}$ update, resulting in

$$
|J| M_{kl} \mathfrak{q}_{lp}(t_1) = |J| M_{kl} \mathfrak{q}_{lp}(t_0) + p_{kp} + c_{kp}. \tag{4.58}
$$

Algorithm 1 ADER integration of the gravitational free surface boundary condition from $t = t_0$ to $t = t_1$. The displacement from the previous time step is $\eta_0 = \eta(t_0)$, and $\Delta t = t_1 - t_0$ is the size of the current time step. For $i = 0, \dots, N$, we use the derivatives vectors $\boldsymbol{D}^i$ to reconstruct the pressure and velocity. This is accomplished by the function `projectAndRotateDerivatives`, which projects the derivative vector of order i at time $t = t_0$ to the face basis and rotates it to the face-aligned coordinate system. For simplicity, we assume that the displacement and its integral are stored in the face-aligned coordinate system. We use equation (4.57) to compute the coefficients of the Taylor series equation (4.55). The function returns both $\eta(t_1)$ and $H = \int_{t_0}^{t^1} \eta(\tau)\,\mathrm{d}\tau$.

```
1:  function ComputeDisplacement($\eta_0, \rho, g, Z, \Delta t, \boldsymbol{D}$)
2:      $\eta_t^{\text{prev}} \leftarrow \eta_0$                      ▷ Initialize previous coefficient
3:      $H \leftarrow \Delta t \eta_0$                                 ▷ Initialize integral of $\eta$
4:      $f_\eta \leftarrow 1$                                          ▷ Constant factor in Taylor series for $\eta$
5:      $f_H \leftarrow \Delta t$                                      ▷ Constant factor in Taylor series for $H$
6:      for $i \in 1 \dots N+1$ do
7:          $f_\eta \leftarrow \frac{\Delta t}{i} f_\eta$              ▷ Update constant factor in Taylor series for $\eta$
8:          $f_H \leftarrow \frac{\Delta t}{i+1} f_H$                  ▷ Update constant factor in Taylor series for $H$
9:          $\frac{\partial^{i-1} v_1}{\partial t^{i-1}}_{|t=t_0}, \frac{\partial^{i-1} p}{\partial t^{i-1}}_{|t=t_0} \leftarrow$ projectAndRotateDerivatives($\boldsymbol{D}^{i-1}$)
10:         $\eta_t \leftarrow \frac{\partial^{i-1} v_1}{\partial t^{i-1}}_{|t=t_0} + \frac{1}{Z}\left(\frac{\partial^{i-1} p}{\partial t^{i-1}}_{|t=t_0} - \rho g \eta_t^{\text{prev}}\right)$   ▷ Evaluate equation (4.57)
11:         $\eta_t^{\text{prev}} \leftarrow \eta_t$
12:         $\eta \leftarrow \eta + f_\eta \eta_t$                     ▷ Update $\eta$
13:         $H \leftarrow H + f_H \eta_t$                              ▷ Update $H$
14:     return $\eta, H$
```

The `predict` kernel first computes the Taylor expansion of the solution using equation (4.46). Then, it computes the update

$$\begin{aligned} p_{kp} = {} & A^*_{pq}\, \mathcal{I}_{lq}(t_0,t_1)|J|K^\xi_{kl} + B^*_{pq}\, \mathcal{I}_{lq}(t_0,t_1)|J|K^\eta_{kl} + C^*_{pq}\, \mathcal{I}_{lq}(t_0,t_1)|J|K^\zeta_{kl} \\ & - \sum_{f\in\mathcal{F}} \left(|S_f|\, A^{*Lf}_{pq}\, \mathcal{I}_{lq}(t_0,t_1) F^{Lf}_{kl}\right) - \sum_{f\in\mathcal{F}^{\text{ext}}} \left(|S_f|\, A^{*Rf}_{pq} \left(\int_{t_0}^{t_1} \hat{\underline{q}}_{lq}(\tau)\,\mathrm{d}\tau\right) \hat{F}^{Rf}_{kl}\right), \end{aligned} \tag{4.59}$$

consisting of the volume and local surface integrals. The `correct` kernel

$$c_{kp} = -\sum_{f\in\mathcal{F}^{\text{int}}} \left(|S_f|\, A^{*Rf}_{pq}\, \mathcal{I}^f_{lq}(t_0,t_1) F^{Rfgh}_{kl}\right) \tag{4.60}$$

computes the contributions of the numerical flux from the neighbors. The predictor requires only element-local data, and the corrector requires only data from the neighbors. In SeisSol, we add $\boldsymbol{p}$ and $\boldsymbol{c}$ directly to the vector of coefficients $\underline{q}$ once we compute the updates, which avoids using additional storage. The resulting scheme is now a two-step update. We will later discuss why this split is beneficial. In contrast to our ADER-DG scheme, a standard Runge-Kutta DG method requires a mesh traversal for each stage. Hence, for higher orders, the ADER-DG method is more efficient.

As we iterate over our elements, we solve the Riemann problem twice for each interior face. This can lead to better performance than iterating over the faces individually [85]. The friction law required for dynamic rupture faces can be expensive, and thus, it is only computed once per face [168].

We used and extended the ADER-DG implementation in the software SeisSol. It achieves high efficiency by pre-computing all matrices and using the code generation software YATeTo [166]. YATeTo maps the tensor expressions to general matrix multiplication (GEMM) kernels. As our matrices are small, SeisSol relies on optimized backends such as LIBXSMM [59] or PSpaMM [172]. We refer the interested reader to [164], which explains how the SeisSol implementation of the ADER-DG scheme can be optimized and implemented with YATeTo. This framework was especially beneficial for our implementation of algorithm 1 because it automatically inferred the sparsity of the derivative coefficients.

We embedded the acoustic wave equation into the elastic wave equation, which allowed us to use the same PDE. The physical interface conditions are included in the numerical flux, as discussed in chapter 3. However, this embedding leads to a computational overhead.

In this chapter, we derived a one-step upgrade scheme for variable linear PDEs of the form given by equation (2.29). We achieved a high order in space with the Discontinuous Galerkin approach (section 4.1.3). Combining this with the ADER predictor-corrector scheme gave us a scheme that is also high-order in time (section 4.2). Finally, we used both components to derive a novel high-order scheme for the gravitational boundary condition (section 4.3).

Chapter 5.

Energy Stability

In this section, we prove the semi-discrete stability of the numerical scheme as outlined in chapter 4 with the numerical flux discussed in chapter 3 for the acoustic wave equation with gravity (section 2.2.2). We follow the approach of [50] and prove that our method is Gustafsson-Kreiss-Sundström (GKS) stable or, more descriptively, energy stable. The idea is that the norm of the numerical solution $\boldsymbol{q}(\boldsymbol{x}, t)$ should not increase during the simulation. We prove this by showing for all time t that the energy rate

$$\frac{\partial \|\boldsymbol{q}\|}{\partial t} \leq 0, \tag{5.1}$$

does not increase, which implies that the norm $\| \cdot \|$ of the solution does not grow [175].

We define the physical energy, similar to equation (2.96), as

$$\begin{aligned} E(\boldsymbol{q}) &= \underbrace{\int_\Omega \frac{1}{2}\rho \boldsymbol{v} \cdot \boldsymbol{v} \,\mathrm{d}\boldsymbol{x}}_{\text{kinetic energy}} + \underbrace{\int_\Omega \frac{1}{2K} p^2 \,\mathrm{d}\boldsymbol{x}}_{\text{acoustic energy}} + \underbrace{\int_{S_O} \frac{1}{2}\rho g \eta^2 \,\mathrm{d}S}_{\text{gravitational energy}} \\ &= \int_\Omega \frac{1}{2} \left(\boldsymbol{P}\boldsymbol{q}\right) \cdot \boldsymbol{q} + \int_{S_O} \frac{1}{2}\rho g \eta^2 \,\mathrm{d}S \\ &= \left(\frac{1}{2}\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{q}\right)_\Omega + \left\langle \frac{1}{2}\rho g \eta,\, \eta \right\rangle_{S_O}, \end{aligned} \tag{5.2}$$

where we used the matrix

$$\boldsymbol{P} = \begin{pmatrix} \frac{1}{K} & 0 & 0 & 0 \\ 0 & \rho & 0 & 0 \\ 0 & 0 & \rho & 0 \\ 0 & 0 & 0 & \rho \end{pmatrix} \tag{5.3}$$

and the shorthand notation $(\cdot, \cdot)_\Omega$ for the interior product over the volume Ω, which has the boundary $\partial\Omega$. The surface $S_O \subset \partial\Omega$ defines the ocean surface on which we apply the gravitational free surface boundary condition. We use the shorthand notation $\langle \cdot,\, \cdot \rangle_{S_O}$ for inner products over this surface. Equation (5.2) defines a norm of the extended solution ($\boldsymbol{q}$ and η), as it is a positive symmetric transformation of the solution.

The energy rate, i.e., the derivative of the energy (equation (5.2)), is given by

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \left(\boldsymbol{P}\boldsymbol{q},\, \frac{\partial \boldsymbol{q}}{\partial t}\right)_\Omega + \left\langle \rho g \eta,\, \frac{\partial \eta}{\partial t} \right\rangle_{S_O} \tag{5.4}$$

which follows directly from the product rule.

The main result of this section is theorem 1.

Theorem 1. *Assuming exact integration, the numerical scheme described in chapters 3 and 4 for the acoustic wave equation with gravity is energy stable because the energy $E(\boldsymbol{q})$ is non-increasing as*

$$\frac{\partial E(\boldsymbol{q})}{\partial t} \leq 0. \tag{5.5}$$

Similar results have been proven in the literature: Uphoff proved the energy stability for a DG discretization of the elastic part of the fully coupled model [164]. Wilcox, Stadler, Burstedde, and Ghattas showed the stability for elastic-acoustic coupling for an essentially identical Riemann solver [175]. Lotto and Dunham proved the energy stability of a summation-by-parts finite difference discretization of a two-dimensional version of the acoustic wave equation, including the gravitational boundary condition [97]. Hence, we restrict this chapter to proving the stability of our DG discretization of the acoustic part, including gravitational effects. To the best of our knowledge, this result is novel.

In the following, we will attack the problem in simple steps. We compute the energy rate in section 5.1 and insert our discretization. Section 5.2 computes the energy flux across all element interfaces and boundary conditions. We compute the gravitational energy rate in section 5.3. Finally, section 5.4 completes the proof.

5.1. Energy

In this section, we derive the energy rate for our numerical flux. We use the hyperbolic PDE

$$\frac{\partial \boldsymbol{q}}{\partial t} = -\boldsymbol{A}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial x} - \boldsymbol{B}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial y} - \boldsymbol{C}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial z} \tag{2.29 revisited}$$

with the flux matrices defined by equation (2.82). To simplify the notation, we denote the flux matrices by $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ instead of $(\boldsymbol{A}^{\text{ac}}, \boldsymbol{B}^{\text{ac}}, \boldsymbol{C}^{\text{ac}})$. In the same spirit, we use $\mathcal{T}$ for the acoustic rotation matrix defined by equation (3.13).

The total energy for our problem is not conserved in the L^2 norm; however, when ignoring boundary conditions, it is conserved in the norm defined by equation (5.2), which we motivated by considering the physical energy. There is an additional argument in favor of this norm: The flux matrices must be symmetric to apply the energy method, as seen in the discussion in [113]. As our flux matrices are not symmetric, we need to symmetrize them. The products of the matrix $\boldsymbol{P}$ with the flux matrices

$$\begin{aligned}
\overline{\boldsymbol{A}}_{ij} &= (\boldsymbol{PA})_{ij} = (\boldsymbol{PA})_{ji} = \delta_{i2}\delta_{j1} + \delta_{i1}\delta_{j2}, \\
\overline{\boldsymbol{B}}_{ij} &= (\boldsymbol{PB})_{ij} = (\boldsymbol{PB})_{ji} = \delta_{i3}\delta_{j1} + \delta_{i1}\delta_{j3}, \\
\overline{\boldsymbol{C}}_{ij} &= (\boldsymbol{PC})_{ij} = (\boldsymbol{PC})_{ji} = \delta_{i4}\delta_{j1} + \delta_{i1}\delta_{j4},
\end{aligned} \tag{5.6}$$

are symmetric and do not depend on the material parameters. In other words, the matrix $\boldsymbol{P}$ simultaneously symmetrizes the flux matrices. With this, we can follow the approach discussed in [77] and write our PDE in the form

$$\boldsymbol{P}(\boldsymbol{x})\frac{\partial \boldsymbol{q}}{\partial t} = -\overline{\boldsymbol{A}}\frac{\partial \boldsymbol{q}}{\partial x} - \overline{\boldsymbol{B}}\frac{\partial \boldsymbol{q}}{\partial y} - \overline{\boldsymbol{C}}\frac{\partial \boldsymbol{q}}{\partial z}, \tag{5.7}$$

where the matrix $\boldsymbol{P}$ captures the (potentially) discontinuous coefficients and thus depends on space. The flux matrices $(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}})$ of this modified form are symmetric and do not depend on space. Equation (5.7) defines a class of variable-coefficient hyperbolic PDEs for which stability can be proven directly by applying the energy method [77]. In detail, their energy (assuming dissipative homogeneous boundary conditions) is bounded in the norm defined by the inner product $\left(\frac{1}{2}\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{q}\right)_{\mathcal{T}}$. We can interpret the energy $\left\langle \frac{1}{2}\rho g \eta,\, \eta\right\rangle_{S_O}$ of the sea surface as a weighted norm of the displacement that has the correct unit. We will use the form equation (2.29) in the following proof because it mirrors our implementation.

We split the energy rate (equation (5.4)) into a sum over all tetrahedra $\mathcal{T}$ by writing

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \left(\sum_{\mathcal{T}} \underbrace{\left(\boldsymbol{P}\boldsymbol{q},\, \frac{\partial \boldsymbol{q}}{\partial t}\right)_{\mathcal{T}}}_{=\frac{\partial E^{\mathcal{T}}}{\partial t}}\right) + \left(\underbrace{\left\langle \rho g \eta,\, \frac{\partial \eta}{\partial t}\right\rangle_{S_O}}_{=\frac{\partial E^{G}}{\partial t}}\right), \tag{5.8}$$

where the inner product $(\cdot,\, \cdot)_{\mathcal{T}}$ denotes the inner product restricted to a tetrahedron $\mathcal{T}$. We now ignore the product over the gravitational surface $(\frac{\partial E^G}{\partial t})$ and focus on the other interfaces and boundary conditions.

We compute the weak form of the energy rate by inserting the PDE, given by equation (2.29), into equation (5.8) and integrating by parts

$$\begin{aligned}\left(\boldsymbol{P}\boldsymbol{q},\, \frac{\partial \boldsymbol{q}}{\partial t}\right)_{\mathcal{T}} &= -\left(\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x} + \boldsymbol{B}\frac{\partial \boldsymbol{q}}{\partial y} + \boldsymbol{C}\frac{\partial \boldsymbol{q}}{\partial z}\right)_{\mathcal{T}} \\ &= \left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial x},\, \boldsymbol{A}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial y},\, \boldsymbol{B}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial z},\, \boldsymbol{C}\boldsymbol{q}\right)_{\mathcal{T}} - \langle \boldsymbol{P}\boldsymbol{q},\, \boldsymbol{f}^*\rangle_{\partial\mathcal{T}},\end{aligned} \tag{5.9}$$

where we introduced the numerical flux $\boldsymbol{f}*$ in the inner product on the tetrahedron's surface, which we denote by $\langle \cdot,\, \cdot\rangle_{\partial\mathcal{T}}$.[1] By integrating the volume term by parts again, we arrive at the strong form

$$\begin{aligned}\left(\boldsymbol{P}\boldsymbol{q},\, \frac{\partial \boldsymbol{q}}{\partial t}\right)_{\mathcal{T}} = &-\left(\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x}\right)_{\mathcal{T}} - \left(\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{B}\frac{\partial \boldsymbol{q}}{\partial y}\right)_{\mathcal{T}} - \left(\boldsymbol{P}\boldsymbol{q},\, \boldsymbol{C}\frac{\partial \boldsymbol{q}}{\partial z}\right)_{\mathcal{T}} \\ &+ \langle \boldsymbol{P}\boldsymbol{q},\, \boldsymbol{f} - \boldsymbol{f}^*\rangle_{\partial\mathcal{T}},\end{aligned} \tag{5.10}$$

with $\boldsymbol{f} = \tilde{\boldsymbol{A}}\boldsymbol{q}$, where we used the rotated flux matrix $\tilde{\boldsymbol{A}}$ defined by equation (3.4). The name "strong form" comes from the strict smoothness assumptions on $\boldsymbol{q}$ [84]. This form

[1]Equation (5.9) is equivalent to using $\boldsymbol{P}\boldsymbol{q}$ as a test function in equation (4.17).

has the attractive property that we can interpret the surface integral as a penalty term that vanishes when $\boldsymbol{q}$ is continuous between elements.

Next, we consider the volume integrals in equation (5.9). We compute

$$\left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial x}, \boldsymbol{A}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial y}, \boldsymbol{B}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\frac{\partial \boldsymbol{q}}{\partial z}, \boldsymbol{C}\boldsymbol{q}\right)_{\mathcal{T}} \tag{5.11}$$

$$= \left(\underbrace{\boldsymbol{A}^T\boldsymbol{P}}_{=\boldsymbol{PA}}\frac{\partial \boldsymbol{q}}{\partial x}, \boldsymbol{q}\right)_{\mathcal{T}} + \left(\underbrace{\boldsymbol{B}^T\boldsymbol{P}}_{=\boldsymbol{PB}}\frac{\partial \boldsymbol{q}}{\partial y}, \boldsymbol{q}\right)_{\mathcal{T}} + \left(\underbrace{\boldsymbol{C}^T\boldsymbol{P}}_{=\boldsymbol{PC}}\frac{\partial \boldsymbol{q}}{\partial z}, \boldsymbol{q}\right)_{\mathcal{T}} \tag{5.12}$$

$$= \left(\boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x}, \boldsymbol{P}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{B}\frac{\partial \boldsymbol{q}}{\partial y}, \boldsymbol{P}\boldsymbol{q}\right)_{\mathcal{T}} + \left(\boldsymbol{C}\frac{\partial \boldsymbol{q}}{\partial z}, \boldsymbol{P}\boldsymbol{q}\right)_{\mathcal{T}} \tag{5.13}$$

$$= \left(\boldsymbol{P}\boldsymbol{q}, \boldsymbol{A}\frac{\partial \boldsymbol{q}}{\partial x}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\boldsymbol{q}, \boldsymbol{B}\frac{\partial \boldsymbol{q}}{\partial y}\right)_{\mathcal{T}} + \left(\boldsymbol{P}\boldsymbol{q}, \boldsymbol{C}\frac{\partial \boldsymbol{q}}{\partial z}\right)_{\mathcal{T}}. \tag{5.14}$$

They are now identical to the negative of the volume integrals of equation (5.10).

We add the weak (equation (5.9)) and strong form (equation (5.10)) of the energy rate, leading to

$$2\left(\boldsymbol{P}\boldsymbol{q}, \frac{\partial \boldsymbol{q}}{\partial t}\right)_{\mathcal{T}} = \langle \boldsymbol{P}\boldsymbol{q}, \boldsymbol{f} - 2\boldsymbol{f}^*\rangle_{\partial\mathcal{T}}. \tag{5.15}$$

This is a crucial step of our proof, as it eliminates the volume term, leaving us only with the contributions from the faces. Dividing by two and inserting the fluxes, we arrive at

$$\left(\boldsymbol{P}\boldsymbol{q}, \frac{\partial \boldsymbol{q}}{\partial t}\right)_{\mathcal{T}} = \left\langle \boldsymbol{P}\boldsymbol{q}, \tilde{\boldsymbol{A}}\left(\frac{1}{2}\boldsymbol{q} - \boldsymbol{q}^*\right)\right\rangle_{\partial\mathcal{T}}, \tag{5.16}$$

where $\boldsymbol{q}^*$ is the solution of the Riemann problem as described in chapter 3. The matrix $\boldsymbol{P}$ fulfills the rotational invariance property

$$\boldsymbol{P} = \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{P}\boldsymbol{\mathcal{T}}, \tag{5.17}$$

which follows from a straightforward computation.

The remaining surface integral is expressed in the global coordinate system; however, we described our Riemann solver (chapter 3) in the face-aligned coordinate system. We can rotate the surface integral with the manipulations

$$\frac{\partial E^{\mathcal{T}}}{\partial t} = \left\langle \boldsymbol{q}, \boldsymbol{P}\tilde{\boldsymbol{A}}\left(\frac{1}{2}\boldsymbol{q} - \boldsymbol{q}^*\right)\right\rangle_{\partial\mathcal{T}} \tag{5.18}$$

$$= \left\langle \boldsymbol{q}, \boldsymbol{P}\boldsymbol{\mathcal{T}}\boldsymbol{A}\boldsymbol{\mathcal{T}}^{-1}\left(\frac{1}{2}\boldsymbol{q} - \boldsymbol{q}^*\right)\right\rangle_{\partial\mathcal{T}} \tag{5.19}$$

$$= \left\langle \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{q}, \boldsymbol{\mathcal{T}}^{-1}\boldsymbol{P}\boldsymbol{\mathcal{T}}\boldsymbol{A}\boldsymbol{\mathcal{T}}^{-1}\left(\frac{1}{2}\boldsymbol{q} - \boldsymbol{q}^*\right)\right\rangle_{\partial\mathcal{T}} \tag{5.20}$$

$$= \left\langle \tilde{\boldsymbol{q}}, \boldsymbol{P}\boldsymbol{A}\left(\frac{\boldsymbol{1}}{\boldsymbol{2}}\tilde{\boldsymbol{q}} - \tilde{\boldsymbol{q}}^*\right)\right\rangle_{\partial\mathcal{T}}. \tag{5.21}$$

Equation (5.19) follows from inserting the rotational invariance of the flux matrix, given by lemma 1 on page 30, and equation (5.20) from multiplying with the matrix $\boldsymbol{\mathcal{T}}^{-1}$, which is valid because the rotation matrix is orthogonal, i.e., $\boldsymbol{\mathcal{T}}^{T} = \boldsymbol{\mathcal{T}}^{-1}$. In the final step, equation (5.21), we inserted the rotated degrees of freedom (equation (3.13)) and used the rotational invariance of the matrix $\boldsymbol{P}$ (equation (5.17)).

To summarize, we have shown in this section that the energy contribution from any tetrahedron only depends on the contribution of the surface integrals. Furthermore, we can directly work in the face-aligned coordinate system.

5.2. Energy Rate from Faces

In this section, we discuss the contributions coming from the face integrals in more detail. We switch the perspective: We split the contributions over tetrahedra in the previous section; in this section, we consider the effect of the faces. This approach is more natural as two elements contribute to an interior face. Thus, we can see how contributions are balanced. For each pair of neighboring tetrahedra with indices m and n, we define an interior face $\mathcal{E}^{\text{int}}_{mn}$ as their intersection. We must ensure that we include each face only once and thus only consider faces with $m < n$. The exterior faces $\mathcal{E}^{\text{ext}}_{l}$, indexed by $l = 0, \ldots$, are boundary faces. They intersect with exactly one tetrahedron. The numbering of all faces is arbitrary and does not affect the overall analysis.

We again use the inner product notation $\langle a,\, b\rangle_{\mathcal{E}^{\text{int}}_{mn}}$ to denote the inner product over the face $\mathcal{E}^{\text{int}}_{mn}$. An interior face has an energy contribution of

$$E^{\mathcal{E}^{\text{int}}_{mn}} = \left\langle \tilde{\boldsymbol{q}}^{\boldsymbol{L}},\, \boldsymbol{P}^{\boldsymbol{L}} \boldsymbol{A}^{\boldsymbol{L}} \left(\frac{1}{2} \tilde{\boldsymbol{q}}^{\boldsymbol{L}} - \tilde{\boldsymbol{q}}^{*} \right) \right\rangle_{\mathcal{E}^{\text{int}}_{mn}} - \left\langle \tilde{\boldsymbol{q}}^{\boldsymbol{R}},\, \boldsymbol{P}^{\boldsymbol{R}} \boldsymbol{A}^{\boldsymbol{R}} \left(\frac{1}{2} \tilde{\boldsymbol{q}}^{\boldsymbol{R}} - \tilde{\boldsymbol{q}}^{*} \right) \right\rangle_{\mathcal{E}^{\text{int}}_{mn}}, \tag{5.22}$$

which we get by summing the contributions of the tetrahedra $\mathcal{T}_n$ and $\mathcal{T}_m$, which are given by equation (5.21) [164]. Without loss of generality, we denote the element with index m as the left (L) and the other with index R. The negative sign in equation (5.22) comes from the definition of the normal vector. The matrices $\boldsymbol{P}$ and $\boldsymbol{A}$ can differ on both sides of the face. However, the product $\boldsymbol{PA}$ does not depend on the material parameters as seen in equation (5.6). Hence, it is identical for all elements.

The contribution of an exterior face is given by

$$E^{\mathcal{E}^{\text{ext}}_{l}} = \left\langle \tilde{\boldsymbol{q}}^{\boldsymbol{L}},\, \boldsymbol{P}^{\boldsymbol{L}} \boldsymbol{A}^{\boldsymbol{L}} \left(\frac{1}{2} \tilde{\boldsymbol{q}}^{\boldsymbol{L}} - \tilde{\boldsymbol{q}}^{*} \right) \right\rangle_{\mathcal{E}^{\text{ext}}_{l}}. \tag{5.23}$$

In contrast to the interior faces, only one tetrahedron $\mathcal{T}_m$ contributes to this face. We use the star states derived in section 3.4.1 for the boundary faces. We are now ready to summarize the contributions from all faces.

Interior faces After inserting the star state, given by equation (3.46), into the energy contribution of an interior face (equation (5.22)), we arrive at

$$E^{\mathcal{E}^{\text{int}}_{mn}} = \int_{\mathcal{E}^{\text{int}}_{mn}} - \frac{Z^L Z^R \left(u^L - u^R\right)^2 + \left(p^L - p^R\right)^2}{Z^L + Z^R} \,\mathrm{d}S \leq 0. \tag{5.24}$$

For a smooth solution, the right and left sides are identical. In this case, the energy contribution is zero.

Velocity inlet We insert equations (3.49) and (3.50) into equation (5.23) and arrive at the energy rate contribution for a velocity inlet

$$E^{\mathcal{E}_l^v} = \int_{\mathcal{E}_l^{\text{ext}}} Z\left(v_1^* v_1^L - \left(v_1^L\right)^2\right) - p^L v_1^* \,\mathrm{d}S \tag{5.25}$$

which reduces to

$$E^{\mathcal{E}_l^{v=0}} = \int_{\mathcal{E}_l^{\text{ext}}} -Z\left(v_1^L\right)^2 \,\mathrm{d}S \leq 0 \tag{5.26}$$

for the zero-velocity boundary condition, where we set $v_1^* = 0$.

Pressure inlet We insert equations (3.52) and (3.53) into equation (5.23) and arrive at the energy contribution

$$E^{\mathcal{E}_l^p} = \int_{\mathcal{E}_l^{\text{ext}}} -\frac{Z p^* v_1^L - p^* p^L + (p^L)^2}{Z} \,\mathrm{d}S. \tag{5.27}$$

For the free surface condition, we set $p^* = 0$. Hence, equation (5.27) reduces to

$$E^{\mathcal{E}_l^{p=0}} = \int_{\mathcal{E}_l^{\text{ext}}} -\frac{(p^L)^2}{Z} \,\mathrm{d}S \leq 0. \tag{5.28}$$

5.3. Energy Rate from Gravity

In the previous section, we discussed how to compute the energy flux across most boundary conditions. We have not yet discussed how to compute the energy rate resulting from the gravitational surface. This is more challenging, as acoustic energy converts to gravitational energy and vice versa. Hence, to achieve an energy-stable scheme, we also need to consider the energy captured on the surface.

We begin with the contribution from the boundary condition. By inserting the states for the modified free surface condition (equations (3.55) and (3.56)) into equation (5.27), we arrive at a contribution of

$$E_1^{\mathcal{E}_l^g} = \int_{\mathcal{E}_l^{\text{ext}}} \rho g \eta \left(\frac{p^L}{Z} - v_1^L\right) - \frac{\left(p^L\right)^2}{Z} \,\mathrm{d}S, \tag{5.29}$$

which may be positive.

Next, we look at the change in gravitational energy. We combine the definition of the displacement

$$\frac{\partial \eta}{\partial t} = v_1^* = v_1^L + \frac{1}{Z}\left(p^L - \rho g \eta\right) \tag{3.57 revisited}$$

with the gravitational energy rate

$$\frac{\partial E^G}{\partial t} = \left\langle \rho g \eta, \frac{\partial \eta}{\partial t} \right\rangle_{S_O}. \tag{5.30}$$

Focusing on one face of the gravitational surface, we arrive at a contribution of

$$E_2^{\mathcal{E}_l^g} = \int_{\mathcal{E}_m^{\text{ext}}} -\frac{\rho^2 g^2 \eta^2 - \rho g \eta (Z v_1^L + p^L)}{Z} \,\mathrm{d}S, \tag{5.31}$$

which, similarly to equation (5.29), is not necessarily negative.

Finally, we look at the overall energy rate of the gravitational part of our system. By adding the energy rate of the gravitational boundary face (equation (5.29)) and equation (5.31), we get an overall energy rate for each gravitational face of

$$E^{\mathcal{E}_l^g} = E_1^{\mathcal{E}_l^g} + E_2^{\mathcal{E}_l^g} = \int_{\mathcal{E}_m^{\text{ext}}} -\frac{\left(\rho g \eta - p^L\right)^2}{Z} \,\mathrm{d}S \leq 0. \tag{5.32}$$

This is non-positive!

5.4. Proof

This section combines our partial results to prove that the energy rate for the acoustic wave equation without source term and with free surface, zero-velocity, and gravitational free surface boundary conditions is non-positive.

Proof of theorem 1. We collect all interior, zero velocity, free surface, and gravitational faces in the sets $\boldsymbol{\mathcal{E}}^{\text{int}}, \boldsymbol{\mathcal{E}}^{\boldsymbol{v}=\mathbf{0}}, \boldsymbol{\mathcal{E}}^{\boldsymbol{p}=\mathbf{0}}$ and $\boldsymbol{\mathcal{E}}^{\boldsymbol{g}}$. Then, by combining the results for interior faces (equation (5.24)), zero-velocity (equation (5.26)) and free surface (equation (5.28)) boundary conditions, with the total gravitational energy contribution (equation (5.32)), we arrive at the total energy rate of

$$\begin{aligned} \frac{\partial E}{\partial t} = & \left(\sum_{\mathcal{E}_{mn}^{\text{int}} \in \boldsymbol{\mathcal{E}}^{\text{int}}} E^{\mathcal{E}_{mn}^{\text{int}}} \right) + \left(\sum_{\mathcal{E}_l^{v=0} \in \boldsymbol{\mathcal{E}}^{\boldsymbol{v}=\mathbf{0}}} E^{\mathcal{E}_l^{v=0}} \right) \\ & + \left(\sum_{\mathcal{E}_l^{p=0} \in \boldsymbol{\mathcal{E}}^{\boldsymbol{p}=\mathbf{0}}} E^{\mathcal{E}_l^{p=0}} \right) + \left(\sum_{\mathcal{E}_l^g \in \boldsymbol{\mathcal{E}}^{\boldsymbol{g}}} E^{\mathcal{E}_l^g} \right) \end{aligned}, \tag{5.33}$$

which is non-positive because all contributions are non-positive. This proves theorem 1! □

The DG method is not strictly energy preserving because it weakly enforces the boundary condition and the continuity between elements. The resulting energy dissipation stabilizes the DG method [84]. To summarize, our numerical flux (chapter 3) combined with the DG discretization (chapter 4) leads to a numerical scheme for which we can guarantee that the energy of the numerical solution cannot increase, assuming exact time-integration. Therefore, our numerical scheme is semi-discrete stable. This result is especially interesting for the gravitational boundary condition, as it reveals why we must define η by equation (3.57), which requires solving an ODE: Other choices would lead to an unstable method.

Chapter 6.

Local Time-Stepping

As we use ADER-DG (chapter 4), an explicit numerical scheme, we must adhere to a CFL-type (equation (4.47)) condition for the time step size. The standard method is to use global time-stepping (GTS): It updates all elements together and uses the smallest global time step size. While this works well for simple cases, in realistic scenarios, we often have vastly different element sizes. We compute regions of interest with greater detail and—to save computational costs—compute less relevant regions with a coarser resolution. For example, we use elements with larger physical sizes to avoid spurious reflections from our not perfectly absorbing boundary conditions. The solution in these elements does not produce relevant outputs. Another reason for heterogeneous time step sizes is that wave speeds vary spatially. For elastic-acoustic coupling, the wave speeds in the fluid (e.g., for ocean water $c \approx 1.5\,\mathrm{km\,s^{-1}}$) are far smaller than in the Earth (e.g., in the crust $c_p \approx 6\,\mathrm{km\,s^{-1}}$) [39]. We can avoid the latter issue to some extent by adapting the element sizes to the wavelength of the medium [175]. A more significant problem stems from the imperfectness of meshing software. When resolving intersections between parts of the mesh, such as the fault and the ocean floor, meshing software can create ill-shaped elements called slivers. Even a single sliver can drastically reduce the time-to-solution [13]!

But we can do better: We use local time-stepping (LTS), where elements are updated with different time step sizes [37]. The ADER-DG scheme laid out in chapter 4 allows us to use heterogeneous time step sizes directly. In this chapter, we explore an elegant, performant, and robust implementation of local time-stepping for ADER-DG. As we consider linear PDEs, the time step size depends only on the material parameters and on the size of each element but not on the numerical solution. Hence, we only need to consider static load balancing, which simplifies the algorithms.

We begin by explaining the overall structure of our local time-stepping scheme in section 6.1. We use a clustered LTS method [13], which groups elements with similar time step sizes in clusters that are updated together. Section 6.2 follows with an explanation of the necessary changes to the numerical scheme and the resulting constraints on the updates. In section 6.3, we describe a simplified algorithm that updates all elements. We introduce an actor model, which combines a state machine, which keeps track of each cluster's state, with message passing, which explicitly manages the communication between clusters and thus makes the information flow clearer. This abstraction allows us to elegantly encode the scheduling constraints. In section 6.5, we describe how we can map our numerical scheme, as described in chapter 4, to our abstractions. Additionally, we briefly describe how we can add shared memory parallelism. We explain the distributed memory parallelism in section 6.6. Section 6.7 combines everything to create an elegant and stable scheduling algorithm optimized for parallelism. Finally, in section 6.8, we show how to automatically fine-tune the LTS clustering to decrease the time-to-solution. We summarize the resulting numerical scheme in section 6.9.

6.1. Clustered LTS

While it is possible to use a separate time step size for each element [37], this leads to a complicated update scheduling, which is hard to parallelize. The theoretical speed-up, which comes directly from reducing the number of total updates, is optimal, but the practical speedup is not. Hence, this approach is not attractive for HPC applications.

Instead, we use the clustered LTS method developed in [13]. The idea is to use time clusters: We update all elements in a cluster with the same time step, which is not larger than the time step required by the CFL condition. Therefore, the resulting scheme is stable but updates some elements more often than strictly required. Each element should thus belong to the cluster with the highest possible time step size because this reduces the computational cost. Let $(\Delta t)_{\min}$ be the globally minimal time step size. We consider a clustering strategy following [13] for which the ith cluster has a time step size of $r^i (\Delta t)_{\min}$, where $r \in \mathbb{N}, r > 0$ is the time step rate.

Consider, for example, a rate-2 scheme with three clusters and a minimum time step size of $(\Delta t)_{\min} = 0.25$:

Cluster 0 has $\Delta t = 2^0\,(\Delta t)_{\min} = 0.25$ and a rate of 2^0.

Cluster 1 has $\Delta t = 2^1\,(\Delta t)_{\min} = 0.5$ and a rate of 2^1.

Cluster 2 has $\Delta t = 2^2\,(\Delta t)_{\min} = 1$ and a rate of 2^2.

We note that cluster 0 has the smallest and cluster 2 has the largest time step. In this case, all elements with a time step size between 0.25 and 0.5 belong to the zeroth cluster, all elements with a time step between 0.5 and 1 belong to the first cluster, and all other elements belong to the second cluster.

We use the constant $\lambda \in (0.5, 1.0]$. For now, we assume that $\lambda = 1$, but we will return to this parameter later. The first cluster contains elements with time step sizes in the interval

$$[\lambda(\Delta t)_{\min},\, 2\lambda(\Delta t)_{\min})\,, \tag{6.1}$$

the second cluster has elements of size

$$[2\lambda(\Delta t)_{\min},\, 4\lambda(\Delta t)_{\min})\,, \tag{6.2}$$

and so on. We assume that we have m clusters. The last cluster can be open, meaning it contains all elements with a time step size larger than the lower end of the last cluster, as done in [12]. Formally, we define the ith cluster as the cluster that collects elements with time steps in the interval

$$\mathcal{C}_i = \begin{cases} \left[2^i \lambda (\Delta t)_{\min},\, 2^{i+1} \lambda (\Delta t)_{\min}\right) & \text{if } i < m-1, \\ \left[2^i \lambda (\Delta t)_{\min},\, \infty\right) & \text{otherwise.} \end{cases} \tag{6.3}$$

We call i the cluster id. Next, we define the operator `clusterId` that matches an element with time step size Δt to a cluster. Formally, the function

$$\text{clusterId}(\Delta t) = \text{argmin}_i \left\{ \mathcal{C}_i \mid \Delta t \in \mathcal{C}_i \right\}, \tag{6.4}$$

returns the id i of the time cluster that contains the time step size Δt.

We assume that our clustering obeys the maximum difference invariance, which tremendously simplifies our implementation by reducing the number of possible cases [13].

Definition 2. *A clustering obeys the maximum difference invariance if for all elements l_1 and all of its neighbors $l_2 \in \mathcal{N}(l_1)$ we have that $|\text{clusterid}(l_1) - \text{clusterid}(l_2)| \leq d$, where d is the maximum allowed distance.*

In our case, $d = 1$ unless l_1 shares a dynamic rupture face with l_2, in which case we set $d = 0$, as in [168].

6.2. Numerical Considerations

As the previous section showed, setting up our LTS clustering is straightforward. However, we must change the numerical scheme, including what data we store. As outlined in section 4.4, we split our computations into predictor and corrector kernels. The predictor is an element-local process. Hence, it requires only data from one element. The corrector kernel requires time-integrated face data of the element's neighbors in addition to its own data. The neighbors may have different time step sizes, in which case the computations involve data from multiple time steps. Our one-step update equation (4.53) already includes this. However, we must ensure that all required data is available before computing a correction and that the predictor does not override any data necessary to correct other clusters.

The predictor computes a Taylor expansion of the numerical solution in time, which is then used by the corrector of this and all neighboring elements. As discussed earlier, the function defined by equation (4.52) allows us to integrate the degrees of freedom between the beginning of a time step t_0 and its end t_1 by summing up a Taylor series.

To efficiently implement local time-stepping, we need to compute the data while keeping the storage overhead as small as possible. In the following, we assume that our element i has a time step size of $(\Delta t)_i$ and that we have an r-rate scheme. We must consider all neighbor elements k, which potentially have a different time step size than i, for the data storage. We only need to differentiate three cases thanks to the maximum difference invariance (definition 2).

The first case is the simplest one. Both our element and the neighbor have the same time step. In this case, both elements are in the GTS configuration. No additional data is needed to reconstruct the solution, as the neighbor k can directly use the integrated quantities of our element i for the correction step. However, the current element must not overwrite its integrated quantities before the neighbor consumes them.

In the second case, the neighboring cell has a time step of $\frac{1}{r}(\Delta t)_k = (\Delta t)_i$. Our element i takes one time step for every r time steps that the neighbor takes. After we perform a prediction, we must store the resulting derivatives in a buffer. The neighbor k then uses these data to compute the integrated solution from the derivatives of i by evaluating equation (4.52). Here, the neighbor must keep track of the expansion point of the Taylor series, i.e., the time at which we evaluated the derivatives of i. Thus, the element i must not perform further predictions until the neighbor has consumed the derivatives.

In the third case, the neighboring element has a time step of $r(\Delta t)_k = (\Delta t)_i$. Our element i takes r time steps for each time step of the neighbor k. In the prediction, our element i computes the integral of the solution over multiple time steps by summing up the integrated solutions. The neighbor k directly uses these integrated values in its correction step. Additionally, the element i must reset its buffer every r time steps—after ensuring that the neighbor has already consumed the data.

As we have seen in this section, while ADER-DG allows for local time-stepping without extensive modifications, data dependencies introduce constraints for the scheduling of the operations. The storage overhead is relatively small. If the elements have a different time step size, we need to store a buffer for the derivatives of one time step or the integral of the solution over multiple time steps. In some cases, for example, when we have one neighbor with a smaller and one with a higher time step rate, we store the derivatives and the integrated solution. We organize the data in our implementation so that each element manages all data it generates. Furthermore, we always store the volume data, even though the corrector only requires the solution at the faces. Thus, each element only writes local data in the predictor step, reducing the necessary memory transfer. However, we move the cost to the correction step, which accesses data from other elements and must compute the face projection. If required, it must reconstruct the integrated numerical solution from the derivatives.

6.3. The Time-Stepping Algorithm

In this section, we show a simplified version of our time-stepping algorithm. We assume we have an arbitrary number of time clusters but do not use distributed memory parallelization. We define a function `advanceInTime` that advances our simulation to the next synchronization point t^{sync}. Examples of synchronization points are the end of the simulation or moments when we want to output data. At these points, all time clusters must reach the same simulation time.

Algorithm 2 shows a possible implementation of `advanceInTime`. It is a simplified version of the algorithm that we use in practice. The pseudocode does not define all operations. Briefly, `setSyncTime` followed by `reset` primes the cluster to start the time-stepping anew with the final goal of reaching the new synchronization time t^{sync}. The function `act` steps the cluster forward in time—but only if it is currently possible.

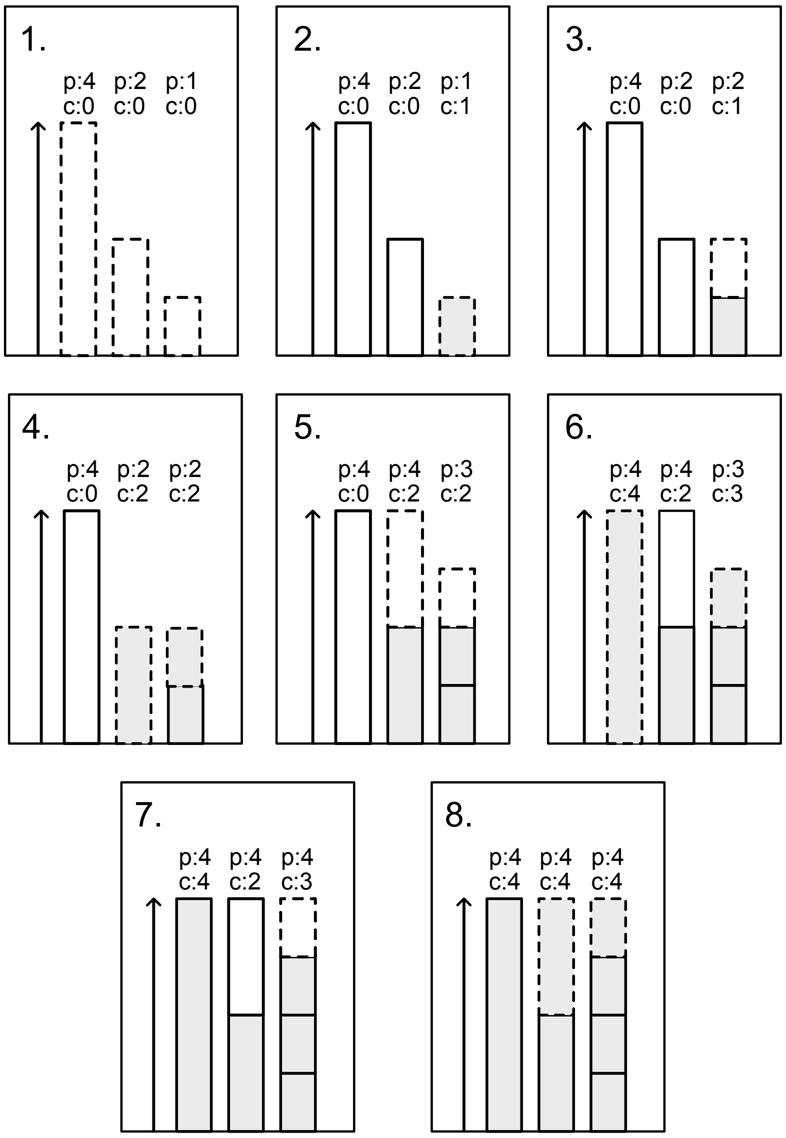


Figure 6.1.: This figure shows how clusters are updated in a rate-2 LTS scheme with three clusters. The left-most cluster has a rate of 4, the center of 2, and the right-most of 1. Each panel shows all updates that are currently possible. Rectangles with dashed borders correspond to planned updates. Solid borders represent already finished updates. Unfilled boxes are predictions, and filled boxes are corrections. Above each cluster, p and c correspond to the number of predictions and corrections, multiplied by the cluster's rate, that the cluster has computed after the planned updates have taken place. We can interpret p, c as the simulation time in the unit of $(\Delta t)_{\text{min}}$. For example, in the first panel, each cluster may predict. In the second panel, only the cluster with the smallest time step size can correct.

Algorithm 2 Simple time-stepping algorithm using our local time-stepping method. We update all clusters in the set $\mathcal{C}$ until $t = t^{\text{sync}}$.

```
function advanceInTimeSimple($\mathcal{C}, t^{\text{sync}}$)
    for cluster $\in \mathcal{C}$ do
        cluster.setSyncTime($t^{\text{sync}}$)
        cluster.reset()
    hasFinished $\leftarrow$ false
    while $\neg$hasFinished do
        hasFinished $\leftarrow$ true
        for cluster $\in \mathcal{C}$ do
            cluster.act()
            hasFinished $\leftarrow$ hasFinished $\land$ cluster.synced()
```

Hence, it needs to ensure that all required data for the update is available and that we do not overwrite data before a neighboring cluster has used it. If the update is legal, `act` computes it. Otherwise, `act` does not perform any action. Hence, this method abstracts both the update and its scheduling. Finally, the method `synced` returns true if and only if the cluster has reached the sync point at $t = t^{\text{sync}}$.

To appreciate the simplicity of this algorithm, we invite the reader to compare it to the original implementation of [13], which iteratively checks which clusters can be updated and then schedules the updates manually. We will later arrive at a more complicated algorithm necessitated by the parallelization strategy and resulting optimizations. However, the spirit of the implementation stays the same.

6.4. The Actor Model

In this section, we set up our time cluster scheduling by defining the missing operations from algorithm 2. Again, we restrict ourselves to the sequential case. The key difference of our implementation to [13] is that we use a state machine approach. We introduce the abstraction of an actor: An actor manages exactly one cluster. Every actor manages its scheduling on its own. Thus, no global scheduling step is necessary. Actors communicate their status to their neighbors directly, resulting in a more robust and simpler implementation compared to [13].

Each actor can be in exactly one of the following states:

Synced (S) This actor has reached the next synchronization point. It lies dormant until a new synchronization point is set.

Predicted (P) The actor has computed a local prediction.

Corrected (C) The actor has computed a correction using information from its neighbors.

We also define a set of actions that an actor can perform. **Correct** and **Predict** perform a correction and prediction, respectively. **Sync** moves an actor into the synchronized

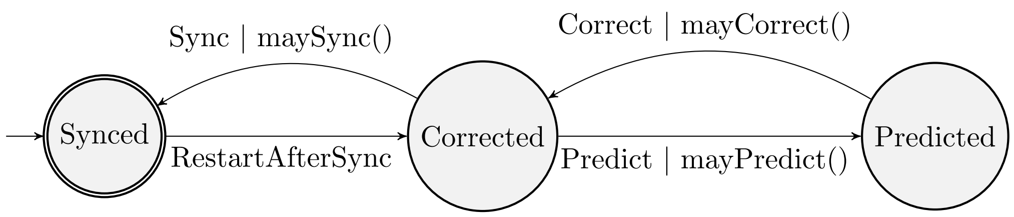


Figure 6.2.: The states (represented by nodes) are connected by actions (edges). **Synced** is both the start and the only accepting state. Only legal actions are shown. The notation for the edges reads as "Action | Requirement". **RestartAfter-Sync** can be considered as an action that is triggered from outside of the cluster by the scheduler.

state, which can be left by restarting it, as indicated by the **RestartAfterSync** action. We define the **Nothing** pseudo-action for convenience: It does not do anything.

Figure 6.2 shows by which actions a cluster can move from one state to another. This diagram is already enough to eliminate a large category of possible bugs: For instance, it is impossible to move from **Predicted** directly to **Synced** without passing through **Corrected**. Furthermore, a prediction cannot follow another prediction but only a correction.

However, as mentioned in section 6.2, there are other constraints on the actions that we need to consider. To keep track of these constraints, we introduce counters and other state variables that keep track of each cluster's progress. Each cluster (indexed by $i = 0, \dots, m-1$) has a time step rate of

$$(\Delta s)_i = r^i. \tag{6.5}$$

For example, if we have a rate-2 LTS scheme with a minimum time step size of $1/2$ and our cluster has a time step size of 1, it has a cluster index of $i = 1$ and thus a rate of $(\Delta s)_1 = 2^1$. Hence, each cluster measures its progress by the number of time steps that the cluster with the smallest time step would have needed. This allows us a simpler notation because we can compare the progress of clusters directly, even if they have a different time step size. Each cluster i counts the predictions s_i^p and corrections s_i^c since the last synchronization point. The variable s_i^s counts the total number of corrections until the next synchronization point. We reset this counter after each synchronization point. It is computed for a cluster i by

$$s_i^s = \left\lceil (\Delta s)_i \frac{t_{n+1}^{\text{sync}} - t_n^{\text{sync}}}{(\Delta t)_i} \right\rceil, \tag{6.6}$$

where t_n^{sync} is the time of the nth synchronization point, $(\Delta t)_i$ is time step size of the cluster and $\lceil \cdot \rceil$ denotes the ceiling function.[1] This is the only part of the scheduling that directly depends on the simulation time. Furthermore, we store the last correction time in

[1]This can lead to some time steps with a time step size of zero. However, this simplifies the implementation drastically and does not lead to significant overhead for typical simulations.

the counter t_i^c. It is needed for the computation of integrals and for other time-dependent kernels.

The counter

$$s_i^n = \max\left(s_i^c + (\Delta s)_i, s_i^s\right) \tag{6.7}$$

gives the number of corrections that the ith cluster has computed after it has completed the next correction. From this, we can extract the adjusted time step rate

$$\left(\overline{\Delta s}\right)_i = \max\left(s_i^n - s_i^c, (\Delta s)_i\right), \tag{6.8}$$

which is the maximum time step rate such that the cluster i does not step over the next sync point.

Each cluster stores these counters for itself but also keeps "shadow counters" that keep track of the state of its immediate neighbors. We denote these by an underbar. For example, the shadow counter of s_i^c is denoted by $\underline{s}_i^c$. We simplify our notation by assuming that each cluster stores all relevant counters in the tuple Ω_i. Furthermore, we often drop the underbar whenever we do not need to differentiate between real and shadow counters.

We define Boolean functions that evaluate the necessary conditions to change into a new state. They only depend on Ω_i. We consider a cluster i with neighbors with indices $k \in \mathcal{N}_i$. The function

$$\text{maySync}(\Omega) = s_i^c \geq s_i^s \tag{6.9}$$

checks whether we have reached the next synchronization point. This is the case if the number of corrections we performed is at least equal to the number of steps until a synchronization point. The function

$$\text{mayPredict}(\Omega) = \begin{cases} \neg\,\text{maySync}(\Omega) \wedge s_i^p < \operatorname{argmin}_{k \in \mathcal{N}_i} s_k^n & \text{if } |\mathcal{N}_i| > 0, \\ \neg\,\text{maySync}(\Omega) & \text{else,} \end{cases} \tag{6.10}$$

checks whether we can compute a local prediction. If the cluster has no neighbors, this is always the case. Otherwise, the cluster needs to ensure that it has not predicted past the next correction time of all neighbors. This avoids overwriting data that a neighboring cluster requires. Furthermore, we only allow a prediction if we are not yet past a synchronization point. The function

$$\text{mayCorrect}(\Omega) = \forall k \in \mathcal{N}_i : (s_i^p \geq s_k^p) \vee (s_k^s \leq s_k^p) \tag{6.11}$$

verifies that we can perform a correction. The first condition checks that we have predicted at least as far as all neighbors k. The second condition considers the case that the neighbor has predicted to the next synchronization point. If at least one of these criteria is fulfilled for all neighbors, we can safely correct as all necessary data are available.

Again, note that we stated all criteria in terms of the number of time steps instead of the absolute simulation time. The latter method was used in the original implementation [13]. It can lead to problems due to floating point inaccuracies, especially when we have

Algorithm 3 Function that returns the next legal action of a cluster. The tuple Ω stores the counters for the current cluster and all relevant neighbors. If no action is legal, it returns **Nothing**. The action **RestartAfterSync** is triggered when the scheduler resets the counter s_i^c.

```
function getNextLegalAction(Ω)
    processMessages()                                ▷ Update shadow states
    switch state do
        case Corrected
            if maySync(Ω) then
                return Sync
            else if mayPredict(Ω) then
                return Predict
        case Predicted
            if mayCorrect(Ω) then
                return Correct
        case Synced
            if $s_i^c = 0$ then          ▷ Check if this is the first step after sync point
                return RestartAfterSync
    return Nothing                                   ▷ No action is legal currently
```

many clusters (leading to a high maximal rate) and a small minimal time step size. In contrast, the only place where the simulation time is taken directly into account in our scheduling method is for handling synchronization points. This is natural and necessary because synchronization points are defined with respect to the simulation time and not the number of time steps.

With these definitions, writing a function that returns the next legal action is trivial. Algorithm 3 defines such a function. Only a maximum of one action is legal at any time. Otherwise, the algorithm would not be deterministic. Here, **Nothing** is treated as a pseudo-action: It is a signal that no action is legal at this moment.

The action is then performed by `unsafePerformAction`, as illustrated in algorithm 4. The functions `enterCorrected` and `enterPredicted`, defined in algorithm 5, are called before the cluster enters the **Corrected** or **Predicted** state, respectively. For now, we treat the concrete implementation of the functions `predict` and `correct`, which perform the actual computations, as black boxes.

We described the necessary state transitions but have not clarified how to update Ω. The local part of Ω_i, i.e., the part that describes the progress of the current cluster, is updated after we compute a correction or prediction. The shadow counters that track the state of the neighboring clusters are updated via message passing. This has the advantage that we directly model the flow of information between clusters. We must ensure that each cluster updates its shadow states before deciding on the next action.

Due to the maximum invariance property (definition 2), each cluster is only connected to a maximum of two other clusters whose time step size differs by a factor of exactly

Algorithm 4 A function that performs one action of one cluster. The naming unsafe indicates that it does not verify that the action is legal. The functions `enterCorrected` and `enterPredicted` are defined by algorithm 5. The function `start` initializes the cluster.

```
function unsafePerformAction(action, $\Omega$)
    switch action do
        case Correct
            enterCorrected()
            state $\leftarrow$ Corrected
        case Predict
            enterPredicted()
            state $\leftarrow$ Predicted
        case Sync
            state $\leftarrow$ Sync
        case RestartAfterSync
            start()
            state $\leftarrow$ Corrected
```

r. This is shown by figure 6.3. We implement the message passing by using message queues. Each cluster manages one queue per neighbor. Consider, for example, the cluster with id $i = 2$ that is connected to clusters 1 and 3. Cluster 2 maintains a queue for incoming messages from cluster 1 and cluster 3. It also holds a reference to the incoming queues of cluster 1 and 3. From the perspective of cluster 2, these are called outgoing queues. Hence, for each pair of connected clusters, we have two queues. As clusters hold a reference to the queues of their neighbors, they can directly push messages to them.

We define Boolean functions that determine whether we should send a message. These functions are evaluated for each neighbor k after we have updated the local parts of Ω_i. A cluster i sends a message to a neighbor k if

$$\begin{aligned}
\text{justBeforeSync}(\Omega, i) &= s_i^s \geq s_i^p \\
\text{maySendMessagePrediction}(\Omega, i, k) &= \text{justBeforeSync}(\Omega, i) \vee s_i^p \geq s_k^n, \\
\text{maySendMessageCorrection}(\Omega, i, k) &= \text{justBeforeSync}(\Omega, i) \vee s_i^c \geq s_k^p.
\end{aligned} \tag{6.12}$$

Each message consists of a tag and an update for the shadow counters. Possible tags are `AdvancedPrediction` and `AdvancedCorrection`, sent after a successful prediction and correction, respectively. Algorithm 6 processes incoming messages. The methods `handleAdvancedCorrectionMessage` and `handleAdvancedPredictionMessage` can be used to update the internal states when receiving a message from a neighbor. As updated counters often result in new legal actions, updating the shadow counters as quickly as possible is crucial for good performance. Hence, before evaluating the transition conditions, we process all messages in line 2 of algorithm 3.

With these ingredients, we can finally state algorithm 7, which updates a cluster if possible. This method has only two self-explanatory lines: First, find the legal action. Second, the function `unsafePerformAction`, defined by algorithm 4, performs this action.

Algorithm 5 The functions `enterCorrected` and `enterPredicted` perform an action, update the local part of Ω, and send messages to all neighboring clusters. The functions `maySendAdvancedCorrectionMessage` and `maySendAdvancedPredictionMessage` are defined by equation (6.12). The functions `correct` and `predict` compute a correction or prediction, respectively. The functions `sendAdvancedCorrectionMessage`(k, s_i^c, t_i^c) and `sendAdvancedPredictionMessage`(k, s_i^p) send a message to the message queue of the cluster with index k. Note that we send the last correction time but do not send the last prediction time, as it is not required by our algorithm. The message contains the updated counter. We define all functions for a cluster with index i.

```
function enterCorrected($\Omega, i$)
    correct()
    $s_i^c \leftarrow s_i^c + (\Delta s)_i$
    $t_i^c \leftarrow t_i^c + (\Delta t)_i$
    for $k \in \mathcal{N}_i$ do
        if maySendAdvancedCorrectionMessage($\Omega_i, i, k$) then
            sendAdvancedCorrectionMessage($k, s_i^c, t_i^c$)
function enterPrediced($\Omega, i$)
    predict()
    $s_i^p \leftarrow s_i^p + (\Delta s)_i$
    for $k \in \mathcal{N}$ do
        if maySendAdvancedPredictionMessage($\Omega_i, i, k$) then
            sendAdvancedPredictionMessage($k, s_i^p$)
```

We have seen how we can describe our state machine algorithmically. Now, we briefly describe how it could be formalized. It is convenient to describe our framework as a state machine. However, state machines typically do not provide strong enough abstractions to be used as a model for complex algorithms where the state transitions depend on multiple factors. We take inspiration from the state chart framework introduced in [53], which extends the finite state machine approach for event-driven algorithms. Our notation deviates from the state chart formalism and follows the standard notation of finite state machines with minor additions. Thus, our notation follows mostly the standard notation for state machines, described, for example, in [142]. The set of actions corresponding to the state machine's alphabet is given by

$$\Sigma = \{\text{RestartAfterSync}, \text{Sync}, \text{Predict}, \text{Correct}\} \tag{6.13}$$

and the set of states as

$$S = \{\text{Synced}, \text{Corrected}, \text{Predicted}\}. \tag{6.14}$$

The initial state is $s_0 = \text{Synced}$ and the set of accepting states is $\boldsymbol{F} = \{\text{Synced}\}$.

Algorithm 6 A function that processes messages and updates Ω accordingly. Here, we marked the shadow counters, i.e., a counter for a neighbor that needs to be manually synchronized, with an underbar. For example, $\underline{s}_k^p$ is the shadow counter for s_k^p. We assume that the message variable is an object that contains counter updates.

```
function PROCESSMESSAGES($\Omega$)
    for $k \in \mathcal{N}_i$ do
        if K.INBOX.HASMESSAGES() then
            message ← k.inbox.pop()                ▷ Get a message from message queue
            switch typeof(message) do
                case AdvancedPredictionMessage
                    $\underline{s}_k^p \leftarrow$ message.$s_k^p$
                    HANDLEADVANCEDPREDICTIONMESSAGE($k$, message)
                case AdvancedCorrectionMessage
                    $\underline{t}_k^c \leftarrow$ message.$t_k^c$
                    $\underline{s}_k^c \leftarrow$ message.$s_k^c$
                    HANDLEADVANCEDCORRECTIONMESSAGE($k$, message)
```

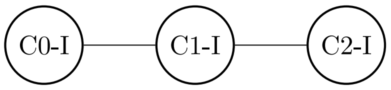


Figure 6.3.: Connection between clusters when we do not use a distributed memory parallelization. The nodes represent clusters. The node labels correspond to the clusters, e.g., "C0-I" is the zeroth cluster, which is connected to the first but not the second cluster. A cluster only communicates with clusters that are directly connected to it. Hence, it does not use data from other clusters and does not need to consider other clusters for its scheduling. This directly reveals that our LTS scheme only requires local constraints and local communication.

Furthermore, we define the higher-order function

$$e(s) = \begin{cases} \text{enterSync} & \text{if } s = \text{Synced}, \\ \text{enterCorrect} & \text{if } s = \text{Corrected}, \\ \text{enterPredict} & \text{if } s = \text{Predicted}, \end{cases} \tag{6.15}$$

that maps a state $s \in S$ to a function. Whenever we enter a state s, we evaluate the function returned by $e(s)$. The function should be considered a non-pure function, i.e., it is allowed to perform arbitrary side effects, including, but not limited to, updating the state of the cluster. This extension was also introduced in [53]. We define the state-transition function $\delta : S \times \Omega \to \Sigma$ that, given a state and cluster status description Ω, returns the next state for this cluster. We can define this implicitly by looking at

Algorithm 7 Function that moves one time cluster forward in time. It is always safe to call it, as it checks which action is legal. If no action is legal, the function does not perform any action.

```
1: function ACT($\Omega$)
2:     nextAction $\leftarrow$ GETNEXTLEGALACTION($\Omega$)
3:     UNSAFEPERFORMACTION(nextAction)
```

figure 6.2, or more concretely, as

$$\begin{aligned} \delta(\text{Synced}, \Omega) &= \begin{cases} \text{Corrected} & \text{if restartAfterSync}(\Omega), \end{cases} \\ \delta(\text{Corrected}, \Omega) &= \begin{cases} \text{Synced} & \text{if maySync}(\Omega), \\ \text{Predicted} & \text{if mayPredict}(\Omega), \end{cases} \\ \delta(\text{Predicted}, \Omega) &= \begin{cases} \text{Corrected} & \text{if mayCorrect}(\Omega). \end{cases} \end{aligned} \tag{6.16}$$

It is a partial function because, sometimes, no action is legal. Re-entering a state is illegal as we omitted the **Nothing** action. Hence, the resulting state graph is deterministic: Only up to one action is legal at any point. Finally, we can define the state machine describing one cluster with the tuple $(S, \Sigma, \delta, s_0, F, e)$, which is identical to a standard state machine description with two significant differences: First, the state-transition function δ depends on the state of the cluster. Second, the description contains the function e responsible for the side effects.

To summarize, the actor model combines state machines with message passing to manage the states of the clusters.

6.5. Computations & Shared Memory Parallelization

We have discussed under which conditions we can move between states. In this section, we finally discuss how we perform the actual computations.

As mentioned before, algorithm 6 calls the methods `handleAdvancedCorrectionMessage` and `handleAdvancedPredictionMessage` whenever an actor receives a message that a neighboring cluster has finished a correction or prediction. For a correction message, a cluster does not need to do anything. Whenever the cluster i receives a prediction message from a neighbor k with a higher time step rate, it must store k's last correction time t_k^c in the shadow counter $\underline{t}_k^c$. As k has a higher time step rate, it provides the coefficients of the Taylor series of its solution. The coefficients are given by the derivative of the solution of k, evaluated at t_k^c. The cluster i must integrate the numerical solution of all elements in the cluster k with equation (4.52), which requires the expansion time of the Taylor series.

We compute the kernels in the methods `correct` and `predict`. The implementation follows the description in section 4.4. As discussed, our computations are grouped in two mesh traversals. In our implementation, which follows the strategy outlined in [13, 168], we parallelize on this level: We update the clusters sequentially and execute the correction and prediction kernels separately. Each mesh traversal is parallelized using OpenMP [23], requiring a single `#pragma omp parallel for` for each loop.

The method `predict` computes the prediction kernel, which consists of using ADER to expand the solution as a Taylor series in time (equation (4.46)), and local updates (equation (4.59)). The method `correct` performs the correction, which consists of solving the Riemann problem between the current element and all of its neighbors on their time-integrated data (equation (4.60)). Hence, the neighbor may need to integrate the Taylor series to reconstruct the time-integrated data if they are not already available. Thus, we only need to store derivatives if a neighboring cluster requires them. In the corrector, we also consider faces with a dynamic rupture boundary condition, as described in [168]. For details about how we store the data, refer to section 6.2 and the description in [13, 164].

Finally, we note that the actor model would allow for another level of parallelism: We could execute actors in parallel, which would require adding mutexes in the message queues and using a parallelism framework that supports nested parallelism. As the naive approach leads to good performance for most scenarios, we restrict ourselves to parallelism on the kernel level.

6.6. Distributed Memory Parallelization

Up to this point, we only discussed the shared memory case. In this section, we describe how we can extend our LTS scheme with distributed memory parallelization. We subdivide our mesh and distribute the work over multiple ranks. Each rank thus handles a subset of the data. We manually synchronize data between ranks by explicit message passing, using the Message Passing Interface (MPI) [109]. The resulting data dependencies are more complex and require modifying our scheme.

For this, we introduce the abstraction of ghost clusters (section 6.6.1). We discuss in section 6.6.2, how we can use them to enable MPI progression. Finally, in section 6.6.3, we briefly describe the challenges generated by elements with a dynamic rupture boundary condition.

6.6.1. Ghost Clusters

For the distributed memory parallelization, we further partition each actor. We divide them into three layers:

Interior (I) These clusters contain all elements for which all neighbors are on the same MPI rank.

Copy (C) These clusters contain all elements for which at least one neighbor is on another MPI rank. We duplicate elements with neighbors on multiple ranks once per neighboring rank [13], simplifying our implementation.

Algorithm 8 Functions for the ghost clusters. Calls that are defined by the superclass `AbstractTimeCluster` are prefixed by `super::`. The functions `receiveGhostLayer` and `sendCopyLayer` create all necessary asynchronous receives and sends and push them to the respective message queues.

```
function ACT($\Omega$)
    TESTFORGHOSTLAYERRECEIVES($\Omega$)
    TESTFORCOPYLAYERSENDS($\Omega$)
    return SUPER::ACT($\Omega$)
function START($\Omega$)
    RECEIVEGHOSTLAYER()
function PREDICT()                                          ▷ Do nothing
function CORRECT()                                          ▷ Do nothing
function MAYCORRECT($\Omega$)
    return TESTFORCOPYLAYERSENDS() $\wedge$ SUPER::MAYCORRECT($\Omega$)
function MAYPREDICT($\Omega$)
    return TESTFORGHOSTLAYERRECEIVES() $\wedge$ SUPER::MAYPREDICT($\Omega$)
function MAYSYNC($\Omega$)
    return TESTFORCOPYLAYERSENDS() $\wedge$ TESTFORGHOSTLAYERRECEIVERS() $\wedge$
SUPER::MAYSYNC($\Omega$)
function HANDLEADVANCEDPREDICTIONTIMEMESSAGE(k, message)
    SENDCOPYLAYER()                              ▷ To all relevant MPI neighbors
function HANDLEADVANCEDCORRECTIONTIMEMESSAGE(k,message)
    $s \leftarrow s_i^c$                     ▷ Number of corrections after next correction
    if state = Predicted then
        $s \leftarrow s_i^n$
    if $s < s_i^s$ then              ▷ Avoid sending duplicate receive before sync point
        RECEIVEGHOSTLAYER()                    ▷ From all relevant MPI neighbors
function TESTFORGHOSTLAYERRECEIVES($\Omega$)
    return TESTQUEUE(receiveQueue)
function TESTFORCOPYLAYERSENDS($\Omega$)
    return TESTQUEUE(sendQueue)
function TESTQUEUE(queue)
    for request $\in$ queue do
        if MPI_TEST(request) then                  ▷ Check if request is finished
            REMOVE(queue, request)
    return EMPTY(queue)
```

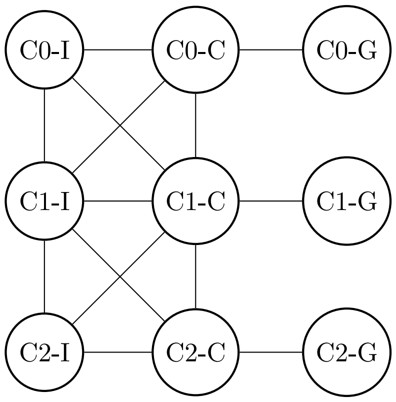


Figure 6.4.: Connection between clusters for the distributed memory parallelization. Actors are represented by nodes. Their labels begin with the character ‘C’, followed by the cluster id. They end with a suffix, which can be either ‘I’, ‘C’, or ‘G’ for interior, copy, or ghost layers. For instance, the label “C2-C” refers to the copy cluster with the id 2.

Ghost (G) These clusters are located on neighboring ranks. They do not perform any computations but rather synchronize with other ranks by sending and receiving data required for computations.

It is important to discuss how these actors are connected. For a visual explanation, refer to figure 6.4. We give each cluster a name of the format “$Ci{-}[I|C|G]$”, where i is the cluster id. The suffixes I, C, and G denote interior, copy, and ghost clusters. All interior and copy clusters are connected to all interior and copy clusters with an id that differs by at most one. Due to the maximum difference property (definition 2), no further data are required. The ghost clusters are only connected to copy clusters of the same rate but not to any interior cluster, as no element in the ghost cluster is next to an element in the interior cluster. No other clusters are connected. Hence, it is directly visible that communication and scheduling are strictly local. If we have only one MPI rank, the copy and ghost clusters are empty, and the cluster connections essentially reduce to the simpler version shown by figure 6.3. We include these empty clusters in our implementation; however, this does not add much overhead as they do not perform any computations.

Furthermore, we duplicate the counters that we introduced for the LTS scheduling. We introduce the additional index $l \in \{\text{int}, \text{copy}, \text{ghost}\}$. For example, in the sequential case, the counter s_i^p counted the predictions for the cluster with id i. Now, for the distributed parallelism case, the cluster i is split into the interior cluster $Ci{-}I$ with counter $s^p_{\text{int},i}$ the copy cluster $Ci{-}C$ with $s^p_{\text{copy},i}$, and the ghost cluster $Ci{-}G$ with $s^p_{\text{ghost},i}$. We summarize the counters in the set $\Omega_{l,i}$. The sets $\mathcal{C}^{\text{int}}, \mathcal{C}^{\text{copy}}$, and $\mathcal{C}^{\text{ghost}}$ contain all interior, copy, and ghost clusters, respectively. Hence, the set of all clusters $\mathcal{C} = \mathcal{C}^{\text{int}} \cup \mathcal{C}^{\text{copy}} \cup \mathcal{C}^{\text{ghost}}$ is defined as their union.

We are now ready to reap the rewards from our modeling: Both interior and copy clusters follow the same logic and implementation outlined before! Due to their different

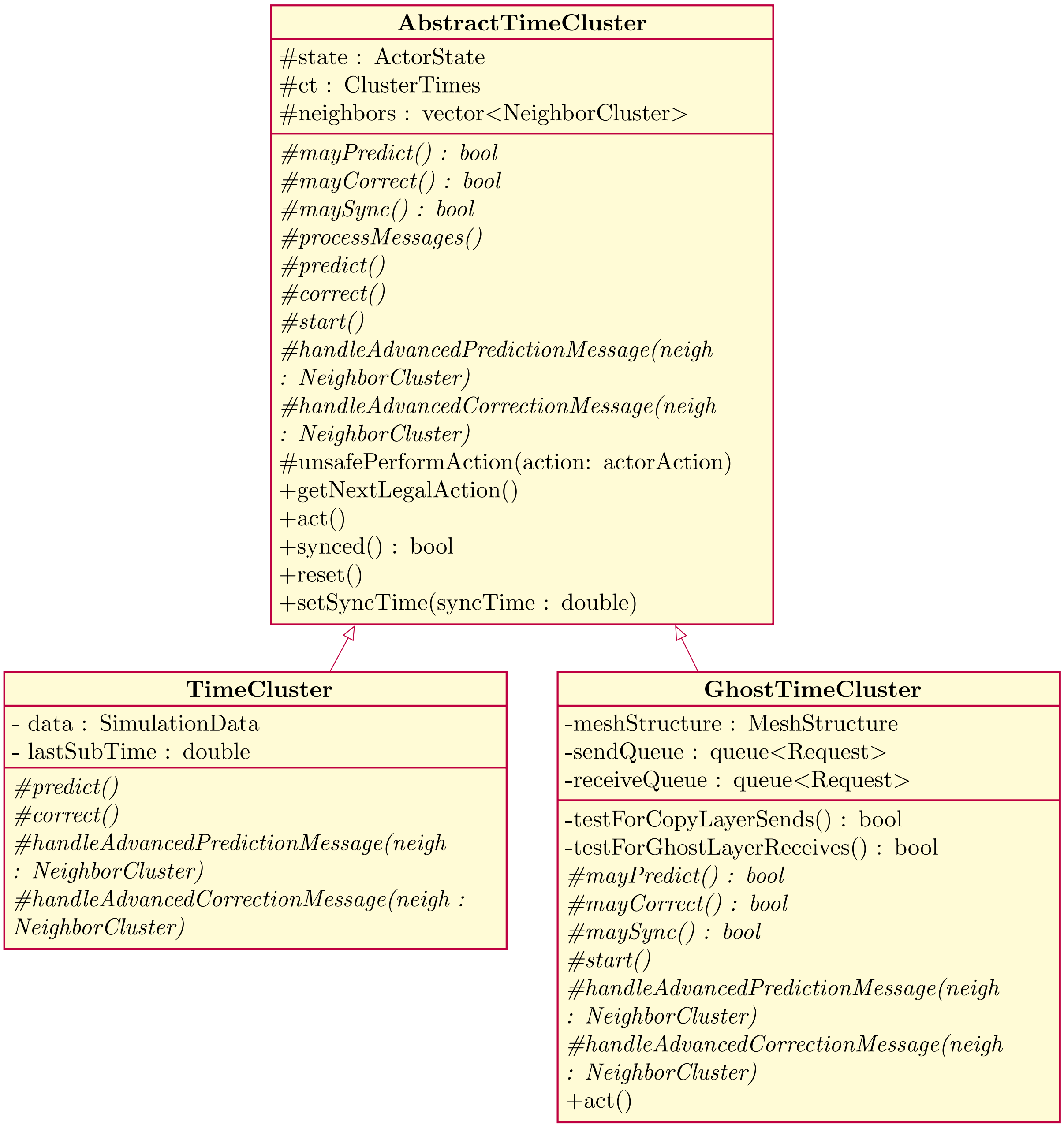

AbstractTimeCluster
#state : ActorState
#ct : ClusterTimes
#neighbors : vector<NeighborCluster>
#mayPredict() : bool
#mayCorrect() : bool
#maySync() : bool
#processMessages()
#predict()
#correct()
#start()
#handleAdvancedPredictionMessage(neigh : NeighborCluster)
#handleAdvancedCorrectionMessage(neigh : NeighborCluster)
#unsafePerformAction(action: actorAction)
+getNextLegalAction()
+act()
+synced() : bool
+reset()
+setSyncTime(syncTime : double)
TimeCluster
- data : SimulationData
- lastSubTime : double
#predict()
#correct()
#handleAdvancedPredictionMessage(neigh : NeighborCluster)
#handleAdvancedCorrectionMessage(neigh : NeighborCluster)
GhostTimeCluster
-meshStructure : MeshStructure
-sendQueue : queue<Request>
-receiveQueue : queue<Request>
-testForCopyLayerSends() : bool
-testForGhostLayerReceives() : bool
#mayPredict() : bool
#mayCorrect() : bool
#maySync() : bool
#start()
#handleAdvancedPredictionMessage(neigh : NeighborCluster)
#handleAdvancedCorrectionMessage(neigh : NeighborCluster)
+act()

Figure 6.5.: UML class diagram that shows the structure of the `TimeCluster` and `GhostTimeCluster` classes, which inherit their functionality from the superclass `AbstractTimeCluster`. The attribute `ct` stores the local counters (Ω), and the vector `neighbors` contains the shadow counters for neighboring clusters and the message queues. The variable `data` is a placeholder for the simulation data (e.g., the numerical solution and buffers required by LTS), and the variable `meshStructure` stores how neighboring ranks are connected. We assume the abstract superclass provides empty implementations for methods such as `predict` and `correct`. Note that both subclasses share the logic for `act`. Hence, they follow the same state machine. Note that the ghost time cluster tests for sends and receives in `act` to ensure MPI progression.

purpose, only the ghost clusters use a slightly different logic. However, the overall structure, as shown in figure 6.2, stays the same. Thus, the same state diagram describes interior, copy, and ghost clusters, simplifying the implementation. Figure 6.5 shows how we can describe this abstraction in an object-oriented manner. We define the superclass `AbstractTimeCluster`, which has two implementations: The class `TimeCluster` is used for both interior and copy clusters; the class `GhostTimeCluster` is used for ghost clusters. The superclass provides an implementation of the counters and the scheduling, as described in algorithms 3 to 7. The inheritance-based approach clarifies that both cluster types follow the same state machine. However, some methods are different. We describe the methods of the ghost time clusters in algorithm 8. As the ghost clusters do not perform any computations, the methods for `predict` and `correct` are trivial: Both do nothing. The method `start` posts the initial receive for the ghost layer. The entire logic happens after receiving a message. As the only neighbor of a ghost cluster is a single copy cluster, any message directly corresponds to an update of the copy cluster. When the ghost cluster receives an `AdvancedPrediction` message, it sends the prediction of the connected copy layer to all relevant MPI neighbors. The logic for an `AdvancedCorrection` message is slightly more complicated. For this, the cluster posts a receive for the ghost layer but only if the next correction does not lead us to a synchronization point. At synchronization points, `start` is called again, which would create a duplicate receive. Furthermore, if the synchronization point marks the end of the simulation, this avoids a dangling receive.

We use non-blocking communication methods to hide the communication behind the computations. For example, we compute the prediction of a copy cluster, which sends an `AdvancedPredictionTimeMessage` to the corresponding ghost cluster. We can then compute a prediction or correction (if available) of another interior or copy cluster. At some point, the ghost cluster uses `processMessages`, and after processing the incoming message, it sends the new prediction to a neighboring MPI rank. As send and receives are now asynchronous, they do not block but rather create a `Request` object that is put into a message queue. Each ghost cluster has two queues, one for its receives and one for its sends. The methods `mayPredict` and `mayCorrect` have the additional constraint that the queues for sends and receives are empty, respectively. We require both queues to be empty for `maySync`. These requirements again stem from the idea that we must not overwrite data that is still required by another operation.

6.6.2. MPI Progression

Using asynchronous message passing is crucial to achieving good performance on modern HPC systems. However, most standard MPI runtimes only transfer data whenever an MPI operation is called. This lack of progression can lead to higher communication delays when using asynchronous requests. Hence, deliberately calling MPI operations can be helpful to ensure communication progress. In our case, we enhance the method `act` of our ghost clusters by running `testForGhostLayerReceives` and `testForCopyLayerSends` on both queues to facilitate a quick MPI progression by calling `MPI_Test`. Furthermore, the methods `mayCorrect` and `mayPredict` also test for sends and receive, respectively.

For example, the issue of MPI progression is discussed in [31, 65], which evaluate using a dedicated communication thread for MPI progression. Multiple studies [13, 168] demonstrated that this improves the performance of SeisSol. Hence, we want to add this to our LTS framework. We introduce the abstraction of a communication manager, whose interface is described by an `AbstractCommunicationManager`. A communication manager contains all ghost clusters in an array. It has a private method `poll` which checks for updates. Its implementation is trivial and reveals the elegance of our abstraction model: It simply calls the method `act` of each ghost cluster, which checks the status of the pending receives and sends and thus leads to MPI progression. The abstract communication manager defines the public method `progression`, which the application can call to indicate that an MPI progression should be done. However, it should be considered a hint, as it may or may not perform this progression directly.

We introduce two different communication strategies: serial and threaded. In the serial case, we use the `SerialCommunicationManager`. Its `progression` method calls `poll` directly, thus ensuring communication.

In the parallel case, we use the `ThreadedCommunicationManager`. Its `progression` method does nothing. Here, progression is realized by a separate communication thread that calls `poll` in an infinite loop. While both variants guarantee that the communication progresses eventually, the threaded version does this quicker and even during computations.

6.6.3. Dynamic Rupture

The dynamic rupture (DR) boundary condition involves the computation of a friction law (equation (2.37)), which requires information from two adjacent faces. We compute it by iterating over all relevant faces. As described in [164, 168], the faces with a dynamic rupture boundary condition are partitioned into copy and interior layers. To avoid confusion, we use the names DR-copy and DR-interior when referring to the partitions of the DR faces. Faces belong to the DR-interior layer if they connect two elements that are in a copy or interior layer. Otherwise, they belong to the DR-copy layer. We can compute the DR-interior before receiving data for the ghost layer. Only the DR-copy layer faces depend on data from neighboring MPI ranks. We require only the DR-interior faces to compute a correction for an interior cluster but require both DR-interior and DR-copy faces to compute a correction for a copy cluster. However, we only want to compute the friction law for the DR-interior faces once. Additionally, we sometimes want to compute information for the fault output, which requires data from the friction law computation of both DR-interior and DR-copy faces.

To introduce these constraints, we use a `DynamicRuptureScheduler`. Each pair of interior and copy clusters with the same cluster id share one scheduler. We keep track of three counters for each pair with cluster id i. We note the last number of correction steps for which the dynamic rupture elements were computed for the interior part ($s^{\mathrm{dr}}_{\mathrm{int},i}$) and the copy layer ($s^{\mathrm{dr}}_{\mathrm{copy},i}$). Finally, we use the counter $s_i^{\mathrm{dr},f}$ to keep track of the last number of corrections for which the fault output was written. Again, to simplify the notation, we include the DR counters in the state $\Omega_{l,i}$, where interior and copy clusters with the id i share the same counters. With these counters, the constraints can be formalized by the functions

$$\text{mayComputeInterior}(\Omega_{l,i}) = s^c_{l,i} > s^{\text{dr}}_{\text{int},i}, \tag{6.17}$$

$$\text{mayComputeFaultOutput}(\Omega_{l,i}) = s^c_{l,i} = s^{\text{dr}}_{\text{int},i} \wedge s^c_{l,i} = s^{\text{dr}}_{\text{copy},i} \wedge s^c_{l,i} > s^{\text{dr},f}_{i}, \tag{6.18}$$

where $l \in \{\text{copy}, \text{int}\}$ is an index for the layer type. We compute the DR-interior faces before a correction whenever `mayComputeInterior` is true. Additionally, we compute the DR-copy faces before every copy layer correction. The fault output is computed whenever `mayComputeFaultOutput` is true.

Alternatively, we could include the dynamic rupture constraints in `mayPredict` and `mayCorrect`. However, we would then need to strictly order copy and interior updates, which our approach avoids because it allows updating copy and interior clusters in any order, which gives us more scheduling flexibility.

6.7. Scheduling

We already discussed the time-stepping loop in section 6.3, where we arrived at algorithm 2. However, we introduced a distributed memory parallelization scheme for which we split each cluster into interior and copy parts. When should we update each cluster?

First, we consider why the order of updates is irrelevant for the shared memory case. We have to compute all predictions and corrections for every time step for every cluster, which is obvious even without considering cluster dependencies. If we skip a time step, we do not arrive at the correct solution. Furthermore, the dependencies are structured such that it is impossible to reach a state where no action is legal for any cluster. And, by construction, it is always possible to execute a legal action without reaching an illegal state. Hence, the order of updates does not matter for the shared memory case.

The same is true to a limited degree when considering distributed memory communication. In whichever order we execute the actions, we are guaranteed to reach the end of the simulation. However, the order in which we process the clusters is essential for good performance, as sending data to neighboring ranks comes with a latency. Without taking special care, this latency could dominate the cost of the actual computations!

Here, we follow the ideas introduced by [13]. We want to hide the communication behind the computation, i.e., we want to keep updating clusters while other clusters wait to send or receive data. We can do this by computing the copy layers first, as they contain all data required by neighboring ranks. Furthermore, we prefer the computation of predictions, as they generate data neighbors require. We prefer clusters with a smaller time step rate, as they require more time steps in total and are thus executed more often. Note that as long as we manage to keep performing computations without blocking for sending or receiving neighboring data, the actual order of updates does not matter. This results in algorithm 9, which performs the same task as algorithm 2 but is more efficient. We first compute all available copy layer predictions, followed by all available copy layer corrections. Next, we compute up to one interior prediction, followed by up to one interior correction. We call the `progression` method of the communication manager to ensure MPI progression.

Algorithm 9 Function to move clusters forward in time until the synchronization point t^{sync}. It executes the same kernels as algorithm 2 but with a scheduling that is better suited for distributed memory parallelization: We prioritize copy clusters over interior clusters and predictions over corrections.

```
function AdvanceInTime($\mathcal{C}$, communicationManager, $t^{\text{sync}}$)
    for cluster $\in \mathcal{C}$ do
        cluster.setSyncTime($t^{\text{sync}}$)
        cluster.reset()
    hasFinished $\leftarrow$ false
    while $\neg$hasFinished do
        for cluster $\in \mathcal{C}^{\text{copy}}$ do
            if cluster.getNextLegalAction() = Predict then
                communicationManager.progression()
                cluster.act()
        for cluster $\in \mathcal{C}^{\text{copy}}$ do
            if cluster.getNextLegalAction() = Correct then
                communicationManager.progression()
                cluster.act()
        for cluster $\in \mathcal{C}^{\text{int}}$ do
            if cluster.getNextLegalAction() = Predict then
                cluster.act()
                break
        for cluster $\in \mathcal{C}^{\text{int}}$ do
            if cluster.getNextLegalAction() = Correct then
                cluster.act()
                break
        hasFinished $\leftarrow$ true
        for cluster $\in \mathcal{C}$ do
            hasFinished $\leftarrow$ hasFinished $\wedge$ cluster.synced()
```

6.8. Wiggle Factor & Cluster Merging

We included a parameter $\lambda \leq 1$, introduced in [12] to improve the LTS clustering, in the definition of the clusters (equation (6.3)) but have not yet described how we can use it. The parameter λ, which we call the wiggle factor, scales both time step limits of all clusters by a constant. For example, the zeroth cluster contains all elements with a time step size of $[(\Delta t)_{\min}, 2(\Delta t)_{\min})$ when not using a wiggle factor. When we use a wiggle factor, the cluster contains all elements with a time step size between $[\lambda(\Delta t)_{\min}, 2\lambda(\Delta t)_{\min})$.[2] Hence, elements in this cluster are updated more often. Thus, at first glance, this seems to reduce the efficiency of our time stepping. In the context

[2]There are, by definition, no elements with a time step size smaller than $(\Delta t)_{\min}$, so the lower limit of the zeroth cluster is $(\Delta t)_{\min}$.

Algorithm 10 Compute the minimal number of clusters while keeping the cost below the admissible cost. It takes the inputs $C, \mathcal{W}, r, c^{\max}, \lambda, (\Delta t)_{\min}$ which are the LTS clustering, the cost of each element, the LTS rate, the maximum admissible cost, the wiggle factor, and the minimum time step size. It uses the function COMPUTEGLOBALCOSTOFCLUSTERING, which computes the cost of the given clustering using equation (6.19). For this, it sums up the local costs of all MPI ranks. The function `enforceMaxClusterId` takes the arguments C and $i^{\max}$, and returns a new clustering where the maximum cluster id is $i^{\max}$.

```
1: function COMPUTEMAXCLUSTERIDAFTERAUTOMERGE($C, \mathcal{W}, r, c^{\max}, \lambda, (\Delta t)_{\min}$)
2:     $i^{\max} \leftarrow \max(C)$
3:     $i^{\max} \leftarrow$ ALLREDUCE($i^{\max}$)
4:     if $r = 1$ then return $i^{\max}$                    ▷ There can only be one cluster
5:     for $i^{\text{cur}} = i^{\max}, i^{\max} - 1, \dots, 1$ do
6:         $C^{\text{new}} \leftarrow$ ENFORCEMAXCLUSTERID($C, i^{\max}$)
7:         $c \leftarrow$ COMPUTEGLOBALCOSTOFCLUSTERING( $C^{\text{new}}$, $\mathcal{W}$, $r$, $\lambda$, $(\Delta t)_{\min}$)
8:         if $c > c^{\max}$ then return $i^{\text{cur}} + 1$
9:     return 0                                            ▷ GTS is acceptable
```

of LTS, however, a smaller λ can lead to a speed-up because it allows us to move some elements to a cluster with a larger time step, thus saving us some cost.

To illustrate, let us consider a simple example. We assume a 2-rate LTS scheme with four elements with time steps $\Delta t = 1, 1.6, 1.6, 3$. We compare two cases:

Without wiggle factor The minimal time step is $(\Delta t)_{\min} = 1$. We set $\lambda = 1$. Hence, we have two clusters with time steps $[1, 2)$ and $[2, 4)$. After clustering, the elements are now updated every 1, 1, 1, and 2 seconds, respectively. On average, for 1 s simulation time, we need $1 + 1 + 1 + 1/2 = 3.5$ updates.

With wiggle Let the wiggle factor be $\lambda = 0.8$. This results in clusters with time steps of $[0.8, 1.6)$ and $[1.6, 3.6)$. After clustering, the elements have time steps of 0.8, 1.6, 1.6, and 1.6. Now, for 1 s simulation time, we need $1.25 + 0.625 + 0.625 + 0.625 = 3.125 < 3.5$ updates.

Hence, even though it may be unintuitive, reducing the time step limits of all clusters may lead to fewer required updates!

We define the cost of a clustering C as

$$\text{cost}(C, \mathcal{W}) = \sum_{i=1,\dots,|C|} \frac{\mathcal{W}_i}{\lambda (\Delta s)_{C_i} (\Delta t)_{\min}} \tag{6.19}$$

where C_i is the cluster id of the ith element and $\mathcal{W}_i$ is its cost [79]. For now, it is not important how we compute the cost. We will discuss it in section 10.3. Note that C depends on the the wiggle factor λ. Hence, the equation (6.19) depends on the clustering because it includes the number of expected updates.

Computing an optimal wiggle factor is difficult. Here, we follow the approach from [12] and perform a simple grid search. We specify a minimum wiggle factor $\lambda^{\min}$ and a step size s, resulting in a list of candidates

$$\mathcal{L} = \lambda^{\min}, \lambda^{\min} + s, \ldots, 1. \tag{6.20}$$

Note that for a rate-r LTS scheme, only values $1/r < \lambda \leq 1$ are reasonable. We denote the clustering resulting from a wiggle factor of λ by C_λ. Then, we simply choose

$$\operatorname{argmin}_\lambda \{\operatorname{cost}(C_\lambda) \mid \lambda \in \mathcal{L}\} \tag{6.21}$$

as our wiggle factor.

Furthermore, we want to be able to enforce a maximum cluster id. For this, the last cluster has a maximum time step size of ∞, as realized in equation (6.3). The advantage of larger clusters is that they can be parallelized more efficiently. Merging the clusters with the highest time steps does not incur a significant additional computational cost. Thus, it can improve the performance. [12] presents a scheme that allows the user to manually set the maximum cluster id. We propose an extension to this: merging clusters automatically. Our method allows the user to define an upper limit by how much the total cost of all clusters (equation (6.19)) is allowed to increase. This results in a maximum admissible cost. We then, as shown in algorithm 10, simply merge clusters until this performance loss is realized.

Finally, we combine the wiggle factor and automatic merging of clusters into one algorithm. However, this is not straightforward. We consider two possible goals of the user: The first is that they start from a simulation with neither wiggle factor nor cluster merging. They then want to find the configuration with the smallest number of time clusters that lose up to a certain percentage of the original performance. The second possible goal is that the user wants to find the best configuration with the smallest number of clusters such that the performance is by a certain percentage smaller than the cost of the optimal wiggle factor without auto-merging. The difference between both models is the baseline used for computing the admissible cost.

We explain how we can realize the first goal and then describe a simple algorithm for the second goal, which uses the first as a subroutine. We start by computing the baseline cost, which is the cost without using either cluster merging or a wiggle factor (i.e., $\lambda = 1$). We compute the cost of this clustering and use it to define the maximum admissible cost by multiplying it with a user-defined factor. Then, we perform the grid search for the wiggle factor λ as outlined. We perform the automatic group merging process for each wiggle factor while staying within the cost limit. This results in tuples $(i_\lambda^{\max}, \lambda, c_\lambda)$, where $i_\lambda^{\max} = \max(C_\lambda)$ is the maximum cluster id of the clustering C_λ and c_λ is its cost. We use a map data structure that maps the number of clusters to the best cost and the wiggle factor λ with which it was achieved. Finally, we choose the wiggle factor with the lowest number of clusters from this map. For this number of clusters, we found the best wiggle factor. Algorithm 11 summarizes this procedure. It contains the search for the best λ without auto-merging as a special case.

Algorithm 11 Function to find the best wiggle factor using automatic merging of LTS clusters (if useAutomerge is true). Input arguments are the cost of each element ($\mathcal{W}$), the LTS rate (r), the maximum admissible cost ($c^{\max}$), and the minimal time step size ($(\Delta t)_{\min}$). The parameters for the grid search are the smallest wiggle factor ($\lambda^{\min} > 1/r$) and the step size s. Additionally, the user may give a maximum cluster id $i^{\max}$. The function returns a tuple consisting of the number of clusters, the best wiggle factor and the resulting cost. We use the functions `computeClustering` to compute the clustering according to equation (6.3) and `enforceMaximumDifference` to enforce the maximum difference invariance (definition 2).

```
 1: function ComputeBestWiggleFactor($\mathcal{W}, r, c^{\max}, (\Delta t)_{\min}, \lambda^{\min}, s, i^{\max}$, useAutomerge)
 2:     $\mathcal{M}^{i^{\max} \mapsto c} \leftarrow$ map()                ▷ Map from max cluster id to best cost
 3:     $\mathcal{M}^{i^{\max} \mapsto \lambda} \leftarrow$ map()                ▷ Map from max cluster id to best wiggle factor
 4:     for $\lambda = \lambda^{\min}, \lambda^{\min} + s, \ldots, 1$ do                ▷ Grid search for $\lambda$ with step size $s$
 5:         $C \leftarrow$ enforceMaximumDifference(computeClusterIds( $r, \lambda, (\Delta t)_{\min}$ ))
 6:         if useAutomerge then
 7:             $i \leftarrow$ min( $i^{\max}$, computeMaxClusterIdAfterAutoMerge($C, \mathcal{W}, r, c^{\max}, \lambda, (\Delta t)_{\min}$ ) )
 8:         else
 9:             $i \leftarrow i^{\max}$
10:         $C \leftarrow$ enforceMaxClusterId($C, i$)
11:         $i \leftarrow$ allReduce(max($C$), max)                ▷ Global maximum cluster id
12:         $c \leftarrow$ computeGlobalCostOfClustering( $C$, $\mathcal{W}$, $r$, $\lambda$, $(\Delta t)_{\min}$ )
13:         if $i^{\max} \notin \mathcal{M}^{i^{\max} \mapsto c} \lor c \leq \mathcal{M}^{i^{\max} \mapsto c}[i]$ then
14:             $\mathcal{M}^{i^{\max} \mapsto c}[i] \leftarrow$ c
15:             $\mathcal{M}^{i^{\max} \mapsto \lambda}[i] \leftarrow \lambda$
16:     if useAutomerge then                ▷ Choose minimal max cluster id with admissible cost
17:         $i^{\text{admissible}} \leftarrow \infty$
18:         for $(i, c) \in \mathcal{M}^{i^{\max} \mapsto c}$ do
19:             if $c < c^{\max} \land i < i^{\text{admissible}}$ then
20:                 $i^{\text{admissible}} \leftarrow$ i
21:     else                ▷ Choose minimal cost
22:         $i^{\text{admissible}} \leftarrow$ argmin($\mathcal{M}^{i^{\max} \mapsto c}$)                ▷ Maximum cluster id with smallest cost
23:     return $\left(i^{\text{admissible}}, \quad \mathcal{M}^{i^{\max} \mapsto \lambda}\left[i^{\text{admissible}}\right], \quad \mathcal{M}^{i^{\max} \mapsto c}\left[i^{\text{admissible}}\right]\right)$
```

Algorithm 12 Function to find the best wiggle factor using automatic merging of LTS clusters (if useAutomerge is true). This algorithm uses algorithm 11 as a sub-routine. The difference is that this algorithm automatically computes the admissible cost $c^{\max}$ by using a baseline cost and a factor m that states how much more expensive the clustering with group merging can be, compared to the baseline. There are two supported baseline models. The first, "bestWiggleFactor", computes the best wiggle factor without automatic merging. The second, "maxWiggleFactor", uses the clustering with $\lambda = 1$ and without automatic merging as the baseline. Other input arguments are the cost of each element ($\mathcal{W}$), the LTS rate (r), the minimal time step size ($(\Delta t)_{\min}$), the smallest wiggle factor ($\lambda^{\min} > 1/r$) and the step size s. Additionally, the user may give a maximum cluster id $i^{\max}$. The function returns a tuple consisting of the number of clusters, the best wiggle factor, and the resulting cost. The implementation is not optimal if automatic cluster merging is not used. In practice, we avoid these computations.

```
1: function FINDBESTWIGGLE($\mathcal{W}, r, m, (\Delta t)_{\min}, \lambda^{\min}, s, i^{\max}$, isAutomergeUsed, baselineCost)
2:     if baselineCost = bestWiggleFactor then                ▷ Find best $\lambda$ without merging
3:         $(i^{\max}, \lambda, c) \leftarrow$ COMPUTEBESTWIGGLEFACTOR($\mathcal{W}, r, \infty, (\Delta t)_{\min}, \lambda^{\min}, s, i^{\max}$, useAutomerge = false)
4:     else                                   ▷ Compute clustering with $\lambda = 1$ as baseline
5:         $C \leftarrow$ ENFORCEMAXIMUMDIFFERENCE(COMPUTECLUSTERIDS( $r, \lambda, (\Delta t)_{\min}$ ))
6:         $C \leftarrow$ ENFORCEMAXCLUSTERID($C, i^{\max}$)
7:         $c \leftarrow$ COMPUTEGLOBALCOSTOFCLUSTERING( $C$, $\mathcal{W}$, $r$, $\lambda$, $(\Delta t)_{\min}$ )
8:     $c^{\max} \leftarrow cm$
9:     return COMPUTEBESTWIGGLEFACTOR($\mathcal{W}, r, c^{\max}, (\Delta t)_{\min}, \lambda^{\min}, s, i^{\max}$, useAutomerge )
```

Next, we use the strategy that we devised for goal 1 to compute the best wiggle factor for goal 2: We want to find the best wiggle factor, and then we are willing to give up a certain percentage of this optimal performance for auto-merging. This can be easily accomplished by algorithm 12: We first search for the best wiggle factor without automatically merging clusters, using algorithm 11. Then, we use the resulting optimal cost as a baseline cost for a second search in which we now enable the auto-merging of clusters. This strategy may sound wasteful: In a naive implementation, we would need to compute the clustering twice, including enforcing the maximum difference. However, with a small modification in line 5 of algorithm 11, we can reduce the cost of the second search to almost zero: We store all computed clusterings after enforcing the maximum difference in a map data structure for each considered λ. This simple memoization strategy requires a small space overhead but turns our naive and expensive algorithm into a performant one.

Another computational challenge arises from the need to enforce the maximum difference between neighboring elements, as described by definition 2. This is computationally expensive, especially with a distributed memory parallelization, because it requires communication with neighboring ranks. Especially with the cluster merging feature, we would need to compute this for every iteration of algorithm 10. We can prove that enforcing a maximal cluster id does not destroy the maximum difference property.

Theorem 2. *Let C be a clustering that satisfies the maximum difference property (definition 2). We enforce a maximum cluster id of m, for example, using the function* `enforceMaxClusterId`. *The resulting clustering is $\hat{C}$. It again satisfies the maximum difference property.*

Proof. Let $l_1, l_2 \in C$ be the original cluster id of two elements that are in arbitrary neighboring clusters. After enforcing the maximum cluster id of m, the cluster ids are given by $\hat{l}_1 = \min(l_1, m)$ and $\hat{l}_2 = \min(l_2, m)$. We denote the resulting clustering by $\hat{C}$.

As the clustering C fulfills the maximum difference invariance, the difference between l_1 and l_2 can be at most one. Hence, we need to consider three cases.

Case 1: $l_1 = l_2$
We compare l_1 to the maximum cluster id m.

Case 1.i: $l_1 \geq m$
The new cluster ids $\hat{l}_1 = \min(l_1, m) = m$ and $\hat{l}_2 = \min(l_2, m) = m$ are identical and thus satisfy the invariance.

Case 1.ii: $l_1 < m$
The new cluster ids $\hat{l}_1 = \min(l_1, m) = l_1$ and $\hat{l}_2 = \min(l_2, m) = l_2 = l_1$ do not change after enforcing the maximum cluster id.

Case 2: $l_2 = l_1 + 1$

Case 2.i: $l_1 \geq m$
The new cluster ids $\hat{l}_1 = \min(l_1, m) = m$ and $\hat{l}_2 = \min(l_2, m) = \min(l_1 + 1, m) = m$ are identical and thus satisfy the invariance.

Case 2.ii: $l_1 < m$
Here, $l_2 = l_1 + 1 \leq m$ and thus does change after enforcing the maximum cluster id. The new cluster ids $\hat{l}_1 = \min(l_1, m) = l_1$ and $\hat{l}_2 = \min(l_2, m) = \min(l_1 + 1, m) = l_1 + 1$ satisfy the invariance.

Case 3: $l_1 = l_2 + 1$
This case follows from case 2 by swapping l_1 and l_2.

Hence, as the maximum difference invariance is preserved for all possible cases, we conclude that enforcing a maximum cluster id respects the maximum difference invariant. Furthermore, as the case 1 showed, if both elements were in the same cluster before merging, they still share the same cluster ($\hat{l}_1 = \hat{l}_2$) after merging. □

6.9. Discussion

In this chapter, we have presented a modern implementation of local time-stepping. We followed a clustered approach (section 6.1), better suited for high-performance computing.

We used the ADER-DG numerical scheme, which, as seen in section 6.2, can naturally be used as part of an LTS scheme. Due to our modeling efforts, we have achieved a simple yet efficient method. The simplified time-stepping loop, as shown in section 6.3, resulted in a straightforward algorithm without manual scheduling. Our new LTS model is state-machine based (section 6.4) and thus easy to understand. Furthermore, the state machine (figure 6.2) excludes a category of bugs because it prevents illegal state transitions. Clusters are synchronized by explicit message passing, which makes dependencies obvious and clearly shows the flow of information. Our new model schedules clusters by comparing the number of normalized time steps they take, which is more robust than comparing the simulation time. Adding ADER-DG kernels to our scheduling with parallel mesh traversals was simple (section 6.5) as we were able to use the same computational logic that we presented in section 4.4. We have demonstrated (section 6.6) how our abstraction can model both the scheduling of computations and the scheduling of the communication, resulting in an elegant distributed memory parallelization. The introduction of copy and ghost layers made it necessary to create a more complicated time-stepping loop. Even with these added factors, the resulting algorithm is still relatively simple (section 6.7). Furthermore, as described in section 6.8, we added functionality to automatically fine-tune the LTS clustering by introducing a wiggle factor λ and by introducing the automatic merging of LTS clusters.

Chapter 7.

Earthquake-Tsunami Coupling

Tsunamis are created by abrupt vertical perturbations of a body of water. Multiple different sources, such as landslides or earthquakes, can cause this. In this thesis, we focus on earthquakes as a source of tsunamis.

During an earthquake, the seafloor moves dynamically in response. This generates seismic waves, ocean acoustic waves, and tsunami waves. Far away from the source, these waves separate, but close to the source, they are superimposed. As the tsunami wave is typically assumed to be the most interesting, many models focus only on this wave [3, 131]. This restriction can become a problem, as instruments such as ocean bottom pressure sensors and off-shore cabled sensor networks are deployed in regions that are in the source area, potentially improving tsunami early warning systems [78, 143, 185].

The current state-of-the-art is using a separate model for earthquakes and tsunamis [102, 178]. The earthquake can be approximated by an analytical solution for the displacement, which typically assumes a homogeneous elastic half-space [114]. For more complex geometries or velocity models, simulations become necessary. For the tsunami, a two-dimensional model and simulation code is typically used [131].

Recently, fully coupled models have emerged that consider the entire wavefield in both Earth and ocean [3, 97, 98, 99, 103]. Chapter 2 described our three-dimensional fully coupled model, which can resolve wavefield features, including tsunami dispersion and acoustic waves in the ocean. As mentioned above, acoustic waves are increasingly important when comparing with measurements. Furthermore, it is well known that dispersion can affect tsunami propagation when the tsunami wavelength is not much longer than the sea depth [131]. For an overview of this, we refer the interested reader to the discussion and the case studies of [49]. Finally, the fully coupled model considers the two-way interaction between earthquake and tsunami, i.e., wave propagation in the ocean can influence wave propagation in the Earth. Hence, it can be used as a reference model because it carries fewer assumptions. It is ideally suited to conduct detailed studies of the complex wavefield close to the earthquake source.

This section gives a brief overview of one-way linked coupling strategies. First, we derive the shallow water equations (section 7.1), a standard two-dimensional model for tsunamis. Next, in section 7.2, we discuss strategies that can be used to couple earthquake and tsunami solvers. Furthermore, we evaluate the strategies with example scenarios and show when each coupling strategy is applicable. In section 7.3, we explain how to approximate the sea surface height from the seafloor displacement. Finally, in section 7.4, we discuss the results and summarize the strengths and weaknesses of the approaches.

7.1. Shallow Water Equations

For tsunami simulations, one common approach is to use the linear long wave equations, also called linear shallow water equations. They can be derived by starting from the ocean part of our fully coupled model, given by section 2.2.2. Our derivation follows the one in [3, 131]. We use the same coordinate system as introduced for the fully coupled model (chapter 2), where the ocean floor is at height $z = -H$ and the ocean surface at rest is at $z = 0$.

First, we assume the ocean is incompressible, meaning $K \to \infty$. Then, equation (2.70) reduces to $\nabla \cdot \boldsymbol{v} = 0$, which is the standard continuity equation for an incompressible fluid. We integrate this in depth, using the linearized boundary conditions on the seafloor (equation (2.90)) and sea surface (equations (2.88) and (2.89)). Introducing the depth-integrated velocities

$$q_1 = \int_{-H}^{0} v_1 \,\mathrm{d}z, \tag{7.1}$$

$$q_2 = \int_{-H}^{0} v_2 \,\mathrm{d}z, \tag{7.2}$$

we arrive at the equation governing the sea surface perturbation

$$\frac{\partial \eta}{\partial t} + \frac{\partial q_1}{\partial x} + \frac{\partial q_2}{\partial y} = \frac{\partial b}{\partial t}, \tag{7.3}$$

where b is the time-dependent bathymetry.

Second, we assume that $\rho \frac{\partial v_3}{\partial t} = 0$, i.e., that the vertical accelerations are negligible. Inserting this into the vertical part of the momentum equation (equation (2.78)) gives the pressure perturbations as $p(x, y, z, t) = \rho g \eta$, i.e., the pressure is in hydrostatic equilibrium. We then insert this pressure into the horizontal part of the momentum equation and again integrate in depth. This results in the equations

$$\frac{\partial q_1}{\partial t} + gH \frac{\partial \eta}{\partial x} = 0, \tag{7.4}$$

$$\frac{\partial q_2}{\partial t} + gH \frac{\partial \eta}{\partial y} = 0, \tag{7.5}$$

which, together with equation (7.3) form the shallow water equations. They describe non-dispersive tsunami propagation. Note that the pressure is still assumed to be hydrostatic and thus depends on z. We can use this to reconstruct a three-dimensional pressure field.

The shallow water equations are often used to simulate tsunamis [102, 178]. When non-linear effects are dominant, for example, near the shore, a non-linear version of the shallow water equations must be used. They can contain terms that handle inundation and run-up [3, 131]. However, this is not in the scope of this thesis. The non-linear version of the equations is obtained by depth-integrating the Euler equations (section 2.2.1) and is, for example, described in [95, Sec. 18.7]. For the rest of this chapter, we only consider the linear shallow water equations, but we will compare our fully coupled model with the non-linear equations later.

7.2. One-Way Coupling Approaches

We introduced the shallow water equations in the previous section. Now, we discuss how to source the tsunami. What kind of initial conditions and source terms do we need? The earthquake simulation results in a moving seafloor, which we can model as a time-dependent bathymetry perturbation $b(x, y, t)$. Initially, the seafloor is at depth $-H(x, y)$, which is perturbed by the earthquake to $z = -H(x, y) + b(x, y, t)$. This seafloor displacement then causes a sea surface displacement. This section investigates how to set these terms, following the discussion in [3] closely.

We first consider methods that work with the seafloor displacement, which can be either computed by an analytical solution or with a numerical earthquake simulation. If this displacement is available as a time series, we can include it as the source term $\frac{\partial b}{\partial t}$ in equation (7.3).

When the displacement is only available at one point in time, e.g., given by the static displacement of an earthquake simulation or from an analytical solution, we need to assume that the earthquake happens instantaneously with a displacement of b^{static}. Formally, an instantaneous uplift corresponds to a change in bathymetry of

$$\frac{\partial b}{\partial t} \approx b^{\text{static}}(x, y)\, \delta(t). \tag{7.6}$$

Integrating equation (7.3) in time with equation (7.6) as the source term and assuming zero velocities leads to the initial condition

$$\begin{aligned} \eta(x, y, t = 0) &= b^{\text{static}}(x, y) \\ q_1(x, y, t = 0) &= 0 \\ q_2(x, y, t = 0) &= 0, \end{aligned} \tag{7.7}$$

where we assumed that the time t is shifted such that $t = 0$ marks the start of the tsunami simulation. Then, we can run the tsunami simulation with the initial condition given by equation (7.7) and by setting the term $\frac{\partial b}{\partial t} = 0$.

We need to discuss the validity of our initial conditions for the velocities and η. While some studies [145, 146, 147] suggest that it is beneficial to include an initial horizontal velocity, studies with a two-dimensional version of the fully coupled model revealed that this leads to negligible differences [98, 99]. Furthermore, using the seafloor uplift as an initial condition for the sea surface height assumes that the ocean does not affect the waves. We will later discuss ways of including this effect in an appropriate manner. We also need to make similar considerations for the time-dependent coupling.

Both approaches discussed thus far have the limitation that they neglect the compressibility of the ocean. Hence, they do not permit the propagation of acoustic waves.

Recently, [131, 133, 134] proposed a one-way coupling strategy to accomplish this. The idea of this model is simple: Instead of recording the seafloor displacement, the method records the sea surface displacement. For this, an earthquake solver must be used that includes an acoustic layer to incorporate the effects of the ocean. The resulting model is similar but not identical to our fully coupled model, which we proposed in section 2.3, as it uses a standard free surface boundary condition on the sea surface, i.e., it assumes that $g = 0$. Then, the displacement on the sea surface, which is generally different from the seafloor displacement, is used as a source term in the shallow water simulation. This approach is called the superposition approach because it decomposes the tsunami into gravity waves (simulated by a two-dimensional shallow water solver) and acoustic wave propagation in the ocean (simulated by a three-dimensional water layer). It follows directly—assuming a linear shallow water solver—from our fully coupled model with a few assumptions. We refer the interested reader to [3] for a derivation of this method. The source term looks similar to the seafloor-based coupling discussed before. We replace the term $\frac{\partial b}{\partial t}$ with the term $\frac{\partial \eta^{g=0}}{\partial t}$ in equation (7.3), where $\eta^{g=0}$ is the sea surface displacement from an earthquake simulation without gravity.

An advantage of the superposition model over the fully coupled model is that it allows more flexibility: The shallow water solver can also be a non-linear model. In this case, the theoretical justification of the superposition decomposition does not hold anymore. However, it works in practice [132]. A non-linear shallow water solver can also include effects such as inundation. These effects, which the fully coupled model cannot resolve, are essential to accurately describe tsunami propagation near the coast.

To summarize, following [3], we define four different modeling strategies:

Method 1 is our fully coupled model.

Method 2 is a one-way linking of an earthquake model with a two-dimensional non-dispersive shallow water solver (equations (7.3) and (7.4)), using equation (7.7) as initial conditions.

Method 3 is the same as method 2 but uses the seafloor displacement $\frac{\partial b}{\partial t}$ as the forcing term instead of an initial condition.

Method 4 is the same as method 3 but uses the sea surface displacement $\frac{\partial \eta^{g=0}}{\partial t}$ as the forcing term.

Next, following [3], we define three scenarios that can be used to compare the different modeling strategies. For all scenarios, we prescribe a vertically moving seafloor with velocity

$$\frac{\partial b}{\partial t} = \frac{1}{\sigma_t \sqrt{2\pi}} \exp\left(-\frac{(t-4\sigma_t)^2}{2\sigma_t^2}\right) \exp\left(-\frac{x^2+y^2}{2\sigma_r^2}\right), \tag{7.8}$$

where the width σ_r and the duration σ_t characterize the source. Equation (7.8) reaches its maximum at time $t = 4\sigma_t$. At $t = 0$, the uplift rate is effectively zero, which allows us to start with an initially unperturbed seafloor. We set the ocean depth $H = 4\,\text{km}$,

Table 7.1.: Table recreated from [3]. Three scenarios prescribe a moving seafloor with velocities given by equation (7.8). The source width σ_r and the source duration σ_t vary between the scenarios. This results in different characteristics which directly impact the validity of coupling methods. The criteria derived in [3] characterize the properties of the source using non-dimensional values.

	Source width σ_r [km]	Source duration σ_t [s]	Instantaneous source? $\sqrt{gH}\sigma_t/\sigma_r \ll 1$	Negligible acoustic wave excitation? $H/(c\sigma_t) \ll 1$	Shallow water limit? $H/\sigma_r \ll 1$
Scenario 1	12.5	125	✗	✓	✓
Scenario 2	12.5	1.25	✓	✗	✓
Scenario 3	1.25	1.25	✓	✗	✗

Table 7.2.: Simulation parameters as used in [3]. The scenarios require a different domain of size $L \times L$ in which the elements have a characteristic length of h. In a cube of size $L^r \times L^r$, centered at the origin, the mesh is refined to a characteristic length h^r. The simulations run until $t = t^s$.

	L[km]	L^r[km]	h[km]	h^r[km]	t^s[s]
Scenario 1	500	300	75	1.50	800
Scenario 2	200	110	75	1.00	400
Scenario 3	400	40	25	0.25	150

acoustic wave speed $c = 1.5\,\mathrm{km\,s^{-1}}$ and gravitational acceleration $g = 9.81\,\mathrm{m\,s^{-2}}$ for all scenarios. The scenarios differ only in the source parameters σ_r and σ_t, found in table 7.1. Scenario 1 is a long-duration source requiring a time-dependent seafloor uplift. Scenario 2 is an impulse source with a wide source area leading to acoustic waves. Finally, scenario 3 is a narrow impulse source that produces short wavelengths and thus likely has significant dispersion and filtering effects from the ocean response.

We vary the refinement, domain size, and simulation time for the scenarios so that important features of the wavefield are visible. Each setup has a refinement zone in which a uniform mesh size is used. Outside of the refinement region, the mesh is coarsened. Table 7.2 shows the simulation parameters.

Figure 7.1 shows the results. For scenario 1, which uses a long-duration source, all methods except method 2, which assumes an instantaneous source, are valid. Scenario 2 contains a source that generates acoustic waves. Only method 1 (fully coupled) and method 4 (superposition) are valid, as they capture this wave type. Note that all methods manage to capture the overall tsunami well. Finally, scenario 3 contains both acoustic waves and dispersion effects. As for scenario 2, the former implies that methods 2 and 3 are no longer valid, as they do not include acoustic waves. Due to the latter, only the fully coupled model (method 1) is valid, as only it includes dispersion effects during tsunami propagation. However, the superposition method (4) could be used with a tsunami solver that includes approximate dispersion [3]. For this scenario, the methods result in strong differences in tsunami amplitudes caused by the filtering of short wavelengths due to the non-hydrostatic ocean response [68]. This example showed the limitations of each model. For an extended discussion, we refer the interested reader to [3].

7.3. The Sea Surface Height

In this section, we discuss how to initialize the sea surface height from an earthquake simulation. We introduce the Tanioka assumption in section 7.3.1, which includes the effects of horizontal displacements into the vertical seafloor displacement. In section 7.3.2, we present transfer functions that approximate the sea surface height from the seafloor displacement. Furthermore, we show that our fully coupled model already includes these approximations.

7.3.1. Tanioka

The coupling strategies that we have explained so far only consider the vertical part of the displacement at the seafloor. However, when the bathymetry is not flat, horizontal displacement can lead to vertical flow. This can have a strong effect on the resulting tsunami, as shown, for example, by the 2011 Tōhoku-Oki earthquake, which excited a large horizontal displacement [67]. Hence, we should also include the effect of this horizontal displacement in the computation of the sea surface height.

We can approximate this effect with the Tanioka correction factor

$$b(x, y, t) \approx \left(u_1 \frac{\partial H}{\partial x} + u_2 \frac{\partial H}{\partial y} + u_3 \right) |_{z=-H}, \tag{7.9}$$

which assumes that the bathymetry varies smoothly, i.e., that $\left(\frac{\partial H}{\partial x}\right)^2 + \left(\frac{\partial H}{\partial y}\right)^2 \ll 1$ [152]. Equation (7.9) can be directly used to include the effect of horizontal displacements for a sloping ocean floor. For a more detailed discussion and applications, we refer the interested reader to the discussion in [131, Sec. 5.4.1] and the references therein.

This factor is automatically included in our fully coupled model, as equation (7.9) follows directly from time-integrating the linearized kinematic boundary condition (equation (2.90)) on the ocean floor [3].

7.3.2. Filtering & Transfer Functions

The one-way coupling method requires approximating the sea surface height from the seafloor displacement. The naive approach of using the same displacement for both neglects the effect that the water has on the waves. A better solution is to use an approximate model for the ocean response. We follow the framework described in [3]: transfer functions. In the following, we denote the Fourier-transformed version of a function f by $\hat{f}$. It changes the variables from the space-time coordinates (x, y, t) to the

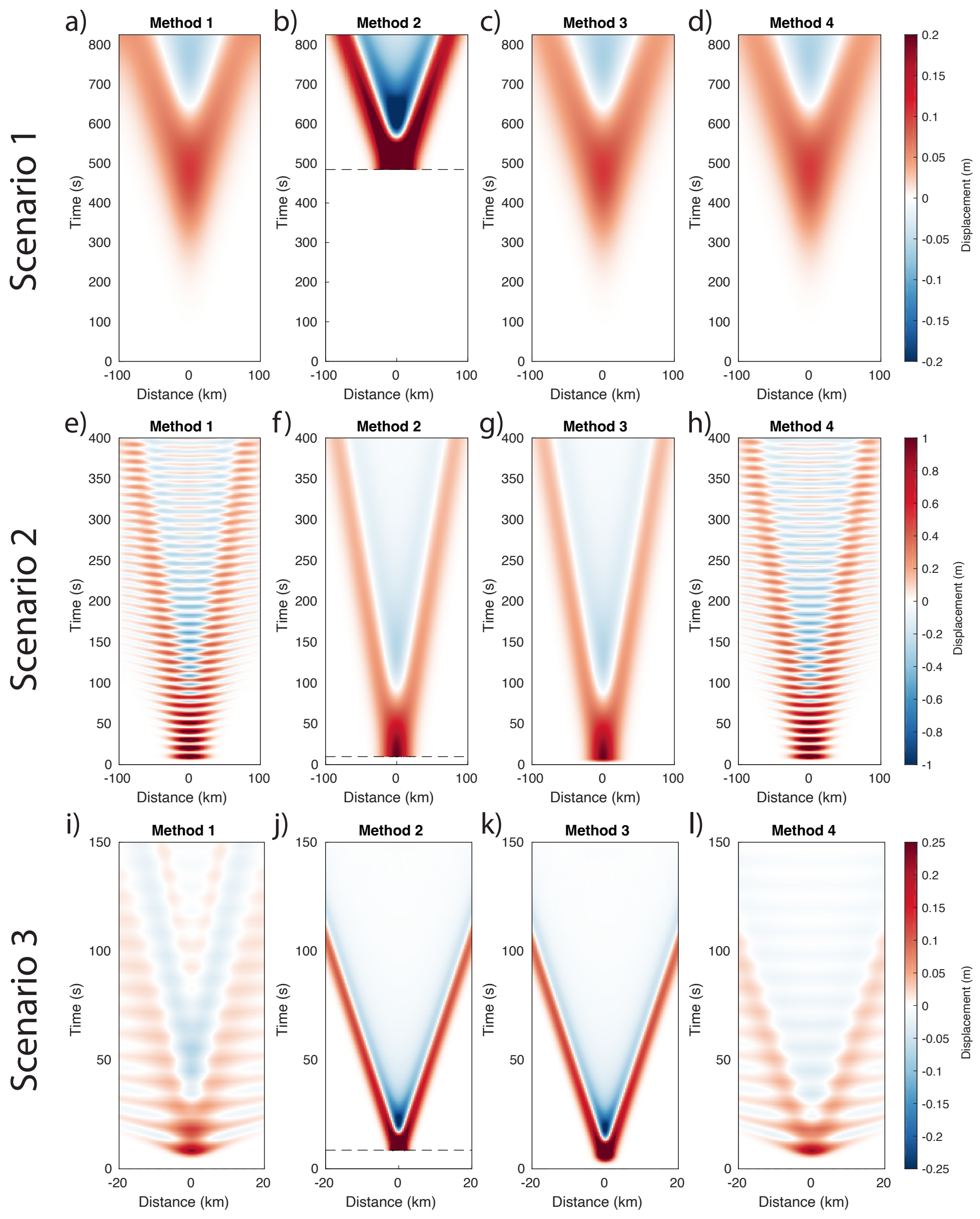


Figure 7.1.: Figure taken from [3]. Sea surface uplift from seafloor uplift given by equation (7.8) with parameters from table 7.1. Method 1 is our fully coupled model, method 2 is one-way linking with an instantaneous source, method 3 is one-way linking with time-dependent sourcing, and method 4 is the superposition coupling.

set of coordinates (k_x, k_y, ω), consisting of the horizontal wavenumbers k_x and k_y, and the angular frequency ω. We define the Fourier transform and its inverse as

$$\hat{f}(k_x, k_y, \omega) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y, t) \exp\left(-i(k_x x + k_y y - \omega t)\right) \mathrm{d}x\, \mathrm{d}y\, \mathrm{d}t, \tag{7.10}$$

$$f(x, y, t) = (2\pi)^{-3} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{f}(k_x, k_y, \omega) \exp\left(i(k_x x + k_y y - \omega t)\right) \mathrm{d}k_x\, \mathrm{d}k_y\, \mathrm{d}\omega. \tag{7.11}$$

Using the radial wavenumber $k = \sqrt{k_x^2 + k_y^2}$, the transfer function

$$T(k, \omega) = \frac{\hat{\eta}(k, \omega)}{\hat{b}(k, \omega)}, \tag{7.12}$$

relates the Fourier transformed bathymetry $\hat{b}$ to the Fourier-transformed sea surface $\hat{\eta}$. Note that the transfer function only depends on k, which is required by the translational invariance of the ocean response. We can use the transfer functions to compute the sea surface height from the bathymetry: First, we compute the Fourier transform of the bathymetry using equation (7.10). Second, we use equation (7.12) to compute the Fourier-transformed sea surface from the bathymetry. Third, we use equation (7.11) to compute the sea surface displacement. Only one question remains: How do we choose the transfer function?

The acoustic part of our fully coupled model, discussed in section 2.2.2, corresponds to the transfer function

$$T_{\mathrm{fc}}(k, \omega) = \frac{1}{\cosh(k^* H) - (g k^*/\omega^2)\sinh(k^* H)} \tag{7.13}$$

with $k^* = \sqrt{k^2 - \omega^2/c^2}$ [3]. Starting from this, we can investigate several special cases.

We now assume the ocean is incompressible, corresponding to the $k^* \to k$ limit. The assumption of incompressibility removes acoustic waves entirely from the model. When we make the further assumption that the ocean uplift is instantaneous, i.e., $\omega \to \infty$, we arrive at

$$T_{\mathrm{inst}}(k) = \frac{1}{\cosh(kH)}, \tag{7.14}$$

which is called the Kajiura filter [68]. This filter is widely used to compute the initial sea surface height for a tsunami simulation [131]. It is a low-pass filter that removes short-wavelength components. Hence, the initial sea surface height is smoother than the seafloor perturbation [131]. While this filter was derived for an ocean with uniform depth, in practice, it can also be applied to scenarios with small depth variations.

When we assume incompressibility and the long wave limit, i.e., $kH \ll 1$, equation (7.13) reduces to the tranfer function for the linear shallow water equations

$$T_{\mathrm{LLW}}(k, \omega) = \frac{1}{1 - gHk^2/\omega^2}, \tag{7.15}$$

which lacks the $(\cosh(kH))^{-1}$ filter. Hence, all wavelengths contained in the seafloor displacement directly transfer to the sea surface. We can apply equation (7.14) before using equation (7.15), which corresponds to the workflow of first applying a filter to the bathymetry before using a tsunami simulation software.

Finally, as discussed earlier, the superposition model (method 4) runs a fully coupled simulation without gravity, followed by a shallow water solver. Thus, due to the sequential nature of the coupling, we can factor the transfer function of the superposition model. The ocean part of the fully coupled simulation in the absence of gravity corresponds to equation (7.13) with $g = 0$ and is given by

$$T_{g=0}(k,\omega) = \frac{1}{\cosh(k^* H)}. \tag{7.16}$$

It is possible to include approximate models of dispersive tsunami propagation by using a Boussinesq solver for the tsunami part. Here, we assume that a linear shallow water equation solver is used for simplicity. We have that

$$T_{\text{sup}}(k,\omega) = \underbrace{\frac{1}{\cosh(k^* H)}}_{=T_{g=0}(k,\omega)} \underbrace{\frac{1}{1 - gHk^2/\omega^2}}_{=T_{\text{LLW}}(k,\omega)}. \tag{7.17}$$

In the incompressible limit ($k^* \to k$), equation (7.16) reduces to the Kajiura filter given by equation (7.14). Thus, the superposition model includes the non-hydrostatic ocean response. We refer the interested reader to the extended discussion in [3].

These transfer functions give us an easy way of analyzing the coupling strategies and result in a simple way of converting the seafloor displacement to an initial tsunami height [131]. We can combine this approach with the Tanioka approximation discussed in section 7.3.1 by convolving equation (7.14) with the result of equation (7.9). This includes both the effect of a sloping seafloor and the effect of the non-hydrostatic ocean response.

7.4. Discussion

We showed how we can derive the shallow water equations in section 7.1, and presented and evaluated four different methods for earthquake-tsunami simulations (section 7.2). Section 7.3 completed the workflow by providing approximations that can be used to convert the seafloor displacement into an initial tsunami height.

What has become clear in these sections is that the fully coupled model is the most general out of all considered models. It includes the full physical effect of the earthquake, wave propagation in the compressible ocean, and tsunami propagation. While acoustic waves often have a very small effect on the tsunami, they can dominate the measurements. Hence, including them in our model allows us to compare better with measurements. Furthermore, our model is fully three-dimensional and does not assume that the pressure is in hydrostatic equilibrium. Thus, it gives us a complete three-dimensional wavefield.

Simpler coupling strategies can capture critical parts of the tsunami. If acoustic waves are relevant but dispersion is not, the superposition method can be an adequate choice. However, its computational cost is not much smaller than the cost of the fully coupled model, as both models require three-dimensional ocean simulation. The computational overhead of the fully coupled model only stems from the gravitational boundary condition, which we can discretize efficiently, as shown in section 4.3. The superposition coupling has other advantages. It allows using an existing shallow water solver (with minimal additions for sourcing) and a standard earthquake solver, assuming it can also simulate acoustic waves. Hence, the superposition method is easier to implement than the fully coupled model. Furthermore, as in all one-way coupling strategies, the shallow water model can be non-linear and can thus include additional physical effects such as inundation, which leads to better results near the coast.

We can use a seafloor displacement-based coupling strategy when acoustic waves are irrelevant. As mentioned in section 7.3, it is advisable to use the Tanioka filter (equation (7.9)) and the Kajiura filter (equations (7.12) and (7.14)) to account for a sloping seafloor and the non-hydrostatic ocean response. The resulting coupling is cheap and can be used with a non-linear shallow water model.

To summarize, the fully coupled model should be considered as a reference model for off-shore earthquake-induced tsunamis. Often, we can use a compromise between the expensive fully coupled model and the cheaper one-way linking models. We use the fully coupled model for the first part of the simulation until the earthquake has stopped and after most acoustic waves have left the domain. Then, we switch to a shallow water solver using the result of the fully coupled model as the initial condition. This gives us the best of both worlds: We get an accurate result for the tsunami generation and then capture the tsunami, which happens on a much longer time scale, with an approximate model.

Chapter 8.

Verification

In this chapter, we verify our fully coupled solver using a variety of analytical solutions. It is well known that we can expect a convergence order of $q = N + 1$ for a DG scheme with polynomial order N [61]. Hence, asymptotically, the error should be of size $\mathcal{O}(h^q)$, where h is the characteristic element length. We consider two meshes: A coarser one with a characteristic length of h_i that leads to an error of e_i and a refined one with a characteristic length of h_{i+1} and an error of e_{i+1}. We measure the error between the numerical solution q_i and the analytical solution q with the integral

$$e = \left(\int_\Omega |q_i - q|^2 \, \mathrm{d}\boldsymbol{x} \right)^{1/2}, \tag{8.1}$$

which is the standard L^2 norm. We only consider the solution for σ_{11}. The numerical order of convergence converges to

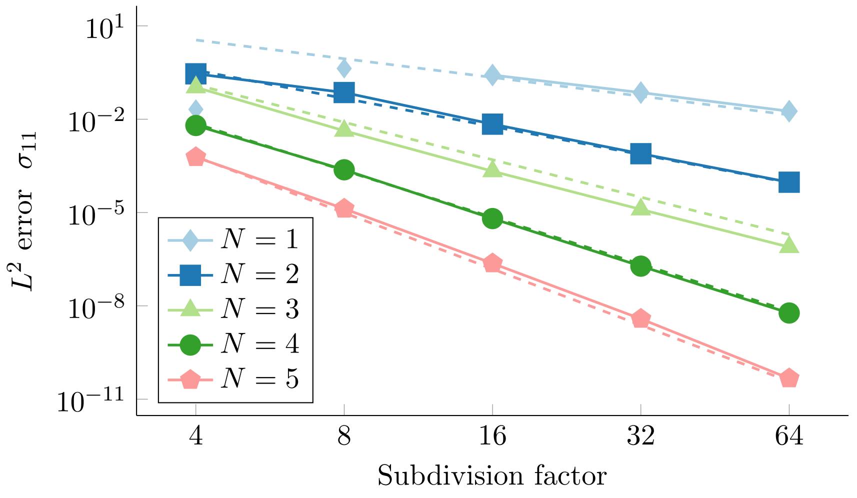
$$q_i \approx \frac{\log\left(e_{i+1}/e_i\right)}{\log\left(h_{i+1}/h_i\right)} \tag{8.2}$$

for $i \to \infty$. We use the linear regression

$$\log(e) \approx m + \hat{q} \log(h), \tag{8.3}$$

to predict the error e from h. The coefficients m, which summarizes constant factors, and $\hat{q}$, the average convergence order, can be found by a linear least squares procedure. We exclude errors in this estimate which are not in the asymptotic region, i.e., h is too small. We also exclude errors near the precision with which we can calculate the analytical solution. In this chapter, we use the Python library statsmodels [139] to fit equation (8.3).

We begin by showing a planar wave solution in section 8.1, demonstrating the convergence of the elastic and acoustic solvers. This setup uses a periodic boundary condition, showing that our scheme simulates the propagation over multiple wavelengths without dissipating too much energy. Next, we show that the elastic-acoustic coupling works using two test cases. The first, Snell's law at the elastic-acoustic interface (section 8.2), considers the reflection of an incoming wave at this material interface. The second, the Scholte wave scenario (section 8.3), shows that our model can capture interface waves correctly.



N	q
1	1.9
2	3.0
3	4.3
4	5.0
5	5.9

Figure 8.1.: Convergence plot for the elastic planar wave. Solid lines indicate the range of errors used to compute the average convergence order. For $N = 2$, we included only subdivision factors larger than 8. The performance for larger factors is an outlier and does not reflect the obtained error accurately. The dashed lines indicate the optimal convergence. We achieved nearly optimal numerical orders of convergence for all considered polynomial orders.

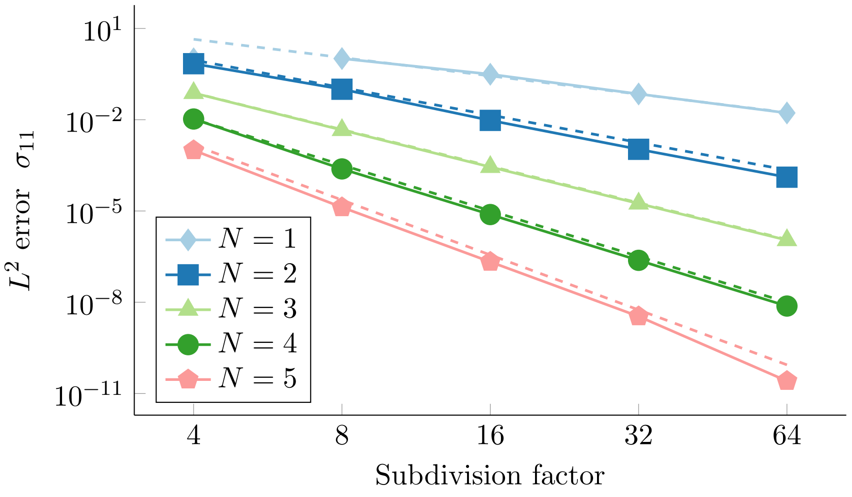
Finally, we use a three-dimensional ocean model (section 8.4) to show that our implementation of the gravitational boundary condition is correct. Combining these scenarios, we show that our implementation achieves high-order convergence for fully coupled elastic-acoustic tsunami simulations. We use a time step size with a factor of $C(N) = 0.4(2N + 1)^{-1}$ in equation (4.47), a rate-2 LTS scheme, double precision, and SeisSol version 1.0.1 [167].

8.1. Planar Waves

The plane wave ansatz is a simple analytical solution for constant coefficient linear partial differential equations. The idea is to express the solution in a representation similar to a Fourier basis [41, Sec. 4.2.1]. It has been applied to SeisSol to test the accuracy for various rheological models, for example, in [35, 181]. We apply this approach to the elastic and acoustic wave equations in this section. We assume an complex ansatz of

$$\boldsymbol{q}(\boldsymbol{x}, t) = \exp\left(i(\omega t - \boldsymbol{k} \cdot \boldsymbol{x})\right) \boldsymbol{q_0}. \tag{8.4}$$

We need to find parameters for the angular frequency ω, and the initial state $\boldsymbol{q_0}$ for an arbitrary direction $\boldsymbol{k} = (k_1, k_2, k_3)$. We derive an equation for these parameters by inserting equation (8.4) into our system of hyperbolic PDEs (equation (2.29)). Using the



N	q
1	2.0
2	3.1
3	4.0
4	5.1
5	6.2

Figure 8.2.: Convergence plot for the acoustic planar wave. Solid lines indicate the range of errors used to compute the average convergence order. For $N = 2$, we included only subdivision factors larger than 4. The dashed lines indicate the optimal convergence. We achieved the optimal numerical order of convergence for all considered polynomial orders.

derivatives of equation (8.4),

$$\frac{\partial \boldsymbol{q}}{\partial t} = \omega i \boldsymbol{q}, \quad \frac{\partial \boldsymbol{q}}{\partial x} = -ik_1 \boldsymbol{q}, \quad \frac{\partial \boldsymbol{q}}{\partial y} = -ik_2 \boldsymbol{q}, \quad \frac{\partial \boldsymbol{q}}{\partial z} = -ik_3 \boldsymbol{q}, \tag{8.5}$$

we arrive at

$$(i\omega - \boldsymbol{A} ik_1 - \boldsymbol{B} ik_2 - \boldsymbol{C} ik_3) \underbrace{\exp\left(i(\omega t - \boldsymbol{k} \cdot \boldsymbol{x})\right) \boldsymbol{q_0}}_{=\boldsymbol{q}} = 0. \tag{8.6}$$

After dividing by $\exp\left(i(\omega t - \boldsymbol{k} \cdot \boldsymbol{x})\right)$, we arrive at

$$(i\omega - \boldsymbol{A} ik_1 - \boldsymbol{B} ik_2 - \boldsymbol{C} ik_3)\, \boldsymbol{q_0} = 0. \tag{8.7}$$

Finally, multiplying with i and reordering results in the eigenproblem

$$\underbrace{(\boldsymbol{A} k_1 + \boldsymbol{B} k_2 + \boldsymbol{C} k_3)}_{=\hat{\boldsymbol{A}}} \boldsymbol{q_0} = \omega \boldsymbol{q_0}, \tag{8.8}$$

where the goal is finding eigenvalues ω and their corresponding eigenvectors $\boldsymbol{q_0}$ of the plane-wave operator $\hat{\boldsymbol{A}}$ which is the same plane-wave operator (equation (2.31)) that we considered in the discussion of Riemann problems.

We only consider the real part of the solution, which reduces to

$$\begin{aligned} \Re\left(\exp(i(\omega t - \boldsymbol{k} \cdot \boldsymbol{x}))\boldsymbol{q_0}\right) = {} & \cos\left(\Re(\omega) t - \boldsymbol{k} \cdot \boldsymbol{x}\right) \exp\left(-\Im(\omega) t\right) \Re(\boldsymbol{q_0}) \\ & - \sin\left(\Re(\omega) t - \boldsymbol{k} \cdot \boldsymbol{x}\right) \exp\left(-\Im(\omega) t\right) \Im(\boldsymbol{q_0}). \end{aligned} \tag{8.9}$$

We used the notation $\Re(\cdot)$ for the real and $\Im(\cdot)$ for the imaginary part of a complex number. Equation (8.9) is a solution of the PDE because it is a linear combination of solutions. The eigenvalues of the plane wave operator $\hat{\boldsymbol{A}}$ are real because our PDE is hyperbolic [95]. Hence, we can normalize the eigenvectors such that they only have real components. Thus, both ω and $\boldsymbol{q_0}$ can be assumed to be real and hence, equation (8.9) reduces to

$$\boldsymbol{q}(\boldsymbol{x}, t) = \cos\left(\omega t - \boldsymbol{k} \cdot \boldsymbol{x}\right) \boldsymbol{q_0}. \tag{8.10}$$

Note that ω and $\boldsymbol{q_0}$ depend on the material properties. The eigenvalues ω correspond to the waves of the PDEs. We use the solution for both acoustic and elastic materials. We impose a linear combination of a left-going S wave and a right-going P wave for elastic materials. We set $\lambda = 2, \mu = 1$ and $\rho = 1$, leading to wave speeds of $c_p = 2, c_s = 1$.

For the acoustic material, we use a right-going acoustic wave. There, we set $K = 4$ and $\rho = 1$, which leads to a wave speed of $c = 2$. For the choice $\boldsymbol{k} = \boldsymbol{a}\pi$, with $\boldsymbol{a} \in \mathbb{Z}^2$, the solution is periodic in the domain $[-1, 1]^3$. Hence, the simulation returns to the original condition for both materials every $\sqrt{3}$ time units. We run the setup for 10 wave propagations, leading to a simulation time of $\approx$17.32. The domain is divided into $4, \ldots, 64$ cubes, split into tetrahedra.

Figure 8.1 shows the observed errors for the elastic material and figure 8.2 for the acoustic setup. We observe that we reach the optimal convergence order for all polynomial orders. For $N = 1$, we require a subdivision factor of 16 to arrive in the asymptotic region. Thus, our implementation achieves high-order convergence for both elastic and acoustic wave propagation.

8.2. Snell's Law at an Elastic-Acoustic Interface

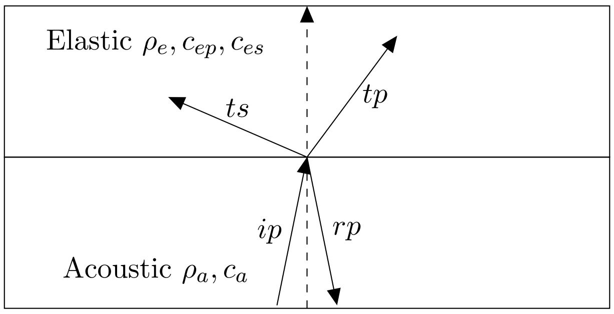


Figure 8.3.: Setup of Snell's law. We have four waves: an incident acoustic wave (ip), a reflected acoustic wave (rp), a transmitted P wave (tp), and a transmitted S wave (ts). The direction vectors are the ones used in our simulation.

We have seen in the previous section that our solver works for both elastic and acoustic media. This section presents a scenario that uses Snell's law at the elastic-acoustic interface to test the elastic-acoustic coupling. It was introduced in [73]. Our description follows [175]. The idea of this analytical solution is simple: We derive a solution for each medium and combine them using interface conditions.

An incident pressure (*ip*) wave travels from the acoustic domain towards the interface. Some parts are reflected as acoustic waves (*rp*). The residual energy enters the elastic domain as a longitudinal wave (*tp*) and a transverse wave (*ts*). Each of these waves has an angle $(\alpha_{ip}, \alpha_{rp}, \alpha_{tp}, \alpha_{ts})$ and strength $(C_{ip}, C_{rp}, C_{tp}, C_{ts})$. Figure 8.3 visualizes the structure of this problem.

Thus, we write the solution as

$$\boldsymbol{u}(\boldsymbol{x}, t) = \begin{cases} \boldsymbol{u_{ip}} + \boldsymbol{u_{rp}} & \text{if in acoustic domain,} \\ \boldsymbol{u_{tp}} + \boldsymbol{u_{ts}} & \text{if in elastic domain.} \end{cases} \tag{8.11}$$

We define the displacements

$$\begin{aligned} \boldsymbol{u_{ip}} &= C_{ip}\boldsymbol{d_{ip}} \cos\left(k_{ap}\left(x\sin(\alpha_{ip}) + z\cos(\alpha_{ip}) - \omega t\right)\right), \\ \boldsymbol{u_{rp}} &= C_{rp}\boldsymbol{d_{rp}} \cos\left(k_{ap}\left(x\sin(\alpha_{rp}) - z\cos(\alpha_{rp}) - \omega t\right)\right), \\ \boldsymbol{u_{tp}} &= C_{tp}\boldsymbol{d_{tp}} \cos\left(k_{ep}\left(x\sin(\alpha_{tp}) + z\cos(\alpha_{tp}) - \omega t\right)\right), \\ \boldsymbol{u_{ts}} &= C_{ts}\boldsymbol{d_{ts}} \cos\left(k_{es}\left(x\sin(\alpha_{ts}) + z\cos(\alpha_{ts}) - \omega t\right)\right), \end{aligned} \tag{8.12}$$

with directions

$$\begin{aligned} \boldsymbol{d_{ip}} &= \left(\sin(\alpha_{ip}), 0, \cos(\alpha_{ip})\right)^T, \\ \boldsymbol{d_{rp}} &= \left(\sin(\alpha_{rp}), 0, -\cos(\alpha_{rp})\right)^T, \\ \boldsymbol{d_{tp}} &= \left(\sin(\alpha_{tp}), 0, \cos(\alpha_{tp})\right)^T, \\ \boldsymbol{d_{ts}} &= \left(-\cos(\alpha_{ts}), 0, \sin(\alpha_{ts})\right)^T. \end{aligned} \tag{8.13}$$

We set $\omega = 2\pi$. The acoustic material has a density and wave speed of $\rho_a = c_{ap} = 1$. The elastic material has a density of $\rho_e = 1$ and wave speeds of $c_{ep} = 3$ and $c_{es} = 2$.

We assume that the incident pressure wave arrives at an angle of $\alpha_{ip} = 0.2$. Next, we apply Snell's law to all waves and arrive at

$$\begin{aligned} \sin(\alpha_{ip})/c_{ap} &= \sin(\alpha_{rp})/c_{ap}, \\ \sin(\alpha_{rp})/c_{ap} &= \sin(\alpha_{tp})/c_{ep}, \\ \sin(\alpha_{tp})/c_{ep} &= \sin(\alpha_{ts})/c_{es}, \end{aligned} \tag{8.14}$$

which defines all angles.

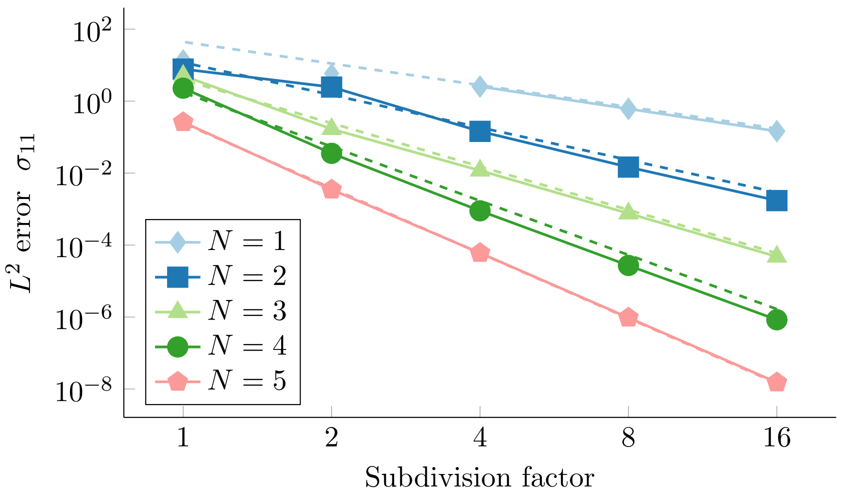
What is left is the computation of the factors C in equation (8.11). We apply the elastic-acoustic interface condition (equation (2.94)). The interface is at $z = 0$ and has the normal vector $\boldsymbol{n} = (0, 0, 1)^T$. With this, the interface conditions are

$$\begin{aligned} \boldsymbol{v_a}(x, y, z = 0) \cdot \boldsymbol{n} &= \boldsymbol{v_e}(x, y, z = 0) \cdot \boldsymbol{n}, \\ \boldsymbol{\sigma_a}(x, y, z = 0) \cdot \boldsymbol{n} &= \boldsymbol{\sigma_e}(x, y, z = 0) \cdot \boldsymbol{n}. \end{aligned} \tag{8.15}$$

We can choose the value for C_{ip} freely. Here, we use $C_{ip} = 1$. The solution of this non-linear system of equations is approximated by

$$\begin{aligned} C_{rp} &= 0.480555914321673987, \\ C_{tp} &= 0.433811888960301440, \\ C_{ts} &= 0.404562109533773730, \end{aligned} \tag{8.16}$$

where all values linearly depend on C_{ip}. We used the computer algebra system Sage [130] to compute the constants. As we use the velocity-stress formulation, we compute the stress tensor and the velocities by analytical differentiation of the displacements (equation (8.11)).



N	q
1	2.1
2	3.2
3	4.1
4	5.3
5	6.0

Figure 8.4.: Convergence plot for Snell's law at the elastic-acoustic interface. Solid lines indicate the range of errors used to compute the average convergence order. For $N = 2$, we included only subdivision factors larger than 2. The dashed lines indicate the optimal convergence. We achieved the optimal numerical orders of convergence for all considered polynomial orders.

We prescribe the analytical solution on all boundaries and let the simulation run until $t = 10$. The domain has size $[-1, 1] \times [-1, 1] \times [-2, 2]$. We divide the domain into cubes. Figure 8.4 shows the results. We achieve the optimal order of convergence for all evaluated polynomial orders. Hence, this scenario shows that our solver can resolve the elastic-acoustic interface satisfactorily.

8.3. Scholte Waves

We discuss an exact solution for Scholte waves in a coupled elastic-acoustic medium in this section. Scholte waves are waves that travel along the elastic-acoustic interface. They decay exponentially away from the interface. Our description follows [73, Sec. 5.2] and [175]. Similarly to the previous section, we define an elastic material ($z < 0$) with material λ_e, μ_e, ρ_e and an acoustic material ($z > 0$) with K_e, ρ_e.

We split the solution into one

$$\begin{aligned} u_1^a(x,y,z,t) &= \Re(ikB_1 \exp(-kb_{ap}z)\exp(i(kx-\omega t))), \\ u_2^a(x,y,z,t) &= 0, \\ u_3^a(x,y,z,t) &= \Re(-1kb_{ap}B_1 \exp(-1kb_{ap}z)\exp(i(kx-\omega t))), \end{aligned} \tag{8.17}$$

for the acoustic and one

$$\begin{aligned} u_1^w(x,y,z,t) &= \Re\left((ikB_2 \exp(kb_{ep}z) - kb_{es}B_3 \exp(kb_{es}z))\exp(i(kx-\omega t))\right), \\ u_2^e(x,y,z,t) &= 0, \\ u_3^e(x,y,z,t) &= \Re\left((kb_{ep}B_2 \exp(kb_{ep}z) + ikB_3 \exp(kb_{es}z))\exp(i(kx-\omega t))\right) \end{aligned} \tag{8.18}$$

for the elastic medium.

We have three body waves: The suffixes *ap* and *ep* denote the P wave in the acoustic and elastic part, and *es* is the S wave in the elastic region. For each of these waves, we define a wave speed

$$c_{ap} = \sqrt{\frac{\lambda_a + 2\mu_a}{\rho_a}}, \quad c_{ep} = \sqrt{\frac{\lambda_e + 2\mu_e}{\rho_e}}, \quad c_{es} = \sqrt{\frac{\mu_e}{\rho_e}}, \tag{8.19}$$

and a decay rate

$$b_{ap} = \sqrt{1 - (c^2/c_{ap}^2)}, \quad b_{ep} = \sqrt{1 - (c^2/c_{ep}^2)}, \quad b_{es} = \sqrt{1 - (c^2/c_{es}^2)}, \tag{8.20}$$

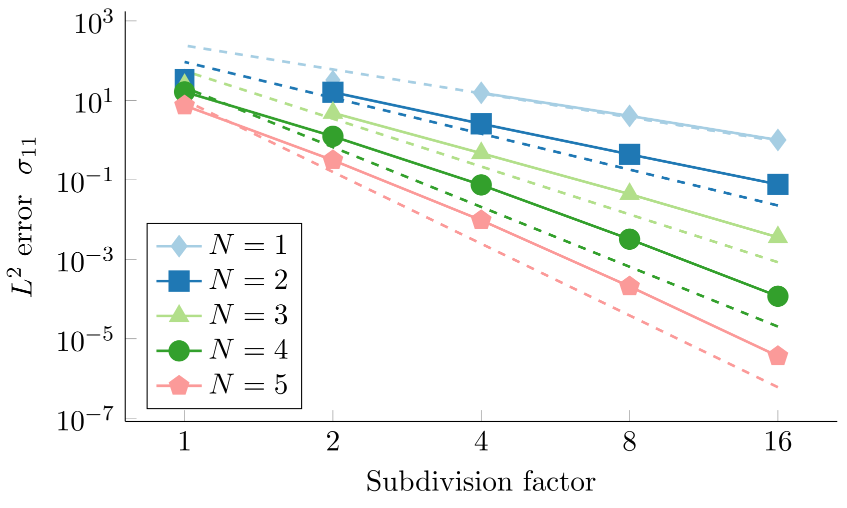
where we introduced the Scholte wave speed c.

Now, we need to combine the solutions for both media. Following [175] inserting the interface conditions (equation (2.94)) into equations (8.17) and (8.18) leads to the coupled system

$$\begin{aligned} 2i\left(1 - \frac{c^2}{c_{ep}^2}\right)^{1/2} B_2 - \left(2 - \left(\frac{c^2}{c_{es}^2}\right) B_3\right) &= 0, \\ \frac{c^2}{c_{es}^2} B_1 + \left(\frac{\rho_e}{\rho_a}\left(2 - \frac{c^2}{c_{es}^2}\right) B_2 + 2i\frac{\rho_e}{\rho_a}\left(1 - \frac{c^2}{c_{es}^2}\right)^{1/2} B_3\right) &= 0, \\ \left(1 - \frac{c^2}{c_{ap}^2}\right)^{1/2} B_1 + \left(1 - \frac{c^2}{c_{ep}^2}\right)^{1/2} B_2 + iB_3 &= 0, \end{aligned} \tag{8.21}$$

which we need to solve for the factors B_1, B_2, and B_3. This system has, in principle, an infinite number of solutions. Hence, following [73], we choose c such that equation (8.21) has a determinant of 0 and thus a non-trivial solution. This requirement leads to the equation

$$\left(\frac{\rho_a}{\rho_e} b_{ep} + b_{ap}\right) r^4 - 4b_{ap}r^2 - 4b_{ap}(b_{ep}b_{es} - 1) = 0, \tag{8.22}$$



N	q
1	2.0
2	2.6
3	3.5
4	4.3
5	5.2

Figure 8.5.: Convergence plot for the Scholte wave scenario. Solid lines indicate the range of errors used to compute the average convergence order. The dashed lines indicate the optimal convergence. We can only report the optimal convergence rate for $N = 2$ for this scenario. However, we can still see exponential convergence for all other orders. Higher orders lead to smaller errors.

where $r = \frac{c}{c_{es}}$, which we can solve to find c. Note that in the case of $\rho_a \to 0$, this equation reduces to the one that determines the Rayleigh wave speed [73].

We use the material parameters $\lambda_a = \rho_a = 1$ for the acoustic and $\lambda_e = \rho_e = \mu_e = 1$ for the elastic region. We solved equations (8.21) and (8.22) with Sage [130], resulting in the constants

$$\begin{aligned} c &= 0.7110017230185816, \\ B_1 &= -0.35944998i, \\ B_2 &= -0.81946427i, \\ B_3 &= 1. \end{aligned} \tag{8.23}$$

We compute the solution in velocity-stress formulation by differentiating equations (8.17) and (8.18).

The domain has size $[-1, 1] \times [-20, 20] \times [-1, 1]$ and is divided into cubes. We prescribe the analytical solution on all boundaries and simulate until $t = 1$. Figure 8.5 shows the results of our convergence study. Our implementation achieves high-order convergence; however, we do not obtain optimal rates for $N > 2$. Nevertheless, higher orders for this scenario have a clear advantage, as they lead to smaller errors.

8.4. Compressible Ocean

We have seen in the previous sections that our implementation works well for simulations of acoustic-elastic coupling. However, we have not yet evaluated the quality of our boundary condition implementation. In this section, we describe a three-dimensional verification test for the acoustic wave equation with gravity. The analytical solution, first described in [3], can be obtained by rotating the two-dimensional solution of [97] around the z axis. It follows from a standard eigenmode-based ansatz, similar to the solution presented in section 8.1. Our derivation follows [3, 97]. We assume a solution of the form

$$\begin{aligned}
p(x,y,z,t) &= \left(\sinh(k_* z) + g\frac{k_*}{\omega^2}\cosh(k_* z)\right)\sin(k_x x)\sin(k_y y)\sin(\omega t),\\
v_1(x,y,z,t) &= \left(\sinh(k_* z) + g\frac{k_*}{\omega^2}\cosh(k_* z)\right)\cos(k_x x)\sin(k_y y)\cos(\omega t)\frac{k_x}{\omega\rho},\\
v_2(x,y,z,t) &= \left(\sinh(k_* z) + g\frac{k_*}{\omega^2}\cosh(k_* z)\right)\sin(k_x x)\cos(k_y y)\cos(\omega t)\frac{k_y}{\omega\rho},\\
v_3(x,y,z,t) &= \left(\cosh(k_* z) + g\frac{k_*}{\omega^2}\sinh(k_* z)\right)\sin(k_x x)\sin(k_y y)\cos(\omega t)\frac{k_*}{\omega\rho},
\end{aligned} \tag{8.24}$$

where k_x, k_y, k_* and ω are parameters.

Integrating the vertical velocity at the surface in time leads to the sea surface displacement

$$\eta(x,y,t) = \int_0^t v_3(x,y,z,\tau)\,\mathrm{d}\tau = \frac{k_\star \sin(\omega t)\sin(k_x x)\sin(k_y y)}{\omega^2 \rho}. \tag{8.25}$$

We use the following boundary conditions:

$$\begin{aligned}
p &= 0 \quad \text{at } x = 0, \text{ and } x = L_x,\\
p &= 0 \quad \text{at } y = 0, \text{ and } y = L_y,
\end{aligned} \tag{8.26}$$

$$p = \rho g \eta(x,y,t) \quad \text{at } z = 0, \tag{8.27}$$

$$\boldsymbol{n}\cdot\boldsymbol{v} = 0 \quad \text{at } z = -H. \tag{8.28}$$

Here, $L_x = L_y = 10\,\text{km}$ are the lengths of the domain, and $H = 1\,\text{km}$ is the water height. Thus, we use a computational domain of size $[0, L_x] \times [0, L_y] \times [-H, 0]$.

Now, we need to find parameters such that equation (8.24) fulfills the boundary conditions (equations (8.26) to (8.28)). To fulfill the free surface boundary conditions on the sides (equation (8.26)), we need to set the horizontal wavenumbers to

$$k_x = \frac{n\pi}{L_x}, \quad k_y = \frac{n\pi}{L_y}, \tag{8.29}$$

for arbitrary positive integers n. From the rigid body boundary condition (equation (8.28)), we get the dispersion relation

$$\omega = \sqrt{g k_\star \tanh(k_\star H)}, \tag{8.30}$$

Table 8.1.: Constants for the water tank test case for the gravity wave mode and the first acoustic wave mode.

Mode	ω	$k_\star$
$n = 0$ Gravity	0.0425599572628432	0.4433813748841239
$n = 1$ Acoustic	2.4523337594491745	1.5733628061766445

which relates the angular frequency ω to the wavenumber $k_\star$. Finally, we get the definition of $k_\star$ by inserting equation (8.27) into the pressure term of equation (8.24)

$$k_\star = \sqrt{k_x^2 + k_y^2 - \frac{\omega^2}{c^2}}. \tag{8.31}$$

We can solve this for ω, leading to

$$\omega^2 = c^2 \left(k_x^2 + k_y^2 - k_\star^2\right). \tag{8.32}$$

Combining equations (8.30) and (8.32), we can compute the values of $k_\star$ by solving

$$c^2 \left(k_x^2 + k_y^2 - k_\star^2\right) - g k_\star \tanh(H k_\star) = 0 \tag{8.33}$$

We can write complex solutions for $k_\star$ as $k_\star = A + Bi$. Using $k_\star^2 = A^2 + 2iAB - B^2$, we rewrite equation (8.33) as

$$c^2 \left(k_x^2 + k_y^2 - A^2 - 2iAB + B^2\right) - g(A + Bi) \tanh\left(H(A + Bi)\right) = 0. \tag{8.34}$$

We first look for the real solution. Thus, setting $B = 0$ and realizing that $\tanh(iB) = i\tan(B)$, results in

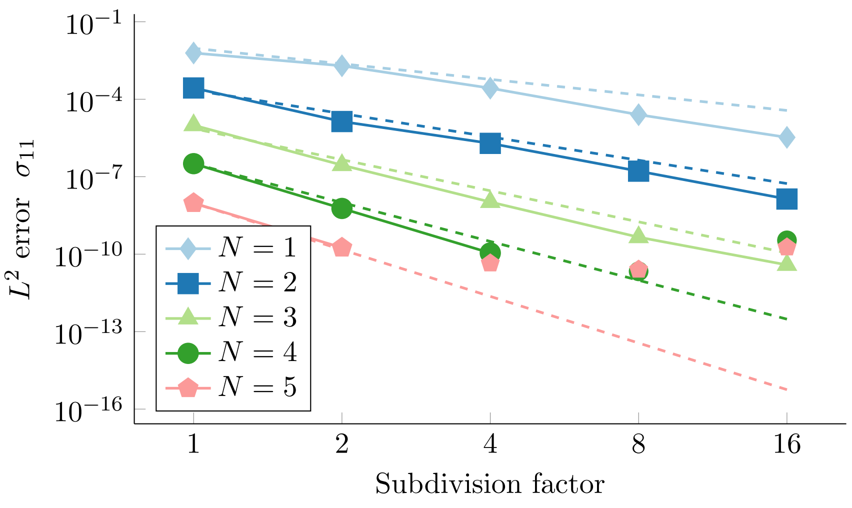
$$c^2 \left(k_x^2 + k_y^2 - A^2\right) - gA \tanh\left(HA\right) = 0. \tag{8.35}$$

We do the same exercise for imaginary solutions by setting $A = 0$, which leads us to

$$c^2 \left(k_x^2 + k_y^2 + B^2\right) + gB \tan\left(HB\right) = 0. \tag{8.36}$$

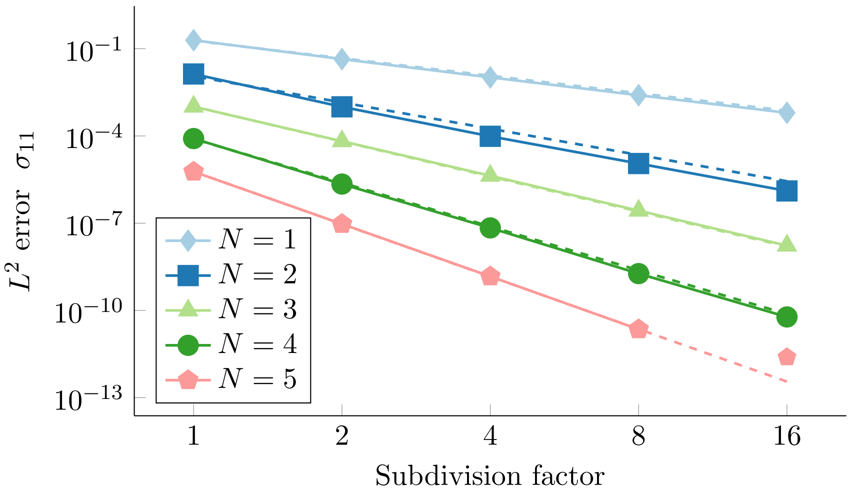
Hence, we need to solve equations (8.35) and (8.36) for k_*. We compute ω by inserting k_* into equation (8.32). There are multiple solutions for these equations. We denote the gravity wave mode, which is the solution of equation (8.35) as mode 0. The acoustic wave modes, which are the solutions of equation (8.36), are numbered sequentially by their value. Computing these constants with as much precision as possible is crucial, as the numerical solution is quite sensitive to these values. Table 8.1 shows the resulting constants. Furthermore, we set $g = 9.81\,\mathrm{m\,s^{-2}}$ and use a material with density $\rho = 1000\,\mathrm{kg\,m^{-3}}$ and acoustic wave speed $c = 1.5\,\mathrm{km\,s^{-1}}$. Note that we set up the simulation in a unit system that uses kilometers for lengths, which is numerically more stable than SI units.

The domain is discretized into cubes. We simulate for a time of $\frac{2\pi}{\omega}$, which corresponds to approximately 147.6 s and 2.6 s for the gravity and first acoustic wave mode, respectively. The results for the gravity wave mode (figure 8.6) show that our solver achieves an excellent



N	q
1	2.8
2	3.5
3	4.5
4	5.7
5	5.7

Figure 8.6.: Convergence plot for gravity wave mode of the ocean scenario. Solid lines indicate the range of errors used to compute the average convergence order. The dashed lines indicate the optimal convergence. Here, we achieve much better convergence orders than expected; however, the solution becomes worse for $N \geq 4$ for higher subdivision factors.



N	q
1	2.1
2	3.3
3	4.0
4	5.1
5	6.0

Figure 8.7.: Convergence plot for the first acoustic wave mode of the ocean scenario. Solid lines indicate the range of errors used to compute the average convergence order. The dashed lines indicate the optimal convergence. We obtain the optimal order of convergence for all considered polynomial degrees.

numerical order of convergence for up to $N = 4$. For $N = 4$ and especially $N = 5$, we note that the results become worse after sufficient refinement. This was also observed in [3], which used our old implementation of the gravitational boundary condition, detailed in [79]. The reason for this probably lies in the numerical instability of the analytical solution. However, the results for the first acoustic wave mode (figure 8.7) are much better. We achieve the optimal order of convergence for all considered polynomial orders. For order $N = 5$, we stop improving after an error of roughly 10^{-10} has been reached, again most likely caused by the error of the computation of the analytical solution. The results show that our implementation of the boundary conditions is correct and that we achieved high-order convergence.

8.5. Discussion

We have seen in this chapter that the implementation of our numerical model is correct by comparing it with multiple analytical solutions. Section 8.1 has shown that our implementation can simulate both elastic and acoustic wave propagation. Sections 8.2 and 8.3 have shown that we can simulate coupled elastic-acoustic scenarios. Finally, we have shown in section 8.4 that the implementation of the gravitational boundary condition is correct. Thus, our numerical scheme achieves high-order convergence for fully coupled earthquake-tsunami simulations.

Chapter 9.

Scenarios

In this section, we present large-scale fully coupled scenarios. We begin with an earthquake-tsunami benchmark (section 9.1) that models an earthquake followed by a tsunami. For this setup, we compare our fully coupled method with a standard two-way coupling method and investigate the differences in the wave structure. We discuss a fully coupled scenario for the Palu, Sulawesi earthquake-tsunami in section 9.2. This scenario models a real-world event with a detailed multiphysics model. Finally, in section 9.3, we simulate a small earthquake induced by the stimulation of an enhanced geothermal system. We use the fully coupled model to compute the sound generated by this event.

In this and the following chapter, we use the following petascale clusters:

Frontera [154] 8368 dual-socket Intel Cascade Lake Xeon Platinum 8280 (28 cores each). The nodes are organized in 101 racks and are connected by a Mellanox HDR InfiniBand interconnect with a fat tree topology.

SuperMUC-NG [93] 6336 dual-socket nodes with Intel Skylake Xeon Platinum 8174 (24 cores each). The nodes are organized in 8 islands and connected by an OmniPath network with a fat tree topology.

Shaheen-II [52] 6174 dual-socket nodes with Intel Haswell Xeon E5-2698v3 (16 cores each). The nodes are connected with an Aries interconnect with Dragonfly topology.

Mahti [138] 1404 nodes with dual socket AMD Rome 7H12 (64 cores each). The nodes are connected with a Mellanox HDR InfiniBand interconnect using a Dragonfly+ topology.

All clusters except Frontera use hyperthreading, i.e., they have two logical threads for each physical core. In this section, we use the computational tools and optimal settings described in chapter 10. We detail the used settings and SeisSol versions in appendix A.

9.1. Earthquake-Tsunami Benchmark

This section follows the discussion in [79, Sec. 6.1]. We extend an earthquake-tsunami benchmark scenario, called “Scenario A”, from [102]. It models a megathrust earthquake with an idealized dynamic rupture earthquake source on a planar fault. The tsunami setup assumes a flat bathymetry and inundation on a linearly sloped beach. We use this setup to compare our fully coupled method with a standard one-way linking method. We

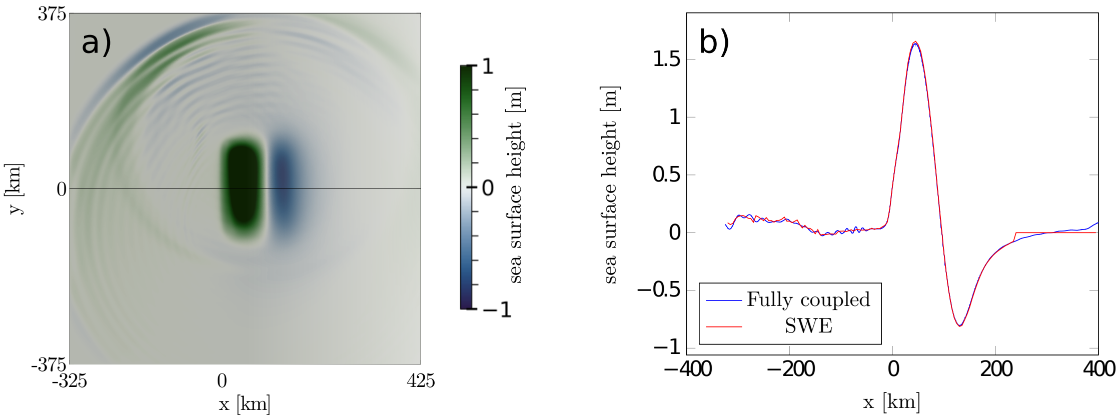


Figure 9.1.: Figure taken from [79]. Sea surface height of ("Scenario A" of [102]). (a) Snapshot of sea surface height (η) of our fully coupled model at $t = 120\,\mathrm{s}$. (b) Comparison of our fully coupled model with the one-way linked model. Cross section (black line in (a)) at $y = 0\,\mathrm{km}$, also at $t = 120\,\mathrm{s}$. Shown in blue is the sea surface height from our fully coupled model, and shown in red is the reference solution using a one-way linking (by time-dependent sourcing) with the non-linear shallow water solver sam(oa)^2-flash. We used the same earthquake simulation results, obtained with polynomial order 5, for both methods.

do not expect our model to agree completely with the results presented in [102]. While we should match the tsunami itself, we expect differences due to ocean acoustic waves excited by the high-frequency seismic waves from the earthquake [3]. Furthermore, our fully coupled model does not include the sloping beach.

We model the crust as a homogeneous medium, with density $\rho = 3775\,\mathrm{kg\,m^{-3}}$ and wave speeds $c_p = 7639.9\,\mathrm{m\,s^{-1}}$ and $c_s = 4229.4\,\mathrm{m\,s^{-1}}$, which represents a typical material estimated for oceanic crusts in subduction zones [176]. We consider an M_W 8.5 earthquake and model the rupture by a dynamic rupture model with a linear slip-weakening friction law [6]. This friction law is computationally relatively inexpensive, and the cost of the friction solver is constant throughout the simulation. The fault lies at a depth of 35 km and connects to the surface. It has a dip of 16°. The rupture stops when reaching the surface, as we set a higher fault strength there. Notably, the mean rupture velocity of $3.5\,\mathrm{km\,s^{-1}}$ is smaller than c_s. Hence, the resulting earthquake is called a subshear earthquake. However, the rupture transitions to supershear speed locally.

We added a water layer of depth 2 km with a density of $\rho = 1000\,\mathrm{kg\,m^{-3}}$ and an acoustic wave speed of $c = 1500\,\mathrm{m\,s^{-1}}$ to this Earth model. On top of the ocean, we use the gravitational free surface boundary condition to include tsunami propagation. Inside the crust, we use an element length of 400 m at the fault and 250 m at the nucleation patch. The water layer is meshed with element lengths up to 2 km; however, because we use conforming meshes, it is also automatically refined at the fault. As we use higher-order polynomials, the actual resolved resolution is lower. In our case, we

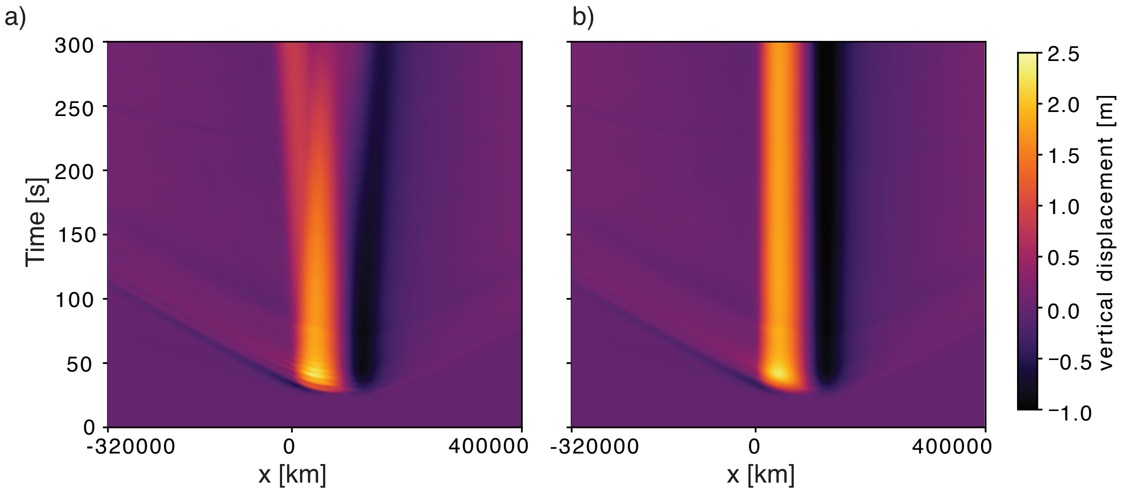


Figure 9.2.: Wave structure of the fully coupled simulation of "Scenario A", from [102]. We used a line at $y = 0\,\mathrm{km}$ with a spatial resolution of 1 km and a time resolution of 50 Hz. a) shows the sea surface vertical displacement and b) shows the seafloor displacement. The effect of the ocean response and the tsunami are visible in a). We observe seismic waves in both a) and b).

consider $N = 5$ polynomials. Hence, the minimal resolution at the nucleation patch is approximately 50 m. The mesh for the earthquake (without the water layer) has roughly 16 million elements. Adding the water layer increases the mesh size to 29.5 million elements. For all simulations with this setup, we use a CFL constant (equation (4.47)) of $C(N) = 0.35(2N + 1)^{-1}$.

In [79], we compared our fully coupled model with a two-step linking approach. For the latter, the displacement from the earthquake simulation was interpolated to a Cartesian grid using bi-linear interpolation. This seafloor displacement was used as a time-dependent source term in the two-dimensional non-linear shallow water equations solver sam(oa)^2-flash, which uses a second-order Runge-Kutta DG method and dynamic adaptive mesh refinement. For details, we refer the interested reader to [102]. The tsunami model contains a linearly-sloping beach at $x = 240\,\mathrm{km}$, which we did not include in our fully coupled model. We simulated this with a rate-2 LTS scheme up to 120 s.

It is clear from the results (figure 9.1) that both modeling strategies capture the same tsunami. However, there are some differences. On the right, we can see that the beach in the one-way linked model leads to slight differences close to the boundary, which do not influence the tsunami elsewhere. On the left, we can see high-frequency oscillations (with period $< 5.3\,\mathrm{s}$) in the fully coupled model. These oscillations have a small wavelength and correspond to reverberating acoustic waves in the ocean. The one-way linked model does not contain these waves, as they are not included in the used physical model, which assumes that the ocean is incompressible.

To better investigate the wave structure, we placed receivers on a $y = 0\,\mathrm{km}$ slice on the seafloor with a spacing of 1000 m and used a longer simulation time of 300 s. We simulated this with SeisSol version 1.0.1 [167]. Figure 9.2 shows a space-time plot of the vertical velocity of our fully coupled model. We compare the displacement at the sea surface (a) with the seafloor (b). The seismic waves are clear in both figures. Acoustic waves can be seen on the sea surface as transient waves. We can observe the beginning of the tsunami propagation, moving with speed $\sqrt{gH} \approx 140\,\mathrm{m\,s^{-1}}$ on the sea surface. Note that we can see some reflections coming in from the boundary of our simulation; however, they do not seem to affect the tsunami. We ran this on 256 nodes of SuperMUC-NG, resulting in a sustained performance of 312.2 TFLOPS. The simulation took roughly 10 hours and 56 minutes. We used a rate-5 LTS scheme for this simulation and a wiggle factor of $\lambda = 1$, which resulted in a clustering (figure 9.3) with an expected speedup of 5 compared to GTS. This LTS configuration demonstrates that our scheme can handle rates other than 2.

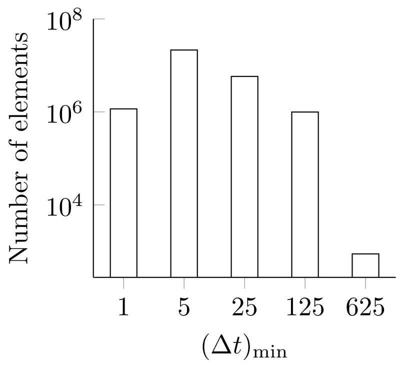


Figure 9.3.: LTS clustering with rate-5 for the earthquake-tsunami benchmark mesh. The y-axis (logarithmically scaled) shows the number of elements per cluster while the x-axis shows their respective minimum time step sizes in terms of the globally minimal time step size $(\Delta t)_{\min} \approx 107.22\,\mu\mathrm{s}$.

To summarize, this experiment shows that our model captures a tsunami that is identical to the one obtained from a standard one-way linked model. However, it also shows that our model includes more waves than the classical coupling strategy because it allows the ocean to be compressible. Furthermore, it demonstrates that we can use a rate-5 LTS scheme.

9.2. Palu, Sulawesi 2018

A devastating M_{W} 7.5 earthquake struck Palu Bay on the island of Sulawesi, Indonesia on September 28, 2018. It triggered an unexpected local tsunami. In this section, we present a fully coupled model of this event, following our paper [79, Sec. 6.2].

This event is interesting from a geophysical perspective for two main reasons. The first is that the rupture happened with supershear speed, i.e., it moved faster than the shear wave speed. The second is that it was a strike-slip earthquake that nevertheless induced a tsunami. This type of mechanism leads to predominantly horizontal displacement. As

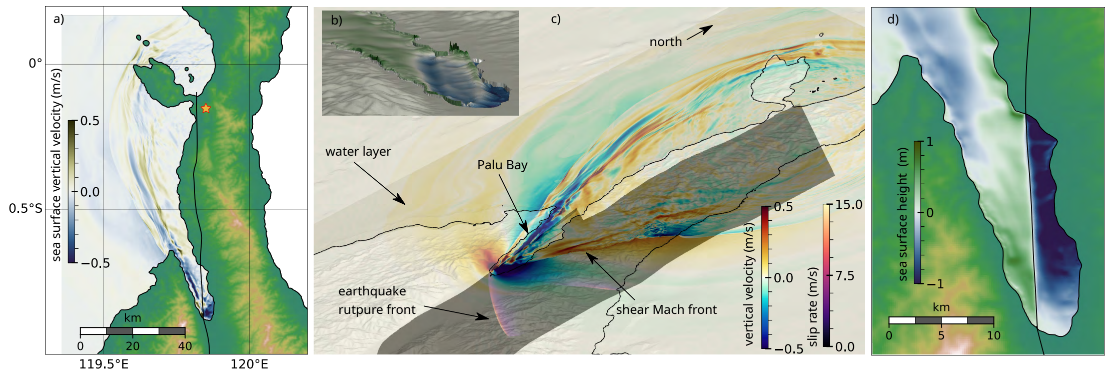


Figure 9.4.: Figure taken from [79]. Results of our fully coupled model for the Palu earthquake tsunami. a) Map view of the resulting tsunami. The vertical sea surface velocity at $t = 15\,\mathrm{s}$ is shown. The black lines indicate the location of faults, and the star is the earthquake's epicenter. b) 3D visualization of the sea surface height in meters at $t = 60\,\mathrm{s}$. The sea surface is exaggerated by a factor of 2000. c) 3D snapshot at $t = 15\,\mathrm{s}$. We show the slip rate on the fault and the vertical velocity on the free surface. On the free surface, the vertical velocity is shown. The earthquake rupture front is highlighted and has a—for supershear earthquakes characteristic—trailing mach front. d) Map view of sea surface height at $t = 15\,\mathrm{s}$.

tsunamis are typically induced by vertical displacement, this tsunami was unexpected. The cause of the tsunami is highly debated [104].

[163] showed that the tsunami can be explained by displacements resulting from the interplay of earthquake rupture and the complex geometry of the bay. However, this paper used a standard one-way linking approach to model the tsunami. As we discussed in chapter 7, this method of tsunami simulations makes strong assumptions. Hence, we present a fully coupled model which can capture the full dynamics of the event, starting from earthquake rupture, to wave propagation in both elastic (Earth) and acoustic (ocean) media, to tsunami propagation. For this, we extended the model published in [163] by adding a water layer on top of the Earth, corresponding to the ocean. We use our gravitational free surface boundary condition to enable tsunami propagation.

Our meshes use a maximum element size of 5 km. We refine in a cuboid with coordinates $x = 5\,\mathrm{km} \pm 70\,\mathrm{km}, y = 0\,\mathrm{km} \pm 180\,\mathrm{km}$ and $z = -8\,\mathrm{km} \pm 34\,\mathrm{km}$, which includes our region of interest and the water layer. We use the following two meshes:

M is a medium-sized mesh with roughly 89 million elements. It has a water layer resolution of 100 m and uses elements with 1 km length in the refinement zone.

L is our large mesh with roughly 519 million elements. It resolves the water layer with a resolution of 50 m and seismic waves in the refinement zone with 500 m.

We run the **M** mesh for 100 s, which is enough to capture the entire earthquake dynamics. The tsunami reaches the coast in this time frame. We run the **L** mesh for 30 s, which is enough to capture the earthquake mechanics, tsunamigenesis, and initial acoustic wave propagation in the ocean. The water layer causes the majority of the computational effort: In the **L** mesh, 453.7 million elements discretize the ocean, increasing the total mesh size by a factor of 8.

We observe frequencies up to 30 Hz for the **L** mesh in the acoustic receivers. Using the heuristic that we require two elements per wavelength to reach an acceptable error, we would expect to resolve frequencies up to 15 Hz in the water layer [71]. The fast-moving supershear dynamic rupture source produces a sharp Mach cone (figure 9.4). The velocity wavefield is highly complex and is a superposition of elastic waves in Earth, acoustic waves in the ocean, for which we expect periods shorter than 1.6 s, and the tsunami. As figure 9.4b shows, the rupture arrives at Palu Bay at 15 s. While the seismic waves strongly transiently affect the wavefield, they do not contribute to the tsunami generation, as seen in the displacement field in figure 9.4d.

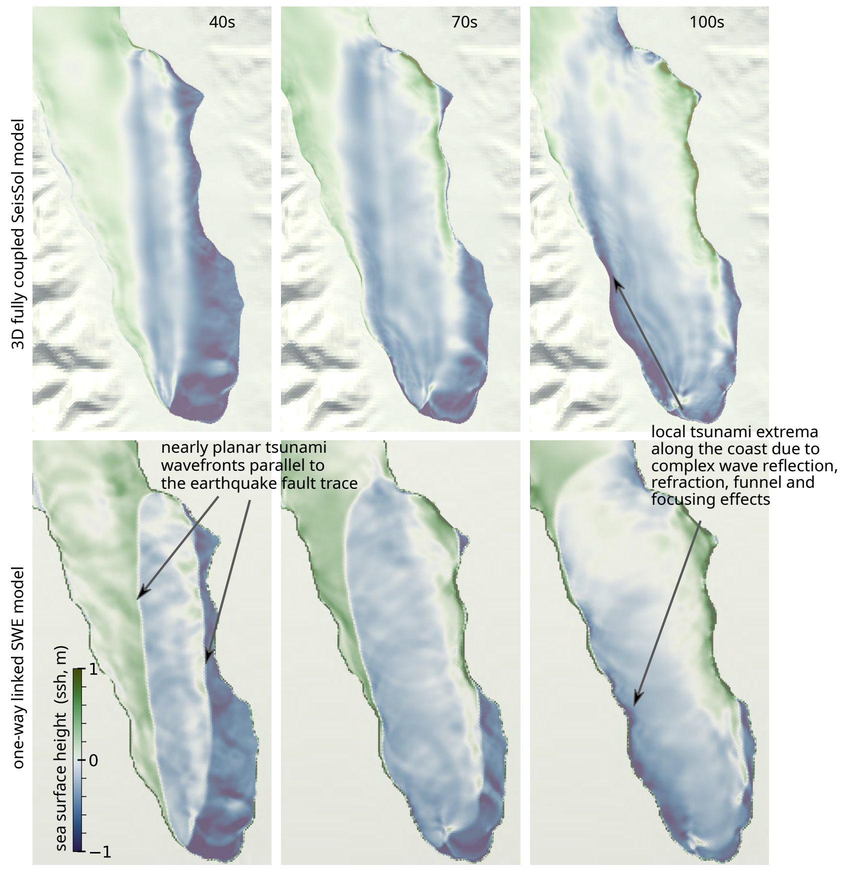

40s
70s
100s
3D fully coupled SeisSol model
one-way linked SWE model
nearly planar tsunami wavefronts parallel to the earthquake fault trace
local tsunami extrema along the coast due to complex wave reflection, refraction, funnel and focusing effects
sea surface height (ssh, m)
1
0
−1

Figure 9.5.: Figure taken from [79]. Comparison between one-way linking (lower row) and fully coupled model (upper row). Both models capture the tsunami. The main difference between both models is the sharpness of the wavefront. Various factors, including the non-hydrostatic ocean response, may cause these.

The results (figure 9.5) of our fully coupled model show several differences from a one-way linking model. We compare the fully coupled model to one-way linking with a non-linear shallow water solver. Southeast of the fault, we observe subsidence, and northwest of the fault, we see the uplift of the ocean. Both are caused by the seafloor displacement, visible in figure 9.4d. The wavefronts are parallel to the fault trace at 40 s. As the water is shallower there, the tsunami propagates slower at the boundaries of the bay where the fault enters or leaves. At 100 s, we no longer see these clear patterns: We can observe more complex behavior, such as wave reflections, along the coast.

While both models result in mostly similar wavefields, there are some differences. The most pronounced one is that the one-way linked model produces a sharper wavefront. One possible reason might be the non-hydrostatic effect of the ocean, as discussed in section 7.3.

We use a CFL constant of $C(N) = 0.35(2N + 1)^{-1}$ for all simulations, as done in [79]. We activated a wiggle factor search for factors $\lambda = 0.51, 0.52, \ldots, 1.0$ and allowed a performance loss of 1 % from the automatic merging of clusters. We deactivated the automatic cluster merging for simulations without wiggle factor ($\lambda = 1$).

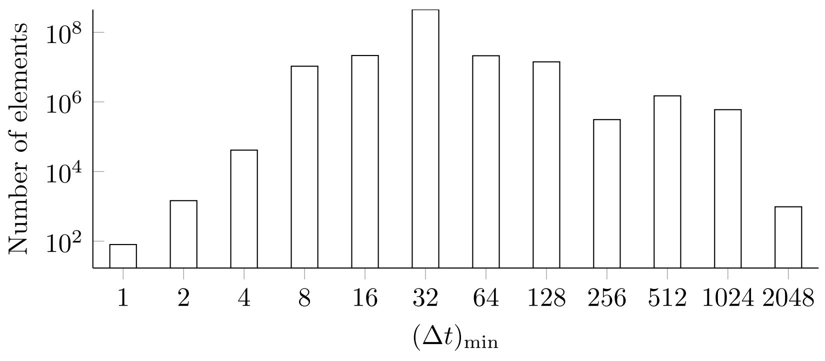
The best wiggle factor for the **M** mesh is $\lambda = 0.65$ with 8 time clusters. The resulting clustering requires 91.3 % of the updates of the clustering with $\lambda = 1$, which has 11 clusters. While the $\lambda = 1$ leads to a speedup of 24.7 compared to GTS, the $\lambda = 0.65$ clustering is 27 times faster than GTS.

The wiggle factor and automatic cluster merging have a stronger effect for the **L** mesh. The rate-2 LTS scheme with $\lambda = 1$ (figure 9.6a) reduced the number of necessary updates by a factor of 30.2, compared to GTS. Most elements, more than 86 %, are in the cluster with time step size $32(\Delta t)_{\text{min}}$. The optimal wiggle factor $\lambda = 0.71$ leads to the clustering shown in figure 9.6b, which requires only 79.3 % of the time steps of the $\lambda = 1$ simulation. Compared to GTS, this optimized clustering is 38.1 times faster. Additionally, the automatic cluster merging reduced the number of clusters from 12 to 8. However, the number of elements in the cluster with time step $(\Delta t)_{\text{min}}$ decreased from 80 to just 8 elements, which could reduce the performance in practice because small clusters are harder to parallelize.

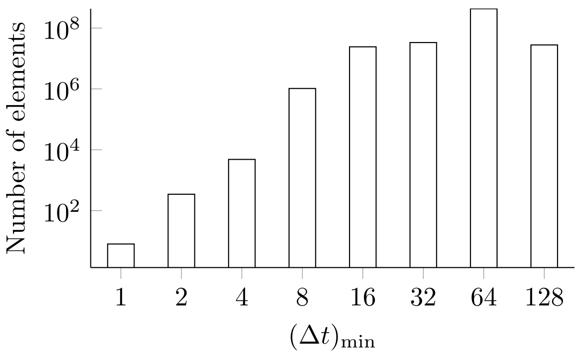
In [79], we used an older version of SeisSol that uses a different implementation of LTS, dynamic rupture, and the gravitational boundary condition. The results of [79] for the **L** mesh were obtained on 3072 nodes of cluster SuperMUC-NG, resulting in approximately 3.14 PFLOPS of sustained average performance and a simulation runtime of 5 hours and 30 minutes. Running the same setup on 6144 nodes of Shaheen-II reached an estimated performance of 2.3 PFLOPS.[1] These performance numbers include free surface output (every 0.1 s) and receiver output (every 0.01 s)

We simulated the **M** mesh simulation with version 1.0.1 of SeisSol [167], the version presented in this thesis, on 1000 nodes of the cluster Frontera. The version with $\lambda = 1$ achieved a performance of 1.3 PFLOPS and took roughly 3 hours and 57 minutes. Using $\lambda = 0.65$ led to a sustained performance of 1.2 PFLOPS and a time-to-solution of 3

[1]The simulation timed out after ≈16 s simulation time. Performance is an extrapolation from the number of calculated FLOPS of the SuperMUC-NG simulation and from the execution time on Shaheen-II.



(a) Reproduced from [79]. $\lambda = 1$, $(\Delta t)_{\min} \approx 6.7\,\mu\mathrm{s}$



(b) $\lambda = 0.71$, automatic cluster merging with up to 1 % performance loss. $(\Delta t)_{\min} \approx 4.7\,\mu\mathrm{s}$

Figure 9.6.: Elements clustering for the **L** mesh. The y-axis (logarithmically scaled) shows the number of elements per cluster while the x-axis shows their respective minimum time step sizes in terms of the globally minimal time step size $(\Delta t)_{\min}$.

hours and 48 minutes. As the performance of supercomputers can vary between runs, this measurement is insufficient to decide which version is faster. The results of both simulations agree with each other and with the results in [79].

To summarize, we have shown in this section how we can apply our fully coupled model to a realistic model of an earthquake-tsunami event. Our results compare well to one-way linking. However, the fully coupled model leads to a smoother tsunami. Our fully coupled model relies on the significant speedup due to LTS, further enhanced by the wiggle factor. Simulations with this setup have reached sustained performance in the petascale range on the cluster Shaheen-II, SuperMUC-NG and Frontera.

9.3. Helsinki Metropolitan Area

This section summarizes [82]. The Otaniemi project is an enhanced geothermal system (EGS) constructed by the company St1 Deep Heat Oy for district heating. It is a geothermal doublet system, situated roughly 6 km below the campus of Aalto University in Otaniemi, a district in Espoo, next to Helsinki. Induced seismicity is a necessary side effect of EGSs, as fluid injection is used to increase the flow rate. The stimulation for this project was done in two phases. We focus here on the first phase, in June and July 2018, in which roughly 18 000 m^3 of water was pumped down. This induced thousands of small earthquakes [90]. Figure 9.7 shows an overview of this. A second, smaller stimulation occurred in May 2020, where 2900 m^3 water was used.

It is crucial to quantify the risk and discomfort caused by these events. Prior work mainly focused on quantifying the risk of ground shaking [17, 75]. However, not only ground shaking is relevant, as sounds such as rumbling have been reported for events of various sizes [25, 40, 62, 151]. For the stimulation experiment conducted in the context of the Otaniemi project, over 300 macroseismic events were collected in 2018 and 2020 [4, 64, 91, 128]. These reports correspond to a certain annoyance level of the public. The association of EGSs with disturbances can lead to a lower acceptance level in the public, as has been demonstrated, for example, in Switzerland [148]. Hence, research into ground shaking and audible signals is required to better inform the public, ideally leading to a higher acceptance for similar projects.

Limiting the strength of these induced events (e.g., measured by magnitude or peak ground velocity) is important to control the hazard and nuisance. The current state of the art is traffic light systems (TLSs) [11]. TLSs assume that real-time monitoring and associated reactions are enough to avoid hazard [9]. However, this requires a fine-tuned earthquake hazard forecasting tool as otherwise large events, such as the 2006 Basel, Switzerland event, can still be excited [54]. These forecast models can be complex multiphysics models [48] or probabilistic models [136, 137]. However, most of these TLS have in common that they work under the belief that such induced earthquakes do not typically lead to audible signals [105]. They thus do not consider noise [170], which is at odds with the macroseismic reports [64] and the perception of the public [148]. In Helsinki, a traffic light system was deployed [4], which, together with an adaptive fluid injection protocol, successfully limited the magnitude of the events [4, 88, 89]: The largest of these events with M_L 1.8 did not exceed the maximum target magnitude of M_L 2.1.

We focus on this event, which corresponds to event 13 of [64]. As it generated audible noise, further research into the mechanism of sound excitation is required. Related work includes research into inaudible sounds, such as infrasound, which can be excited by coupled effects of P waves, S waves, and surface waves [42, 60, 112, 140]. However, sound can also be generated by secondary sources (e.g., due to topography) [8, 91] or by weather phenomena. The latter effect and the related influence of temperature can be included in ray tracers [8] or other approaches [173, 174]. Infrasound can be measured by a mature network of infrasound sensors [56, 66] or by temporarily deployed sensors [171]. On the other hand, measurement networks are not as mature for audible sound; hence, observations are rather sparse. It is well known that small to medium-sized earthquakes

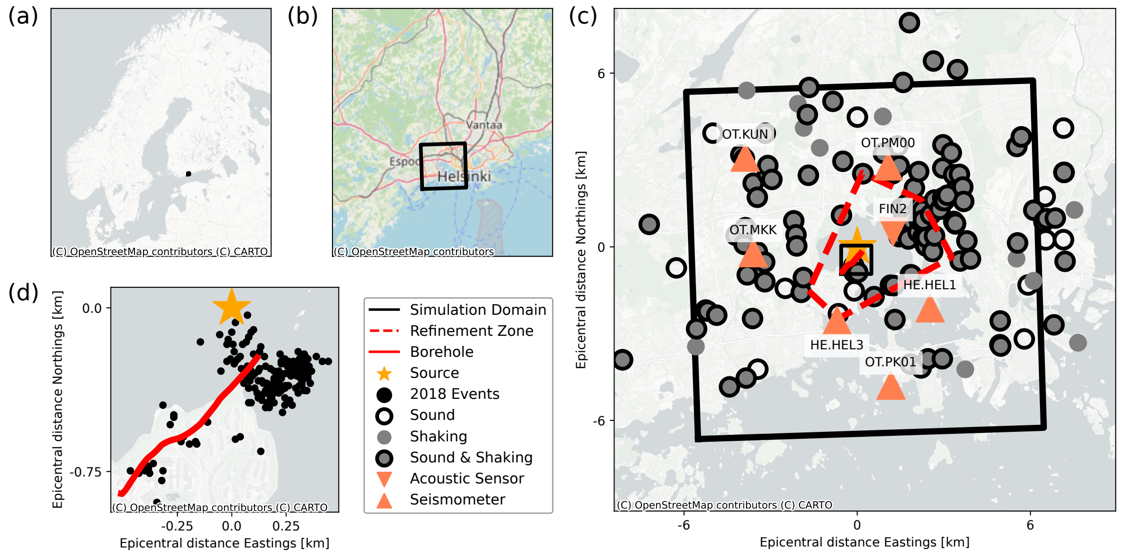


Figure 9.7.: Figure taken from [82]. Overview of induced seismicity from the St1 Otaniemi project and our simulation setup. This figure uses the Web Mercator coordinate system, which differs from the simulation's map projection. (a) Study area location in Northern Europe, marked by a black symbol corresponding to the black rectangle in (b) and (c). (b) Study area in the Helsinki metropolitan area. The black square marks the computational domain. (c) As in (b), the large black square denotes the 12 km × 12 km simulation domain. The circles represent 220 macroseismic reports stimulations in 2018: Black-outlined circles indicate sound observations, gray-filled circles represent shaking sensations, and gray circles with black outlines signify simultaneous sound and shaking. The dashed red polygon marks the refinement area of our simulation, which contains the source region and neighboring area. The FIN2 microphone array location is shown by an inverted triangle [91], while other triangles represent selected seismic stations used for data comparison. The star marks the location of the largest induced M_{L} 1.8 event 13 in [64]. We use its location as the origin of all maps. (d) Area surrounding the source. The red line is the 2018 borehole trajectory, which is not perfectly vertical. After it reached a depth of 5 km, it traveled northeast. The black dots are 203 large, manually revised event locations [64].

can also lead locally to audible noise [106, 150, 155].

These sound excitations are often approximated by the relationship between the vertical ground motion and the sound pressure level [63, 91, 161]. We want to estimate the sound pressure level (SPL) in the air induced by a seismic wave. Consider a plane wave moving up in the z-direction and crossing the elastic-acoustic interface. Using the planar wave ansatz (equation (8.10)),

$$p(z,t) = \cos(ct - z), \quad v_3(z,t) = \frac{1}{\rho c}\cos(ct - z), \tag{9.1}$$

is a solution to the acoustic wave equation (equation (2.79)). Furthermore, we know from equation (2.95) that both traction and vertical velocity are continuous across the elastic-acoustic interface. Using this assumption, we can write equation (9.1) as

$$\Delta p = \rho c v_3, \tag{9.2}$$

where ρ and c are the density and acoustic wave speed in the air and v_3 is the vertical velocity measured in the solid. Equation (9.2) directly relates measurements done on Earth (v_3) to the pressure perturbation measured close to the surface in the air (Δp). Of course, in practice, a realistic source never only generates strictly vertically propagating planar waves. However, if the vertical distance to the source dominates the horizontal distance, the incident angle of the P wave is nearly normal. For this situation, equation (9.2) is approximately valid.

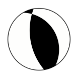
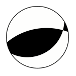
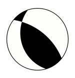
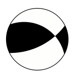
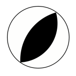
The human hearing range is limited to sounds in the range of 20 Hz to 20 kHz [44]. Sound that is below this range is called infrasound [110]. It can be excited by various anthropogenic sources or by natural phenomena such as earthquakes. Even though infrasound is typically not heard, it can be perceived by humans as vibrations, given a sufficiently high sound pressure level [110]. In this section, we focus on quantifying the SPL and do not consider how humans perceive the sounds. However, we correlate the spatial distribution of the SPL with reports of heard sounds.

For the earthquakes induced in the Helsinki metropolitan region, the macroseismic reports show a strong spatial variation with fine-scale features [64], which figure 9.7 illustrates. These small scale features and the minimum frequency of 20 Hz, which is required to resolve audible sound, require a very fine resolution [71], which leads to a high computational workload. Thus, we must use high-performance computing tools capable of simulating elastic-acoustic wave propagation, such as the one developed in this thesis.

We compute high-resolution fully coupled elastic-acoustic simulations for the largest $M_L 1.8$ earthquake event 13 in [64] that was induced during the stimulation phase of the Otaniemi EGS. Our setup contains a 12 km × 12 km area with a sub-element refinement of up to 2.3 m. We use a local velocity model and realistic topography to set up our simulation. We compare with measurements of both seismic and acoustic signals, and we evaluate the agreement with the macroseismic reports. Furthermore, we use our numerical laboratory to investigate the effect of the orientation of the moment tensor on results.

In section 9.3.1, we discuss the setup of our numerical experiments and introduce a novel workflow to compute high-resolution peak SPL maps. In section 9.3.2, we discuss

Table 9.1.: This table (taken from [82]) shows the five investigated earthquake source mechanisms. Event 13 is the reference M_{L} 1.8 event induced during the stimulation on the 16th of July 2018. We used the source mechanism described in [64]. The mechanisms Strike + 90, Dip + 90, and Rake + 90 are constructed by rotating the reference event by 90°. Finally, the slip vector of the Orthogonal mechanism is orthogonal to Event 13 and Strike + 90.

Event	Event 13	Strike + 90	Dip + 90	Rake + 90	Orthogonal
Strike (°)	328	58	148	328	216
Dip (°)	31	31	59	31	52
Rake (°)	71	71	289	161	91
Beachball					

the results: We compare our synthetics with observations Furthermore, we discuss the effect of the source orientation on the resulting peak ground velocity (PGV) and SPL maps. Finally, in section 9.3.3, we summarize the application and discuss the implications of our model and its limitations.

9.3.1. Numerical Experiments

Source

We focus on the M_{L} 1.8 event 13 of [64], which happened on 2018-07-16 at 17:26:03 UTC. Its hypocenter was located at 60.196°N, 24.837°E at a depth of 6.1 km. Following [90, 100], we convert the local magnitude to the seismic moment by the relation

$$M_0 = 10^{(M_{\mathrm{L}}+7.98)/0.83}. \tag{9.3}$$

We model this event by a point source using the Brune source time function

$$S(t) = \begin{cases} \exp\left(-(t-t_0)/T\right)(t-t_0)/T^2 & (t-t_0) > 0 \\ 0 & \text{else} \end{cases}, \tag{9.4}$$

where $t_0 = 0.05$ s is the onset time and $T = 0.02$ s governs the source duration [18, 101]. The source has a corner frequency of around 24 Hz. Similar short source durations have been reported for events of similar size [158]; however, it is difficult to constrain the corner frequencies of such small events [1]. Hence, we must experimentally verify whether our choice fits the data well. Furthermore, we created multiple rotations of this source (table 9.1), which we use to perform a parameter study.

Mesh

We use a domain of size 12 km × 12 km × 15 km that is centered on the source location. On top of the Earth, we put a 2 km thick air layer. We incorporated accurate topography data with 2 m resolution from the National Land Survey of Finland to model the elastic-acoustic interface. We created two meshes, one that is fully coupled and includes an air layer and one that only contains the Earth but has a higher resolution.

In the fully coupled mesh, we focus on a cone-shaped refinement region (figure 9.7), which includes the earthquake source and the Munkkivuori neighborhood. We used element sizes of 97 m (Earth) and 14 m (air) in this refinement region. This small resolution is required to accurately resolve the high-frequency content of the solution [69]. We gradually decrease the mesh resolution outside the refinement area to a maximum element size of 2 km. The larger domain size and the coarser resolution serve as a cost-effective sponge layer that helps mitigate reflections from the imperfectly absorbing boundary conditions. Near the elastic-acoustic interface, the mesh is automatically refined to account for topography details, leading to smaller element sizes. Furthermore, we use a conforming mesh, which restricts the flexibility of the mesh generation. Hence, some elements may be much smaller than the target resolution: 1 % of all element edges are smaller than 7.04 m! We use polynomials of degree five, achieving sixth-order accuracy in space and time and leading to an effective resolution of 16.2 m (Earth) and 2.3 m (air) in our refinement region. Our mesh comprises 40.9 million elements, with the computational cost primarily attributed to modeling acoustic wave propagation in air, similarly as observed for the Palu scenario (section 9.2).

In the Earth-only setup, which does not include an air layer, we use a uniform mesh resolution of 70 m in the Earth. This mesh consists of 32.5 million elements. Including a high-resolution air layer for the entire domain would pose significant computational challenges.

Velocity model

Below the surface, we adopt a one-dimensional seismic velocity model [94], obtained by vertical seismic profiling at the injection well. It describes the P wave velocities, which increase from $5.9\,\mathrm{km\,s^{-1}}$ at the surface to $6.5\,\mathrm{km\,s^{-1}}$ at 3 km depth, followed by a decrease to $6\,\mathrm{km\,s^{-1}}$ at 6 km depth. The S wave speed is $v_P/v_S = 1.71$. We use a constant density of $2700\,\mathrm{kg\,m^{-3}}$ in the Earth. In the air, we use a constant acoustic wave speed ($c = 340.5\,\mathrm{m\,s^{-1}}$) and density ($\rho = 1.225\,\mathrm{kg\,m^{-3}}$).

Output & post-processing

We placed a grid of receivers in the fully coupled simulation with a spacing of 100 m within the high-resolution refinement area at an elevation of 0.5 m above the elastic-acoustic interface, capturing synthetic acoustic fields. Additionally, we placed receivers just below the acoustic grid at a depth of 0.05 m to capture the seismic wavefield. Thus, we have a grid of 1386 receiver pairs, which allows us to compare the seismic wavefield with the acoustic wavefield. Furthermore, we added receivers at all ST1 borehole sensor locations and the surface stations of the 2018 HE and OT networks [64]. We marked the subset of these sensors used for data comparison as triangles in figure 9.7. We placed the FIN2 acoustic sensors [91], visualized as inverted triangles in figure 9.7, in our simulation at a height of 0.5 m. All receivers record 200 times per second.

We write free surface output every 0.1 s unless we want to compute the horizontal peak ground velocity with the Earth-only setup, in which case we write it every 1 µs. In addition, we activate the volume output for some simulations, which we write every 0.1 s.

Calibration

We assume that we can approximate the peak pressure in the air by the peak ground velocity measured directly below the interface, using the plane wave assumption given by equation (9.2). We try to validate this rule of thumb and calibrate it using the synthetics obtained from our fully coupled simulation. Hence, we assume that the pressure perturbation Δp follows the relationship

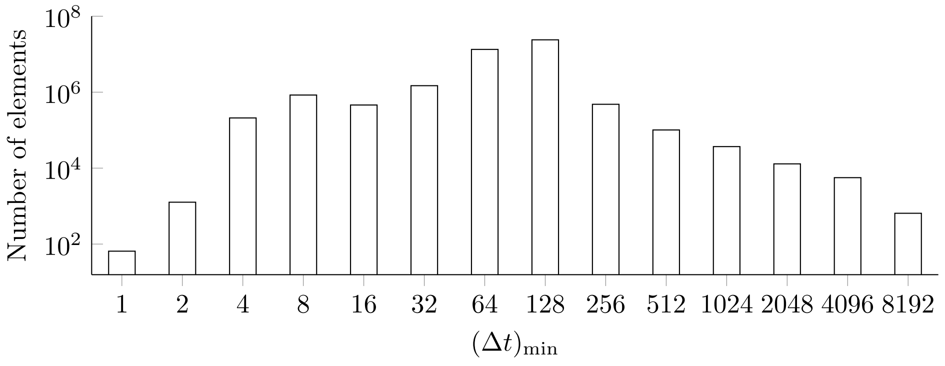
$$(\Delta p)^{\text{peak}} = c_0 + c_1 v_3^{\text{peak}} + \varepsilon, \tag{9.5}$$

where $(\Delta p)^{\text{peak}}$ is the peak SPL and v_3^{peak} is the peak velocity on the free surface. The model parameters are the intercept c_0, summarizing factors such as topography or source effects, and the slope c_1. Finally, ε is a normally distributed error term. Equation (9.5) is a simple linear regression model and can thus be fit by a standard least-squares minimization. If equation (9.2) were perfectly accurate, this would result in the parameters $c_0 = 0\,\text{Pa}$ and $c_1 = \rho c \approx 417.1\,\text{Pa}\,\text{s}\,\text{m}^{-1}$ [21, 91, 161].

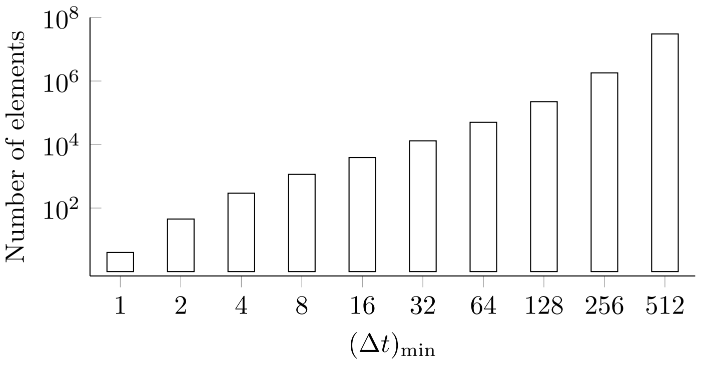
We compute the coefficients with the data obtained in the refinement zone of our fully coupled simulation, which consists of the peak ground vertical velocity just below the ground and the peak SPL just above the ground. Note that we must fit a model for each source rotation (table 9.1). As a result, we get optimized values for c_0 and c_1 for each source, which we can use to predict the peak SPL from the peak ground vertical velocity. The resulting two-step workflow combines the results from the fully coupled model with the high-resolution Earth-only model: First, we use the data obtained from the refinement region of the fully coupled model to fit equation (9.5). Second, we use the peak vertical ground velocity obtained from the high-resolution Earth-only model to predict the peak SPL. This workflow results in highly accurate approximate sound pressure levels without too much computational cost.

Phase estimation

We want to examine the influence of the P wave and S wave on the disturbance patterns. To do this, we segment the wavefield into a part belonging to each wave. We assume that the S wave arrives at a point with distance d to the source at $t^S(d) = d/c^S_{\max}$, where $c^S_{\max} = 3.83\,\text{km}\,\text{s}^{-1}$ is the fastest S wave speed in our velocity model. Using this, we assume that at a location with distance d, seismic signals stem from the P wave for all $t \in (0, t^S(d))$ and from the S wave for all other times. As we use an upper bound for the S wave arrival time, we underestimate the duration of the P wave. However, this does not significantly bias our results because the P wave coda has a much smaller amplitude than the S wave.



(a) LTS clustering for the fully coupled setup, $(\Delta t)_{\min} \approx 1.8\,\mu s$



(b) LTS clustering for the Earth-only setup, $(\Delta t)_{\min} = 68.4\,\mathrm{ns}$

Figure 9.8.: LTS clustering for the Helsinki setups. Note that this figure uses a log-log axis.

LTS & computational aspects

We use a rate-2 LTS scheme and a CFL constant of $C(N) = 0.2(2N + 1)^{-1}$ and $C(N) = 0.1(2N + 1)^{-1}$ for the fully coupled and Earth-only simulations respectively. For this scenario, the newly developed LTS scheduling algorithm, as described in chapter 6, was crucial because the simulations did not run with SeisSol's older LTS implementation. This is because the stark differences in resolution and wave speeds lead to a small minimum time step and many clusters. In detail, the fully coupled mesh has a minimum time step size of $(\Delta t)_{\min} = 1.8\,\mu s$ and a clustering, depicted in figure 9.8a, that leads to a speedup of 64.8 compared to GTS. Even though we targeted a uniform mesh resolution for the Earth-only model, it showcases the LTS speedup very well. Figure 9.8b shows the resulting clustering, resulting in a speedup of 586.7 over the GTS scheme. The scenario has a minimum time step size of only $(\Delta t)_{\min} = 68.4\,\mathrm{ns}$. We use a CFL constant of 0.1 and 0.2 for the Earth-only and fully coupled simulations, which are much smaller than the CFL constants that we typically use. The small time step sizes, combined with the high LTS speedups, hint that the quality of the meshes is not optimal. Using a wiggle factor of $\lambda = 0.86$ would lead to a minuscule speedup of 1.2 % for the fully coupled model. For the Earth-only model, a wiggle factor of $\lambda = 0.51$ leads to an even smaller speedup of 0.2 %. Hence, we did not use the wiggle factor for these simulations.

We simulated up to 3 s and used the clusters SuperMUC-NG and Mahti. Using 200 nodes of SuperMUC-NG, the fully coupled simulation took around 1.25 hours.

9.3.2. Results

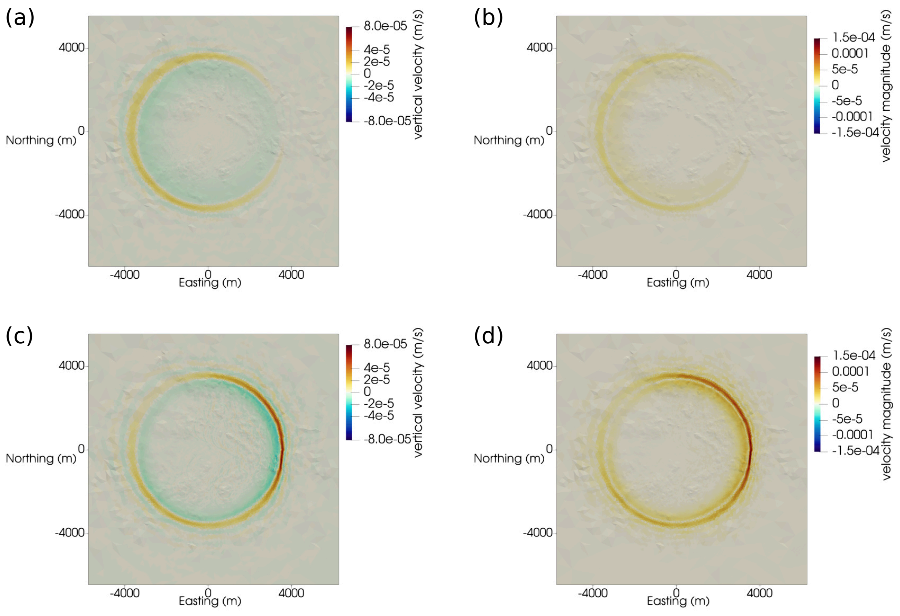


Figure 9.9.: Figure taken from [82]. Shown are the vertical ground velocity (a,c) and the surface velocity magnitude (b,d) for the M_L 1.8 event 13. (a,b) are at 1.2 s, which is the time where the effect of the P wave is clearly visible. (c,d) show the effect of the S wave, at roughly 2 s.

We are now ready to discuss the results. Figure 9.9 shows the vertical velocity and velocity magnitude at the Earth's surface, showcasing source effects, radiation patterns, and the maximum 10 m high topography, which leads to scattering that does not cause strong decoherence of the wavefronts but leads to visible coda effects [121, 153].

Additionally, we present the three-dimensional fully coupled wavefield (figures 9.10 and 9.11) at selected time steps. Figure 9.10a sketches the computational mesh used in our simulation. We can see the P wave energy in figure 9.10b and the S wave energy in figure 9.10c. Figure 9.11 highlights the interaction of elastic and acoustic waves at the interface: While the P wave has already left the domain at 2 s, we still see the reflected S waves in the Earth and the transmitted acoustic wave in the air.

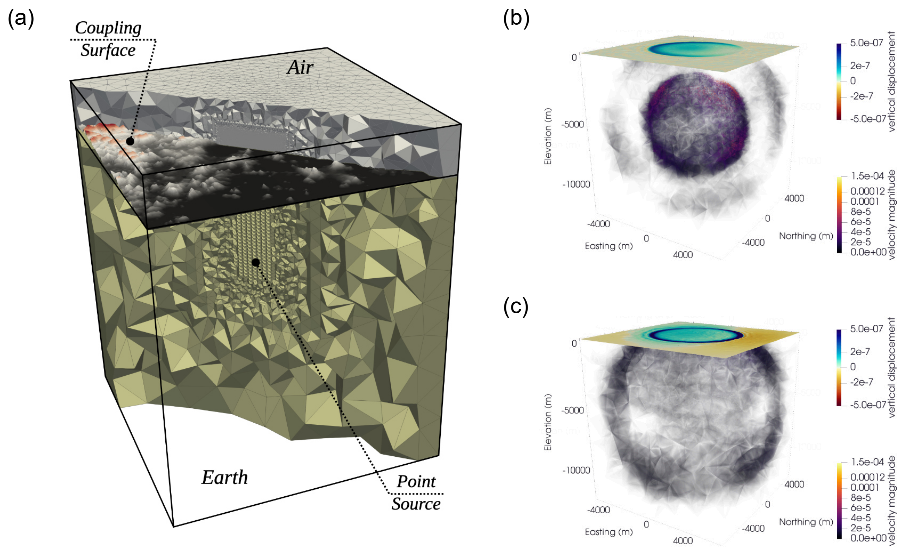


Figure 9.10.: Figure taken from [82]. a) shows the computational mesh with elastic and acoustic layers and the refinement zone. b) and c) show the vertical displacement at the surface and the velocity magnitude in the Earth at 1.2 s and 2.0 s, respectively. The displacement in c), associated with the S wave, is larger than in b), caused by the P wave.

Comparison with measurements

We compare two types of measurements: seismograms and acoustic data. We begin with the former. Figure 9.12 shows three-component seismograms of selected stations (two broadband, four short-period). We filtered both synthetics (red) and observations (black) in the range of 1 Hz to 10 Hz, which allows us to focus on first-order features such as P and S wave travel times and relative amplitude. Our synthetics are an excellent fit for the observations, which shows that our model is reasonable. Additionally, the arrival times at the stations indicate that the local velocity model is a good match for the geological situation in the area. We manually aligned the synthetics and measurements in figure 9.12 such that the P arrival time matches. While we aligned each station separately, the required time shift varies only by up to 0.02 s, again demonstrating that the velocities in the region are mostly homogeneous.

Furthermore, we can see that the source moment tensor, obtained from first-order polarity data [64] and waveform inversion [127], matches the true source well. This leads to the observed nearly perfect consistency of the polarities and the P wave to S wave amplitude ratio of synthetics and observations. However, some small structural heterogeneities manifest in differences between measurements and synthetics. For this, consider, for example, the difference in the S wave in the HEL1, PK01, KUN, and MKK stations in the N channel.

We also compare our synthetics with acoustic measurements from [91] and focus on the FIN2 array, as the FIN1 array had a lower data quality. Similarly to the elastic

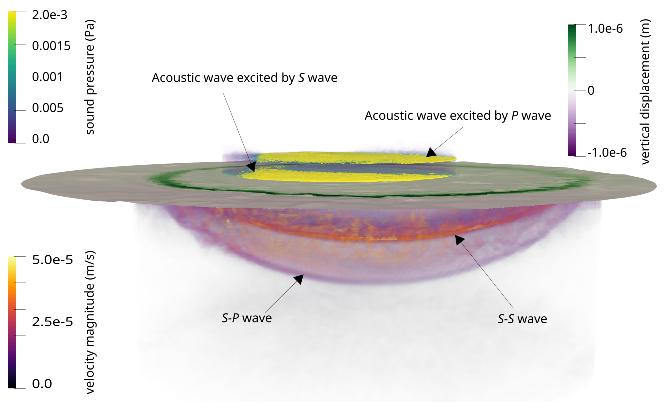


Figure 9.11.: Figure taken from [82]. Volume visualization showing the acoustic and elastic wavefields within specific radii around the source at 2 s. The acoustic wavefield is presented in a cylindrical region with a radius of 2 km, while the seismic wavefield is displayed within a radius of 4 km. It shows two distinct wavefronts in the acoustic layer: The upper wavefront corresponds to the P wave excitation, and the lower wavefront represents the excitation caused by the S wave. In the elastic layer, the visualization captures the reflected S wave (S–S) and the reflected P wave (S–P). These reflections originate from the interaction between the incident S wave and the elastic-acoustic interface.

measurements, we are interested in the first-order properties of our acoustic data. Hence, we consider waveform envelopes (figure 9.13) and spectrograms (figure 9.14). The FIN2 array comprises four sensors deployed within an 80 m distance. In contrast to the seismograms, the acoustic measurements have much higher inter-sensor variability, which our synthetics do not match (figure 9.13). However, the (not normalized) envelopes are consistent with the measurements as the absolute amplitudes of the P wave ($\approx$0.005 Pa) and S wave ($\approx$0.01 Pa) obtained from our fully coupled simulation are in the range of the observed values. The P wave and S wave arrival times match well, and both observations and measurements show that the S wave has a higher amplitude than the P wave. The main differences between synthetics and observed values are that we neither match the intra-array difference nor the energy content of the P and S wave coda.

Figure 9.14 compares the spectrograms of synthetics and observations. Here, we observe the same patterns: The difference between stations is significant, and we match the P and S wave arrivals well but do not match the coda. Furthermore, the S wave carries more acoustic energy than the P wave. We limit ourselves to frequencies of up to around 25 Hz, as these are resolved well by our simulation. The energy content in higher frequency bands of our simulation is likely corrupted by numerical issues such as dissipation and dispersion, which, in part, is caused by the Gibbs phenomenon due to the high-order approximation scheme that we employed [61].

Hence, to summarize, we match the first-order properties of both seismic and acoustic

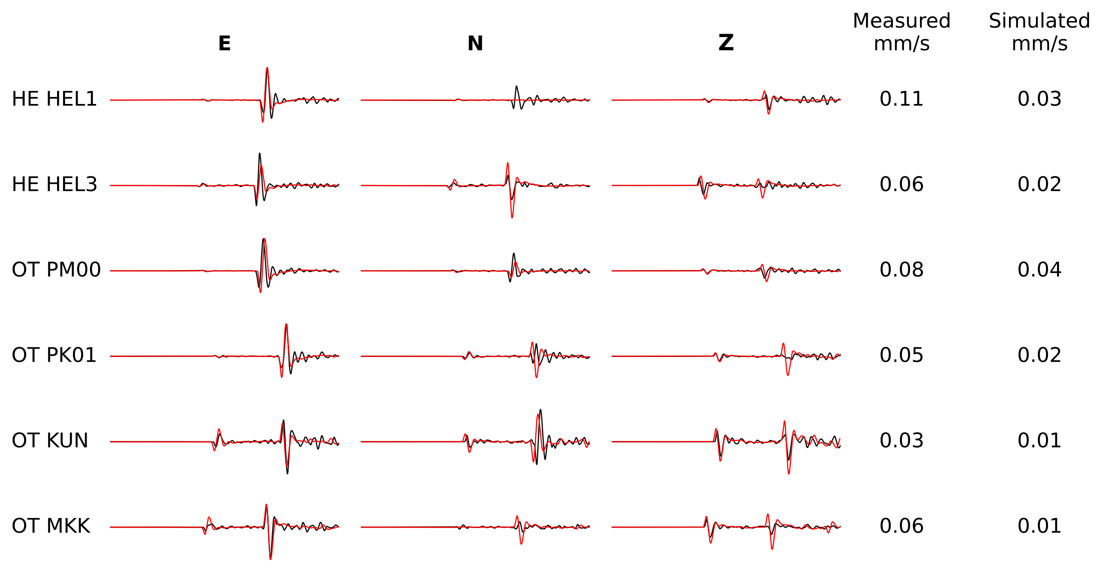


Figure 9.12.: Figure taken from [82]. We compare observed (black) and synthetic (red) velocity waveforms at two permanent broadband (HE) and four temporary short-period seismic stations (OT). We processed the seismic records by removing the instrument response using pre-filter corner frequencies of 0.5 and 40 Hz, with a bandpass filter (1 to 10 Hz) applied to both data and synthetics. We manually aligned synthetic and observed waveforms based on the P wave arrival time to account for the arbitrarily chosen onset of our source time function. This alignment varies by up to 0.02 s between stations to accommodate for velocity heterogeneities absent in our one-dimensional velocity model. We normalized the synthetic and observed velocity waveforms individually by the peak velocity indicated in the last two columns.

measurements well. However, this analysis shows the limits of our computational setup.

Calibration

We fit the linear regression, given by equation (9.5), using the statistical library statsmodels [139]. Table 9.2 shows the obtained coefficients for all considered sources. The intercept is non-zero for all sources, which implies that there are non-linear effects. The slope is smaller than the theoretical prediction of equation (9.2). The intercept and the slope are statistically significant at the $p < 0.001$ level. As the coefficient of determination r^2 indicates, the fit is good for all sources.

For the M_L 1.8 event 13, the values $c_0 = 0.00118 \pm 0.00036$ and $c_1 = 393.64 \pm 9.11$, where $\pm$ gives the uncertainty with respect to the 95 % confidence interval, result in a good approximation. Figure 9.15 shows the excellent agreement of the predicted SPL with the SPL obtained from the fully coupled model. However, there are some outliers,

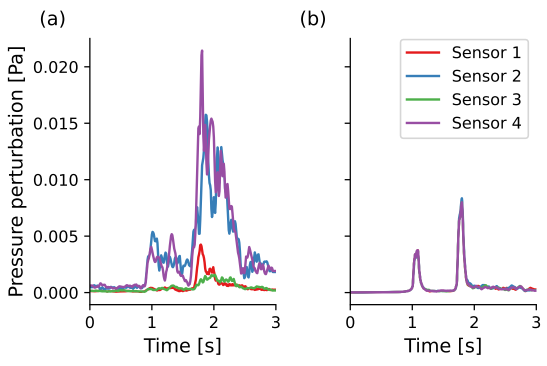


Figure 9.13.: Figure taken from [82]. (a) Envelope of the acoustic measurements of the FIN2 stations [91]. Even though the distance between stations is only about 10 m to 30 m, we see stark differences between the stations. (b) shows the envelopes of our synthetic measurements. Here, the data for all sensors agree with each other. For both (a,b), we used a 1 Hz high-pass filter and afterward smoothed the envelope with a centered moving average filter with a window size of 0.04 s, similarly as done in [91].

especially for high peak vertical velocities.

Table 9.2.: Results of the linear regression for all considered source rotations. The symbol $\pm$ indicates a 95 % Student-t confidence interval and r^2 is the coefficient of determination.

Source	Intercept	Slope	r^2
Event 13	0.00118 ± 0.00036	393.64 ± 9.11	0.839
Strike + 90	0.00243 ± 0.00048	356.22 ± 13.34	0.665
Dip + 90	0.00155 ± 0.00046	381.97 ± 12.30	0.728
Rake + 90	0.00030 ± 0.00019	414.62 ± 7.97	0.883
Orthogonal	0.00265 ± 0.00055	354.70 ± 14.24	0.633

P & S waves

While it is commonly assumed that the P wave is responsible for the sound generation [63], the situation is more complex in our case. Figure 9.16 shows the ratio of the peak SPL during the P wave and S wave, using the windows which we defined earlier. For areas close to the epicenter, the P wave is the dominating wave. Especially up to around 1 km epicentral distance, the S wave leads to a smaller SPL than the P wave. However, for larger distances, the S wave becomes more dominant.

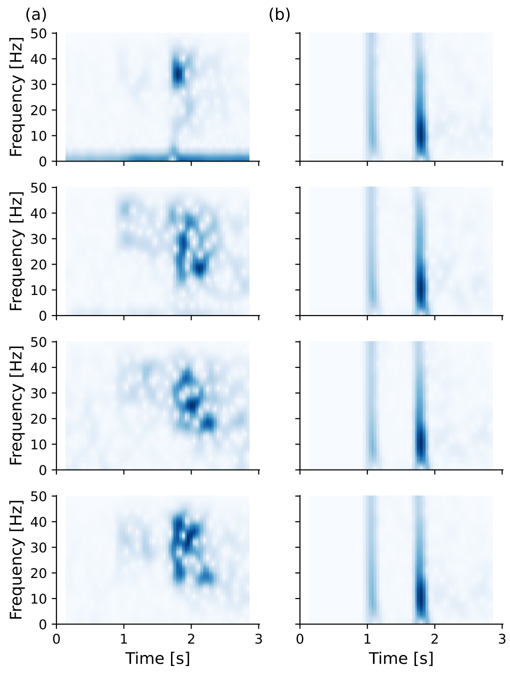


Figure 9.14.: Figure taken from [82]. Spectrograms measured at the FIN2 stations [91] (a) versus the synthetic spectrograms (downsampled by a factor of two) of our simulation (b). We show weak and strong squared power densities as light and dark colors. As we only resolve up to roughly 25 Hz, the frequency content in the upper half of our synthetic may be caused by numerical noise.

Maps

We now discuss the results of our two-step workflow. Using the calibration results, we now show the approximate SPL for the simulation domain, depicted as the black square in figure 9.7. In detail, we use the linear regression coefficients from table 9.2 to predict the peak SPL from the measured peak vertical ground velocity. As mentioned before, this combines the fully coupled simulations with the better-resolved Earth-only simulations. Figure 9.17 depicts the such generated maps. All maps use the Web Mercator projection. We interpolated the pointwise SPL values to an equidistant grid centered at the source. With an area of 8 km × 8 km, it is smaller than the simulation domain, which helps to minimize reflections stemming from our not perfectly absorbing boundary conditions. Furthermore, the first row of figure 9.17 shows the macroseismic response distribution associated with the M_{L} 1.8 event 13. The markers correspond to reports of audible (‘x’), shaking (diamonds), and combined (circles) sensations.

The first column shows the peak horizontal ground velocity (PGV) in mm s^{-1}. The PGV is a standard measurement in earthquake engineering as it is a good proxy for perceptible ground motion. We can see the effects of the source mechanism. Furthermore, we observe topography effects.

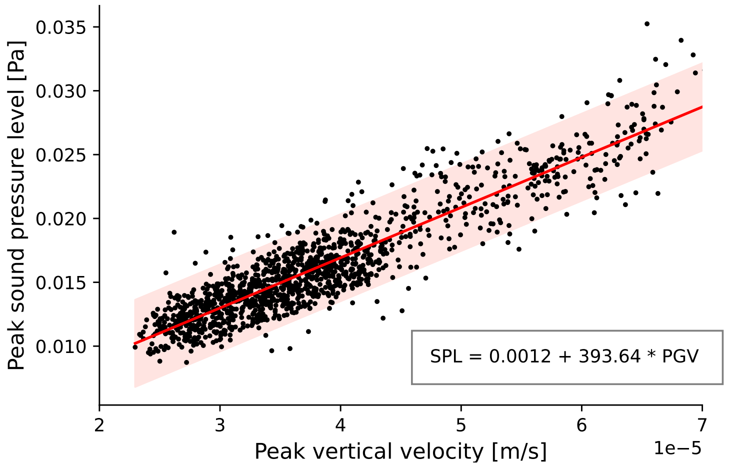


Figure 9.15.: Figure taken from [82]. Comparison of the peak vertical velocity with the peak sound pressure level for our fully coupled model of event 13 inside the refinement zone. The red line shows the linear regression fit, also shown as an equation in the box. We represent a 95 % prediction interval by the shaded area.

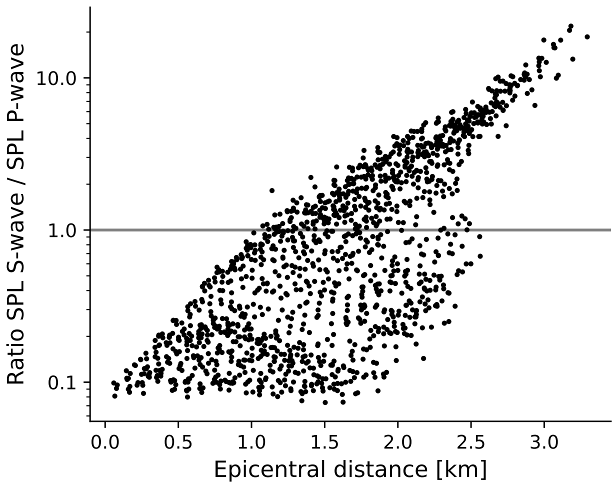


Figure 9.16.: Figure taken from [82]. Comparison of the epicentral distance (x-axis) with the ratio between the peak sound pressure level for the S and P wave. The black line marks a ratio of 1. Close to the epicenter, the P wave is primarily responsible for sound generation, while further away, the effect of the S wave is more dominant.

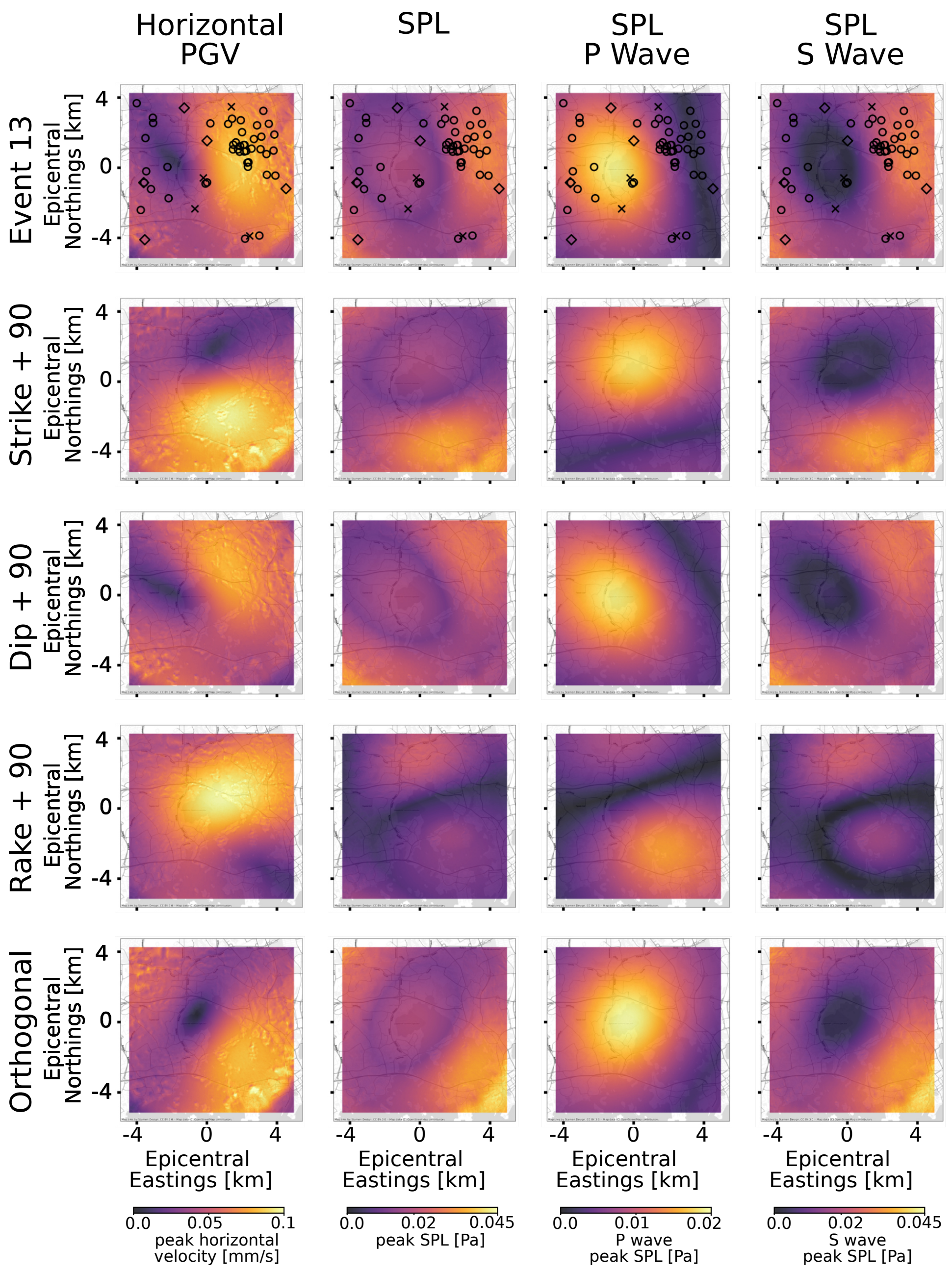


Figure 9.17.: Figure taken from [82]. Five fully coupled simulations with variable source geometries. Top row markers indicate observations and results tied to the observed M_L 1.8 event mechanism [64]: The symbols ('x', diamonds, circles) denote audible, shaking, and combined sensations. The columns "Horizontal PGV", "SPL", "SPL P wave", and "SPL S wave" display peak ground velocity ($\mathrm{m\,s^{-1}}$) and sound pressure level (Pa) estimates. Rows two to five correspond to modified orientations of the original moment tensor point source (table 9.1).

The second column shows the reconstructed peak SPL in Pa. Again, the patterns depend on the focal mechanism. Some regions have low sound excitation, which, due to our linear approximation (equation (9.5)), implies that the peak vertical ground velocity is small. However, the horizontal PGV is not necessarily zero in the same regions, which can help explain why there are some regions where only audible disturbances were reported without a corresponding shaking.

The last two columns show the peak SPL during the P and S wave windows. Here, we use a different colormap for both waves, which hides the fact that the peak SPL induced by the P wave is typically much smaller (compare also with figure 9.16).

The first row of figure 9.17 shows the M_L 1.8 event 13. The source mechanisms are governed primarily by the response to the stress condition in the reservoir [89]. Thus, most observed induced events were similar to our reference event [64, 94, 127]. As demonstrated, the PGV and peak SPL patterns are directly related to the resulting faulting mechanism. We also considered (rows two to five) other source orientations, illustrating that different reservoir structures would impact the induced events. While these sources are unrealistic for the stimulations in the Otaniemi project context, they can be considered a case study of different reservoir configurations. Hence, the subsurface response is another potential variable to consider when planning EGSs.

9.3.3. Conclusion

This section presented a model for an earthquake induced by the stimulation of an EGS. We showed how it can be modeled using a realistic velocity model, topography, and a realistic point source. We presented two setups: a fully coupled elastic-acoustic model that directly captures the sound generation and a more refined Earth-only model. Our models resolve frequencies that overlap with the lower limit of human hearing. Furthermore, we presented a novel workflow that uses the synthetic sound pressure levels computed from the fully coupled model to fit a linear regression model, which we then used to predict the peak sound pressure levels from the peak vertical ground velocity obtained from our Earth-only model.

We demonstrated that our synthetic seismograms and acoustic time series match the observations well. Contrasting the usual assumption, the S wave led to the most intense sound generation. Our results show that our implementation of elastic-acoustic coupling can be used for this application. It could become an essential tool for the planning, execution, and analysis of future EGS stimulations.

9.4. Summary

We have shown three setups for elastic-acoustic coupling. We used the first, the earthquake-tsunami benchmark (section 9.1), to discuss the effects of acoustic waves and to compare our fully coupled model with one-way linking. The second, the Palu, Sulawesi, earthquake-tsunami scenario (section 9.2), is a fully coupled setup for a real-life event. It showed that our model can capture the entire dynamics of a tsunamigenic earthquake, from frictional fault failure, to wave propagation in the Earth and the ocean, to tsunami propagation.

Our results compare well with the one-way linked reference model, and production runs for this scenario achieved petascale performance on three supercomputers. The third scenario, the St1 Otaniemi EGS in the Helsinki metropolitan area (section 9.3), showed that we can use the elastic-acoustic coupling to simulate sound generated from small, induced earthquakes. We presented a novel workflow to combine coarser fully coupled simulations with highly resolved Earth-only models. These scenarios show that our fully coupled model can be used to simulate realistic scenarios.

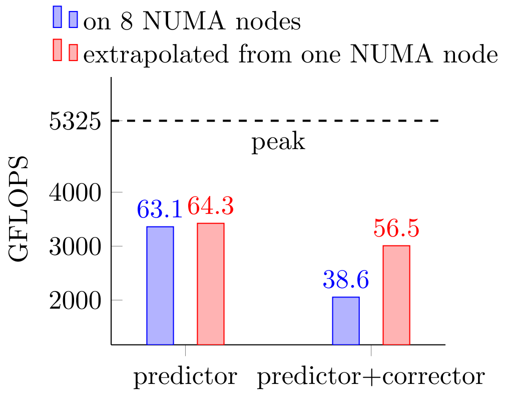
Furthermore, all scenarios used LTS to drastically reduce the time-to-solution. The earthquake-tsunami benchmark used a rate-5 LTS to achieve a speedup of five compared to GTS. The Palu scenario had an LTS speedup of 27 (**M**) and 38.1 (**L**). Finally, the Helsinki scenario had a speedup of 64.8 and 586.7 for fully coupled and Earth-only setups, respectively. The results show that our new LTS implementation works reliably for different LTS rates, wiggle factors, and cluster merging configurations.

Chapter 10.

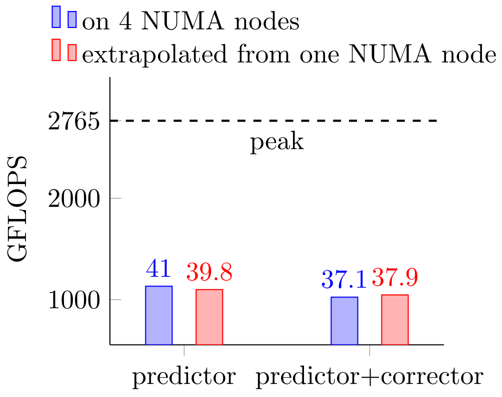
High-Performance Computing

In this chapter, we present high-performance computing aspects of our implementation. We begin by discussing the single node performance on AMD Rome and Fujitsu A64FX CPUs in section 10.1. Following these results, section 10.2 introduces an algorithm to set the affinity of our computation and communication threads so that they respect non-uniform memory access (NUMA). In section 10.3, we summarize how we can optimize SeisSol's graph-based mesh partitioning method for the fully coupled model: Elements can have different computational costs, for example, due to the gravitational boundary condition or dynamic rupture. We must include this cost in the mesh partitioning to achieve good load balancing. Finally, we evaluate the scalability of the Palu scenario in section 10.4 on the cluster Frontera.

10.1. Single Node Performance



(a) Mahti, dual-socket AMD EPYC 7H12



(b) Isambard, Fujitsu A64FX

Figure 10.1.: Single node performance of SeisSol-proxy. The blue bars correspond to the performance obtained on the entire node. For the red bars, we ran our benchmark on one NUMA node and multiplied the resulting performance by the number of NUMA nodes. Hence, this corresponds to the performance in the absence of NUMA effects. The numbers above bars correspond to the reached percentage of the peak performance.

In this section, we investigate single node performance on the AMD Rome and Fujitsu A64FX architectures. We ported SeisSol to the AMD Rome architecture in [79]. As mentioned in section 4.4, SeisSol uses the code generator YATeTo [166] to map the computational kernels to efficient code. It maps tensor expressions to subroutines for small GEMM operations, which are then executed by optimized backends. On the AMD Rome architecture, we use the backend LIBXSMM [59]. As this backend does not support the Rome architecture directly, we generate code for the Intel Haswell architecture, which uses the AVX2 vector instruction set that is also supported by the Rome CPU. To evaluate our success, we use a performance proxy, called "SeisSol-proxy", which executes the computational kernels used in the wave propagation part of the solver on random data [168]. The kernels are split into predictor and corrector, as outlined in section 4.4. The predictor is a strictly local computation; the corrector also depends on data from neighboring elements. In this section, we ran the proxy for 100 loops, each with 300 000 elements.

Furthermore, we evaluate the effects of non-uniform memory access (NUMA). NUMA architectures divide the CPU into multiple NUMA nodes. Each NUMA node owns a part of the memory. Accessing nearby data, i.e., data directly connected to a NUMA node, is fast, while accessing further away data is more expensive. Hence, the memory bandwidth and latency depend on the location of the data. We expect these effects in the corrector kernel but not in the predictor kernel. SeisSol uses the "first-touch policy", which allocates memory on the NUMA node where it is first accessed, ensuring that each core processes elements located on the nearest NUMA node. However, this cannot help us much with the corrector kernel, as data from other elements can be located on other NUMA nodes.

In [79], we tested this hypothesis while simultaneously evaluating the single node performance on the AMD Rome architecture. We used a single node of the Mahti cluster. It is a dual-socket AMD Rome 7H12 system with a base frequency of 2.6 GHz. Each CPU has 64 cores and 4 NUMA nodes. We always use the entire node to ensure that the clock rate does not increase to the turbo frequency of 3.3 GHz. Hence, we assume a peak performance of 5325 GFLOPS per node. Our results are shown in figure 10.1a. We compare running the kernels on all available cores or on one single NUMA node. Scaling the latter performance with the number of NUMA nodes results in the extrapolated performance, i.e., the performance we would get without NUMA effects. When running only the predictor, we achieved a performance of 3360 GFLOPS (63 % of peak) on all nodes, the extrapolated performance was 8×428 GFLOPS = 3424 GFLOPS, corresponding to 64 % of the peak performance. The difference between both results is insignificant, so the predictor does not suffer from NUMA effects. The situation drastically differs for the entire wave propagation kernel, i.e., running predictor and corrector. Here we achieved 2053 GFLOPS (38 % of peak) on all codes and 8×376 GFLOPS = 3008 GFLOPS (56 % of peak) extrapolated performance. Thus, the corrector kernel exhibits strong NUMA effects.

Using one MPI rank per NUMA node may make sense for this architecture. Of course, this only moves the NUMA effects to the message passing; however, the kernels will not suffer from NUMA effects. As mentioned in sections 6.6 and 6.7, we can hide the communication behind the computations. Thus, it seems likely that we could also hide the cost of NUMA data transfer.

Additionally, we ported SeisSol to the Fujitsu A64FX CPU, which uses an Arm instruction set with Scalable Vector Instructions (SVE). We used the just-in-time compilation interface of LIBXSMM and a modified version of PSpaMM, which we extended to generate SVE instructions, as code generation backends. We investigated the resulting performance on the cluster Isambard [51]. Each of its compute nodes has one A64FX CPU at 1.8 GHz with 48 threads, resulting in a peak performance of 2765 GFLOPS. The CPU is divided into 4 NUMA nodes. An interesting detail about the A64FX CPU is that it uses high-bandwidth memory, which should lead to faster memory access. To check this, we measured the memory bandwidth with the software likwid [162]. Out of five measurements, the best was a memory bandwidth of approximately $885\,\mathrm{GB\,s^{-1}}$. To contrast, on one Mahti node, we measured only $340\,\mathrm{GB\,s^{-1}}$, less than half of the bandwidth of one Isambard node. Considering that a Mahti node has a peak performance that is roughly twice the peak performance of one Isambard node, it becomes clear that the A64FX CPU has a significantly different machine balance.

The machine balance influences the performance of our kernels. To evaluate the resulting effect, we repeated the experiment that we performed on Mahti: We ran SeisSol-proxy on one A64FX CPU using 48 threads.[1] Figure 10.1b shows the results. We achieved a performance (best out of five runs) of 1134 GFLOPS for the predictor and 1027 GFLOPS for both kernels, corresponding to 41 % and 37.1 % of peak, respectively. When running only on one NUMA node, we achieved an extrapolated performance of $4 \times 274\,\mathrm{GFLOPS} = 1099\,\mathrm{GFLOPS}$ and $4 \times 261\,\mathrm{GFLOPS} = 1045\,\mathrm{GFLOPS}$. Thus, we observe no significant NUMA effects on this architecture. Furthermore, the difference between the performance of corrector and predictor kernels is smaller than on the AMD Rome architecture, which is most likely caused by the different machine balance of the A64FX CPU.

10.2. Pinning

SeisSol's default operation mode used one MPI rank per node with NUMA-aware memory initialization and a communication thread to ensure MPI progression. This approach, combined with OpenMP for shared memory parallelism, led to good performance on previous-generation supercomputers (see, e.g., the results in [13, 58, 168]). However, as the single node results in section 10.1 suggest, running multiple MPI ranks per node might increase the performance on some architectures.

We explained implementation strategies for the communication thread in section 6.6. It runs an infinite loop that calls `MPI_Test`, which is computationally expensive. Hence, we need to ensure that the communication threads, realized as POSIX threads (pthreads), do not run on cores used for the OpenMP workers.[2] We also want to restrict the freedom of the operating system scheduler to move worker threads to different cores, as this could

[1]We used the SeisSol version with commit hash `5a932232047967b4f4fbfa129855c699e5cd829f`.

[2]The same principle also holds for threads used for asynchronous output by SeisSol [126].

Algorithm 13 Functions that pin all threads in a NUMA-aware manner. We assume that the OpenMP threads are pinned to CPU threads by the OpenMP runtime. We use the Linux functions `sched_getaffinity` and `sched_setaffinity` to get and set the affinity of a process. The functions `CPU_SET` and `CPU_ISSET` set and check an entry of the CPU mask. We use `getZeroMask` to get an empty CPU mask and `CPU_OR` to compute the logical or of two CPU masks. Finally, `numa_node_of_cpu` returns the NUMA node id of a thread. Note that we simplified the interface of these methods. We define the functions `getWorkerUnionMask` and `getNodeMask` that create a mask that stores which CPUs of the current MPI process or compute node, respectively, are used by an OpenMP worker thread. We use `getNodeCommunicator` to communicate with all MPI ranks that reside on the same compute node. The function `getFreeCPUsMask` returns a mask of all unoccupied CPUs that share a NUMA node with any worker thread of this process. Finally, we pin the current thread to this mask with `pinToFreeCPUs`.

```
function pinToFreeCPUs()
    freeMask ← getFreeCPUsMask()
    sched_setaffinity(0, freeMask)                        ▷ Pin current thread
function getFreeCPUsMask()
    openMPMask ← getWorkerUnionMask()
    nodeOpenMPMask ← getNodeMask()
    freeMask ← getZeroMask()                              ▷ Set mask to zero
    numaNodesOfThisProcess ← set(int)
    for cpu = $0, 1, \dots$ do
        if CPU_ISSET(cpu, openMPMask) then
            insert(numaNodesOfThisProcess, numa_node_of_cpu(cpu))
    for cpu = $0, 1, \dots$ do
        if ¬CPU_ISSET(cpu, nodeOpenMPMask) then
            numaNode ← numa_node_of_cpu(cpu)
            if numaNode ∈ numaNodesOfThisProcess then
                CPU_SET(cpu, &freeMask)
    return freeMask
function getNodeMask()
    workerMask ← getWorkerUnionMask()
    return allReduce(workerMask, ∨, getNodeCommunicator() )
function getWorkerUnionMask()
    workerUnion ← getZeroMask()
    parallel (shared(workerUnion)) do             ▷ Run on all OpenMP workers
        worker ← sched_getaffinity(0, worker)     ▷ Get affinity of current thread
        critical do                                   ▷ Perform sequentially
            workerUnion ← CPU_OR(workerUnion, worker)
    return workerUnion
```

lead to performance degradation and, thus, to load balancing issues. Furthermore, if the scheduler moves a thread to another NUMA node, this could create NUMA issues even for the predictor kernel. Hence, we must manually pin worker and communication threads to cores. We developed a simple and portable algorithm to correctly pin communication threads using multiple ranks per node in [79].

In the following, we assume that our machine has p physical cores and a hyperthreading factor of s. The latter factor determines how many logical threads are available per physical core. We use hyperthreading for all clusters that support it. The first step is pinning the worker threads, for which we rely on OpenMP's mechanism for thread affinity. In detail, we set the number of OpenMP threads to $s(p-1)$ with the environmental variable `OMP_NUM_THREADS` $= s(p-1)$. Furthermore, we pin the OpenMP threads to logical cores using `OMP_PLACES` $= p-1$.

Now, we need to find all cores that are still free, and that share a NUMA domain with at least one worker thread. We first compute a CPU mask on each rank that stores whether the worker threads use a core. We use `MPI_COMM_TYPE_SHARED` to split the MPI communicator. This communicator contains all ranks that share a compute node. Next, we use this communicator to combine all masks of one node by using a logical `or`. Then, we use libnuma to compute a list of all NUMA nodes used by our OpenMP threads on this rank. Finally, we pin the communication thread to all free logical cores that share a NUMA node with the compute threads of the current MPI rank. Note that this relies on the MPI runtime to pin our program to the correct part of the compute node. Alternatively, we can pin the entire program manually to a part of the node, e.g., using numactl. We thus ensure, in a portable way, that pinning of pthreads is NUMA aware and that they do not interfere with the worker threads. Algorithm 13 summarizes the resulting algorithm.

10.3. Mesh Partitioning

We introduced the distributed memory parallelization for our actor model in section 6.6. For this to work well, we must distribute the work fairly between processors when using distributed memory parallelism. The approach in SeisSol is to use static load balancing. We facilitate this by using the graph-based partitioning software parMETIS [135]. Hence, we create a dual graph of our mesh by defining a vertex for each element and an edge for all elements connected by a face. We can use vertex weights, which model the cost per element, and edge weights, which model the communication cost. Selecting well-approximated weights is crucial because each element's expected cost can vary. The most significant factor is the LTS, which results in some elements being updated more often than others. Another factor is the increasingly complex multiphysics models that we are using: dynamic rupture and the gravitational boundary condition. In this section, we summarize our work in [79]

We define the total weight of an element by

$$w_{\text{element}} = \underbrace{2^{c_{\max}-c_v}}_{\text{time step factor}} \left(w_{\text{base}} + n_{\text{DR}} w_{\text{DR}} + n_{\text{G}} w_{\text{G}}\right), \tag{10.1}$$

which is consistent with the cost we assumed for the clustering, defined by equation (6.19). It contains multiple weights. First, we compute the (integer) cost per time step. It consists of a base weights w_{base}, which we assume to be constant for all elements. The cost of faces with dynamic rupture boundary conditions is reflected in the number of such faces n_{DR}, and the cost of each face w_{DR} [164]. We do the same for the n_{G} faces with gravity, which each are assumed to have a cost of w_{G}. Second, we compute a factor that considers how often the element needs to be updated. We weigh each element with the update rate, given by the term $2^{c_{\max}-c_v}$, where c_v is the cluster number of the element v and $c_{\max}$ is the id of the largest time cluster [13]. Hence, elements in the cluster with the highest time step size have a factor of one, as they are most seldom updated.

It is a common observation that modern supercomputers exhibit strong performance fluctuations. The performance of each node can change during the simulation, and the performance of nodes can differ. While the first problem would require dynamic load balancing, the second problem can be solved by static load balancing. The critical insight is that we must treat supercomputers as heterogeneous hardware [108, 183]. Hence, we assign node weights to each node, as discussed in [164]. This is done by benchmarking our computational kernels with a simple proxy application. We use the (normalized) inverse computational time as a node weight, which we feed into ParMETIS (`tpwgts`).

10.4. Strong Scaling

In this section, we evaluate the strong scaling of the fully coupled Palu scenario (section 9.2) using our implementation of elastic-acoustic coupling and our new LTS scheme. SeisSol has been shown to scale well for large-scale dynamic-rupture simulations [58, 168]. Here, we present an extension of the strong scaling section of [79]. We begin by summarizing the effect of NUMA for the strong scaling on the cluster Mahti and then demonstrate the strong scaling of the implementation presented in this thesis. Additionally, we compute the load balancing weights introduced in section 10.3.

We measure the performance in the unit hardware GFLOPS per node. This includes multiplications with zeros because we embed the acoustic wave equation in the elastic wave equation and because some matrices are sparse. To compute the performance in GFLOPS, we divide the total number of floating point operations by the time between the first and last time step. Hence, we do not include time spent to initialize the simulation. In this section, we write GFLOPS for the performance in GFLOPS normalized with the number of nodes. We used order five polynomials for all simulations and deactivated all output.

Figure 10.2 shows the strong-scaling results we obtained on Mahti, as described in [79]. The results use an older version of SeisSol, available at [80].[3] As discussed in section 10.1, we notice strong NUMA effects on modern architectures. Hence, we also want to evaluate how many MPI ranks we should use per node. We evaluated four different configurations: One MPI rank per node, one per socket (2 per node) and one per NUMA node (4 per node). Note that we sacrifice one OpenMP thread per MPI rank to facilitate asynchronous

[3]We cannot re-run this setup with the current version, as our "grand-challenge" cluster allocation on Mahti has expired.

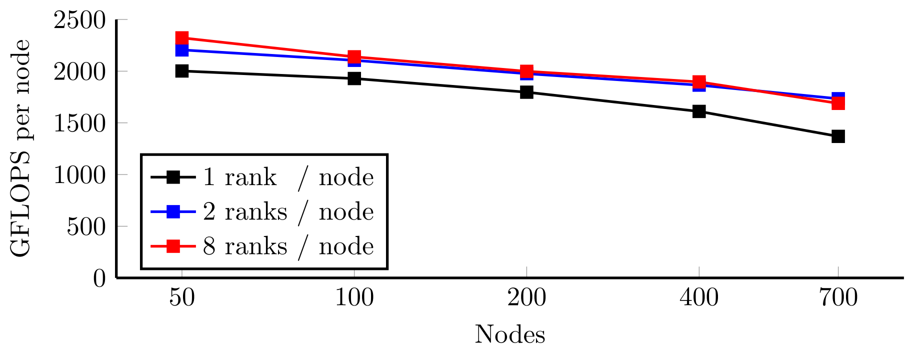


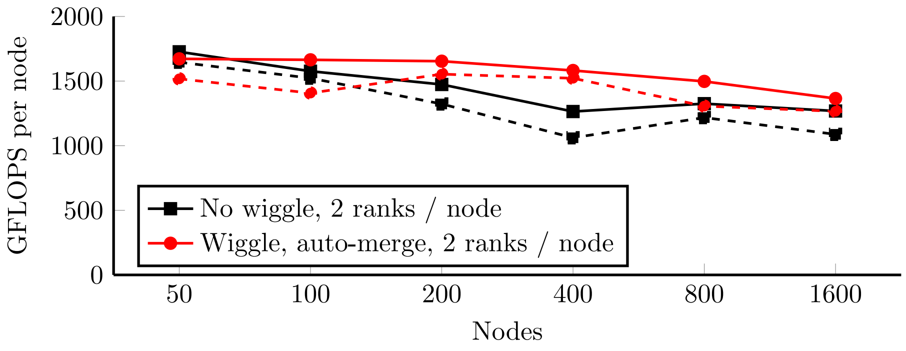
Figure 10.2.: Figure reproduced from [79]. Parallel efficiency for the **M** Palu mesh on the cluster Mahti for 50 to 700 nodes and 1,2 or 8 MPI ranks per node. Note that this uses an older version of SeisSol and, thus, a different implementation of the gravitational boundary condition and a different LTS implementation.

output and MPI progression due to the communication thread (sections 6.6.2 and 10.2) and the output thread [126].

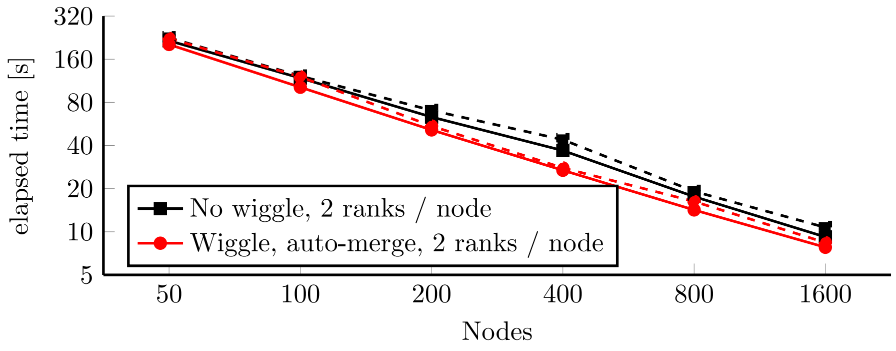
For this test, we report the maximal performance over multiple runs to avoid performance fluctuations. We run the **M** setup for the Palu earthquake-tsunami (section 9.2) for 0.03 s. As figure 10.2 shows, we achieved the best performance using 8 MPI ranks per node. For this case, we achieved 2322 GFLOPS on 50 nodes and 1689 GFLOPS on 700 nodes. This translates to a parallel efficiency of approximately 73 % and a maximum total performance of ≈1.18 PFLOPS on Mahti.

We performed strong scaling simulations on the cluster Frontera with SeisSol version 1.0.1 [167], which includes all features presented in this thesis. We simulated the first 0.1 s of the Palu scenario and conducted experiments with and without wiggle factor (introduced in section 6.8). For the wiggle factor experiments, we used a minimum wiggle factor of 0.51, a grid search step size of 0.01, and automatic cluster merging with an allowed cost overhead of 0.01. The optimal settings for this are $\lambda = 0.65$ and eight clusters for the **M** mesh and $\lambda = 0.71$ and eight clusters for the **L** mesh, as described in section 9.2.

Before we continue with the strong scaling, we must find an optimal set of element weights, which we introduced in section 10.3. To set them, we ran a grid search on Frontera for the weights in the range of $w_{\text{G}} = 0, \ldots, 100$, using fixed weights of $w_{\text{base}} = 100$ and $w_{\text{DR}} = 200$. We used 50 nodes and the **M** Palu mesh. We ran five simulations, each for 0.1 s simulated time. First, we consider the best performance reached out of all five simulations. The best choice, $w_{\text{G}} = 50$, led to a performance 1693 GFLOPS, and the worst choice, $w_{\text{G}} = 25$, resulted in 1654 GFLOPS. However, the difference between the best and worst performance of all runs with $w_{\text{G}} = 50$ is 154 GFLOPS, which is larger than the difference between all considered weights! Additionally, we performed a similar grid search on SuperMUC-NG, which resulted in a similar performance with an optimal weight of 75 or 100. Hence, we decided to use a weight of $w_{\text{G}} = 75$ as a compromise.



(a) Parallel efficiency



(b) Time-to-solution. Note: We use a log-log scale.

Figure 10.3.: Performance using the **M** mesh on Frontera. The solid lines indicate the best achieved performance. The dashed lines are the worst-case performance.

Note, however, that this is an approximate weight.

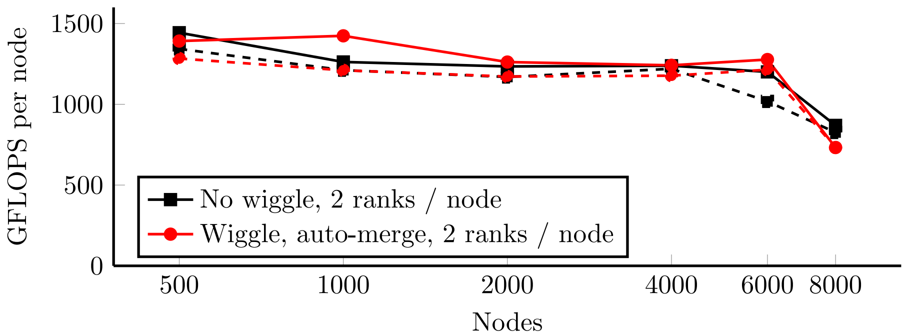
To contrast this, for the implementation in [79], we used a weight of $w_{\text{G}} = 300$. This hints that the implementation of the gravitational boundary condition presented in this thesis is more efficient than the one described in [79]. Further problems arise with w_{DR}: First, we only simulate for a short period, but the cost of dynamic rupture varies with the numerical solution as we need to compute the slip across the fault, which involves solving a non-linear system with Newton's method. The number of Newton iterations and, thus, the computational cost typically increases after the fault ruptured. Second, we compute the fault slip rate twice when a dynamic rupture face is adjacent to two partitions. Hence, the cost of a dynamic rupture face also depends on the partitioning. Thus, perfect load balancing would require dynamic load balancing. A similar grid search procedure for the parameter w_{DR} in [79] did not reveal a clear choice. The results of [79] were obtained with a weight of $w_{\text{DR}} = 200$. Considering both grid searches, we set the values $w_{\text{G}} = 75$ and $w_{\text{DR}} = 200$, which results in an adequate parallel efficiency. However, the performance results could be tuned even further by adjusting these weights. This is, as demonstrated, challenging and a procedure that would need to be repeated for every cluster and every setup.

Using these settings, we performed a strong scaling study on Frontera. We started with the **M** mesh and scaled from 50 nodes to 1600 nodes, similarly as done in our original experiments on SuperMUC-NG and Mahti, described in [79]. For this, we performed three simulations each. We report the result of the best runs; however, figure 10.3 also shows the worst performance. First, we consider the performance in GFLOPS per node (figure 10.3a). Without a wiggle factor, we achieved 1727 GFLOPS per node on 50 nodes and 1268 GFLOPS on 1600 nodes, equivalent to a parallel efficiency of roughly 73 %. When using the optimal wiggle factor and auto-merging of clusters, we achieved 1672 GFLOPs on 50 nodes and 1365 GFLOPS on 1600 nodes, resulting in a parallel efficiency of 81.6 %. We further note that the simulation performance decreased much more slowly than when using no wiggle factor. While the simulation without wiggle factor incurred a significant slowdown when moving from 50 to 100 nodes (1576 GFLOPS) and from 100 to 200 nodes (1474 GFLOPS), the wiggle factor, and especially the auto-merging seems to have a stabilizing effect: Moving to 100 nodes (1665 GFLOPS) and 200 (1654 GFLOPS) for this simulation barely changed the performance!

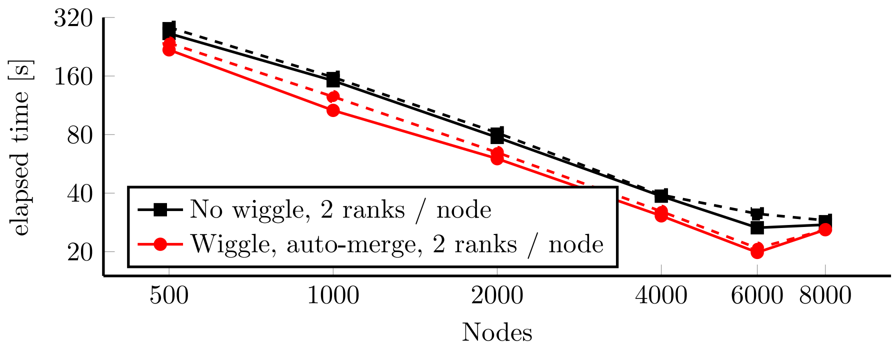
The wiggle factor has the additional effect of decreasing the time-to-solution (figure 10.3b). On 50 nodes, the simulation with wiggle factor took 202.8 s and thus required 94.1 % of the runtime of the simulation without wiggle factor (215.4 s). A similar situation occurred for 1600 nodes, where the simulation with wiggle factor took 7.8 s, which is only 84.7 % of the time of the simulation with wiggle factor (9.2 s) Hence, increasing the number of nodes from 500 to 1600 (a factor of 32) decreased the time-to-solution by a factor of 26.0 and 23.4 with and without wiggle factor, respectively.

In addition, we performed scaling runs on up to 8000 nodes for the **L** mesh. As this has been done during a large-scale allocation, the number of runs we could perform was restricted to a smaller number. We ran three simulations each for all runs with up to 4000 nodes. For the simulations using a wiggle factor, we ran two simulations for 6000 nodes and one simulation for 8000 nodes. We repeated the simulations without a wiggle factor three times for 6000 nodes and two times for 8000 nodes. Figure 10.4 shows the results. First, the results for 8000 nodes are poor for both mesh sizes. They are likely an outlier and might be caused by cluster configuration issues. We thus focus primarily on scaling up to 6000 nodes. Without wiggle factor, we achieve a performance of about 1443 GFLOPS on 500 nodes and 1201 GFLOPS on 6000 nodes. This is equivalent to a parallel efficiency of 83 %. When using both wiggle factor and auto-merging, we have a performance of 1392 GFLOPS on 50 nodes and 1227 GFLOPS on 6000 nodes, leading to a parallel efficiency of 91.8 %. The difference between both settings is not necessarily significant, as system jitter effects can be quite pronounced at this scale. These effects translate directly to unreliable performance estimates for SeisSol [164]. However, we achieved an excellent parallel performance.

The strength of the wiggle factor is even more apparent when we consider the time-to-solution, as shown in figure 10.4b. With the optimal wiggle factor, the simulation took 218.1 s on 500 nodes and 19.8 s on 6000 nodes. Without a wiggle factor, it took 264.7 s on 500 nodes and 26.6 s on 6000 nodes. Hence, the simulation with a wiggle factor required only 82.4 % and 74.7 % of the runtime of the simulation without a wiggle factor



(a) Parallel efficiency



(b) Time-to-solution. Note: We use a log-log scale.

Figure 10.4.: Performance using the **L** mesh on Frontera. The solid lines indicate the best achieved performance, and the dashed lines are the worst-case performance.

on 500 and 6000 nodes, respectively. Furthermore, increasing the number of nodes from 50 to 6000 (a factor of 12) decreased the time-to-solution by 11 and 10 with and without wiggle factor, respectively.

To summarize, we achieved an excellent scaling of both **M** and **L** mesh. The setups that used a wiggle factor resulted in a significantly better time-to-solution and scalability for both mesh sizes.

10.5. Discussion

In this section, we discussed HPC aspects. We optimized SeisSol on the AMD Rome and Fujitsu A64FX architectures. On AMD Rome, we achieved a single node performance of 56.5 % of the peak when neglecting NUMA effects. When considering NUMA effects, we only achieved a performance of 38 % of the peak. However, for the Fujitsu A64FX architecture, we achieved a similar performance with (37.1 % of peak) or without NUMA effects (41 % of peak). These results clarified that we must use multiple MPI ranks per node on some architectures to combat the strong NUMA effects. Hence, section 10.2

introduced an algorithm to automatically pin threads to NUMA nodes. Furthermore, the fully coupled scenarios are complex multiphysics simulations that include many moving parts with different costs. As demonstrated in section 10.3, we can expand the static mesh partitioning with element weights, allowing us to handle varying element costs. We evaluated this in section 10.4 and noticed that it is hard to get robust estimates for the optimal parameters due to the machine jiggle. However, we explained how we set the parameters heuristically. The strong scaling study demonstrated that this simple strategy resulted in excellent parallel scalability on up to 6000 nodes. Furthermore, the wiggle factor allowed us to gain a significant speedup due to better scalability and better time-to-solution: For example, scaling the **L** Palu setup from 500 to 6000 nodes of Frontera led to a parallel efficiency of 91.8 %.

Chapter 11. Conclusion

In this thesis, we introduced a scalable and stable three-dimensional fully coupled elastic-acoustic model discretized with the numerical method ADER-DG. Furthermore, we presented a novel local-time-stepping framework that is elegant and efficient.

We began by establishing the physical model (chapter 2). For this, we derived equations that govern the elastic and acoustic parts of the domain. The former uses the standard elastic wave equations, while the latter uses a linearized acoustic model that includes tsunami waves with a linearized free surface condition. We embed the acoustic equations in the elastic PDE, simplifying the implementation.

Chapters 3 and 4 introduced the ADER-DG discretization using an exact Riemann solver. The resulting method has a high-order of convergence in both space and time. We introduced a Taylor-series-based ADER-integration approach for our gravitational boundary conditions. As chapter 5 showed, the combination of boundary conditions, Riemann solver, and space-discretization culminated in a semi-discrete stable scheme as the energy of the numerical solution is guaranteed to decrease, assuming exact time-integration.

In earthquake simulations, it is typical to have different wave speeds as the material is inhomogeneous. Furthermore, meshing software can lead to small element sizes, especially at the intersection between layers. Both effects are more apparent in fully coupled scenarios. For example, the wave speed in the acoustic layer ($1500\,\mathrm{m\,s^{-1}}$ in the ocean or $343\,\mathrm{m\,s^{-1}}$ in the air) is much smaller than the wave speed in the Earth (around $6000\,\mathrm{m\,s^{-1}}$). This leads to the need for local time-stepping.

SeisSol's original LTS implementation did not work for these scenarios, as it used a scheduling method based on absolute time differences. We presented a new actor model (chapter 6), which combines a state-machine to handle the state of each cluster, with explicit message passing, to make the communication of the cluster states more obvious. This resulted in an elegant abstraction that describes computations and communication. We introduced a wiggle factor to the LTS method, which automatically fine-tunes the cluster distribution by automatically shifting the boundaries of the clusters. Furthermore, we added a feature that automatically merges clusters with large time steps. Both features can be enabled at the same time.

We presented multiple applications for the fully coupled model and the new local time-stepping algorithm. Chapter 7 introduced and compared multiple earthquake-tsunami coupling workflows. We demonstrated the strengths and weaknesses of our fully coupled model and presented standard approximations for the sea surface height, including the

Tanioka approximation and the Kajiura filter. As we showed, these directly result from the fully coupled model. Hence, our proposed model can serve as a reference model for the simulation of off-shore tsunami generation and propagation.

Chapter 8 used a carefully curated selection of analytical solutions to verify our model. We tested the wave propagation in uncoupled media using elastic and acoustic planar waves and the propagation in coupled waves using two elastic-acoustic coupling test cases. The first one, Snell's law, demonstrated that we correctly simulate the interaction of body waves with the elastic-acoustic interface. The second, a Scholte wave, showed that our model correctly handles an elastic-acoustic surface wave. Finally, we verified our discretization of the gravitational boundary condition by comparing it with an exact solution for wave propagation in the ocean. All scenarios achieved high-order convergence up to certain limits imposed by floating-point accuracies. Hence, we are confident that the implementation of our physical model is correct.

In chapter 9, we introduced three scenarios. The first, an earthquake-tsunami benchmark, compared the fully coupled method with a one-way linked method: The fully coupled method captures the tsunami correctly but leads to a more detailed wave structure than one-way linking. The second scenario, our first real-world application, the Palu scenario, models a M_{W} 7.5 supershear earthquake, which triggered a localized and unexpected tsunami. We created large fully coupled models (81 million for the **M** and 518 million elements for the **L** mesh) for this earthquake-tsunami event. While our results compare well with one-way linking, the resulting tsunami differed slightly. We achieved sustained petascale performance for this scenario on the clusters Shaheen-II, SuperMUC-NG, and Frontera. The third scenario showed that our model can also simulate earthquake-sound coupling. We considered a small earthquake induced by an enhanced geothermal system in the metropolitan region of Helsinki. Our novel workflow used linear regression to approximate peak sound pressure level from peak ground vertical velocity. This can be considered a calibrated version of a common rule of thumb. We achieved good results: The first-order features of our results, such as arrival times and the P to S wave ratio, compare well for both elastic and acoustic measurements. Finally, we created maps for the peak ground velocity and the sound pressure distribution. We evaluated the difference between P and S wave: The S wave leads to higher peak sound pressure levels, contrasting the typical observation that the P wave dominates the sound field.

Finally, for all considered scenarios, LTS led to a significant speedup. In detail, for the **L** Palu setup, LTS is roughly 30 times faster than GTS. For the fully coupled Helsinki scenario, LTS requires 64.8 times fewer updates than LTS. Hence, LTS is required to simulate both setups!

Finally, chapter 10 detailed the high-performance computing aspects required for using our fully coupled model on modern clusters. First, we discussed optimizations and the resulting performance for AMD Rome and Fujitsu A64FX CPUs. On AMD Rome, we achieved 38 % of the peak performance when running on all NUMA nodes and up to 56 % when running on one NUMA node. For the Fujitsu A64FX architecture, we achieved a similar performance with (37.1 % of peak) or without NUMA effects (41 % of peak). These results showed that we must consider NUMA effects on some architectures. Hence, we introduced a new way of automatically pinning all threads to the correct NUMA nodes. We discuss modifying the mesh partitioning to accommodate new constraints from the fully coupled model. Finally, we showed strong scaling results demonstrating excellent parallel performance on Frontera for both Palu scenarios. We scaled the **M** mesh from 50 nodes to 1600. Using neither wiggle factor nor cluster merging led to a parallel efficiency of 73 %; using both led to a parallel efficiency of 81.6 %. As an additional effect, the wiggle factor improved the time-to-solution, which was only 84.7 % of the one without the wiggle factor when using 1600 nodes. We did the same for the **L** mesh, which we scaled from 500 to 6000 nodes. Again, using neither wiggle factor nor cluster merging led to a parallel efficiency of 83 %, while enabling both features led to a parallel efficiency of 91.8 %. Similarly to the smaller setup, the time-to-solution when using both features is only 74.7 % of the run without wiggle factor when running on 6000 nodes. This demonstrates that our implementation is highly scalable and that the wiggle factor can drastically reduce the time-to-solution of real-world setups.

To summarize, we presented a three-dimensional fully coupled model and have shown that it is stable and achieves a high order of convergence for multiple scenarios. Together with the validation for multiple application scenarios, this proves that our model works. Our local time-stepping implementation is reliable, and we used it for all application scenarios. Finally, our strong scaling study proved that our implementation works on large supercomputers.

Our implementation of the three-dimensional fully coupled model is used for other applications. [87] applies it to model the Húsavík-Flatey fault zone in North Iceland. A fully coupled model for a tsunami in the Hellenic Arc is a work in progress [179]. Both scenarios extend dynamic rupture models to include a water layer, resulting in models that comprehensively describe the process of tsunamigenesis. These recent applications demonstrate that the model and its implementation, developed in this thesis, are a valuable contribution to the field.

It would be possible to further improve the performance of our implementation by directly simulating the acoustic part of the domain with the acoustic wave equation, i.e., not using the embedding presented in section 2.3. Computational costs often lie primarily in the acoustic layer, for example, for the Palu scenario (section 9.2). Thus, using computations with fewer degrees of freedom in the acoustic region would directly translate to a significant speedup of the entire simulation. However, supporting multiple PDEs within a single simulation poses several challenges, such as load balancing and local time-stepping. Therefore, significant alterations to SeisSol would be necessary. This thesis presented a significant building block for this: We could extend the concept of LTS clusters to differentiate not only between time step sizes but also between PDEs. While our strong scaling experiments demonstrated excellent parallel performance, computing clusters in parallel could enhance the performance and scalability further. The actor model can accommodate this as the clusters manage their respective constraints. However, a challenge remains in carefully fine-tuning this added parallelism layer.

Appendix A. Simulations

This appendix briefly describes the configuration we used for our production runs. Table A.1 summarizes parameters used to obtain the results in chapter 9.

Table A.1.: SeisSol parameters used for our production runs. If the name column contains a citation, we used the results of this citation. Otherwise, the results are new. We used time step sizes of $C(N) = \text{CFL}(2N+1)^{-1}$. We control the mesh partitioning by the weights w_G and w_DR, which we introduced in section 10.3. Both are zero for the HEL setups, as neither uses the gravitational boundary or dynamic rupture. We describe the LTS configuration (chapter 6) by the rate r, the wiggle factor λ, and merging, which indicates whether the simulation used the automatic merging of LTS clusters.

Name	CFL	w_G	w_DR	r	λ	Merging
Earthquake-tsunami benchmark [79]	0.35	300	100	2	1	✗
Earthquake-tsunami benchmark, v1.0.1	0.35	100	100	5	1	✗
Palu [79]	0.35	300	200	2	1	✗
Palu, no wiggle, v1.0.1	0.35	75	200	2	1	✗
Palu, wiggle, v1.0.1	0.35	75	200	2	0.65	✓
Helsinki, fully coupled [82]	0.2	0	0	2	1	✗
Helsinki, Earth-only [82]	0.1	0	0	2	1	✗

The setups for the earthquake-tsunami benchmark and the Palu earthquake are available in the supplement [80] to [79]. An updated version of these setups that supports SeisSol version 1.0.1 can be found in [81]. The setup for the Helsinki EGS earthquake is available in [83], which is the supplement to [82]. Parameter files for the convergence studies presented in chapter 8 are available in the GitHub repository (`https://github.com/SeisSol/Examples/`) with commit hash `2359667d871bf2068f672ca25f357f97477e2e72`. The setup used for the method comparison in section 7.2 is available in the supplement [2] to [3].

We obtained all previously unpublished results presented in this thesis with SeisSol v1.0.1 [167]. We produced the Helsinki results of [82] with an earlier version of SeisSol, which includes the new LTS scheme. In [79], we computed the earthquake-tsunami benchmark and Palu results with a different implementation of dynamic rupture, gravitational boundary, and LTS. We always use double-precision floating point accuracy.

Bibliography

[1] Rachel E. Abercrombie. “Resolution and Uncertainties in Estimates of Earthquake Stress Drop and Energy Release”. In: *Philosophical Transactions of the Royal Society A* 379.2196 (2021), p. 20200131. DOI: 10.1098/rsta.2020.0131.

[2] Lauren Abrahams, Lukas Krenz, Eric Dunham, Alice-Agnes Gabriel, and Tatsuhiko Saito. *Comparison of Methods for Coupled Earthquake and Tsunami Modeling, Data*. Stanford Digital Repository, 2022. DOI: 10.25740/JV404DC0795.

[3] Lauren S. Abrahams, Lukas Krenz, Eric M. Dunham, Alice-Agnes Gabriel, and Tatsuhiko Saito. “Comparison of Methods for Coupled Earthquake and Tsunami Modelling”. In: *Geophysical Journal International* 234.1 (2023), pp. 404–426. DOI: 10.1093/gji/ggad053.

[4] Thomas Ader, Michael Chendorain, Matthew Free, Tero Saarno, Pekka Heikkinen, Peter Eric Malin, Peter Leary, Grzegorz Kwiatek, Georg Dresen, Felix Bluemle, and Tommi Vuorinen. “Design and Implementation of a Traffic Light System for Deep Geothermal Well Stimulation in Finland”. In: *Journal of Seismology* 24.5 (2019), pp. 991–1014. DOI: 10.1007/s10950-019-09853-y.

[5] Keiiti Aki and Paul G. Richards. *Quantitative Seismology*. 2. edition, corrected printing. University Science Books, 2009. ISBN: 978-1-891389-63-4.

[6] D. J. Andrews. “Rupture Velocity of Plane Strain Shear Cracks”. In: *Journal of Geophysical Research (1896-1977)* 81.32 (1976), pp. 5679–5687. DOI: 10.1029/JB081i032p05679.

[7] Paola F. Antonietti, Francesco Bonaldi, and Ilario Mazzieri. “A High-Order Discontinuous Galerkin Approach to the Elasto-Acoustic Problem”. In: *Computer Methods in Applied Mechanics and Engineering* 358 (2020), p. 112634. DOI: 10.1016/j.cma.2019.112634.

[8] Stephen J. Arrowsmith, Jeffrey B. Johnson, Douglas P. Drob, and Michael A. H. Hedlin. “The Seismoacoustic Wavefield: A New Paradigm in Studying Geophysical Phenomena”. In: *Reviews of Geophysics* 48.4 (2010). DOI: 10.1029/2010RG000335.

[9] Stefan Baisch, Christopher Koch, and Annemarie Muntendam-Bos. “Traffic Light Systems: To What Extent Can Induced Seismicity Be Controlled?” In: *Seismological Research Letters* 90.3 (2019), pp. 1145–1154. DOI: 10.1785/0220180337.

[10] Jordan W. Bishop, David Fee, Ryan Modrak, Carl Tape, and Keehoon Kim. “Spectral Element Modeling of Acoustic to Seismic Coupling Over Topography”. In: *Journal of Geophysical Research: Solid Earth* 127.1 (2022). DOI: 10.1029/2021JB023142.

[11] Julian J. Bommer, Stephen Oates, José Mauricio Cepeda, Conrad Lindholm, Juliet Bird, Rodolfo Torres, Griselda Marroquín, and José Rivas. "Control of Hazard Due to Seismicity Induced by a Hot Fractured Rock Geothermal Project". In: *Engineering Geology* 83.4 (2006), pp. 287–306. DOI: 10.1016/j.enggeo.2005.11.002.

[12] Alexander Breuer and Alexander Heinecke. "Next-Generation Local Time Stepping for the ADER-DG Finite Element Method". In: *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 2022, pp. 402–413. DOI: 10.1109/IPDPS53621.2022.00046.

[13] Alexander Breuer, Alexander Heinecke, and Michael Bader. "Petascale Local Time Stepping for the ADER-DG Finite Element Method". In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. 2016, pp. 854–863. DOI: 10.1109/IPDPS.2016.109.

[14] Alexander Breuer, Alexander Heinecke, Michael Bader, and Christian Pelties. "Accelerating SeisSol by Generating Vectorized Code for Sparse Matrix Operators". In: *Parallel Computing* (2014). DOI: 10.3233/978-1-61499-381-0-347.

[15] Alexander Breuer, Alexander Heinecke, Sebastian Rettenberger, Michael Bader, Alice-Agnes Gabriel, and Christian Pelties. "Sustained Petascale Performance of Seismic Simulations with SeisSol on SuperMUC". In: *Supercomputing*. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 1–18. DOI: 10.1007/978-3-319-07518-1_1.

[16] F. Brezzi, L. D. Marini, and E. Süli. "Discontinuous Galerkin methods for first-order hyperbolic problems". In: *Mathematical Models and Methods in Applied Sciences* 14.12 (2004), pp. 1893–1903. DOI: 10.1142/S0218202504003866.

[17] Edward M. Brooks, Seth Stein, Bruce D. Spencer, Leah Salditch, Mark D. Petersen, and Daniel E. McNamara. "Assessing Earthquake Hazard Map Performance for Natural and Induced Seismicity in the Central and Eastern United States". In: *Seismological Research Letters* 89.1 (2018), pp. 118–126. DOI: 10.1785/0220170124.

[18] James N. Brune. "Tectonic Stress and the Spectra of Seismic Shear Waves from Earthquakes". In: *Journal of Geophysical Research (1896-1977)* 75.26 (1970), pp. 4997–5009. DOI: 10.1029/JB075i026p04997.

[19] John Charles Butcher. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. Wiley-Interscience, 1987.

[20] Bernardo Cockburn, George E. Karniadakis, and Chi-Wang Shu. "The Development of Discontinuous Galerkin Methods". In: *Discontinuous Galerkin Methods*. Lecture Notes in Computational Science and Engineering. Springer, 2000, pp. 3–50. DOI: 10.1007/978-3-642-59721-3_1.

[21] Richard K. Cook. "Infrasound Radiated During the Montana Earthquake of 1959 August 18". In: *Geophysical Journal International* 26.1-4 (1971), pp. 191–198. DOI: 10.1111/j.1365-246X.1971.tb03393.x.

[22] R. Courant, K. Friedrichs, and H. Lewy. “Über die partiellen Differenzengleichungen der mathematischen Physik”. In: *Mathematische Annalen* 100.1 (1928), pp. 32–74. DOI: `10.1007/BF01448839`.

[23] L. Dagum and R. Menon. “OpenMP: An Industry Standard API for Shared-Memory Programming”. In: *IEEE Computational Science and Engineering* 5.1 (Jan.-March/1998), pp. 46–55. DOI: `10.1109/99.660313`.

[24] David Keyes. *Efficient Computation through Tuned Approximation.* SIAM Activity Group on Supercomputing. Nov. 2, 2022. URL: `https://siag-sc.org/media/files/DK-slides.pdf` (visited on 08/24/2023).

[25] Charles Davison. “Earthquake Sounds”. In: *Bulletin of the Seismological Society of America* 28.3 (1938), pp. 147–161. DOI: `10.1785/BSSA0280030147`.

[26] Steven M. Day, Luis A. Dalguer, Nadia Lapusta, and Yi Liu. “Comparison of Finite Difference and Boundary Integral Solutions to Three-Dimensional Spontaneous Rupture”. In: *Journal of Geophysical Research* 110.B12 (2005), B12307. DOI: `10.1029/2005JB003813`.

[27] J. De La Puente, J.-P. Ampuero, and M. Käser. “Dynamic Rupture Modeling on Unstructured Meshes Using a Discontinuous Galerkin Method”. In: *Journal of Geophysical Research* 114.B10 (2009), B10302. DOI: `10.1029/2008JB006271`.

[28] Josep De la Puente. “Seismic Wave Simulation for Complex Rheologies on Unstructured Meshes”. PhD thesis. LMU, 2008.

[29] Josep De La Puente, Michael Dumbser, Martin Käser, and Heiner Igel. “Discontinuous Galerkin Methods for Wave Propagation in Poroelastic Media”. In: *GEOPHYSICS* 73.5 (2008), T77–T97. DOI: `10.1190/1.2965027`.

[30] Josep de la Puente, Martin Käser, Michael Dumbser, and Heiner Igel. “An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes - IV. Anisotropy”. In: *Geophysical Journal International* 169.3 (2007), pp. 1210–1228. DOI: `10.1111/j.1365-246X.2007.03381.x`.

[31] Alexandre Denis, Julien Jaeger, and Hugo Taboada. “Progress Thread Placement for Overlapping MPI Non-blocking Collectives Using Simultaneous Multithreading”. In: *Euro-Par 2018: Parallel Processing Workshops.* Lecture Notes in Computer Science. Springer International Publishing, 2019, pp. 123–133. DOI: `10.1007/978-3-030-10549-5\_10`.

[32] Ravil Dorozhinskii and Michael Bader. “SeisSol on Distributed Multi-GPU Systems: CUDA Code Generation for the Modal Discontinuous Galerkin Method”. In: *The International Conference on High Performance Computing in Asia-Pacific Region.* ACM, 2021, pp. 69–82. DOI: `10.1145/3432261.3436753`.

[33] Michael Dumbser. “Arbitrary High Order PNPM Schemes on Unstructured Meshes for the Compressible Navier–Stokes Equations”. In: *Computers & Fluids* 39.1 (2010), pp. 60–76. DOI: `10.1016/j.compfluid.2009.07.003`.

[34] Michael Dumbser, Dinshaw S. Balsara, Eleuterio F. Toro, and Claus-Dieter Munz. “A Unified Framework for the Construction of One-Step Finite Volume and Discontinuous Galerkin Schemes on Unstructured Meshes”. In: *Journal of Computational Physics* 227.18 (2008), pp. 8209–8253. DOI: 10.1016/j.jcp.2008.05.025.

[35] Michael Dumbser and Martin Käser. “An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes - II. The Three-Dimensional Isotropic Case”. In: *Geophysical Journal International* 167.1 (2006), pp. 319–336. DOI: 10.1111/j.1365-246X.2006.03120.x.

[36] Michael Dumbser, Martin Käser, and Josep De La Puente. “Arbitrary High-Order Finite Volume Schemes for Seismic Wave Propagation on Unstructured Meshes in 2D and 3D”. In: *Geophysical Journal International* 171.2 (2007), pp. 665–694. DOI: 10.1111/j.1365-246X.2007.03421.x.

[37] Michael Dumbser, Martin Käser, and Eleuterio F. Toro. “An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes - V. Local Time Stepping and p-Adaptivity”. In: *Geophysical Journal International* 171.2 (2007), pp. 695–717. DOI: 10.1111/j.1365-246X.2007.03427.x.

[38] Kenneth Duru and Eric M. Dunham. “Dynamic Earthquake Rupture Simulations on Nonplanar Faults Embedded in 3D Geometrically Complex, Heterogeneous Elastic Solids”. In: *Journal of Computational Physics* 305 (2016), pp. 185–207. DOI: 10.1016/j.jcp.2015.10.021.

[39] Adam M. Dziewonski and Don L. Anderson. “Preliminary Reference Earth Model”. In: *Physics of the Earth and Planetary Interiors* 25.4 (1981), pp. 297–356. DOI: 10.1016/0031-9201(81)90046-7.

[40] John E. Ebel, Vladimir Vudler, and Michael Celata. “The 1981 Microearthquake Swarm near Moodus, Connecticut”. In: *Geophysical Research Letters* 9.4 (1982), pp. 397–400. DOI: 10.1029/GL009i004p00397.

[41] Lawrence C. Evans. *Partial Differential Equations*. 2nd ed. Graduate Studies in Mathematics v. 19. American Mathematical Society, 2010. ISBN: 978-0-8218-4974-3.

[42] L. G. Evers, D. Brown, K. D. Heaney, J. D. Assink, P. S. M. Smets, and M. Snellen. “Evanescent Wave Coupling in a Geophysical System: Airborne Acoustic Signals from the Mw 8.1 Macquarie Ridge Earthquake”. In: *Geophysical Research Letters* 41.5 (2014), pp. 1644–1650. DOI: 10.1002/2013GL058801.

[43] Paola F. Antonietti, Francesco Bonaldi, and Ilario Mazzieri. “Simulation of Three-dimensional Elastoacoustic Wave Propagation Based on a Discontinuous Galerkin Spectral Element Method”. In: *International Journal for Numerical Methods in Engineering* 121.10 (2020), pp. 2206–2226. DOI: 10.1002/nme.6305.

[44] Hugo Fastl and Eberhard Zwicker. *Psychoacoustics: Facts and Models*. Vol. 22. Springer Science & Business Media, 2006. DOI: 10.1007/978-3-540-68888-4.

[45] María R. Fernández-Ruiz, Marcelo A. Soto, Ethan F. Williams, Sonia Martin-Lopez, Zhongwen Zhan, Miguel Gonzalez-Herraez, and Hugo F. Martins. "Distributed Acoustic Sensing for Seismic Activity Monitoring". In: *APL Photonics* 5.3 (2020), p. 030901. DOI: 10.1063/1.5139602.

[46] Andreas Fichtner. *Full Seismic Waveform Modelling and Inversion.* Springer Science & Business Media, 2010. ISBN: 978-3-642-15807-0.

[47] Gregor Gassner, Michael Dumbser, Florian Hindenlang, and Claus-Dieter Munz. "Explicit One-Step Time Discretizations for Discontinuous Galerkin and Finite Volume Schemes Based on Local Predictors". In: *Journal of Computational Physics* 230.11 (2011), pp. 4232–4247. DOI: 10.1016/j.jcp.2010.10.024.

[48] Emmanuel Gaucher, Martin Schoenball, Oliver Heidbach, Arno Zang, Peter A. Fokker, Jan-Diederik van Wees, and Thomas Kohl. "Induced Seismicity in Geothermal Reservoirs: A Review of Forecasting Approaches". In: *Renewable and Sustainable Energy Reviews* 52 (2015), pp. 1473–1490. DOI: 10.1016/j.rser.2015.08.026.

[49] S. Glimsdal, G. K. Pedersen, C. B. Harbitz, and F. Løvholt. "Dispersion of Tsunamis: Does It Really Matter?" In: *Natural Hazards and Earth System Sciences* 13.6 (2013), pp. 1507–1526. DOI: 10.5194/nhess-13-1507-2013.

[50] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time Dependent Problems and Difference Methods.* Vol. 24. John Wiley & Sons, 1995.

[51] GW4 and the UK Met Office. *A64FX - Fujitsu — GW4-Isambard Documentation.* URL: https://gw4-isambard.github.io/docs/user-guide/A64FX.html (visited on 09/15/2023).

[52] Bilel Hadri, Samuel Kortas, Saber Feki, Rooh Khurram, and Greg Newby. "Overview of the KAUST's Cray X40 System–Shaheen II". In: *Proceedings of the 2015 Cray User Group* 3 (2015).

[53] David Harel. "Statecharts: A Visual Formalism for Complex Systems". In: *Science of Computer Programming* 8.3 (1987), pp. 231–274. DOI: 10.1016/0167-6423(87)90035-9.

[54] Markus O. Häring, Ulrich Schanz, Florentin Ladner, and Ben C. Dyer. "Characterisation of the Basel 1 Enhanced Geothermal System". In: *Geothermics* 37.5 (2008), pp. 469–495. DOI: 10.1016/j.geothermics.2008.06.002.

[55] Ami Harten, Bjorn Engquist, Stanley Osher, and Sukumar R Chakravarthy. "Uniformly High Order Accurate Essentially Non-Oscillatory Schemes, III". In: *Journal of Computational Physics* 71.2 (1987), pp. 231–303. DOI: 10.1016/0021-9991(87)90031-3.

[56] M. A. H. Hedlin, K. Walker, D. P. Drob, and C.D. de Groot-Hedlin. "Infrasound: Connecting the Solid Earth, Oceans, and Atmosphere". In: *Annual Review of Earth and Planetary Sciences* 40.327 (2012), p. 2012. DOI: 10.1146/annurev-earth-042711-105508.

[57] Alexander Heinecke, Alexander Breuer, Michael Bader, and Pradeep Dubey. “High Order Seismic Simulations on the Intel Xeon Phi Processor (Knights Landing)”. In: *High Performance Computing*. Lecture Notes in Computer Science. Springer International Publishing, 2016, pp. 343–362. DOI: 10.1007/978-3-319-41321-1_18.

[58] Alexander Heinecke, Alexander Breuer, Sebastian Rettenberger, Michael Bader, Alice-Agnes Gabriel, Christian Pelties, Arndt Bode, William Barth, Xiang-Ke Liao, Karthikeyan Vaidyanathan, Mikhail Smelyanskiy, and Pradeep Dubey. “Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers”. In: *SC ’14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2014, pp. 3–14. DOI: 10.1109/SC.2014.6.

[59] Alexander Heinecke, Greg Henry, Maxwell Hutchinson, and Hans Pabst. “LIBXSMM: Accelerating Small Matrix Multiplications by Runtime Code Generation”. In: *SC ’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2016, pp. 981–991. DOI: 10.1109/SC.2016.83.

[60] Bruno Hernandez, Alexis Le Pichon, Julien Vergoz, Pascal Herry, Lars Ceranna, Christoph Pilger, Emanuele Marchetti, Maurizio Ripepe, and Rémy Bossu. “Estimating the Ground-Motion Distribution of the 2016 Mw 6.2 Amatrice, Italy, Earthquake Using Remote Infrasound Observations”. In: *Seismological Research Letters* 89.6 (2018), pp. 2227–2236. DOI: 10.1785/0220180103.

[61] Jan S. Hesthaven and Tim Warburton. *Nodal Discontinuous Galerkin Methods*. Springer New York, 2008. DOI: 10.1007/978-0-387-72067-8.

[62] David P. Hill. “What Is That Mysterious Booming Sound?” In: *Seismological Research Letters* 82.5 (2011), pp. 619–622. DOI: 10.1785/gssrl.82.5.619.

[63] David P. Hill, Fredrick G. Fischer, Karen M. Lahr, and John M. Coakley. “Earthquake Sounds Generated by Body-Wave Ground Motion”. In: *Bulletin of the Seismological Society of America* 66.4 (1976), pp. 1159–1172. DOI: 10.1785/BSSA0660041159.

[64] Gregor Hillers, Tommi A. T. Vuorinen, Marja R. Uski, Jari T. Kortström, Päivi B. Mäntyniemi, Timo Tiira, Peter E. Malin, and Tero Saarno. “The 2018 Geothermal Reservoir Stimulation in Espoo/Helsinki, Southern Finland: Seismic Network Anatomy and Data Features”. In: *Seismological Research Letters* 91.2A (2020), pp. 770–786. DOI: 10.1785/0220190253.

[65] Torsten Hoefler and Andrew Lumsdaine. “Message Progression in Parallel Computing - to Thread or Not to Thread?” In: *2008 IEEE International Conference on Cluster Computing*. 2008, pp. 213–222. DOI: 10.1109/CLUSTR.2008.4663774.

[66] P. Hupe, L. Ceranna, A. Le Pichon, R. S. Matoza, and P. Mialle. "International Monitoring System Infrasound Data Products for Atmospheric Studies and Civilian Applications". In: *Earth System Science Data* 14.9 (2022), pp. 4201–4230. DOI: 10.5194/essd-14-4201-2022.

[67] Yoshihiro Ito, Takeshi Tsuji, Yukihito Osada, Motoyuki Kido, Daisuke Inazu, Yutaka Hayashi, Hiroaki Tsushima, Ryota Hino, and Hiromi Fujimoto. "Frontal Wedge Deformation near the Source Region of the 2011 Tohoku-Oki Earthquake". In: *Geophysical Research Letters* 38.7 (2011). DOI: 10.1029/2011GL048355.

[68] Kinjiro Kajiura. "The Leading Wave of a Tsunami". In: *Bulletin of the Earthquake Research Institute, University of Tokyo* 41.3 (1963), pp. 535–571.

[69] Martin Käser and Michael Dumbser. "A Highly Accurate Discontinuous Galerkin Method for Complex Interfaces between Solids and Moving Fluids". In: *GEOPHYSICS* 73.3 (2008), T23–T35. DOI: 10.1190/1.2870081.

[70] Martin Käser, Michael Dumbser, Josep de la Puente, and Heiner Igel. "An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes – III. Viscoelastic Attenuation". In: *Geophysical Journal International* 168.1 (2007), pp. 224–242. DOI: 10.1111/j.1365-246X.2006.03193.x.

[71] Martin Käser, Verena Hermann, and Josep de la Puente. "Quantitative Accuracy Analysis of the Discontinuous Galerkin Method for Seismic Wave Propagation". In: *Geophysical Journal International* 173.3 (2008), pp. 990–999. DOI: 10.1111/j.1365-246X.2008.03781.x.

[72] Martin Käser, P. Martin Mai, and Michael Dumbser. "Accurate Calculation of Fault-Rupture Models Using the High-Order Discontinuous Galerkin Method on Tetrahedral Meshes". In: *Bulletin of the Seismological Society of America* 97.5 (2007), pp. 1570–1586. DOI: 10.1785/0120060253.

[73] Alexander A Kaufman and Anatoli L Levshin. *Acoustic and Elastic Wave Fields in Geophysics: III.* Vol. 32. Elsevier, 2005.

[74] Katsuyoshi Kawaguchi, Yoshiyuki Kaneda, and Eiichirou Araki. "The DONET: A Real-Time Seafloor Research Infrastructure for the Precise Earthquake and Tsunami Monitoring". In: *OCEANS 2008 - MTS/IEEE Kobe Techno-Ocean.* 2008, pp. 1–4. DOI: 10.1109/OCEANSKOBE.2008.4530918.

[75] Sabrina Keil, Joachim Wassermann, and Tobias Megies. "Estimation of Ground Motion Due to Induced Seismicity at a Geothermal Power Plant near Munich, Germany, Using Numerical Simulations". In: *Geothermics* 106 (2022), p. 102577. DOI: 10.1016/j.geothermics.2022.102577.

[76] Dimitri Komatitsch and Jean-Pierre Vilotte. "The Spectral Element Method: An Efficient Tool to Simulate the Seismic Response of 2D and 3D Geological Structures". In: *Bulletin of the Seismological Society of America* 88.2 (1998), pp. 368–392. DOI: 10.1785/BSSA0880020368.

[77] David A. Kopriva, Gregor J. Gassner, and Jan Nordström. "Stability of Discontinuous Galerkin Spectral Element Schemes for Wave Propagation When the Coefficient Matrices Have Jumps". In: *Journal of Scientific Computing* 88.1 (2021), p. 3. DOI: 10.1007/s10915-021-01516-w.

[78] Jeremy E. Kozdon and Eric M. Dunham. "Constraining Shallow Slip and Tsunami Excitation in Megathrust Ruptures Using Seismic and Ocean Acoustic Waves Recorded on Ocean-Bottom Sensor Networks". In: *Earth and Planetary Science Letters* 396 (2014), pp. 56–65. DOI: 10.1016/j.epsl.2014.04.001.

[79] Lukas Krenz, Carsten Uphoff, Thomas Ulrich, Alice-Agnes Gabriel, Lauren S. Abrahams, Eric M. Dunham, and Michael Bader. "3D Acoustic-Elastic Coupling with Gravity: The Dynamics of the 2018 Palu, Sulawesi Earthquake and Tsunami". In: *SC' 21: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* 2021, pp. 1–14. DOI: 10.1145/3458817.3476173.

[80] Lukas Krenz, Carsten Uphoff, Thomas Ulrich, Alice-Agnes Gabriel, Lauren S. Abrahams, Eric M. Dunham, and Michael Bader. *Supplementary Material for 3D Acoustic-Elastic Coupling with Gravity: The Dynamics of the 2018 Palu, Sulawesi Earthquake and Tsunami.* Zenodo, Apr. 9, 2021. DOI: 10.5281/ZENODO.5159333.

[81] Lukas Krenz, Carsten Uphoff, Thomas Ulrich, Alice-Agnes Gabriel, Lauren S. Abrahams, Eric M. Dunham, and Michael Bader. *Supplementary Material for 3D Acoustic-Elastic Coupling with Gravity: The Dynamics of the 2018 Palu, Sulawesi Earthquake and Tsunami (Updated for SeisSol v1.0.1).* Zenodo, Apr. 9, 2021. DOI: 10.5281/ZENODO.8341965.

[82] Lukas Krenz, Sebastian Wolf, Gregor Hillers, Alice-Agnes Gabriel, and Michael Bader. "Numerical Simulations of Seismoacoustic Nuisance Patterns from an Induced M 1.8 Earthquake in the Helsinki, Southern Finland, Metropolitan Area". In: *Bulletin of the Seismological Society of America* (2023). DOI: 10.1785/0120220225.

[83] Lukas Krenz, Sebastian Wolf, Gregor Hillers, Alice-Agnes Gabriel, and Michael Bader. *Supplementary Material: Numerical Simulations of Seismoacoustic Nuisance Patterns from an Induced M 1.8 Earthquake in the Helsinki, Southern Finland, Metropolitan Area.* Zenodo, June 19, 2023. DOI: 10.5281/ZENODO.8056416.

[84] Martin Kronbichler. "The Discontinuous Galerkin Method: Derivation and Properties". In: *Efficient High-Order Discretizations for Computational Fluid Dynamics.* CISM International Centre for Mechanical Sciences. Springer International Publishing, 2021, pp. 1–55. DOI: 10.1007/978-3-030-60610-7_1.

[85] Martin Kronbichler and Katharina Kormann. "Fast Matrix-Free Evaluation of Discontinuous Galerkin Finite Element Operators". In: *ACM Transactions on Mathematical Software* 45.3 (2019), 29:1–29:40. DOI: 10.1145/3325864.

[86] Pijush K. Kundu, Ira M. Cohen, David R. Dowling, and Grétar Tryggvason. *Fluid Mechanics.* Sixth edition. Elsevier/AP, 2016. ISBN: 978-0-12-405935-1.

[87] Fabian Kutschera, Alice-Agnes Gabriel, Sara Aniko Wirp, Bo Li, Thomas Ulrich, Claudia Abril, and Benedikt Halldórsson. "Linked and Fully-Coupled 3D Earthquake Dynamic Rupture and Tsunami Modeling for the Húsavík-Flatey Fault Zone in North Iceland". In: *EGUsphere [preprint]* (2023), pp. 1–43. DOI: 10.5194/egusphere-2023-1262.

[88] Grzegorz Kwiatek, Patricia Martinez Garzon, Jörn Davidsen, Peter Malin, Aino Karjalainen, Marco Bohnhoff, and Georg Dresen. "Limited Earthquake Interaction during a Geothermal Hydraulic Stimulation in Helsinki, Finland". In: *Journal of Geophysical Research: Solid Earth* 127.9 (2022). DOI: 10.1029/2022JB024354.

[89] Grzegorz Kwiatek, Tero Saarno, Thomas Ader, Felix Bluemle, Marco Bohnhoff, Michael Chendorain, Georg Dresen, Pekka Heikkinen, Ilmo Kukkonen, Peter Leary, et al. "Controlling Fluid-Induced Seismicity During a 6.1 km-Deep Geothermal Stimulation in Finland". In: *Science Advances* 5.5 (2019).

[90] Grzegorz Kwiatek, Tero Saarno, Thomas Ader, Felix Bluemle, Marco Bohnhoff, Michael Chendorain, Georg Dresen, Pekka Heikkinen, Ilmo Kukkonen, Peter Leary, Maria Leonhardt, Peter Malin, Patricia Martínez-Garzón, Kevin Passmore, Paul Passmore, Sergio Valenzuela, and Christopher Wollin. "Controlling Fluid-Induced Seismicity during a 6.1-Km-Deep Geothermal Stimulation in Finland". In: *Science Advances* 5.5 (2019). DOI: 10.1126/sciadv.aav7224.

[91] Oliver D. Lamb, Jonathan M. Lees, Peter E. Malin, and Tero Saarno. "Audible Acoustics from Low-Magnitude Fluid-Induced Earthquakes in Finland". In: *Scientific reports* 11.1 (2021), pp. 1–8. DOI: 10.1038/s41598-021-98701-6.

[92] Lev D. Landau and Evgenij M. Lifšic. *Theory of Elasticity.* 3. English edition. 7. Elsevier, Butterworth-Heinemann, 2009. ISBN: 978-0-7506-2633-0.

[93] Leibniz-Rechenzentrum (LRZ). *Hardware of SuperMUC-NG.* URL: https://doku.lrz.de/hardware-of-supermuc-ng-11482553.html (visited on 09/15/2023).

[94] Maria Leonhardt, Grzegorz Kwiatek, Patricia Martínez-Garzón, Marco Bohnhoff, Tero Saarno, Pekka Heikkinen, and Georg Dresen. "Seismicity During and After Stimulation of a 6.1km Deep Enhanced Geothermal System in Helsinki, Finland". In: *Solid Earth Sciences Library* 12.3 (2021), pp. 581–594. DOI: 10.5194/se-12-581-2021.

[95] Randall J. LeVeque. *Finite Volume Methods for Hyperbolic Problems.* Cambridge University Press, 2002. DOI: 10.1017/cbo9780511791253.

[96] Jiaxuan Li, Taeho Kim, Nadia Lapusta, Ettore Biondi, and Zhongwen Zhan. "The Break of Earthquake Asperities Imaged by Distributed Acoustic Sensing". In: *Nature* (2023), pp. 1–7. DOI: 10.1038/s41586-023-06227-w.

[97] Gabriel C. Lotto and Eric M. Dunham. "High-order Finite Difference Modeling of Tsunami Generation in a Compressible Ocean from Offshore Earthquakes". In: *Computational Geosciences* 19.2 (2015), pp. 327–340.

[98] Gabriel C. Lotto, Tamara N. Jeppson, and Eric M. Dunham. "Fully Coupled Simulations of Megathrust Earthquakes and Tsunamis in the Japan Trench, Nankai Trough, and Cascadia Subduction Zone". In: *Pure and Applied Geophysics* 176.9 (2019), pp. 4009–4041. DOI: `10.1007/s00024-018-1990-y`.

[99] Gabriel C. Lotto, Gabriel Nava, and Eric M. Dunham. "Should Tsunami Simulations Include a Nonzero Initial Horizontal Velocity?" In: *Earth, Planets and Space* 69.1 (2017), p. 117. DOI: `10.1186/s40623-017-0701-8`.

[100] B. Lund, Marianne Maria Malm, Päivi Birgitta Mäntyniemi, Kati Johanna Oinonen, T. Tiira, Marja Riitta Uski, and Tommi Antton Tapani Vuorinen. *Evaluating Seismic Hazard for the Hanhikivi Nuclear Power Plant Site, Seismological Characteristics of the Source Areas, Attenuation of Seismic Signal, and Probabilistic Analysis of Seismic Hazard.* Ed. by J. Saari. Vol. Report NE-4459. ÅF-Consult Ltd, 2015.

[101] Raul Madariaga. "Earthquake Scaling Laws". In: *Extreme Environmental Events.* Springer New York, 2011, pp. 364–383. DOI: `10.1007/978-1-4419-7695-6_22`.

[102] E. H. Madden, M. Bader, J. Behrens, Y. van Dinther, A.-A. Gabriel, L. Rannabauer, T. Ulrich, C. Uphoff, S. Vater, and I. van Zelst. "Linked 3-D Modelling of Megathrust Earthquake-Tsunami Events: From Subduction to Tsunami Run Up". In: *Geophysical Journal International* 224.1 (2021), pp. 487–516. DOI: `10.1093/gji/ggaa484`.

[103] Takuto Maeda and Takashi Furumura. "FDM Simulation of Seismic Waves, Ocean Acoustic Waves, and Tsunamis Based on Tsunami-Coupled Equations of Motion". In: *Pure and Applied Geophysics* 170.1 (2013), pp. 109–127. DOI: `10.1007/s00024-011-0430-z`.

[104] P. Martin Mai. "Supershear Tsunami Disaster". In: *Nature Geoscience* 12.3 (2019), pp. 150–151. DOI: `10.1038/s41561-019-0308-8`.

[105] Ernie Majer, James Nelson, Ann Robertson-Tait, Jean Savy, and Ivan Wong. *Protocol for Addressing Induced Seismicity Associated with Enhanced Geothermal Systems.* Tech. rep. DOE/EE–0662, 1219482. 2012, DOE/EE–0662, 1219482. DOI: `10.2172/1219482`.

[106] Päivi Birgitta Mäntyniemi. "Revisiting Svenskby, Southeastern Finland: Communications Regarding Low-Magnitude Earthquakes in 1751–1752". In: *Geosciences* 12.9 (2022). DOI: `10.3390/geosciences12090338`.

[107] Satoshi Matsuoka, Jens Domke, Mohamed Wahib, Aleksandr Drozd, and Torsten Hoefler. "Myths and Legends in High-Performance Computing". In: *The International Journal of High Performance Computing Applications* 37.3-4 (2023), pp. 245–259. DOI: `10.1177/10943420231166608`.

[108] John D. McCalpin. "HPL and DGEMM Performance Variability on the Xeon Platinum 8160 Processor". In: *SC' 18: International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 2018. DOI: `10.1109/SC.2018.00021`.

[109] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard Version 4.0.* Manual. 2021.

[110] Henrik Møller and Christian Sejer Pedersen. "Hearing at Low and Infrasonic Frequencies". In: *Noise & Health* 6.23 (2004), pp. 37–57.

[111] Gordon E. Moore. "The Experts Look Ahead: Cramming More Components onto Integrated Circuits". In: *Electronics* 38.8 (1965), pp. 114–119.

[112] J. Paul Mutschlecner and Rodney W. Whitaker. "Infrasound from Earthquakes". In: *Journal of Geophysical Research: Atmospheres* 110.D1 (2005), pp. 1–11. DOI: 10.1029/2004JD005067.

[113] Jan Nordström. "A Roadmap to Well Posed and Stable Problems in Computational Physics". In: *Journal of Scientific Computing* 71.1 (2017), pp. 365–385. DOI: 10.1007/s10915-016-0303-9.

[114] Yoshimitsu Okada. "Surface Deformation Due to Shear and Tensile Faults in a Half-Space". In: *Bulletin of the Seismological Society of America* 75.4 (1985), pp. 1135–1154.

[115] Emile A. Okal. "The Generation of T Waves by Earthquakes". In: *Advances in Geophysics.* Vol. 49. Elsevier, 2008, pp. 1–65. DOI: 10.1016/S0065-2687(07)49001-X.

[116] Tom Parker, Sergey Shatalin, and Mahmoud Farhadiroushan. "Distributed Acoustic Sensing – a New Tool for Seismic Applications". In: *First Break* 32.2 (2014). DOI: 10.3997/1365-2397.2013034.

[117] Ryan Paulik, Aditya Gusman, James H. Williams, Gumbert Maylda Pratama, Sheng-lin Lin, Alamsyah Prawirabhakti, Ketut Sulendra, Muhammad Yasser Zachari, Zabin Ellyni Dwi Fortuna, Novita Barrang Pare Layuk, and Ni Wayan Ika Suwarni. "Tsunami Hazard and Built Environment Damage Observations from Palu City after the September 28 2018 Sulawesi Earthquake and Tsunami". In: *Pure and Applied Geophysics* 176.8 (2019), pp. 3305–3321. DOI: 10.1007/s00024-019-02254-9.

[118] C. Pelties, A.-A. Gabriel, and J.-P. Ampuero. "Verification of an ADER-DG Method for Complex Dynamic Rupture Problems". In: *Geoscientific Model Development* 7.3 (2014), pp. 847–866. DOI: 10.5194/gmd-7-847-2014.

[119] Christian Pelties, Josep De La Puente, Jean-Paul Ampuero, Gilbert B. Brietzke, and Martin Käser. "Three-dimensional Dynamic Rupture Simulation with a High-Order Discontinuous Galerkin Method on Unstructured Tetrahedral Meshes". In: *Journal of Geophysical Research: Solid Earth* 117.B2 (2012). DOI: 10.1029/2011JB008857.

[120] Kaare B. Petersen and Michael S. Pedersen. *The Matrix Cookbook.* Technical University of Denmark, 2012.

[121] Arben Pitarka, Aybige Akinci, Pasquale De Gori, and Mauro Buttinelli. “Deterministic 3D Ground-Motion Simulations (0–5 Hz) and Surface Topography Effects of the 30 October 2016 Mw6.5 Norcia, Italy, Earthquake”. In: *Bulletin of the Seismological Society of America* 112.1 (2022), pp. 262–286. DOI: 10.1785/0120210133.

[122] Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics.* Vol. 37. Texts in Applied Mathematics. Springer New York, 2007. DOI: 10.1007/b98885.

[123] Marlon D. Ramos, Prithvi Thakur, Yihe Huang, Ruth A. Harris, and Kenny J. Ryan. “Working with Dynamic Earthquake Rupture Models: A Practical Guide”. In: *Seismological Research Letters* 93.4 (2022), pp. 2096–2110. DOI: 10.1785/0220220022.

[124] William H. Reed and Thomas R. Hill. *Triangular Mesh Methods for the Neutron Transport Equation.* Tech. rep. Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[125] Anne Reinarz, Dominic E. Charrier, Michael Bader, Luke Bovard, Michael Dumbser, Kenneth Duru, Francesco Fambri, Alice-Agnes Gabriel, Jean-Matthieu Gallard, Sven Köppel, Lukas Krenz, Leonhard Rannabauer, Luciano Rezzolla, Philipp Samfass, Maurizio Tavelli, and Tobias Weinzierl. “ExaHyPE: An Engine for Parallel Dynamically Adaptive Simulations of Wave Problems”. In: *Computer Physics Communications* 254 (2020), p. 107251. DOI: 10.1016/j.cpc.2020.107251.

[126] Sebastian Rettenberger and Michael Bader. “Optimizing I/O for Petascale Seismic Simulations on Unstructured Meshes”. In: *2015 IEEE International Conference on Cluster Computing.* IEEE, 2015, pp. 314–317. DOI: 10.1109/CLUSTER.2015.51.

[127] A. Rintamäki, G. Hillers, S. Heimann, T. Dahm, and A. Korja. “Centroid Full Moment Tensor Analysis Reveals Fluid Channels Opened by Induced Seismicity at EGS, Helsinki Region, Southern Finland”. In: *EGU General Assembly Conference Abstracts.* 2023, EGU23–12756. DOI: 10.5194/egusphere-egu23-12756.

[128] Annukka E. Rintamäki, Gregor Hillers, Tommi A. T. Vuorinen, Tuija Luhta, Jonathan M. Pownall, Christina Tsarsitalidou, Keith Galvin, Jukka Keskinen, Jari T. Kortström, Tzu-Chi Lin, Päivi B. Mäntyniemi, Kati J. Oinonen, Tahvo J. Oksanen, Pirita J. Seipäjärvi, George Taylor, Marja R. Uski, Ahti I. Voutilainen, and David M. Whipp. “A Seismic Network to Monitor the 2020 EGS Stimulation in the Espoo/Helsinki Area, Southern Finland”. In: *Seismological Research Letters* 93.2A (2021), pp. 1046–1062. DOI: 10.1785/0220210195.

[129] Barbara Romanowicz, Richard Allen, Knute Brekke, Li-Wei Chen, Yuancong Gou, Ivan Henson, Julien Marty, Doug Neuhauser, Brian Pardini, Taka’aki Taira, Stephen Thompson, Junli Zhang, and Stephane Zuzlewski. “SeaFOAM: A Year-Long DAS Deployment in Monterey Bay, California”. In: *Seismological Research Letters* (2023). DOI: 10.1785/0220230047.

[130] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.6).* https://www.sagemath.org. 2022.

[131] Tatsuhiko Saito. *Tsunami Generation and Propagation.* Springer Geophysics. Springer Japan, 2019. DOI: 10.1007/978-4-431-56850-6.

[132] Tatsuhiko Saito, Toshitaka Baba, Daisuke Inazu, Shunsuke Takemura, and Eiichi Fukuyama. "Synthesizing Sea Surface Height Change Including Seismic Waves and Tsunami Using a Dynamic Rupture Scenario of Anticipated Nankai Trough Earthquakes". In: *Tectonophysics* 769 (2019), p. 228166. DOI: 10.1016/j.tecto.2019.228166.

[133] Tatsuhiko Saito and Takashi Furumura. "Three-Dimensional Tsunami Generation Simulation Due to Sea-Bottom Deformation and Its Interpretation Based on the Linear Theory". In: *Geophysical Journal International* 178.2 (2009), pp. 877–888. DOI: 10.1111/j.1365-246X.2009.04206.x.

[134] Tatsuhiko Saito and Hiroaki Tsushima. "Synthesizing Ocean Bottom Pressure Records Including Seismic Wave and Tsunami Contributions: Toward Realistic Tests of Monitoring Systems". In: *Journal of Geophysical Research: Solid Earth* 121.11 (2016), pp. 8175–8195. DOI: 10.1002/2016JB013195.

[135] Kirk Schloegel, George Karypis, and Vipin Kumar. "Parallel Static and Dynamic Multi-Constraint Graph Partitioning". In: *Concurrency and Computation: Practice and Experience* 14.3 (2002), pp. 219–240. DOI: 10.1002/cpe.605.

[136] Ryan Schultz, Gregory C. Beroza, and William L. Ellsworth. "A Risk-Based Approach for Managing Hydraulic Fracturing–Induced Seismicity". In: *Science* 372.6541 (2021), pp. 504–507. DOI: 10.1126/science.abg5451.

[137] Ryan Schultz, Vince Quitoriano, David J. Wald, and Gregory C. Beroza. "Quantifying Nuisance Ground Motion Thresholds for Induced Earthquakes". In: *Earthquake Spectra* 37.2 (2021), pp. 789–802. DOI: 10.1177/8755293020988025.

[138] CSC – IT Center for Science Ltd. *Mahti - Docs CSC.* URL: https://docs.csc.fi/computing/systems-mahti/ (visited on 09/15/2023).

[139] Skipper Seabold and Josef Perktold. "Statsmodels: Econometric and Statistical Modeling with Python". In: *9th Python in Science Conference.* 2010. DOI: 10.25080/Majora-92bf1922-011.

[140] Shahar Shani-Kadmiel, Gil Averbuch, Pieter Smets, Jelle Assink, and Läslo Evers. "The 2010 Haiti Earthquake Revisited: An Acoustic Intensity Map from Remote Atmospheric Infrasound Observations". In: *Earth and Planetary Science Letters* 560 (2021), p. 116795. DOI: 10.1016/j.epsl.2021.116795.

[141] Peter M. Shearer. *Introduction to Seismology.* Third. Cambridge University Press, 2019. DOI: 10.1017/9781316877111.

[142] Michael Sipser. *Introduction to the Theory of Computation.* 2nd ed. Thomson Course Technology, 2006. ISBN: 978-0-534-95097-2.

[143] A. Sladen, D. Rivet, J. P Ampuero, L. De Barros, Y. Hello, G. Calbris, and P. Lamare. “Distributed Sensing of Earthquakes and Ocean-Solid Earth Interactions on Seafloor Telecom Cables”. In: *Nature Communications* 10.1 (2019), p. 5777. DOI: `10.1038/s41467-019-13793-z`.

[144] J. S. Sochacki, J. H. George, R. E. Ewing, and Scott B. Smithson. “Interface Conditions for Acoustic and Elastic Wave Propagation”. In: *Geophysics* 56.2 (1991), pp. 168–181. DOI: `10.1190/1.1443029`.

[145] Y. Tony Song, L. -L. Fu, Victor Zlotnicki, Chen Ji, Vala Hjorleifsdottir, C. K. Shum, and Yuchan Yi. “The Role of Horizontal Impulses of the Faulting Continental Slope in Generating the 26 December 2004 Tsunami”. In: *Ocean Modelling* 20.4 (2008), pp. 362–379. DOI: `10.1016/j.ocemod.2007.10.007`.

[146] Y. Tony Song and Shin-Chan Han. “Satellite Observations Defying the Long-Held Tsunami Genesis Theory”. In: *Remote Sensing of the Changing Oceans*. Springer Berlin Heidelberg, 2011, pp. 327–342. DOI: `10.1007/978-3-642-16541-2_17`.

[147] Y. Tony Song, Ali Mohtat, and Solomon C. Yim. “New Insights on Tsunami Genesis and Energy Source”. In: *Journal of Geophysical Research: Oceans* 122.5 (2017), pp. 4238–4256. DOI: `10.1002/2016JC012556`.

[148] Michael Stauffacher, Nora Muggli, Anna Scolobig, and Corinne Moser. “Framing Deep Geothermal Energy in Mass Media: The Case of Switzerland”. In: *Technological Forecasting and Social Change* 98 (2015), pp. 60–70. DOI: `10.1016/j.techfore.2015.05.018`.

[149] Walter A. Strauss. *Partial Differential Equations: An Introduction.* 2nd ed. Wiley, 2007. ISBN: 978-0-470-05456-7.

[150] Matthieu Sylvander and Dorin G. Mogos. “The Sounds of Small Earthquakes: Quantitative Results from a Study of Regional Macroseismic Bulletins”. In: *Bulletin of the Seismological Society of America* 95.4 (2005), pp. 1510–1515. DOI: `10.1785/0120040197`.

[151] Matthieu Sylvander, Christian Ponsolles, Sébastien Benahmed, and Jean-François Fels. “Seismoacoustic Recordings of Small Earthquakes in the Pyrenees: Experimental Results”. In: *Bulletin of the Seismological Society of America* 97.1B (2007), pp. 294–304. DOI: `10.1785/0120060009`.

[152] Yuichiro Tanioka and Kenji Satake. “Tsunami Generation by Horizontal Displacement of Ocean Bottom”. In: *Geophysical Research Letters* 23.8 (1996), pp. 861–864. DOI: `10.1029/96GL00736`.

[153] Taufiqurrahman Taufiqurrahman, Alice-Agnes Gabriel, Thomas Ulrich, Lubica Valentova, and Frantisek Gallovič. “Broadband Dynamic Rupture Modeling with Fractal Fault Roughness, Frictional Heterogeneity, Viscoelasticity and Topography: The 2016 Mw 6.2 Amatrice, Italy Earthquake”. In: *Geophysical Research Letters* 49.22 (2022). DOI: `10.1029/2022GL098872`.

[154] Texas Advanced Computing Center (TACC). *Frontera - TACC HPC Documentation*. URL: https://docs.tacc.utexas.edu/hpc/frontera/#system (visited on 09/15/2023).

[155] François Thouvenot, Liliane Jenatton, and Jean-Pierre Gratier. “200-m-Deep Earthquake Swarm in Tricastin (Lower Rhône Valley, France) Accounts for Noisy Seismicity over Past Centuries”. In: *Terra Nova* 21.3 (2009), pp. 203–210. DOI: 10.1111/j.1365-3121.2009.00875.x.

[156] V. A. Titarev and E. F. Toro. “ADER: Arbitrary High Order Godunov Approach”. In: *Journal of Scientific Computing* 17.1 (2002), pp. 609–618. DOI: 10.1023/A:1015126814947.

[157] V.A. Titarev and E.F. Toro. “ADER Schemes for Three-Dimensional Non-Linear Hyperbolic Systems”. In: *Journal of Computational Physics* 204.2 (2005), pp. 715–736. DOI: 10.1016/j.jcp.2004.10.028.

[158] J. Tomic, R. E. Abercrombie, and A. F. Do Nascimento. “Source Parameters and Rupture Velocity of Small M$\leq$2.1 Reservoir Induced Earthquakes”. In: *Geophysical Journal International* 179.2 (2009), pp. 1013–1023. DOI: 10.1111/j.1365-246X.2009.04233.x.

[159] E.F. Toro and V.A. Titarev. “Derivative Riemann Solvers for Systems of Conservation Laws and ADER Methods”. In: *Journal of Computational Physics* 212.1 (2006), pp. 150–165. DOI: 10.1016/j.jcp.2005.06.018.

[160] Eleuterio F. Toro. *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Berlin Heidelberg, 2009. DOI: 10.1007/b79761.

[161] Patrizia Tosi, Valerio De Rubeis, Andrea Tertulliani, and Calvino Gasparini. “Spatial Patterns of Earthquake Sounds and Seismic Source Geometry”. In: *Geophysical Research Letters* 27.17 (2000), pp. 2749–2752. DOI: 10.1029/2000GL011377.

[162] Jan Treibig, Georg Hager, and Gerhard Wellein. “LIKWID: A Lightweight Performance-Oriented Tool Suite for X86 Multicore Environments”. In: *2010 39th International Conference on Parallel Processing Workshops*. IEEE, 2010, pp. 207–216. DOI: 10.1109/ICPPW.2010.38.

[163] T. Ulrich, S. Vater, E. H. Madden, J. Behrens, Y. van Dinther, I. van Zelst, E. J. Fielding, C. Liang, and A.-A. Gabriel. “Coupled, Physics-Based Modeling Reveals Earthquake Displacements Are Critical to the 2018 Palu, Sulawesi Tsunami”. In: *Pure and Applied Geophysics* 176.10 (2019), pp. 4069–4109. DOI: 10.1007/s00024-019-02290-5.

[164] Carsten Uphoff. “Flexible Model Extension and Optimisation for Earthquake Simulations at Extreme Scales”. PhD thesis. Technische Universität München, 2020.

[165] Carsten Uphoff and Michael Bader. “Generating High Performance Matrix Kernels for Earthquake Simulations with Viscoelastic Attenuation”. In: *2016 International Conference on High Performance Computing & Simulation (HPCS)*. 2016, pp. 908–916. DOI: 10.1109/HPCSim.2016.7568431.

[166] Carsten Uphoff and Michael Bader. “Yet Another Tensor Toolbox for Discontinuous Galerkin Methods and Other Applications”. In: *ACM Transactions on Mathematical Software* 46.4 (2020), pp. 1–40. DOI: `10.1145/3406835`.

[167] Carsten Uphoff, Lukas Krenz, Thomas Ulrich, Sebastian Wolf, Adrian Knoll, Duo Li, Alexander Heinecke, Ravil Dorozhinskii, Stephanie Wollherr, Marius Bohn, Nico Schliwa, Gilbert Brietzke, Taufiq Taufiqurrahman, Sebastian Anger, Sebastian Rettenberger, Frédéric Simonis, Alice Gabriel, Viktoria Pauw, Alexander Breuer, Fabian Kutschera, Kadek Hendrawan Palgunadi, Leonhard Rannabauer, Lukas van de Wiel, Bo Li, Calum Chamberlain, Jeena Yun, John Rekoske, Yonatan G, and Michael Bader. *SeisSol*. 2023. DOI: `10.5281/ZENODO.7837068`.

[168] Carsten Uphoff, Sebastian Rettenberger, Michael Bader, Elizabeth H Madden, Thomas Ulrich, Stephanie Wollherr, and Alice-Agnes Gabriel. “Extreme Scale Multi-Physics Simulations of the Tsunamigenic 2004 Sumatra Megathrust Earthquake”. In: *SC' 17: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM. 2017, p. 21. DOI: `10.1145/3126908.3126948`.

[169] Bram van Leer. “Upwind and High-Resolution Methods for Compressible Flow: From Donor Cell to Residual-Distribution Schemes”. In: *16th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, 2003. DOI: `10.2514/6.2003-3559`.

[170] James P. Verdon and Julian J. Bommer. “Green, Yellow, Red, or out of the Blue? An Assessment of Traffic Light Schemes to Mitigate the Impact of Hydraulic Fracturing-Induced Seismicity”. In: *Journal of Seismology* 25.1 (2021), pp. 301–326. DOI: `10.1007/s10950-020-09966-9`.

[171] F. Vernon, J. Tytell, M. A. H. Hedlin, K. Walker, R. Busby, and R. Woodward. “Integration of Infrasound, Atmospheric Pressure, and Seismic Observations with the NSF EarthScope USArray Transportable Array”. In: *EGU General Assembly Conference Abstracts*. 2012, p. 10770.

[172] Peter Wauligmann, Nathan Brei, Alexander Puscas, and Jonas Schreier. *PSpaMM*. URL: `https://github.com/SeisSol/PSpaMM` (visited on 08/01/2023).

[173] Roger Waxler and Jelle Assink. “Propagation Modeling through Realistic Atmosphere and Benchmarking”. In: *Infrasound Monitoring for Atmospheric Studies: Challenges in Middle Atmosphere Dynamics and Societal Benefits*. Springer International Publishing, 2019, pp. 509–549. DOI: `10.1007/978-3-319-75140-5_15`.

[174] Roger Waxler, Claus H. Hetzer, Jelle D. Assink, and Philip Blom. “A Two-Dimensional Effective Sound Speed Parabolic Equation Model for Infrasound Propagation with Ground Topography”. In: *The Journal of the Acoustical Society of America* 152.6 (2022), pp. 3659–3669. DOI: `10.1121/10.0016558`.

[175] Lucas C. Wilcox, Georg Stadler, Carsten Burstedde, and Omar Ghattas. “A High-Order Discontinuous Galerkin Method for Wave Propagation through Coupled Elastic–Acoustic Media”. In: *Journal of Computational Physics* 229.24 (2010), pp. 9373–9396. DOI: 10.1016/j.jcp.2010.09.008.

[176] Nadine G. Reitman William J. Stephenson and Stephen J. Angster. *U.S. Geological Survey Open-File Report 2017–1152: P- and S-wave Velocity Models Incorporating the Cascadia Subduction Zone for 3D Earthquake Ground Motion Simulations, Version 1.6—Update for Open-File Report 2007–1348*. Tech. rep. U.S. Geological Survey, 2017. DOI: 10.3133/ofr20171152.

[177] Andrew Wilson and Shuo Ma. “Wedge Plasticity and Fully Coupled Simulations of Dynamic Rupture and Tsunami in the Cascadia Subduction Zone”. In: *Journal of Geophysical Research: Solid Earth* 126.7 (2021). DOI: 10.1029/2020JB021627.

[178] Sara Aniko Wirp, Alice-Agnes Gabriel, Maximilian Schmeller, Elizabeth H. Madden, Iris van Zelst, Lukas Krenz, Ylona van Dinther, and Leonhard Rannabauer. “3D Linked Subduction, Dynamic Rupture, Tsunami, and Inundation Modeling: Dynamic Effects of Supershear and Tsunami Earthquakes, Hypocenter Location, and Shallow Fault Slip”. In: *Frontiers in Earth Science* 9 (2021). DOI: 10.3389/feart.2021.626844.

[179] Sara Aniko Wirp, Thomas Ulrich, Lukas Krenz, Michael Bader, Stefano Lorito, and Alice-Agnes Gabriel. “Earthquake Scenarios for the Hellenic Arc from 3D Dynamic Rupture Modeling: Implications for Tsunami Hazard”. In: *EGU General Assembly Conference Abstracts*. 2022. DOI: 10.5194/egusphere-egu22-9486.

[180] Sebastian Wolf, Alice-Agnes Gabriel, and Michael Bader. “Optimization and Local Time Stepping of an ADER-DG Scheme for Fully Anisotropic Wave Propagation in Complex Geometries”. In: *Computational Science – ICCS 2020*. Vol. 12139. Springer International Publishing, 2020, pp. 32–45. DOI: 10.1007/978-3-030-50420-5_3.

[181] Sebastian Wolf, Martin Galis, Carsten Uphoff, Alice-Agnes Gabriel, Peter Moczo, David Gregor, and Michael Bader. “An Efficient ADER-DG Local Time Stepping Scheme for 3D HPC Simulation of Seismic Waves in Poroelastic Media”. In: *Journal of Computational Physics* 455 (2022). DOI: 10.1016/j.jcp.2021.110886.

[182] Stephanie Wollherr, Alice-Agnes Gabriel, and Carsten Uphoff. “Off-Fault Plasticity in Three-Dimensional Dynamic Rupture Simulations Using a Modal Discontinuous Galerkin Method on Unstructured Meshes: Implementation, Verification and Application”. In: *Geophysical Journal International* 214.3 (2018), pp. 1556–1584. DOI: 10.1093/gji/ggy213.

[183] Brian J. N. Wylie. “Exascale Potholes for HPC: Execution Performance and Variability Analysis of the Flagship Application Code HemeLB”. In: *2020 IEEE/ACM International Workshop on HPC User Support Tools (HUST) and Workshop on Programming and Performance Visualization Tools (ProTools)*. IEEE, 2020. DOI: 10.1109/hustprotools51951.2020.00014.

[184] N. Yamamoto, K. Hirata, S. Aoi, W. Suzuki, H. Nakamura, and T. Kunugi. "Rapid Estimation of Tsunami Source Centroid Location Using a Dense Offshore Observation Network". In: *Geophysical Research Letters* 43.9 (2016), pp. 4263–4269. DOI: 10.1002/2016GL068169.

[185] Zhongwen Zhan, Mattia Cantono, Valey Kamalov, Antonio Mecozzi, Rafael Müller, Shuang Yin, and Jorge C. Castellanos. "Optical Polarization–Based Seismic and Water Wave Sensing on Transoceanic Cables". In: *Science* 371.6532 (2021), pp. 931–936. DOI: 10.1126/science.abe6648.