Technische Universität München
Fakultät für Informatik

Earthquake and Tsunami Simulation with high-order ADER-DG methods

Leonhard Andreas Rannabauer

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende(r):	Prof. Dr. Francisco Javier Esparza Estaun
Prüfer der Dissertation:	1. Prof. Dr. Michael Georg Bader
	2. Prof. Dr. Michael Dumbser

Die Dissertation wurde am 19.01.2022 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 16.05.2022 angenommen.

I do not know what I may appear to the world, but to myself I seem to have been only like a boy playing on the seashore, and diverting myself in now and then finding a smoother pebble or a prettier shell than ordinary, whilst the great ocean of truth lay all undiscovered before me.

Isaac Newton

Acknowledgments

I want to thank Michael Bader for giving me the opportunity to do this exciting journey, for his support, guidance and for throwing me in cold water at the right moments.

My gratitude goes to Kenneth Duru, for the long hours we worked together on our joint project, for his patience and mentoring.

I thank the whole ExaHyPE crew Alice, Anne, Ben, Dominic, Duo, Jean-Matthieu, Maurizo, Michael, Philipp and Tobias, for making the project not only a success but also enjoyable.

My friend Carsten for the four years in our office that we spent with creative discussions and joint cursing. I owe you a lot.

My friend Lukas for his support, ideas and book recommendations, who especially made the last year more enjoyable.

Thanks go to my school mates, without which I probably would have finished this thesis one year earlier, but with way less fun.

My parents for their unconditional support and food supply.

And finally and most important Mirjam, for going with me through the ups and downs of the past five years. Without you this whole work would not haven been possible.

What I enjoyed most about my time in academia was the collaboration with motivated and inspiring people, who understand the importance of good scientific work and dedicate their careers to that cause.

- Thanks to all of you.

Abstract

Automated, efficient codes are the key to enable novel approaches in earthquake and tsunami science as urgent computing, probabilistic seismic and tsunami analysis or uncertainty quantification.

In this work I introduce numerical solvers based on the ADER discontinuous Galerkin (DG) method to model earthquakes and tsunamis. The characterizing feature of the ADER-DG method is its element local time-integration, that easily allows to reach high convergence orders in time. High arithmetic intensity of its main kernels, which lead to high performance on current supercomputing architectures and a reduced communication overhead are additional advantages of the method.

I present the realization of an ADER-DG method and a Runge-Kutta DG method of second order for the shallow water equations in the dynamic adaptive mesh framework sam(oa)^2. I render the a posterior finite volume limiter of the ADER-DG scheme to be well-balanced and positivity preserving. In a time-to-solution comparison for hardware optimized implementations of both schemes, I break down their performance in several factors based on the meshing infrastructure, on hardware optimizations and on characteristics of the simulated problem.

In order to enable the sourcing of tsunamis with the computed displacements from seismic simulations, I introduce a pipeline for linked simulations. The pipeline includes a novel Fourier filter to erase fast traveling seismic waves from the earthquake output that cannot be represented properly in the tsunami model.

I introduce the ExaSeis framework for dynamic rupture earthquake simulations with the ExaHyPE-Engine on dynamically refined curvilinear meshes. As the established artificial boundary conditions produce spurious reflections, ExaSeis provides an implementation of the first discontinuous Galerkin method with perfectly matched layers. Finally, I introduce a new automated meshing approach, based on the transfinite interpolation to represent faults and topographies with curvilinear meshes.

Contents

CHAPTER

1

Introduction

The origin of computer simulations lies in the second world war and its aftermath: The first Turing-complete, digital computer ENIAC was used by the allied forces for the optimization of ballistic curves and computer simulations on the MANIAC system played a fundamental role in the Manhattan project for the development of the atom and the H-Bomb [52]. Since the second world war, computer hardware and numerical methods have considerably evolved and have become a decisive element in the assessment of natural disasters and the prevention of destruction. Large scale simulations forecast the weather, model climate change, and assess the hazard of tsunamis and earthquakes [14, 31, 78, 87].

Compared to the early simulations of the US-military, the available computing power has exponentially increased, closely following the predictions of Moore's law. While the ENIAC system could perform approximately 500 floating point operations per second in 1946 [62], the upcoming high performance computing (HPC) system El Capitan reaches a theoretical 2.0×10^{18} floating point operations per second [125]. In this early stage of the exascale area, forward models can be simulated in very high detail and with complex physical models [106, 132].

In this work, we consider the simulation of linked earthquake-tsunami events. For example, the devastating Indian Ocean tsunami was seismically triggered by the Sumatra-Andaman earthquake that occurred on December 26th 2004. The tsunami caused more than 283,000 deaths in densely populated, poor areas at the coasts of Indonesia, Thailand, Sri Lanka and

11 other countries [85]. Alongside the enormous size of the event, the high number of casualties was caused by unpreparedness and the lack of a working early-warning system, which did not exist as earthquakes and tsunamis in this magnitude were not expected in the Indian ocean [114].

The need for large-scale simulations originates in the highly varying spatial scales that earthquake and tsunami forward simulations have to incorporate: In case of earthquake simulations, fracture mechanics on faults have to be resolved in meters [79]. Additionally, a high spatial resolution is required to represent the high frequency content of emerging seismic waves. In contrast, the study region usually includes several countries and is in the range of thousands of kilometers. For the simulation of tsunamis the inundation process at a coast has to be resolved as detailed as possible, while the modeled tsunami can travel hundreds of kilometers before reaching it. To handle the resulting large number of unknowns large-scale, high performance codes are required.

While earthquake and tsunami prediction using computers is still out of reach [16], exascale computing facilitates plausible hazard assessment due to earthquake and tsunami events: For example, in *urgent computing* a dedicated infrastructure of monitoring facilities and HPC resources is build to automatically reproduce earthquakes or tsunamis, based on measured initial conditions, within hours after an event happened [102].

The available computing power allows to look at events in a wider angle: In *Probabilistic Seismic Hazard Analysis* (PSHA) and *Probabilistic Tsunami Hazard Analysis* (PTHA), the goal is to estimate the probability[1] that an area is destructed by earthquakes or tsunamis [16, 61, 65, 116].

Closely related to PTHA and PSHA is the field of *uncertainty quantification* (UQ): As the exact parameters of models are usually based on assumptions, they come with a certain degree of uncertainty. In UQ we produce a probability distribution of event outcomes, based on the uncertainty of model parameters [115].

In PTHA, PSHA and UQ the continuous space of possible events or parameters has to be discretized, which leads to the necessity to run up to millions of simulations, as for example required for Bayesian and Monte Carlo methods [58, 115].

In this work we address several requirements that urgent computing, PTHA, PTSA, and UQ share: First, the numerical codes must provide an accurate solution as quickly as possible, which means that the numerical scheme and its implementation have to optimize time-to-solution, while

[1] ie. the probability distribution to a so called intensity measure

including all available data. In case of earthquake simulations, this includes that the scheme is able to represent the topography and fault structures accurately, as the structure of a fault has a direct influence on the rupture mechanics of the earthquake and the topography influences seismic waves on the surface with wave-scattering and amplification effects [10, 56]. For tsunami simulations, inundation at the coast has to be resolved properly and tsunamis have to be initialized with accurate sources [15, 141]. Second, the set-up of initial conditions and meshes for single simulations should be done with minimal effort and preferably be automated. In case of urgent computing this aspect is required by design, in UQ, PTHA and PSHA the set-up of millions of single simulations quickly multiplies to a considerable effort.

The earthquake and tsunami model are both based on hyperbolic partial differential equations: We use the linear elastic wave equation with spontaneous dynamic rupture for the simulation of earthquakes [3, 86]. Tsunamis are modeled with the non-linear shallow water equations [86, 137].

To solve both hyperbolic partial differential equations (PDEs), our choice falls on the high order discontinuous Galerkin (DG) methods. Their element local high order approximation leads to high convergence orders and a high arithmetic intensity of the main kernels, compared to low order finite volume (FV) methods [18, 48, 131]. Compared to finite-difference methods, it is easy to include topography [39]. As only the adjacent neighbors are considered in the update of an element, the method is well suitable to be parallelized with a domain decomposition approach.

We focus on a subset of the discontinuous Galerkin method, the arbitrary high order discontinuous Galerkin (ADER-DG) method [33, 55, 126]. The unique characteristic of ADER-DG is the integration in time, which is solved element-locally in a single predictor step. The convergence order of the ADER-DG method is determined by a one-dimensional interpolation rule in time, such that we can easily reach higher orders in space and time by increasing the element-local polynomial representation. However, in case of non-linear PDEs the element-local solution of the scheme is prone to instabilities caused by emerging shocks. In order to address this problem an *a posteriori* sub-cell finite volume limiter is provided for the scheme [36, 38].

For applications in computational seismology, linear high order ADER-DG improves time-to-solution [20]. As we cannot translate this result to ADER-DG for the non-linear SWE, we perform an extensive analysis in time-to-solution as part of this work. In order to see how well the method compares to established approaches, we provide a second order Runge-Kutta

DG method as reference [135, 136]. In case of the ADER-DG method we resolve inundation with the *a posteriori* sub-cell finite volume limiter, the Runge-Kutta method a priori applies a Barth Jespersen typed limiter. To show the whole spectrum of time-to-solution, we decompose it into factors attributed to the underlying meshing-infrastructure, factors of the numerical schemes and factors of their hardware optimization.

While wetting and drying is a solved problem for finite volume methods [57, 127], only tailored DG methods based on RK time-stepping exist [59, 117, 118, 142]. The original version of the finite volume limiter of the ADER-DG scheme is not positivity preserving nor well-balanced, such that we have to render to fulfill these constraints.

To numerically solve the elastic wave equation, we introduce the linear ADER-DG method on curvilinear meshes. With curvilinear meshes, we can represent structures and topography accurately with splines of high order polynomials [43]. As the common absorbing boundary conditions cause reflections and disturb the solution, we introduce a novel perfectly matched layers approach [42]. Finally we automate the generation of meshes with an inductive domain splitting approach.

We implement the non-linear ADER-DG method, for the simulation of tsunamis with the SWE, in the dynamic adaptive mesh framework sam(oa)^2 [95, 96, 103]. In order to simulate earthquakes, we implement the linear ADER-DG method in the ExaHyPE-Engine to solve for the linear elastic wave equation in its first order formulation [104, 126].

sam(oa)^2 as well as the ExaHyPE-Engine are frameworks for the solution of hyperbolic partial differential equations in two (sam(oa)^2 and ExaHyPE-Engine) and three dimensions (ExaHyPE-Engine only) [95, 96, 104]. The frameworks come with finished parallelization and load-balancing strategies for shared and distributed memory, based on OpenMP (sam(oa)^2), TBB (ExaHyPE-Engine) and MPI (both). In order to highly resolve significant features of a solution, as tsunami fronts or rupture mechanics on a fault, both frameworks enable the simulation on dynamically refined meshes.

In order to generate realistic sources for tsunami simulations, we introduce a linking pipeline. The pipeline transforms the displacements generated by an earthquake simulation, into a compatible source for the subsequent tsunami. As the earthquake simulation contains waves, we can not represent in the tsunami simulation, we develop a novel Fourier filtering approach.

This thesis is organized in three main parts: Chapter 2 to Chapter 4 consider the simulation of tsunamis with high order DG methods in the sam(oa)^2 framework. The linking of earthquake and tsunami methods is

the content of Chapter 5 to Chapter 6. ADER-DG methods to simulate earthquakes with the ExaHyPE-Engine and the introduction of ExaSeis are presented in Chapter 7 and Chapter 8.

In Chapter 2, we give a brief introduction to the shallow water equations and the challenges for their numerical solution. We then introduce the sam(oa)^2-framework and a Runge-Kutta DG method of second order with Barth-Jespersen typed limiting and an ADER-DG methods with our version of *a posteriori* finite volume limiting. In Chapter 3 we verify both methods against a series of established benchmarks and reproduce the tsunamis subsequent to the Sumatra-Andaman earthquake. A comparison in time-to-solution for both methods follows in Chapter 4.

The linking pipeline to transform results from earthquake simulations into tsunami sources is presented Chapter 5. Our novel filtering approach to remove fast seismic waves is introduced in Section 5.3. The ASCETE Framework in Chapter 6 applies the developed techniques in a series of benchmarks.

We introduce an ADER-DG method for the simulation of dynamic rupture events, using the ExaHyPE-Engine as back-end, in Chapter 7. Here we introduce perfectly matched layers in order to avoid reflections form boundaries and present an automated meshing approach. The method is finally verified in a series of community benchmarks in Chapter 8.

CHAPTER

2

High order discontinuous Galerkin methods for the simulation of tsunamis

2.1 Introduction

In this first chapter, we want to look into two discontinuous Galerkin schemes for the shallow water equations to simulate tsunamis, that we realized in sam(oa)^2. Both methods have successfully been applied for the simulation of tsunami events [4, 92, 103].

First we want to put our schemes in the right context and highlight their main characteristics. For that reason, we enumerate some of the most established codes and approaches for the SWE: One of the most notable codes to simulate tsunamis is the *geoclaw* package. The SWE are discretized by a second order finite volume REA (Reconstruct-Evolve-Average) scheme on Cartesian meshes [86]. Wetting and drying are treated by advanced Riemann solvers, e.g. with the augmented Riemann solver by George [57]. In order to avoid oscillations, slopes in the reconstruction step are limited with TVD (Total Variation Diminishing) limiters [29]. To track wave fronts and inundated regions with high resolution, the mesh can be dynamically refined with a block structured approach.

Similar to *geoclaw* is the *Tsunami-HySEA* code. *Tsunami-HySEA* employs a second order finite volume Polynomial Viscosity Matrix (PVM) scheme [48] to solve the two-layer SWE [48]. Here, well-balancedness and positivity are achieved by a special Roe linearization. In addition to the

simulation of tsunamis generated by earthquakes, *Tsunami-HySEA* can model landslide-tsunamis. Domains are discretized with statically structured Cartesian meshes, where refinement is acquired by nesting grids. This approach does not allow dynamically refined meshes and regions that need high resolution have to be estimated a priori. However its regularity suites current GPU architectures [7].

Both codes are based on low-order finite volume schemes, which usually show a very high robustness against oscillations and allow the resolution wet and dry fronts easily. However, their low-order approximation leads to low arithmetic intensity of the main kernels which usually is doomed to remain memory bound on current CPU architectures. In contrast, we know that high order discontinuous Galerkin methods can reach the compute bound regime, from several examples in other fields [20, 131].

High order DG methods for the SWE are usually a variation of Runge-Kutta discontinuous Galerkin (RK-DG) methods [59, 117, 118, 142]. While the high order representation allows us to reach high arithmetic intensity, it comes with several disadvantages: The intra-element high order polynomial representation is sensitive to oscillations caused by Gibb's phenomenon. Modeling wetting and drying cells becomes more complicated, as inner element nodes have to be treated separately. For convergence orders above four, the number of RK stages grows super-linear, which is known as Butcher barrier [23, 24]. To encounter the problems arising for inundation and oscillations, RK-DG schemes are combined with some type of slope limiting [142]. As these limiters work with the solution after a RK stage, the whole mesh has to be iterated a second time.

In this work we introduce two discontinuous Galerkin methods that follow new, alternative approaches to limit solutions. The first method is a second order RK-DG method that resolves oscillations with a newly developed Barth Jespersen typed limiter [135, 136]. The second is an arbitrary high oder discontinuous Galerkin (ADER-DG) method with *a posteriori* finite volume limiting [33, 36, 103].

In this chapter, we first introduce the sam(oa)^2 framework in Section 2.2, which is the base for both methods. After we introduced the shallow water equations in Section 2.3, we derive the numerical schemes in Section 2.4, and look at the different time integration methods in Section 2.4.2. The Barth-Jespersen typed limiter used in the RK-DG method is introduced in Section 2.5.2, our modifications for the *a posteriori* limiter to suite the SWE in Section 2.6.2.

2.2 The sam(oa)^2 framework

sam(oa)^2 is a framework designed for the simulation of hyperbolic partial differential equations on dynamically refined triangular meshes in two dimensions, created by Oliver Meister and Kaveh Rahnema [95, 96]. The basic geometrical element in sam(oa)^2 is a right triangle, which is refined by newest vertex bisection.

sam(oa)^2 acts as an all-in-one solution to several problems: It manages dynamic adaptive mesh refinement on the cell level, where it ensures that no hanging nodes are generated and the mesh remains conforming. The framework acts as memory manager. Cells, edges and nodes of the mesh are stored and loaded by a stack and stream approach. Dynamic load balancing over MPI ranks and OpenMP threads is accomplished on a subset of cells, called a section. Load weights are determined by a chains on chains approach which recently has been extended to a work stealing approach by Samfass et al. [112].

Compared to frameworks such as p4est [22], which provides no memory management, or Dune [12, 13], where several grid implementations are allowed, sam(oa)^2 unifies these concepts for a single application set. For the price of less flexibility, all concepts are tailored particularly for the underlying triangular mesh. A similar approach for Cartesian meshes can be found in the Peano framework [138].

To separate the underlying concepts from an actual numerical algorithm, a hook layer for data types and operations on elements is provided.

2.2.1 Space filling curve and stack and stream approach

To recapture the properties and advantages of the Sierpinski space filling curve (SSFC) and the underlying memory management, we look at an example adaptive refined mesh in Figure 2.1 (a). Starting at two triangles forming a square, each of the cells in the mesh is the result of several newest vertex bisections of coarser triangles. The color of the cells indicates the number of refinements that have been performed, between two (white cells) and five times (black cells). We can represent the mesh as a binary refinement graph, if we interpret every bisected coarse triangle as a node and the two resulting refined triangles as its children (b). In the resulting graph, triangles that are present in the mesh appear as leaves, inner nodes implicitly represent bisected triangles and the two roots correspond to the two initial level triangles.

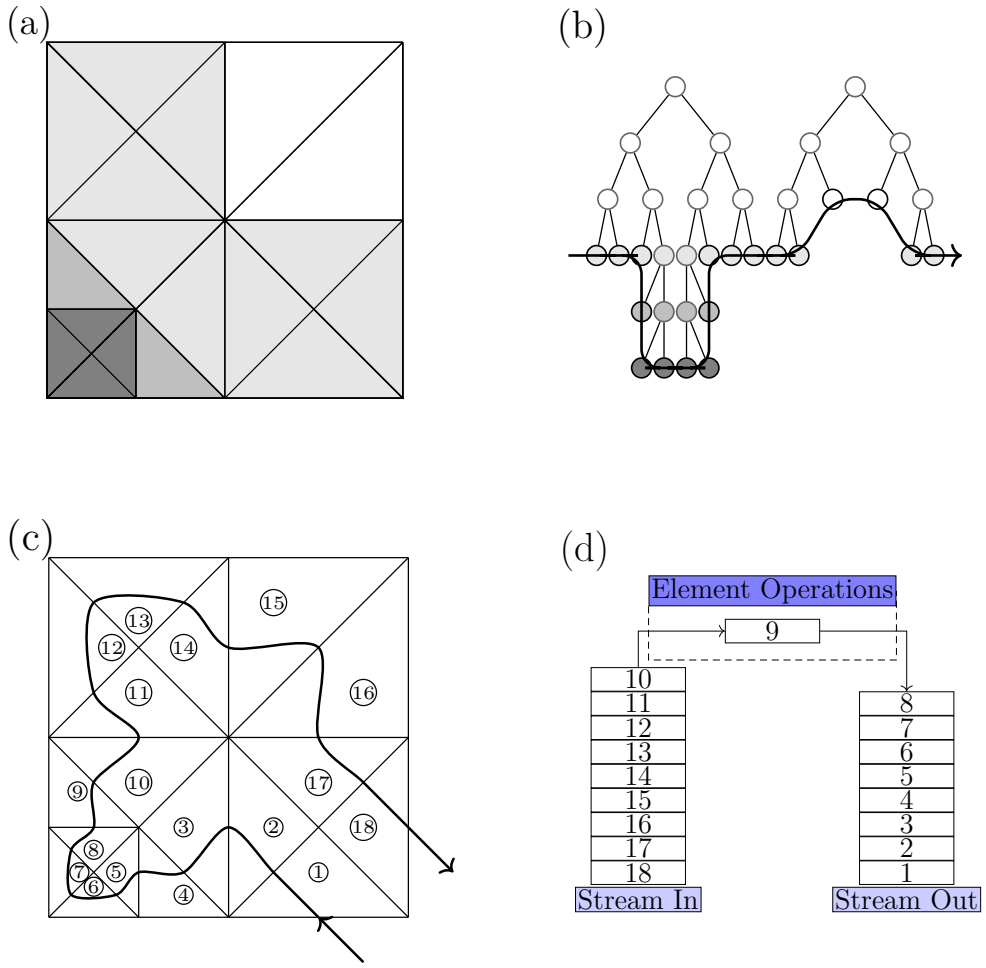


Figure 2.1.: (a) An example triangular mesh in sam(oa)^2. The grayscale indicates the refinement depth of the cell. (b) Corresponding binary tree. Inner nodes correspond to a bisected triangle. Depth first search results in the Sierpinski space filling curve in (c). The ordered cells are stored in streams, from which they are processed and stored in output streams (d).

If we apply depth first search on the graph, we impose a one dimensional ordering on its leaves, which is equivalent to mapping all triangles of the mesh on the SSFC (c). The biggest advantage of this method is that we can find such an SSFC for every arbitrary mesh that has been constructed by newest vertex bisection.

Having a one dimensional ordering allows us to store all triangles in a

cell stream (d), which can be realized as an array. Besides metadata for its geometry, each triangle holds information that we have to store persistently for the numerical scheme we want to realize, like degrees of freedom or material parameters.

To process the data stored in each triangle, we traverse through the cell stream. Each triangle we take from the cell stream is loaded from memory, updated with an element operator and stored back on an output stream. The output stream, storing triangles in reverse order, is then used as input stream of the next traversal. The cell stream approach avoids direct, possibly non-contiguous, memory accesses between elements.

In order to update cells, we require data from neighboring elements, as for example in the computation of fluctuations. For this reason sam(oa)^2 provides an interface to project data from triangles on adjacent edges and to merge them to a cell update afterwards. Illustrated in Figure 2.2 is the SSFC of the previous example with a focus on the order of edge accesses. In case edges are crossed by the SSFC (black line), both adjacent triangles are loaded contiguously, which allows us to directly perform projections onto the edge and compute an element update.

All edges that are not crossed by the SSFC are either located on the right (red) or left (green) side of the curve. For both types sam(oa)^2 introduces two additional input streams. On all colored edges we persistently keep updated projected solutions of both adjacent triangles. On edges of the same color, the SSFC imposes a stack-like access pattern: An edge is loaded the first time, when the first of its adjacent triangles is traversed (Figure 2.2 a). The edge holds the projected solution of both adjacent triangles, stored in the previous traversal, from which we compute an update for both triangles (Figure 2.2 b). We can directly apply the update to the first adjacent cell and store it for the second on the edge. Finally, we project the updated solution of the first cell onto the edge. After all operations have been performed on the first adjacent triangle the edge is pushed on an edge-stack. The stack-property of the SSFC ensures that it is the top element of the stack when its second adjacent triangle is traversed (Figure 2.2 c). With the traversal of the second triangle, we pop the edge from the stack and apply the stored element update (Figure 2.2 d). After that, we project the updated solution of the second cell onto the edge, such that it now stores projections of both updated solutions. Finally, the edge is pushed on an edge output-stream, which is used as the input-stream of the next traversal.

This update scheme matches the element and edge operators for finite volume and discontinuous Galerkin methods and allow us to realise the discontinuous Galerkin methods in chapter Section 2.5 and Section 2.6.

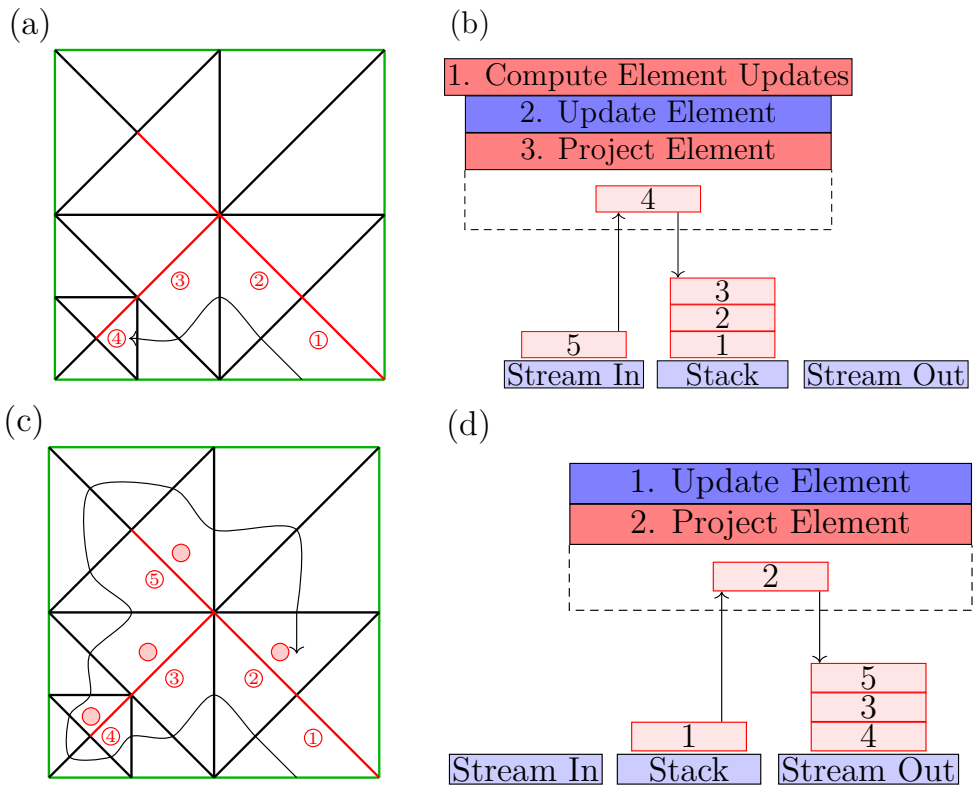


Figure 2.2.: Edge traversal and stack in sam(oa)^2. (a) When the first triangle adjacent to a colored edge is loaded, the edge is loaded from an input stream. (b) We compute an element update for both adjacent triangles, with the cell projections form the last iteration. The triangle is updated and its projection on the edge updated. We push the edge onto a stack. (c) When the second adjacent triangle is loaded the edge is now on top of the stack. (d) We can pull it and update the second adjacent triangle with the element update we computed in (b). Finally, we project the second triangle onto the edge and store it on an output stream.

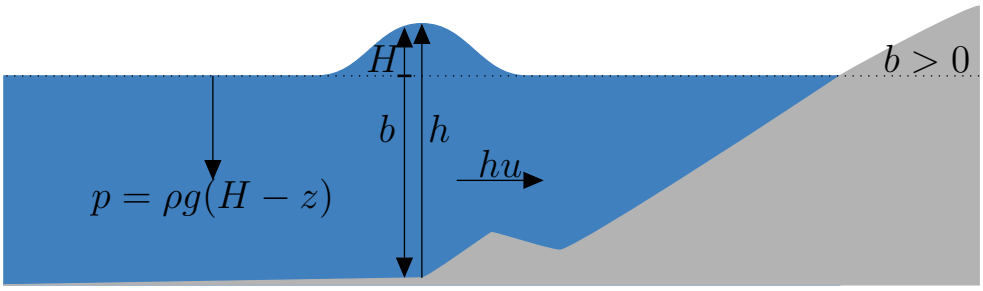


Figure 2.3.: Illustration of the shallow water equations and its physical quantities.

2.3 The shallow water equations

In order to model tsunamis, we introduce the shallow water equations. An overview of their most important characteristics can be found in [86] and a detailed derivation from the incompressible Navier Stokes equations (NSE) are presented in [137]. In this chapter we shortly summarize the key ingredients of this approach.

Sea-water is modeled as incompressible Newtonian fluid for which we assume that the density ρ is constant.[1] We look at the continuity equation and the conservation of momentum, with velocity components u, v, w in x, y and z, and the pressure p [137]:

$$u_x + v_y + w_z = 0 \tag{2.1}$$
$$\rho u_t + \rho(u^2)_x + \rho(uv)_y + \rho(uw)_z + p_x = 0 \tag{2.2}$$
$$\rho v_t + \rho(uv)_x + \rho(v^2)_y + \rho(vw)_z + p_y = 0 \tag{2.3}$$
$$\rho w_t + \rho(uw)_x + \rho(vw)_y + \rho w_z^2 + p_z = g\rho. \tag{2.4}$$

Here we neglect all acting external forces, such as friction or wind, and all body forces except the gravitational acceleration g in vertical direction.

The fundamental idea of the shallow water equations, illustrated in Figure 2.3, is that in case of tsunamis vertical scales are significantly smaller compared to horizontal scales. With this constraint, we can neglect all terms depending on the vertical velocity w in Equation (2.2) to Equation (2.4) and average all other components in depth.

The vertical boundaries of the sea-water are defined by the sea floor,

[1] This implies that we assume that salinity and temperature of the sea-water are constant.

also called bathymetry $b(x, y, t)$ and the vertical water height or water level $H(x, y, t)$. The height of the water column h is defined as their difference,

$$h(x, y, t) := H(x, y, t) - b(x, y, t). \tag{2.5}$$

Within the water column, we can compute the averaged horizontal velocities $\overline{u}(x, y, t), \overline{v}(x, y, t)$,

$$\overline{u} := \frac{1}{h} \int_b^H u(x, y, z, t) \, \mathrm{d}z \,, \qquad \overline{v} := \frac{1}{h} \int_b^H v(x, y, z, t) \, \mathrm{d}z \,. \tag{2.6}$$

No particles can cross the vertical boundaries of the fluid, such that we get kinematic boundary conditions at the surface and the sea floor

$$ub_x + vb_y = -b_t + w, \qquad \text{at } z = b(x, y, t) \text{ and} \tag{2.7}$$

$$uH_x + vH_y = H_t + w, \qquad \text{at } z = H(x, y, t). \tag{2.8}$$

If we neglect all vertical velocities in the vertical momentum conservation Equation (2.4), we can describe the pressure distribution in the vertical direction z with

$$p_z = \rho g. \tag{2.9}$$

Here the change of pressure in z only depends on the gravitational acceleration and the density ρ. Knowing that the atmospheric pressure at the surface is approximately zero, we can compute the pressure at depth z_0

$$p(x, y, z_0, t) = \int_{z_0}^{H(x,z,t)} p_z \, \mathrm{d}z = \rho g(H(x, y, t) - z_0) \tag{2.10}$$

and find the derivatives of the pressure in the horizontal directions x and y,

$$p_x(x, y, z_0, t) = \rho g H_x(x, y, t), \qquad p_y(x, y, z_0, t) = \rho g H_y(x, y, t). \tag{2.11}$$

By integrating the derivatives in depth we end up with:

$$\overline{p_x}(x, y, t) = \int_b^H p_x(x, y, z_0, t) \, \mathrm{d}z_0 = \rho g H_x(x, y, t)(H(x, y, t) - b(x, y, t)) = \rho g \left(\frac{1}{2} h(x, y, t)^2 \right)_x - \rho g h(x, y, t) b(x, y, t)_x, \tag{2.12}$$

which we perform similarly in y.

In order to derive the shallow water equations, we replace the velocity

components in Equation (2.2) and Equation (2.3) with their averages and integrate all three terms in depth. For the horizontal conservation of momentum (Equation (2.2), Equation (2.3)) we get:

$$\rho(hu)_t + \rho(hu^2)_x + \rho(huv)_y + \overline{p}_x = 0 \tag{2.13}$$
$$\rho(hv)_t + \rho(huv)_x + \rho(hv^2)_y + \overline{p}_y = 0. \tag{2.14}$$

The pressure terms are replaced with Equation (2.12) and we cancel out the density. Finally we integrate the continuity equation in depth,

$$\int_{b(x,y,t)}^{H(x,y,t)} u_x + v_y + w_z \,\mathrm{d}z\,. \tag{2.15}$$

Using Leibniz rule we can rewrite the integrals of the horizontal velocities as

$$\int_{b(x,y,t)}^{H(x,y,t)} u_x \,\mathrm{d}z = u|_H H_x - u|_b b_x + \frac{\partial}{\partial x}\int_{b(x,y,t)}^{H(x,y,t)} u \,\mathrm{d}z \tag{2.16}$$
$$= u|_H H_x - u|_b b_x + (h\overline{u})_x, \tag{2.17}$$

$$\int_{b(x,y,t)}^{H(x,y,t)} v_y \,\mathrm{d}z = v|_H H_y - v|_b b_y + \frac{\partial}{\partial y}\int_{b(x,y,t)}^{H(x,y,t)} v \,\mathrm{d}z \tag{2.18}$$
$$= v|_H H_y - v|_b b_y + (h\overline{u})_y. \tag{2.19}$$

We take Equation (2.16) and Equation (2.18) to get

$$\int_{b(x,y,t)}^{H(x,y,t)} u_x + v_y + w_z \,\mathrm{d}z =$$
$$u|_H H_x - u|_b b_x + (h\overline{u})_x + v|_H H_y - v|_b b_y + (h\overline{u})_y + w|_H - w|_b =$$
$$H_t - b_t + (h\overline{u})_x + (h\overline{u})_y = 0, \tag{2.20}$$

where we used the kinematic conditions Equation (2.7) and Equation (2.8) in the last step.

The depth averaged continuity equation Equation (2.20), and the depth averaged conservation of momentum Equation (2.13) and Equation (2.14),

define the shallow water equations, as illustrated in Figure 2.3:

$$h_t = (hu)_x + (hv)_y \tag{2.21}$$

$$hu_t = \left(\frac{1}{2}gh^2 + hu^2\right)_x + (huv)_y - gb_x h \tag{2.22}$$

$$hv_t = (huv)_x + \left(\frac{1}{2}gh^2 + hv^2\right)_y - gb_y h. \tag{2.23}$$

We dropped the notation for the average, such that h is the height of the water column, hu and hv are the depth-averaged flow-rates of sea-water (also called discharge) b is the bathymetry, and g the gravitational acceleration.

The eigenvalues of the resulting system Equation (2.21) to Equation (2.23) are

$$\lambda_i \in \left\{u \pm \sqrt{gh}, u, v, v \pm \sqrt{gh}\right\}. \tag{2.24}$$

In contrast to the common definition of the SWE, we included a time-dependent bathymetry $b(x, y, t)$. This is crucial for the sourcing of tsunamis from earthquakes, which we introduce in Section 5.1.

We can rewrite Equation (2.21) to Equation (2.23) in terms of flux $\mathrm{F}(\mathrm{Q})$ and source $\mathrm{S}(\mathrm{Q})$ terms,

$$\frac{\partial}{\partial t}\mathrm{Q} + \nabla \cdot \mathrm{F}(\mathrm{Q}) = \mathrm{S}(\mathrm{Q}), \tag{2.25}$$

where the divergence operator is defined by

$$\nabla \cdot \mathrm{F}(\mathrm{Q}) = \frac{\partial}{\partial x}\mathrm{F}_1(\mathrm{Q}) + \frac{\partial}{\partial y}\mathrm{F}_2(\mathrm{Q}). \tag{2.26}$$

Finally, the physical quantities of the SWE are stored in the vector Q and the flux and source term are defined with

$$\mathrm{Q} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \qquad \mathrm{S}(q) = \begin{pmatrix} 0 \\ -gb_x h \\ -gb_y h \end{pmatrix} \tag{2.27}$$

$$\mathrm{F}_1(\mathrm{Q}) = \begin{pmatrix} hu \\ \frac{1}{2}gh^2 + hu^2 \\ huv \end{pmatrix}, \qquad \mathrm{F}_2(\mathrm{Q}) = \begin{pmatrix} hv \\ huv \\ \frac{1}{2}gh^2 + hv^2 \end{pmatrix}.$$

2.4 Nonlinear DG for the SWE on triangular meshes

The ADER-DG and Runge-Kutta DG schemes that we present in this chapter are variations of the discontinuous Galerkin method. Base of both schemes are the SWE as we introduced them in Section 2.3,

$$\frac{\partial}{\partial t}\mathrm{Q} + \nabla \cdot \mathrm{F}\left(\mathrm{Q}\right) = \mathrm{S}\left(\mathrm{Q}\right). \tag{2.25}$$

The first step to discretize Equation (2.25) into an iterative scheme is the construction of the semi-discrete scheme. We discretize Equation (2.25) in space only to obtain an ordinary differential equation (ODE) in time. The way the ODE is numerically solved is the characterizing difference between both DG methods.

2.4.1 The semi-discrete scheme

The sub-steps to obtain the semi-discrete scheme have been extensively worked out in literature (a very detailed introduction is given in [70]). In our case we can summarize them with:

1. Following the triangular meshes in sam(oa)^2, we divide the domain $\mathbf{\Omega}$ into $\mathcal{M}$ distinct triangles $\Omega^{(m)}$, $m \in [0, 1, ..., \mathcal{M}]$.

2. On each triangle we use a polynomial basis to approximate Q locally.

3. We solve the PDE on a set of element local test functions.

4. In order to couple elements with their neighbors, we introduce fluctuations.

5. Using integration by substitution, we transform the local integrals from each element onto a reference element. The transformation allows us to express element local operators with a set of general operators defined on the reference element.

6. Operators on the reference element are precomputed, which results in a set of small dense matrix kernels.

Local polynomial interpolation

For a set of interpolation points $(x_i^{(m)}, y_i^{(m)}), i = 1 \dots M$ placed on a triangle $\Omega^{(m)}$, we create the interpolatory basis $\phi_i^{(m)}(x, y)$ such that:

$$\phi_i^{(m)} \left(x_j^{(m)}, y_j^{(m)} \right) = \delta_{ij}, \forall i, j \in 0, ..., M. \tag{2.28}$$

We approximate Q with the polynomial interpolation rule :

$$\mathrm{Q}|_{\Omega^{(m)}} \approx \sum_{i=0}^{M} \phi_i^{(m)}(x, y) \mathrm{q}_i^{(m)}(t) =: \phi_i^{(m)}(x, y) \mathrm{q}_i^{(m)}(t). \tag{2.29}$$

Here we introduced the famous Einstein notation: Indices that appear twice in a multiplication are summed up afterwards [44]. The time-dependent coefficients are denoted with:

$$\mathrm{q}_i^{(m)}(t) = \left(h_i^{(m)}(t), hu_i^{(m)}(t), hv_i^{(m)}(t) \right) \tag{2.30}$$

and correspond to the basis function $\phi_i^{(m)}$. The coefficients are called degrees of freedom (dofs), in which we store the nodal values of all three physical quantities. In order to represent a polynomial of degree N on a triangle, we require $M = (N + 1)(N + 2)/2$ interpolation points. With the same interpolation rule we can approximate the nonlinear flux and source term,

$$\phi_i^{(m)} \frac{\partial}{\partial t} \mathrm{q}_i^{(m)} + \left(\nabla_d \phi_i^{(m)} \right) \mathrm{F}_d \left(\mathrm{q}_i^{(m)} \right) - \phi_i^{(m)} \mathrm{S} \left(\mathrm{q}_i^{(m)} \right) = 0. \tag{2.31}$$

In Equation (2.31) we reinterpreted the divergence operator as tensor contraction

$$\nabla \cdot \left(\phi_i^{(m)} \mathrm{F} \left(\mathrm{q}_i^{(m)} \right) \right) =: \left(\nabla_d \phi_i^{(m)} \right) \mathrm{F}_d \left(\mathrm{q}_i^{(m)} \right), \tag{2.32}$$

where ∇_d is the gradient and $d \in 1, 2$ corresponds to the two spatial dimensions.

For a spatial approximation with polynomials of degree N, we can expect a convergence of $N + 1$ [33, 73, 105].

The weak solution

If we knew that the solution was smooth, the spatial discretization would be finished at this point. Unfortunately, one of the core characteristics of nonlinear hyperbolic PDEs is that discontinuities can evolve in the solution even when the initial and boundary conditions are smooth [86]. Sobolev

spaces are the mathematical tool to capture this aspect, as they introduce weak-derivatives for discontinuous functions [2]. The details of Sobolev spaces are out of this work's scope, for us it is important that in a Sobolev space we look for the weak-solution of Equation (2.31). The weak-solution is defined as the integral over $\Omega^{(m)}$ of Equation (2.31) and a set of arbitrary smooth test functions with compact support on $\Omega^{(m)}$. The DG ansatz in particular considers the weak-solution only on the reduced subset of basis functions $\phi_j^{(m)}$. Under the assumption that the solution is differentiable over elements, this leads to the integral form of the semi-discrete scheme

$$\begin{aligned} \forall j = 0, .., M : & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \phi_i^{(m)} \, \mathrm{d}\Omega^{(m)} \right) \frac{\partial}{\partial t} \mathrm{q}_i^{(m)} + \\ & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \left(\nabla_d \phi_i^{(m)} \right) \mathrm{d}\Omega^{(m)} \right) \mathrm{F}_d \left(\mathrm{q}_i^{(m)} \right) - \\ & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \phi_i^{(m)} \, \mathrm{d}\Omega^{(m)} \right) \mathrm{S} \left(\mathrm{q}_i^{(m)} \right) = 0. \end{aligned} \tag{2.33}$$

Fluctuations

Equation (2.33) is only valid for solutions that are differentiable in the whole domain. However, after the elementwise discretization, we have to address discontinuities that occur on the interfaces of elements. In terms of hyperbolic PDEs these discontinuities state a Riemann problem on each interface, leading to fluctuations between adjacent elements [86]. We incorporate those fluctuations in Equation (2.33) by introducing face integrals over a numerical flux $\mathcal{F}(\cdot,\cdot)\cdot\vec{n}$. The numerical flux is computed on the three edges $\Gamma^{(m,k)}$ of a triangle, between the solution on the element $\phi_i^{(m)} \mathrm{q}_i^{(m)}$ and the solution of the adjacent neighbor $\phi_i^{(m_k)} \mathrm{q}_i^{(m_k)}$

$$\sum_{k=1}^{3} \int_{\Gamma^{(m,k)}} \phi_j^{(m)} \mathcal{F} \left(\phi_i^{(m)} \mathrm{q}_i^{(m)}, \phi_i^{(m_k)} \mathrm{q}_i^{(m_k)} \right) \cdot \vec{n} \, \mathrm{d}\Gamma^{(m,k)}, \tag{2.34}$$

where we denote the index of the adjacent neighbor on the edge k-th edge with m_k and the index of the edge in the neighboring element with e_k. In order to approximate Equation (2.34) we use a one-dimensional polynomial interpolation rule which we transform on the edge, such that it corresponds to interpolation points $(x_l^{(m,k)}, y_l^{(m,k)})$ on the k-th edge.

The projection matrices $\mathcal{P}_{il}^{(m,k)}$ are used to evaluate the basis functions on the k-th edge

$$\mathcal{P}_{il}^{(m,k)} := \phi_i^{(m)} \left(x_l^{(m,k)}, y_l^{(m,k)} \right). \tag{2.35}$$

By denoting the transformed interpolation polynomial with $\varphi_i^{(m,k)}(x,y)$ we get

$$\sum_{k=1}^{3} \int_{\Gamma^{(m,k)}} \phi_j^{(m)} \varphi_i^{(m,k)} \mathcal{F}\left(\mathcal{P}_{il}^{(m,k)} \mathrm{q}_l^k, \mathcal{P}_{il}^{(m_k,e_k)} \mathrm{q}_l{}^{(m_k)}\right) \cdot \vec{n} \, \mathrm{d}\Gamma^{(m,k)} \,. \tag{2.36}$$

There is a vast number of approaches to compute numerical fluctuations [32, 57, 86, 126]. The most common ones are based on an approximate solution to the Riemann problem on the interface. For the SWE we mostly rely on the Rusanov flux, which be briefly introduce in Section 2.4.3. We finally end up with the strong formulation of the semi-discrete scheme [70]

$$\begin{aligned} \forall j : & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \phi_i^{(m)} \, \mathrm{d}\Omega^{(m)}\right) \frac{\partial}{\partial t} \mathrm{q}_i^{(m)} + \\ & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \left(\nabla_d \phi_i^{(m)}\right) \mathrm{d}\Omega^{(m)}\right) \mathrm{F}_d\left(\mathrm{q}_i^{(m)}\right) + \\ \sum_{k=1}^{3} & \left(\int_{\Gamma^{(m,k)}} \phi_j^{(m)} \varphi_i^{(m,k)} \, \mathrm{d}\Gamma^{(m,k)}\right) \mathcal{F}\left(\mathcal{P}_{il}^{(m,k)} \mathrm{q}_l^{(m)}, \mathcal{P}_{il}^{(m_k,k)} \mathrm{q}_l^{(m_k)}\right) \cdot \vec{n} - \\ & \left(\int_{\Omega^{(m)}} \phi_j^{(m)} \phi_i^{(m)} \, \mathrm{d}\Omega^{(m)}\right) \mathrm{S}\left(\mathrm{q}_i^{(m)}\right) = 0, \forall j = 0, .., M. \end{aligned} \tag{2.37}$$

The only contribution to Equation (2.37) from neighboring elements is in the numerical flux. Hence, the only exchange of information between elements happens through fluctuations.

Matrix kernels and integration by substitution

At this point, all unknowns remain as functions in time and independent from the spatial dimensions, which allows us to solve all integrals in advance: Instead of computing all matrices for each single element in the mesh, we can use integration by substitution to transform all basis functions to the reference triangle $\Omega = \{(x,y) : (x,y) \in [0,1]^2, y \leq x\}$. Due to the mesh structure of sam(oa)^2, all transformations $\tau^{(m)} : \Omega^{(m)} \to \Omega$ are purely affine,

$$\phi^{(m)}(x,y) = \phi(\tau^{(m)}(x,y)) \tag{2.38}$$

$$\tau_r^{(m)}(x,y) = \Delta x^{(m)} A_{rd}^{(m)} \begin{pmatrix} x \\ y \end{pmatrix}_d + \vec{t}_d^{(m)}, \text{ for } d, r \in \{0,1\}. \tag{2.39}$$

The transformation from $\Omega^{(m)}$ is defined by the length of its catheti $\Delta x^{(m)}$, the rotation matrix $A_{lk}^{(m)}$ (for meshes in sam(oa)^2 only the angles $\alpha = \frac{k\pi}{4}, k = 0, ..., 7$ appear), and the transposition $\vec{t}_l^{(m)}$.

In order to compute derivatives using the operator of the reference element, we change the variables of the derivative to the reference coordinates ξ, η on Ω,

$$\nabla_d = \left(\frac{\partial}{\partial\eta}, \frac{\partial}{\partial\xi}\right)_r \left(D\tau^{(m)}\right)_{rd} =: \bar{\nabla}_r \Delta x^{(m)} A_{rd}^{(m)}. \tag{2.40}$$

Here we introduced the derivative on the reference element $\bar{\nabla}_r$.

We define the spatial mass M_{ji}, stiffness K_{ji}, boundary B_{imj} and projection matrices $\mathcal{P}_{il}^{(k)}$ on the reference element,

$$\begin{aligned} \mathrm{M}_{ji} &:= \int_\Omega \phi_j \phi_i \,\mathrm{d}\Omega & \mathrm{K}_{jri} &:= \int_\Omega \phi_j \left(\bar{\nabla}_r \phi_i\right) \mathrm{d}\Omega \\ \mathrm{B}_{ji}^{(k)} &:= \int_{\Gamma^{(k)}} \phi_j \varphi_i^{(k)} \,\mathrm{d}\Gamma^{(k)} & \mathcal{P}_{il}^{(k)} &:= \phi_i\left(x_l^{(k)}, y_l^{(k)}\right). \end{aligned} \tag{2.41}$$

By substituting $\Omega^{(m)}$ with Ω in Equation (2.37), we construct the matrix form of the semi-discrete scheme,

$$\begin{aligned} &\forall j : \Delta x^{(m)} \mathrm{M}_{ji} \frac{\partial}{\partial t} \mathrm{q}_i^{(m)} + \mathrm{K}_{jri} A_{rd}^{(m)} \mathrm{F}_d\left(\mathrm{q}_i^{(m)}\right) + \\ &\sum_{k=1}^{3} \mathrm{B}_{ji}^{(k)} \mathcal{F}\left(\mathcal{P}_{il}^{(k)} \mathrm{q}_l^{(m)}, \mathcal{P}_{il}^{(e_k)} \mathrm{q}_l^{(m_k)}\right) \cdot \vec{n} - \mathrm{M}_{ji} \mathrm{S}\left(\mathrm{q}_i^{(m)}\right) = 0. \end{aligned} \tag{2.42}$$

For the sake of brevity, we summarize the semi-discrete scheme with a volume $\mathrm{Vol}\,(\cdot)$ and a boundary $\sum_{k=1}^{3} \mathrm{Bnd}\,(\cdot)$ operator,

$$\frac{\partial}{\partial t} \mathrm{q}_i^{(m)} = \frac{1}{\Delta x^{(m)}} \left(\mathrm{Vol}\left(\mathrm{q}_i^{(m)}\right) + \sum_{k=1}^{3} \mathrm{Bnd}\left(\mathrm{q}_i^{(m)}, \mathrm{q}_i^{(m_k)}\right)\right). \tag{2.43}$$

2.4.2 Integration in time

Integration in time is the most significant difference between our numerical methods.

Runge-Kutta DG

The most common approach to integrate Equation (2.43) in time are given by explicit Runge-Kutta (RK) methods, where the ODE is evaluated at several intermediate time-steps [107]. In case of a second order RK method the dofs $\mathrm{q}_i^{(n,m)}$ at t_n are updated in two stages, for a time-step size of $\Delta t = t_{n+1} - t_n$,

$$\begin{aligned} {}^*\mathrm{q}_i^{(n,m)} =& \mathrm{q}_i^{(n,m)} + \frac{\Delta t}{2\Delta x}\left(\mathrm{Vol}\left(\mathrm{q}_i^{(n,m)}\right) + \sum_{k=1}^{3}\mathrm{Bnd}\left(\mathrm{q}_i^{(n,m)}, \mathrm{q}_i^{(n,m_k)}\right)\right) \\ \mathrm{q}_i^{(n+1,m)} =& \frac{1}{2}\mathrm{q}_i^{(n,m)} + \\ & \frac{1}{2}\left({}^*\mathrm{q}_i^{(n,m)} + \frac{\Delta t}{\Delta x}\left(\mathrm{Vol}\left({}^*\mathrm{q}_i^{(n,m)}\right) + \sum_{k=1}^{3}\mathrm{Bnd}\left({}^*\mathrm{q}_i^{(n,m)}, {}^*\mathrm{q}_i^{(n,m_k)}\right)\right)\right). \end{aligned} \tag{2.44}$$

This approach is comparably easy to realize as the same implementation of Equation (2.43) can be used in every stage. However RK methods come with two significant downsides:

- The Butcher barrier lets the number of stages, that are required to reach high order convergence, grow superlinear [23, 24].
- As we have to compute numerical fluctuations at every stage, we have to traverse the whole mesh for each stage.

ADER-DG

The Butcher barrier is one of the main reasons for the development of the Arbitrary High Order Derivative DG method (ADER-DG) [33, 126]. Instead of directly solving Equation (2.43), we find an element-local prediction of the evolution of $\mathrm{q}_i^{(m)}$

$$\mathrm{q}_i^{(m)}(t) \approx \varphi_l^{(n)}(t)\mathrm{p}_{il}^{(n,m)}, \tag{2.45}$$

with a one-dimensional quadrature rule $\varphi_l^{(n)}$ at N time-steps $t_l \in [t^n, t^{n+1}]$, such that $t_0 = t^n$ and $t_N = t^{n+1}$. The coefficients p_{il} are independent from space and time. The Predictor represents the solution in an element under the assumption that the global solution is smooth across boundaries. In this assumption lies the main motivation for the ADER-DG scheme, in case of a smooth solution the temporal convergence-rate is determined by the polynomial degree of the interpolation rule.

To compute the DG-Predictor, we take Equation (2.37) and set all fluctuations to zero $\mathcal{F}(,) \cdot \vec{n} = 0$. We solve the nonlinear equation system for the DG-Ansatz in time, with test functions $\varphi_m^{(n)}(t)$:

$$\begin{aligned} \Delta x^{(m)} \mathrm{M}_{ji} \int_{t^n}^{t^{n+1}} \varphi_m^{(n)} \frac{\partial}{\partial t} \varphi_l^{(n)} \, \mathrm{d}t \cdot \mathrm{p}_{il}^{(n,m)} + \\ \mathrm{K}_{jri} \int_{t^n}^{t^{n+1}} \varphi_m^{(n)} \varphi_l^{(n)} \, \mathrm{d}t \cdot \mathrm{F}_r \left(\mathrm{p}_{il}^{(n,m)} \right) + \\ \mathrm{M}_{ji} \int_{t^n}^{t^{n+1}} \varphi_m^{(n)} \varphi_l^{(n)} \, \mathrm{d}t \cdot \mathrm{S} \left(\mathrm{p}_{il}^{(n,m)} \right) = 0 \end{aligned} \tag{2.46}$$

There are several analytically equivalent solutions to solve Equation (2.46). We follow the continuous Galerkin predictor approach [55].

As the coefficients are independent of time, we can precompute the temporal mass ${}^t\mathrm{M}$ and stiffness ${}^t\mathrm{K}$ matrices. To simplify the integral, we use integration by substitution to transform the interval $[t_n, t_{n+1}]$ to $[0, 1]$. We get the discrete nonlinear equation system:

$$\begin{aligned} \Delta x^{(m)} {}^t\mathrm{K}_{ml} \mathrm{M}_{ji} \mathrm{p}_{il}^{(n,m)} = \\ \Delta t \left({}^t\mathrm{M}_{ml} \mathrm{K}_{jri} \cdot \mathrm{F}_r \left(\mathrm{p}_{il}^{(n,m)} \right) + {}^t\mathrm{M}_{ml} \mathrm{M}_{ji} \cdot \mathrm{S} \left(\mathrm{p}_{il}^{(n,m)} \right) \right). \end{aligned} \tag{2.47}$$

As in Equation (2.43), we can again move all spatial matrices to the right. On the left side we split the temporal stiffness matrix and the DG-Predictor into the part belonging to t_n at $l = 0$ and $l = 1, ..., N$ for all other intermediate time-steps. For $l = 0$ we know by definition that $\mathrm{p}_{i0}^{(n,m)} = \mathrm{q}_i^{(n,m)}$,

$${}^t\mathrm{K}_{ml} = \left({}_0^t\mathrm{K}_m, {}_1^t\mathrm{K}_{mL} \right) \qquad \mathrm{p}_{il}^{(n,m)} = \left(\mathrm{q}_i^{(n,m)}, {}_1\mathrm{p}_{iL}^{(n,m)} \right), \tag{2.48}$$

with $L \in 0, \ldots, N-1$. Finally, we can solve Equation (2.47) for ${}_1\mathrm{p}_{iL}^{(n,m)}$

$$\begin{aligned} {}_1\mathrm{p}_{jL}^{(n,m)} = & \frac{\Delta t}{\Delta x^{(m)}} {}_1^t\mathrm{K}_{Lm}^{-1} {}^t\mathrm{M}_{ml} \mathrm{M}_{iJ}^{-1} \mathrm{K}_{Jki} \cdot \mathrm{F}_k \left(\mathrm{p}_{il}^{(n,m)} \right) + \\ & \frac{\Delta t}{\Delta x^{(m)}} {}_1^t\mathrm{K}_{Lm}^{-1} {}^t\mathrm{M}_{ml} \cdot \mathrm{S} \left(\mathrm{p}_{jl}^{(n,m)} \right) - {}_1^t\mathrm{K}_{Lm0}^{-1} {}_0^t\mathrm{K}_m \mathrm{q}_j^{(n,m)} =: \mathrm{P} \left(\mathrm{p}_{jL}^{(n,m)}, \mathrm{q}_j^{(n,m)} \right). \end{aligned} \tag{2.49}$$

With P we summarize the whole right side of the equation. Dumbser et al. show that for bound flux and source terms, P is a self contraction [33]. After Banach's fixed-point theorem, this allows us to solve Equation (2.49)

with a fixed-point iteration

$$ {}^{i+1}\mathrm{p}_{iL}^{(n,m)} = \mathrm{P}\left({}^{i}\mathrm{p}_{iL}^{(n,m)}, \mathrm{q}_{j}^{(n,m)}\right). \tag{2.50} $$

As initial values[2] we use ${}^{0}\mathrm{p}_{iL}^{(n,m)} = \mathrm{q}_{i}^{(n,m)}$ for all L. To compute the predictor we repeat Equation (2.50) until the change in one iteration is below a predefined threshold

$$ ||{}^{i+1}\mathrm{p}_{iL}^{(n,m)} - {}^{i}\mathrm{p}_{iL}^{(n,m)}||_2 < \epsilon_{pred}, \tag{2.51} $$

where we use $\epsilon_{pred} = 1.0 \times 10^{-16}$ throughout this thesis. Equivalently to the Picard Iteration in a continuous setting, the fixed-point iteration finds a discrete solution to the ODE, we defined in Equation (2.46). This is why it is often referred to as the Picard loop or directly called Picard Iteration. As result, we obtain the element local evolution of the solution spanned on a grid in time and space.

The corrector step

To get a time-stepping scheme and couple the local solution with adjacent elements we take the semi-discrete scheme and integrate it in time:

$$ \int_{t_n}^{t_{n+1}} \frac{\partial}{\partial t}\mathrm{q}_{i}^{(m)}\,\mathrm{d}t \;=\; \frac{1}{\Delta x^{(m)}} \int_{t_n}^{t_{n+1}} \mathrm{Vol}\left(\mathrm{q}_{i}^{(m)}\right) \;+\; \sum_{k=1}^{3} \mathrm{Bnd}\left(\mathrm{q}_{i}^{(m)}, \mathrm{q}_{i}^{(m_k)}\right) \mathrm{d}t\,. \tag{2.52} $$

The left hand side is evolved with the fundamental theorem of calculus, such that we get $\mathrm{q}_{i}^{(n+1,m)} - \mathrm{q}_{i}^{(n,m)}$. Flux, source and fluctuations at the boundary now have to be computed in space and time [126]. We interpolate the flux, source and boundary terms with the same interpolation rule that we used in Equation (2.45). All integrals can be easily transformed to the interval $[0, 1]$, which again allows us to precompute them:

$$ \begin{aligned} \mathrm{q}_{i}^{(n+1,m)} &= \mathrm{q}_{i}^{(n,m)} + \frac{\Delta t}{\Delta x^{(m)}} \\ &\left(\left(\int_0^1 \varphi_j\,\mathrm{d}t\right)\mathrm{Vol}\left(\mathrm{p}_{ij}^{(n,m)}\right) + \left(\int_0^1 \varphi_j\,\mathrm{d}t\right)\sum_{k=1}^{3}\mathrm{Bnd}\left(\mathrm{p}_{ij}^{(n,m)}, \mathrm{p}_{ij}^{(n,m_k)}\right)\right). \end{aligned} \tag{2.53} $$

The resulting scheme is called the Corrector-step and allows us to propagate

[2] In case of a resting lake the initial condition is the solution.

the solution in time, having the predictor of an element and its adjacent neighbors.

The time-step size and CFL condition

In order to ensure stability of the Runge-Kutta time-stepping and the DG-Predictor, we keep Δt within the limit given by the CFL condition [28, 33]

$$\Delta t_n \leq \min_{\Omega^{(m)}} \Delta x^{(m)} \left((2N+1) \max_i \lambda_i \left(\phi_j \mathrm{q}_j^{(n,m)} \right) \right)^{-1}, \tag{2.54}$$

where $\lambda_i(\mathrm{q}^{(n,m)})$ are the eigenvalues of the SWE evaluated at t_n for triangle $\Omega^{(m)}$,

$$\lambda_0 = u - \sqrt{gh} \quad \lambda_1 = u + \sqrt{gh} \quad \lambda_2 = v - \sqrt{gh} \quad \lambda_3 = v + \sqrt{gh} \tag{2.55}$$

As we need the time-step size a priori, we estimate it for t_n based on the nodal values in each cell at t_{n-1} and scale the result by a factor $c \in]0, 1]$ to account for possible jumps in λ between the time-steps

$$\Delta t_n = c \cdot \min_{\Omega^{(m)}} \Delta x^{(m)} \left((2N+1) \max_{i,j} \lambda_i \left(\mathrm{q}_j^{(n-1,m)} \right) \right)^{-1}. \tag{2.56}$$

2.4.3 Fluctuations

In both codes we compute fluctuations with the Rusanov flux [86]

$$\mathcal{F}\left(\mathrm{q}_i^L, \mathrm{q}_i^R\right) \cdot \vec{n} = \frac{1}{2}\left(\mathrm{F}\left(\mathrm{q}_i^L\right) - \mathrm{F}\left(\mathrm{q}_i^R\right)\right) \cdot \vec{n} + \frac{1}{2}\bar{\mathrm{B}}\left(\mathrm{q}_i^L, \mathrm{q}_i^R\right) \cdot \vec{n} - \frac{\alpha}{2}\left(\mathrm{q}_i^R - \mathrm{q}_i^L\right), \tag{2.57}$$

where q_i^L and q_i^R are the two projected dofs from Equation (2.36) and α is the maximal absolute eigenvalue. $\bar{\mathrm{B}}$ accounts for the hydrostatic pressure and the source term, and is in our case defined by

$$\bar{\mathrm{B}}\left(\mathrm{q}_i, \mathrm{q}_i^R\right) \cdot \vec{n} = \begin{pmatrix} 0 \\ g(h_i^L + h_i^R)\left(\max(h_i^L + b_i^L - \bar{b}_i, 0) - \max(h_i^R + b_i^R - \bar{b}_i, 0)\right)\vec{n}_1 \\ g(h_i^L + h_i^R)\left(\max(h_i^L + b_i^L - \bar{b}_i, 0) - \max(h_i^R + b_i^R - \bar{b}_i, 0)\right)\vec{n}_2 \end{pmatrix}, \tag{2.58}$$

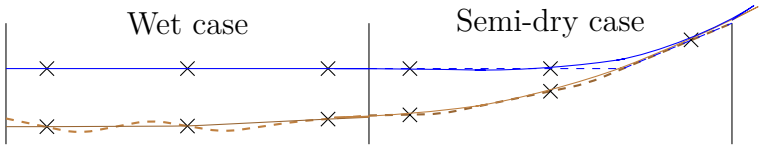


Figure 2.4.: Water level (blue) and bathymetry (brown) at a coast in the analytic (dashed) and discrete setting. Quadrature nodes are marked as crosses. For a wet area (left) and at a coast (right), in which the discrete setting becomes the semi-dry case.

where $\bar{b}_i = \max\left(b^L, b^R\right)$. The Rusanov flux has to fulfill the well-balanced criterion, for identical water level $h_i^L + b_i^L = h_i^R + b_i^R$ but different water columns $h_i^L \neq h_i^R$ and zero velocity on both sides. While Equation (2.58) ensures that velocities remain zero in this case, fluctuations are generated in the diffusive term of the water column as $h_i^R - h_i \neq 0$. To enforce well-balancedness we change it to

$$\left(\max(h_i^L + b_i^L - \bar{b}_i, 0) - \max(h_i^R + b_i^R - \bar{b}_i, 0)\right). \tag{2.59}$$

In case of the ADER-DG method we have to take the time integral from the Corrector (Equation (2.53)) into account

$$\begin{aligned} &\frac{1}{2}\int_0^1 \phi_j \left(\mathrm{F}\left(\mathrm{p}_{ij}^L\right) - \mathrm{F}\left(\mathrm{q}_i^R\right)\right)\cdot\vec{n}\,\mathrm{d}t + \\ &\frac{1}{2}\int_0^1 \phi_j \bar{\mathrm{B}}\left(\mathrm{p}_{ij}^L, \mathrm{p}_{ij}^R\right)\cdot\vec{n}\,\mathrm{d}t - \int_0^1 \phi_j \frac{\alpha}{2}\left(\mathrm{p}_{ij}^R - \mathrm{p}_{ij}^L\right)\mathrm{d}t\,. \end{aligned} \tag{2.60}$$

In this version of the numerical flux, we would have to store the whole projected predictor on each edge, to later evaluate the flux and source term. Instead we exploit that the Rusanov flux is linear in $\mathrm{F}\left(\mathrm{p}_{ij}^L\right)\cdot\vec{n}$ and in p and only store the time-integrated values:

$$\frac{1}{2}\left(\mathcal{F}_i^L - \mathcal{F}_i^R\right)\cdot\vec{n} + \frac{1}{2}\bar{\mathrm{B}}\left(\mathcal{Q}_i^L, \mathcal{Q}_i^R\right)\cdot\vec{n} - \frac{\alpha}{2}\left(\mathcal{Q}_i^R - \mathcal{Q}_i^L\right), \tag{2.61}$$

with

$$\begin{aligned} \mathcal{Q}_i = \int_0^1 \phi_j \mathrm{p}_{ij}\,\mathrm{d}t\,, \mathcal{F}_i = \int_0^1 \phi_j \mathrm{F}\left(\mathrm{p}_{ij}\right)\mathrm{d}t\,, \\ \bar{\mathrm{B}}\left(\mathcal{Q}_i^L, \mathcal{Q}_i^R\right) \approx \int_0^1 \phi_j \bar{\mathrm{B}}\left(\mathrm{p}_{ij}^L, \mathrm{p}_{ij}^R\right)\cdot\vec{n}\,\mathrm{d}t\,. \end{aligned} \tag{2.62}$$

2.5 The sam(oa)^2-flash code

The first numerical method we realized in sam(oa)^2 is called sam(oa)^2-flash, which is based on the work by Vater et al. [136]. sam(oa)^2-flash is a second order RK-DG scheme: Second order convergence in space is acquired through a linear polynomial interpolation on Gauss-Legendre nodes (See Equation (2.29)). Time integration is performed with Heun's method, such that the numerical scheme is equivalent to the one we derived in Equation (2.44).

On the boundary of the elements, the well-balanced positivity preserving Rusanov flux from Section 2.4.3 defines numerical fluctuations and is interpolated on three Gauss-Legendre quadrature nodes (Equation (2.35), see Section 2.4.1). The method's characterizing feature is the Barth-Jespersen type limiter, which is used to adjust the solution after each RK-stage

$$
\begin{aligned}
&\text{1. Traversal:} && {}^{*}\mathrm{q}_i^n = \mathrm{q}_i^n + \frac{\mathrm{d}t}{\mathrm{d}x}\left(\mathrm{Vol}\left(\underline{\mathrm{q}}_i^n\right) - \mathrm{Bnd}\left(\underline{\mathrm{q}}_i^n, \underline{\tilde{\mathrm{q}}}_i^n\right)\right)\\
&\text{2. Traversal:} && {}^{*}\underline{\mathrm{q}}_i^n = \text{BJ-Limiter}({}^{*}\mathrm{q}_i^n, {}^{*}\hat{\mathrm{q}}_i^n) \qquad (2.63)\\
&\text{3. Traversal:} && \mathrm{q}_i^{n+1} = \mathrm{q}_i^n + 0.5\frac{\mathrm{d}t}{\mathrm{d}x}\left({}^{*}\underline{\mathrm{q}}_i^n + \mathrm{Vol}\left({}^{*}\underline{\mathrm{q}}_i^n\right) - \mathrm{Bnd}\left({}^{*}\underline{\mathrm{q}}_i^n, {}^{*}\underline{\tilde{\mathrm{q}}}_i^n\right)\right)\\
&\text{4. Traversal:} && \underline{\mathrm{q}}_i^{n+1} = \text{BJ-Limiter}\left(\mathrm{q}_i^{n+1}, \hat{\mathrm{q}}_i^{n+1}\right).
\end{aligned}
$$

To compute the update ($^{*}\mathrm{q}_i^n$,q_i^{n+1}) in each RK-Stage, we require the limited results of all adjacent elements from the previous stage ($^{*}\underline{\mathrm{q}}_i^n$,$\underline{\mathrm{q}}_i^{n+1}$). The limiter on the other hand uses the updates from all adjacent elements. In total we require four traversals to iterate the scheme by one time-step.

2.5.1 Well-balancedness through flux modification

Well balancedness in wet cells is ensured by the exact Gauss-Legendre interpolation of the scheme, used for the approximation of the flux and source term in Equation (2.31), and the well-balanced Riemann solver: In the case of a wet cell with zero velocities $\mathrm{F}(\mathrm{Q})$ and $\mathrm{S}(\mathrm{Q})$ are quadratic functions of Q. After the discretization of Q to a linear polynomial, flux and source become polynomials of degree two, which we can interpolate exactly with the Gauss-Legendre quadrature rule (compare to the wet case in Figure 2.4).

In order to keep the semi-dry case well-balanced, we force the gravitational acceleration to zero, such that no momenta are generated and the solution stays constant. To detect the semi-dry case, we assume that for a

resting lake with ascending coast the highest water level $H_i = b_i + h_i$ is at a dry node on land (compare to the semi-dry case in Figure 2.4).

With this assumption we can define a criterion to detect semi-dry cells

$$\max_{x_i \in \Omega}(H_i) - \max_{x_i \in \Omega}(b_i) < \text{TOL}, \tag{2.64}$$

where TOL is a predefined tolerance, usually the dry-tolerance.

There are several exceptions where the criterion erroneously marks cells as semi-dry, however these false positives do not show a significant influence on the solution [135].

2.5.2 A Barth-Jespersen typed limiter for wetting and drying cells.

Slope-Limiters are a common tool to resolve oscillations in DG methods for the SWE [29, 142]. While these approaches reliably resolve discontinuities, their interaction with a wetting and drying mechanism can lead to instabilities [21, 141]. This is the main motivation for the newly developed limiter, which is based on the well-balanced Barth-Jespersen limiter and a positivity operator by Bunya et al. [11, 21]. The limiter exists as "edge-based" and "vertex-based" version, where we look only at the "vertex-based" realization, as Vater et al. show that this approach leads to better results for several benchmarks [136].

Limiting fluid depth

Equivalent to the linear interpolatory polynomial basis, we can represent the water level in each cell by its average $\overline{H}$ and slope ∇H

$$H(x) = \overline{H} + \nabla H \cdot (x - x_c)\,, \tag{2.65}$$

where x_c is the centroid of the cell.

Within the neighborhood $\mathcal{N}$ of a cell, which contains the cell itself and all cells that share at least one of its vertices $\mathcal{V}_j$, we compute the minimal and maximal average,

$$\overline{H}_{\min} = \min_{C \in \mathcal{N}}(\overline{H}_C), \qquad \overline{H}_{\max} = \max_{C \in \mathcal{N}}(\overline{H}_C). \tag{2.66}$$

The limiter scales the slope ∇H, by a factor $\alpha \in [0, 1]$, such that all values

in the cell are within these two bounds,

$$\overline{H}_{\min} \leq \overline{H} + \alpha \nabla H \cdot (x - x_c) \leq \overline{H}_{\max}. \tag{2.67}$$

Following this constraint, α is computed by the rule

$$\alpha = \min_{\mathcal{V}_j} \begin{cases} \min(1, \frac{\overline{H}_{min} - \overline{H}}{H_j - \overline{H}}) & , H_j < \overline{H} \\ \min(1, \frac{\overline{H}_{max} - \overline{H}}{H_j - \overline{H}}) & , H_j > \overline{H} \\ 1 & \text{else.} \end{cases} \tag{2.68}$$

In order to avoid negative water columns, a positivity operator defined by Bunya et al. is applied to the limited nodal water columns [21]. The operator equalizes negative water heights, by subtracting the missing values equally from positive water columns. First, all three nodal values for the water-column h_0, h_1, h_2 are sorted in ascending order, then the water columns are rebalanced. For $h_i < h_j < h_k$:

$$\begin{aligned} \underline{\mathrm{h}}_i &:= \max(0, h_i), \\ \underline{\mathrm{h}}_j &:= \max(0, h_j - \frac{\Delta h_i}{2}), \\ \underline{\mathrm{h}}_k &:= \max(0, h_k - \frac{\Delta h_i}{2} - \Delta h_j), \end{aligned} \tag{2.69}$$

where $\Delta h_i = \underline{\mathrm{h}}_i - h_i$ is the amount we equalized in i.

Velocity limiting

As the computation of the velocity $u = \frac{hu}{h}$ is ill-conditioned for small water columns $h \to 0$, it can lead to very high values close to the coastline. While on one hand these high velocities are usually unphysical, they also reduce the maximal admissible time-step size significantly, as they lead to increased wave speeds (Section 2.4.2). In order to avoid high velocities, limiting is also applied to the velocity. While we could apply the same slope-liming that was used for the water level, the limiter uses a different approach, that focuses on the reduction of the variability of velocities within a DG cell. As both velocity components are limited independently with the same algorithm, we show it for v only:

We first compute the cell averages of the water columns $\overline{h}$, momentum $\overline{hv}$ and velocity $\overline{v}$ and determine their minimum $\overline{v}_{min}$ and maximum $\overline{v}_{max}$ in the neighborhood $\mathcal{N}$. On each vertex the velocity is forced within the

maximum and minimum averages,

$$\underline{\mathrm{v}}_i = \min\left(\max\left(v_i, \bar{v}_{min}\right), \bar{v}_{max}\right). \tag{2.70}$$

After limiting we want to keep the average momentum in the cell, such that we can only choose to limit two of the velocities $\underline{\mathrm{v}}_j, \underline{\mathrm{v}}_k$ with Equation (2.70). The third velocity has to be recomputed with

$$\underline{\mathrm{v}}_i^b = \frac{\overline{hv} + (\overline{hv} - \underline{\mathrm{h}}_j\underline{\mathrm{v}}_j) + (\overline{hv} - \underline{\mathrm{h}}_k\underline{\mathrm{v}}_k)}{\overline{h}}. \tag{2.71}$$

This leaves us with three potential configurations, from which we want to pick the one for which we end up with the smallest variation,

$$\Delta v_i = \max\left(\underline{\mathrm{v}}_i^b, \underline{\mathrm{v}}_j, \underline{\mathrm{v}}_k\right) - \min\left(\underline{\mathrm{v}}_i^b, \underline{\mathrm{v}}_j, \underline{\mathrm{v}}_k\right). \tag{2.72}$$

To find the optimal configuration we have to compute Δv_i for all three possible configurations, from which we then can choose the one with the smallest variation. Finally, the limited momenta are reproduced from the limited water columns and velocities,

$$\underline{\mathrm{hv}}_i = \underline{\mathrm{h}}_i\underline{\mathrm{v}}_i^b,\ \underline{\mathrm{hv}}_j = \underline{\mathrm{h}}_j\underline{\mathrm{v}}_j,\ \underline{\mathrm{hv}}_k = \underline{\mathrm{h}}_k\underline{\mathrm{v}}_k. \tag{2.73}$$

2.6 An ADER-DG method for the SWE

The second method we realized in sam(oa)^2 is an ADER-DG method with *a posteriori* finite volume limiting [103]. We implemented the method following the Predictor-Corrector scheme we derived in Section 2.4.2.

Its spatial interpolation rule (Equation (2.29)) is based on the alpha-optimized quadrature nodes from [70], in time interpolated with Gauss-Lobatto nodes (Equation (2.45)). On the boundary we use the time-averaged well-balanced version of the Rusanov flux (Section 2.4.3), sampled on the N+1 alpha-optimized nodes that are located on each edge (Equation (2.35)).

To limit oscillating cells and resolve wetting and drying, we use the *a posteriori* finite volume sub-cell limiting approach by Dumbser et al. [36]. The limiter requires the implementation of an additional FV scheme, that we take from the work by Ferreira et al. [50]. In the scheme wetting and drying cells are treated by the augmented Riemann solver by George [57].

The limiter is applied *a posteriori* to the solution after the corrector step and by design does not require an additional traversal. Hence we can realize

the scheme in two traversals of the mesh

$$\begin{aligned} &\text{1. Traversal:} && \mathrm{p}_i^{n+1} = \text{Predictor}\left(\mathrm{q}_i^n\right) \\ &\text{2. Traversal:} && \mathrm{q}_i^{n+1} = \mathrm{q}_i^n + \text{Corrector}\left(\mathrm{q}_i^n, \mathrm{p}_i^{n+1}, \hat{\mathrm{p}}_i^{n+1}\right) \\ & && \underline{\mathrm{q}}_i^{n+1} = \text{FV-Limiter}\left(\mathrm{q}_i^{n+1}, \hat{\mathrm{q}}_i^n\right). \end{aligned} \tag{2.74}$$

2.6.1 Well-balancedness through flux splitting

As we resolve the coast and semi-dry cells with the limiting scheme, the ADER-DG method is only applied in areas of deep-water. Hence, we only have to ensure *well-balancedness* for the wet case. The integration rule, based on the alpha-optimized quadrature nodes is not accurate for polynomials of order $2N$ and without a proper modification, the scheme would be *ill-balanced*. To resolve this issue we split the flux of the SWE into a conservative flux modeling the conservation of mass and momentum and a non-conservative flux for the gravitational acceleration

$$\nabla \cdot \mathrm{F}\left(\mathrm{Q}\right) + \mathrm{B}\left(\mathrm{Q}\right) \cdot \nabla \mathrm{Q} + \mathrm{S}\left(\mathrm{Q}\right) = \begin{pmatrix} hu \\ \frac{(hu)^2}{h} \\ \frac{(huhv)}{h} \end{pmatrix}_x + \begin{pmatrix} hv \\ \frac{(huhv)}{h} \\ \frac{(hv)^2}{h} \end{pmatrix}_y + \begin{pmatrix} 0 \\ ghh_x \\ ghh_y \end{pmatrix} - \begin{pmatrix} 0 \\ ghb_x \\ ghb_y \end{pmatrix}.$$

The non-conservative part is treated numerically equivalently to the source term, which allows us to add both terms. After the spatial discretization the terms cancel each other in the resting-lake case as $h_x - b_x = H_x = 0$.

2.6.2 Finite volume limiting: A state machine for troubled or wetting and drying cells

In order to resolve wetting and drying cells and to limit oscillations, our ADER-DG implementation relies on an adjusted realization of the *a posteriori* FV sub-cell limiter by Dumbser et al. [36, 38]. The driving idea behind the limiter is that in the presence of discontinuities FV schemes are more robust and wetting and drying cells can easily be resolved with a suitable Riemann solver. Where a cell is oscillating or has to resolve inundation, we fall back to a FV scheme to evolve the solution. In our case, the first order FV scheme on patches is given by the work of Ferreira et al. [49] and inundation is resolved with the augmented Riemann solver by George [57].

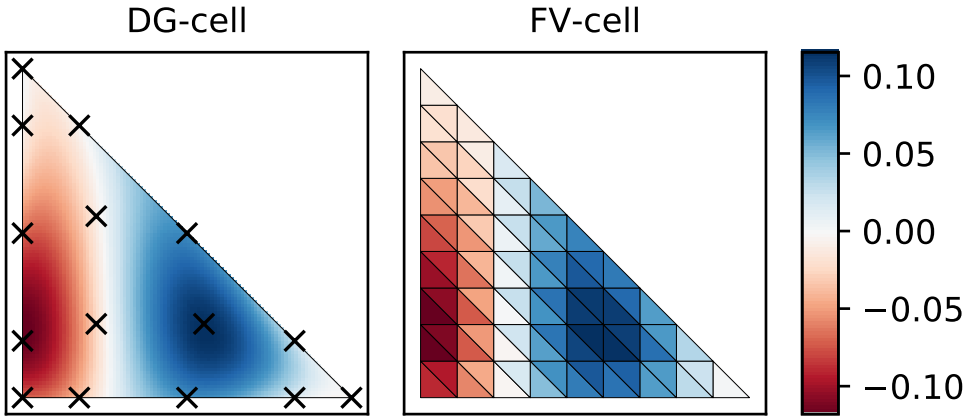


Figure 2.5.: Representation of the water level in a DG-cell (left) an a FV-cell (right). In the DG-cell the solution is represented by a polynomial of degree $N = 4$ on 15 nodes (black crosses). In the FV-cell the solution is represented with constant values in 81 sub-cells.

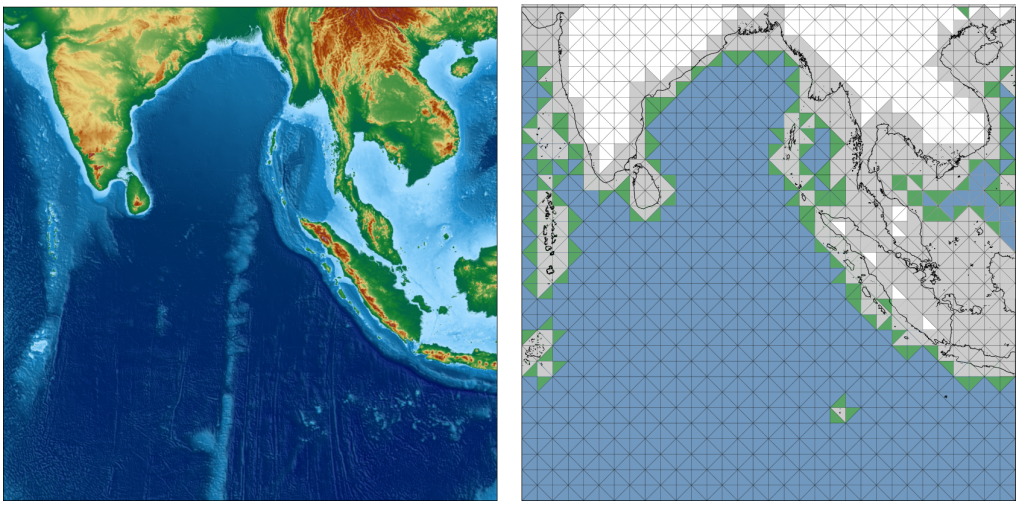


Figure 2.6.: Illustrative example of a hybrid mesh for the Sumatra bathymetry. FV cells (grey) on the coast are detected with the criterion in Equation (2.75). To couple DG cells (blue) and FV cells (grey) we define an interface layer of cells holding both representations (green).

The sub-steps of the limiter in its original version are summarized as:

- **Compute a Candidate Solution** With the ADER-DG method from Section 2.4.2 we compute a candidate solution at the next time-step t_{n+1} for each cell.
- **Detect troubled Cells** All candidate solutions are tested by physical and numerical conditions. In case of the SWE we test if the result contains unphysical negative water columns. To detect oscillations we use the discrete maximum principle, which is introduced in Section 2.6.3.
- **Correct troubled cells** A cell that fails the previous tests is marked as troubled and its candidate solution discarded. We project the solution of the cell and all adjacent neighbors at the current time-step t_n onto a FV patch with $(2N+1)^2$ cells in each dimension (See Figure 2.5 (right) and Section 2.6.4). The FV solution on the whole patch is advanced to t_{n+1} with a robust FV scheme.
- **Reconstruct the DG-Solution** For troubled cells the DG solution is reconstructed from the FV-Solution (See Section 2.6.4).

We will see in Section 2.6.4, that the reconstruction and projection from DG to FV representation and vice versa are neither well-balanced nor positivity preserving. These two issues lead to our version of the limiter: Instead of projecting and reconstructing the solution for cells at the coast in every time-step, we allow cells to persistently store the representation as FV patches.

The resulting mesh contains three types of cells: *DG-cells* that hold a DG representation (Figure 2.5 left), *FV-cells* that hold a FV patch representation (Figure 2.5 right) of the solution and *interface-cells*, that are used to couple DG and FV-cells. An example mesh for the Sumatra benchmark (Section 3.7) is presented in Figure 2.6. The mesh shows DG-cells (blue) in the deep ocean and FV-cells along the coast (grey). Placed between both is a layer of interface-cells (green). Cells on land that are dry and are surrounded by dry cells (white) can be skipped in the computation. To detect cells that are at the coast we initially check the bathymetry at the nodal values of the DG representation for a predefined threshold B_{coast}. The solution of a cell is persistently represented as FV patch if the absolute bathymetry is below this threshold

$$\exists i \in 1, ..., M : |b_i| < B_{coast}. \tag{2.75}$$

While this simple criterion allows us to define a ribbon of FV-cells around the coast line, it can be easily replaced with a more advanced approach whenever needed.

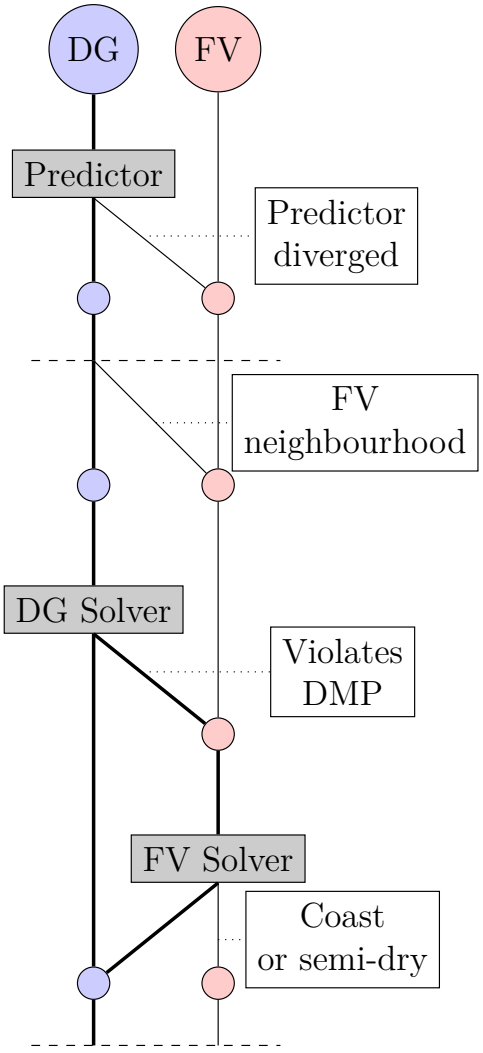


Figure 2.7.: State machine for the representation in a cell of the mesh.

In order to keep track of the various cell types, we implemented a state-machine as presented by the control-flow graph in Figure 2.7.

FV-cells are only advanced with the underlying FV-Solver. The DG-Predictor is explicitly not computed, as in the presence of wetting and drying cells, the fixed-point iteration from Equation (2.49) is prone to diverge as the flux term is potentially not bound. This implies that cells that are adjacent to FV-cells cannot be advanced with the ADER-DG method, as the neighboring DG-Predictor is required to compute fluctuations (See Equation (2.53)). Instead we use cells that are next to FV-cells, but do not resolve a coast or are troubled themselves, as interface-cells to couple between DG- and FV-cells. Interface-cells simultaneously hold a FV and DG representation. From the DG representation, we can evaluate the Picard-Iteration and use the resulting DG-Predictor to compute fluctuations for the neighboring DG-cells. The FV representation on the other hand allows us to compute fluctuations with the neighboring FV-cells. To update interface-cells, we compute the FV representations of all neighboring cells and compute an update using the FV-Solver. Finally the DG representation is reconstructed after every time-step.

For DG-cells we add an additional check for the convergence of the Picard-

Iteration as the iteration might diverge for water columns close to zero, which can appear in scenarios like the oscillating lake (See Section 3.5), where the coastline is constantly moving. We also keep the DMP check for oscillations in DG-cells (thick line).

2.6.3 The discrete maximum principle

Oscillations are detected with the relaxed discrete maximum principle (DMP), which was first defined by Dumbser et al. [36]. It is motivated by the analytic maximum-principle, which states that for the scalar conservation law, the maximal and minimal values of a solution must lie within the minimal and maximal values of its initial condition at any time [144].

The relaxed discrete maximum principle describes this property for the DG-approximation and discrete time-steps: We defined the neighborhood $\mathcal{N}$ of $\Omega^{(m)}$, equivalently to Section 2.5.2, as all elements that share a vertex with $\Omega^{(m)}$ and include $\Omega^{(m)}$. At time-step t_{n+1} the DG solution $\phi_i(x)^{(m)}q_i^{(n+1,m)}$ in element $\Omega^{(m)}$ must remain within the extrema of the neighborhood at t_n, $\forall(x,y) \in \Omega^{(m)}$:

$$\min_{\nu\in\mathcal{N}} \phi_i^{(\nu)} q_i^{(n,\nu)} - \delta \leq \phi_i^{(m)} q_i^{(n+1,m)} \leq \max_{\nu\in\mathcal{N}} \phi_i^{(\nu)} q_i^{(n,\nu)} + \delta. \tag{2.76}$$

The *relaxation parameter* δ introduces a buffer around the extrema. Within the buffer new maxima and minima are allowed to emerge

$$\delta = \delta_s \max\left(\delta_0, \max_{\nu\in\mathcal{N}} \phi_i^{(\nu)} q_i^{(n,\nu)} - \min_{\nu\in\mathcal{N}} \phi_i^{(\nu)} q_i^{(n,\nu)}\right). \tag{2.77}$$

Here, δ_s scales the buffer and δ_0 defines a minimal size of the buffer, which is required to not detect cells where minima and maxima are close to each other as for example in a resting lake.

In case of a shock front, we assume that the occurring oscillations cause the solution to exceed the previous extrema and violate Equation (2.76). Unfortunately, the maximum principle does not hold for arbitrary smooth solutions: For the SWE a simple example where the principle does not hold, is given by a perturbation in the water level of an initially resting lake. Gravity causes the generation of velocity, which violates the DMP compared to the initially zero velocity, if the tolerance is set too small. We see in Chapter 5 that this case is the initial condition for all tsunami simulations. In order to avoid this erroneous limiting for this case, we apply the DMP only to the water column of the solution. In Section 3.4 we test

the difference of applying the DMP for all quantities against applying it to the water column, with the example of the radial dam break.

The computation of extrema in Equation (2.76) is a non-trivial task as it requires the solution of a non-linear equation system. Instead we compare the maximal and minimal coefficients of the DG-representation and the FV-representation in each DG-cell.

2.6.4 Projection an reconstruction

The last components that are missing for the limiter are operators to switch between the DG representation and FV representation in a cell. Here we derive the operators on the reference element only, as after we applied integration by substitution all metric terms are canceled out. Analogous to their original definition by Dumbser et al. [36] we name the operators projection (for DG to FV representation) and reconstruction (for FV to DG representation).

To avoid any loss or gain of mass or energy, both operators ensure that the integrals of physical quantities on the whole cell are conserved. On a sub-cell level, both operators minimize the L_2-difference of both representations.

For a polynomial representation $\phi_i(x,y)\,\mathrm{q}_i$ defined on the reference triangle Ω and constant FV solutions v_i on a set of uniform sub-cells t_i within Ω, these constrains are

$$\sum_i \frac{1}{\|t_i\|} \int_\Omega \sum_j \phi_j(x,y)\,\mathrm{q}_j \mathrm{d}\Omega \stackrel{!}{=} \sum_i v_i, \tag{2.78}$$

$$\left\| \sum_i \frac{1}{\|t_i\|} \int_{t_i} \sum_j \phi_j(x,y)\,\mathrm{q}_j\,\mathrm{d}t_i - v_i \right\|_2 \stackrel{!}{=} \min. \tag{2.79}$$

In case of the projection from DG to FV-representation, we average the DG-representation on every sub-cell to exactly solve Equation (2.79),

$$\frac{1}{\|t_i\|} \int_{t_i} \phi_j \mathrm{q}_j\,\mathrm{d}\vec{x} = \frac{1}{\|t_i\|} \left(\int_{t_i} \phi_j\,\mathrm{d}\vec{x} \right) \mathrm{q}_j = v_i, \forall i. \tag{2.80}$$

We can precompute the projection as a matrix $\Phi \in \mathbb{R}^{(2N+1)^2 \times M}$,

$$\Phi_{ij} = \frac{1}{\|t_i\|} \left(\int_{t_i} \phi_j\,\mathrm{d}\vec{x} \right). \tag{2.81}$$

For the reconstruction on the other hand, the problem is over-determined

as the number of sub-cells $(2N+1)^2$ exceeds the number of nodes in a DG-cell $M = (N+1)\cdot(N+2)/2$. We solve Equation (2.79) with an L2-optimization and enforce the overall conservation in Equation (2.78) to obtain a constrained least squares problem. With λ as Lagrange multiplier and the projection matrix Φ_{ij}, which we use to simplify the minimization term of Equation (2.79), we get the minimization problem

$$\min_{\mathrm{q}_i,\lambda} \left(\Phi_{ij}\mathrm{q}_j - v_i\right)^T \cdot \left(\Phi_{ij}\mathrm{q}_j - v_i\right) - \lambda\left(\mathbf{\Phi}_j\mathrm{q}_j - \sum_i v_i\right). \tag{2.82}$$

With $\mathbf{\Phi}_j$, we denote the row-vector holding the integrals of all basis functions on Ω: $\mathbf{\Phi}_j := \sum_i \Phi_{ij} = \vec{1}_i \cdot \Phi_{ij}$.

As shown in [35] we find the unique solution to Equation (2.82) by calculating the roots of its derivative in q and λ,

$$\begin{aligned} \frac{\delta}{\delta \mathrm{q}} &: 2\Phi_{ij}^T \cdot \Phi_{jk}\mathrm{q}_k - v_j^T \cdot \Phi_{ji} - \Phi_{ij}^T \cdot v_j - \lambda\mathbf{\Phi}_j\mathrm{q}_j = 0 \\ \frac{\delta}{\delta \lambda} &: \mathbf{\Phi}_j\mathrm{q}_j - \sum_i v_i = 0. \end{aligned} \tag{2.83}$$

By moving all dependencies of v to the right, we get the linear equation system

$$\begin{pmatrix} 2\Phi^T \cdot \Phi & \mathbf{\Phi}^T \\ \mathbf{\Phi} & 0 \end{pmatrix} \cdot \begin{pmatrix} \left(\mathrm{q}_j\right) \\ \lambda \end{pmatrix} = \begin{pmatrix} 2\Phi^T \\ \vec{1} \end{pmatrix} \cdot \left(v_i\right). \tag{2.84}$$

For a closed matrix-matrix form of the reconstruction, we invert the matrix on the left side and erase the row which solves for λ. For the resulting matrix we use μ^{-1} as notation,

$$\mathrm{q}_j = \mu_{ji}^{-1} \cdot v_i, \text{ where } \mu^{-1} \in \mathbb{R}^{M\times(2N+1)^2}. \tag{2.85}$$

If we set $v_i = \Phi_{ij}\mathrm{q}_j$ in Equation (2.84) we can solve it with $\lambda = 0$, hence the reconstruction is the left inverse to the projection,

$$\mathrm{q}_k = \mu_{kj}^{-1} \cdot \Phi_{ji} \cdot \mathrm{q}_i. \tag{2.86}$$

Showing that the projection is not the right inverse to the reconstruction is trivial. We know that Equation (2.79) cannot be solved exactly for a

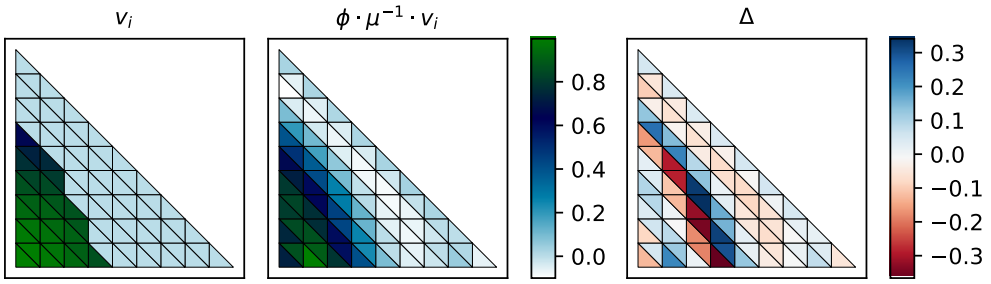


Figure 2.8.: Effects of projection and reconstruction on the water level on a discontinuous coast (left, center). As result artificial waves are generated, shown in the difference on the right.

discontinuous v_i, hence

$$0 \neq \Phi_{kj}\mu_{ji}^{-1}v_i - v_i \rightarrow \Phi_{kj}\mu_{ji}^{-1}v_i \neq v_k. \tag{2.87}$$

Well-balancedness

It is trivial to prove that zero velocities and constant water levels in a cell are conserved, as both operators are linear. This allows us to easily switch between both representations in the deep-water, without violating well-balancedness.

In the semi-dry case, the original formulation of the limiter would lead to a constant switching between FV and DG-representation. We know from Equation (2.87) that the projection is not the right inverse to the reconstruction. This is why it leads to the generation of spurious waves at the interface.

The example in Figure 2.8 shows the water level at a discontinuous synthetic coast (left) compared to the water level after it was reconstructed to a DG- and projected back to a FV-representation (center). The difference of both (right) shows that artificial waves are generated in the process. By keeping cells close to the coast constantly as FV-cells, we can bypass this problem.

Positivity preservation

Looking at the matrices for projection and reconstruction reveals that both hold negative entries. We can thus not be certain that positive values in the polynomial representation of a water column lead to positive values

in the FV representation and vice versa. In case the reconstruction from FV to DG representation results in a negative water column, we simply use our state machine to keep the cell in FV representation until it is fully "repaired".

For the projection, we introduce a mechanism that ensures that negative water columns are equalized and the FV representation contains zero or positive water columns only: Given the conservation constraint Equation (2.78), we know that the overall sum of water columns has to be positive. This allows us to extend the postivity operator of Bunya et al. (see Equation (2.69)) to FV cells and distribute the negative balance equally among sub-cells with positive water height [21].

In Algorithm 1, we describe the exact procedure to rebalance negative water columns. Initially we count the absolute amount we have to distribute among positive water columns. After that, we repeatedly subtract the missing balance equally from all sub-cells. In case a sub-cell holds less than the amount we want to subtracted, we set its water column to zero. In practice, this case is mostly important for cells close to dry areas, where a high variation between nodal values leads to negative values in the polynomial.

ALGORITHM 1: Rebalancing algorithm for negative water columns

```
input  : h(:) potentially negative FV water columns
output: hr(:) positive rebalanced FV water columns
for i ← 0 to len(h) do
    if h(i) < 0 then
        negativeBalance ← negativeBalance - h(i);
        hr(i) ← 0;
    else
        hr(i) ← h(i);
    end
end
nextNegativeBalance ← negativeBalance;
while negativeBalance > 0 do
    for i ← 0 to len(h) do
        Δ ← min(negativeBalance/len(h),hr(i));
        hr(i) ← hr(i) - Δ;
        nextNegativeBalance ← nextNegativeBalance - Δ;
    end
    negativeBalance ← nextNegativeBalance;
end
```

CHAPTER

3

Verification of the tsunami methods

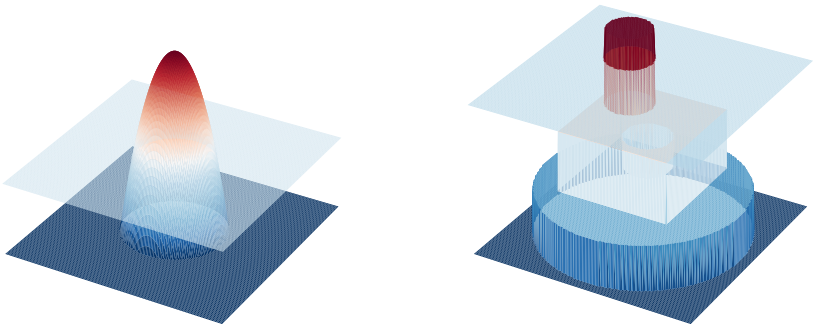
3.1 Introduction

In this chapter we want to verify the numerical schemes we derived in the last chapter. We base the selection of benchmarks on the work by Vater et al. [136] and extend it by test cases highlighting the limiting approaches of both methods. We start with standard benchmarks for well-balancedness (Section 3.2) and the resolution of wetting and drying cells (Section 3.3). As stress test for the limiting approaches we look at a discontinuous problem with highly oscillating solution (Section 3.4) and a moving shoreline (Section 3.5). Finally we reproduce the Okushiri laboratory experiment (Section 3.6) and the Sumatra-Andaman tsunami (Section 3.7).

3.2 The resting lake scenario

The first scenario we use to validate our models is the resting lake scenario. This test is the base of all further analysis. It ensures that for a flat sea surface and zero velocities, no spurious waves or flows are generated by the numerical scheme or its implementation. The benchmark tests the well-balancedness of the schemes.

We look at two versions of the benchmark, based on a smooth (Figure 3.1a) and a discontinuous bathymetry (Figure 3.1b). Both benchmarks are designed to mimic typical coast patterns of complex topographies.



(a) Smooth resting-lake **(b)** Discontinuous resting-lake

Figure 3.1.: Sketches for bathymetry and water level of the resting lake benchmarks.

In the domain $\Omega = [0, 1] \times [0, 1]$ the smooth bathymetry is defined by a Gaussian bell curve

$$b_1(x, y) = 0.25 \cdot \exp\left(-40.0 \cdot \|(x, y) - (0.5, 0.5)\|_2^2\right) - 0.1. \tag{3.1}$$

The discontinuous bathymetry consists of four stacked obstacles

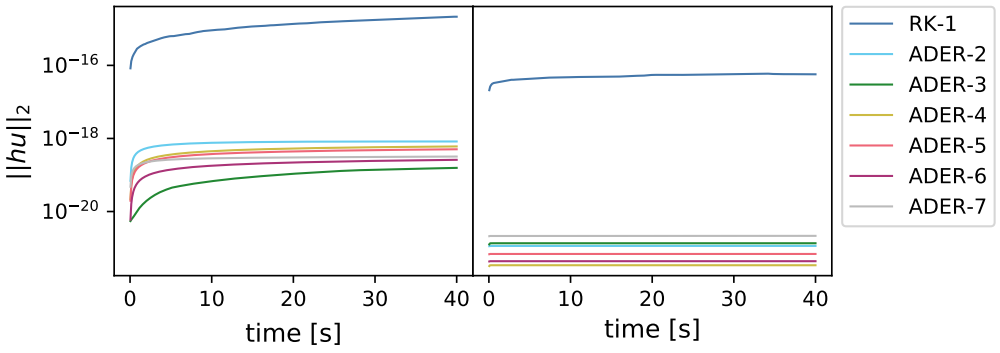
$$b_2(x, y) = \begin{cases} 0.02 & \|(x, y) - (0.35, 0.65)\|_2 < 0.1, \\ -0.05 & \|(x, y) - (0.55, 0.45)\|_2 < 0.1, \\ -0.03 & \|(x - 0.47)\| \wedge \|(y - 0.55)\| < 0.25, \\ -0.07 & \|(x, y) - (0.5, 0.5)\|_2 < 0.45, \\ -0.1 & \text{else.} \end{cases} \tag{3.2}$$

Velocities and water level are set to zero

$$h_0(x, y) = \max(0.0, -b_i(x, y)), \qquad hu_0(x, y) = hv_0(x, y) = 0. \tag{3.3}$$

We run the benchmark on a uniform mesh of 2^{10} cells for 40 s and measure the momentum every 50 ms. In all runs the dry tolerance is set to 1.0×10^{-9} m. To resolve the coast for the ADER-DG method, we set B_{coast} to -0.01 m.

Figure 3.2 (a) and (b) show the resulting L_2-norm of the momentum



(a) Smooth resting-lake (b) Discontinuous resting-lake

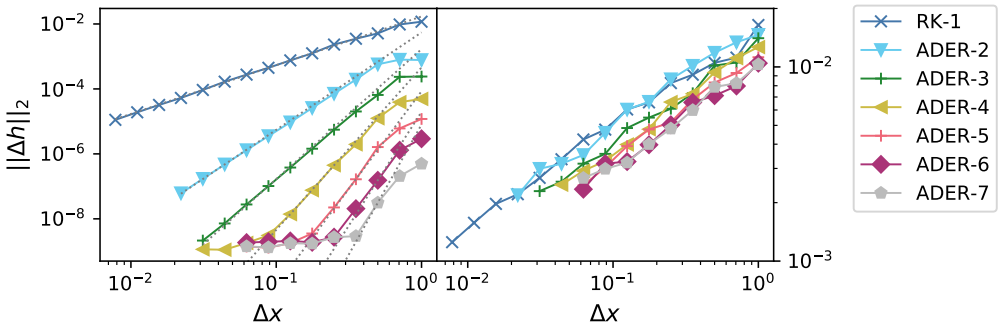
Figure 3.2.: L_2-norm of the momentum over the time of 40 s.

for both benchmarks of the Runge-Kutta method (RK-1) and the ADER-DG method from order 2 to 7 (ADER-2 to ADER-7). For all ADER-DG methods we see that the momentum increases in time with decreasing slope and remains below 1.0×10^{-18} in the smooth resting lake scenario. For the discontinuous resting lake the momentum stays constantly below 1.0×10^{-21}.[1] For RK-1, the momentum slowly increases for the smooth and the discontinuous resting lake and remains at values close to 1.0×10^{-15} after 40 s. In all cases the generated momenta are negligible. We can thus be certain that the solution stays constant for all methods. To test the spatial convergence rate of the methods, we perform the same benchmarks on grids from 2^2 up to 2^{15} cells and compute the L_2-norm for the error of the water column in the numerical approximation to its analytic reference.

In Figure 3.3, we show the resulting comparison of mesh size to L_2 error. In the smooth resting lake, we see that the dry tolerance of 1.0×10^{-9} m sets a lower bound of the error functions of all methods. For a DG method of order N, we reach a convergence rate of $N + 1$, hinted by the underlying dashed lines. While ADER-7 has reached the error limit for 2^6 cells and a cell size of $\Delta x \approx 0.088$, ADER-2 and RK-1 do not reach the limit even with 2^{15} cells.

In the discontinuous case, the convergence rate degenerates to linear convergence. Increasing the order of the method, while keeping the cell size constant, decreases the error. However no method reaches the convergence limit within the tested cell resolutions.

[1] This value lies below machine precision, as we compute the numerical error without considering the volume of single cells and later multiply it analytically.



(a) Smooth resting-lake (b) Discontinuous resting-lake

Figure 3.3.: L_2-norm of the absolute error of the water column for decreasing mesh sizes. Ideal convergence rates are hinted by dashed lines

3.3 A single wave on a sloping beach

To test how well the solvers are able to represent inundation, we compare against the one dimensional analytic solutions given by Carrier et al. [25]. Carrier solved the shallow water equations for several initial waves, propagating towards a linear sloping beach. The values of the initial waves are set such that "the wave does not brake", or in our terms no shocks are generated. The test shows how well the schemes are able to resolve wetting and drying cells. In case of the ADER-DG method, the benchmark also shows if the transition from DG to FV-cells introduces errors to the solution.

We compare the numerical results against a semi-analytic result set, which contains the water level and velocity at 160, 175 and 220 s, as well as the evolution of the shoreline from 90 to 350 s [71]. In the domain $x \in [-400, 50000]$ the corresponding set-up is given by a linear-sloping bathymetry and an initial wave

$$\begin{aligned} h_0(x) = 3.0 \exp\left(-0.4444 \cdot \left(\frac{x}{5000.0} - 4.1209\right)^2\right) + \\ 9.0 \exp\left(-4.0000 \cdot \left(\frac{x}{5000.0} - 1.6384\right)^2\right) - b(x) \qquad (3.4) \\ u_0(x) = 0.0 \qquad b(x) = -0.1 \cdot x. \end{aligned}$$

To solve the problem in two dimensions, we take the initial condition from Equation (3.4) and set it constant along the y-axis in a squared domain.

Table 3.1.: Mesh resolutions for the Runge-Kutta method (RK-1) and ADER-DG methods of order 2 to 7 (ADER-2 to ADER-7), for the single wave on a sloping beach benchmark.

Method	Cells	Δx_c		RK-1	2^{20}	24.6
ADER-2	2^{18}	16.4		ADER-5	2^{16}	16.4
ADER-3	2^{18}	12.3		ADER-6	2^{14}	28.1
ADER-4	2^{16}	19.7		ADER-7	2^{14}	24.6

In case of the RK method we use a mesh size $\Delta x = 25\,\text{m}$. To reach the same resolution on the coast, for each order of the ADER-DG method, we pick the mesh size Δx such that the sub-cell resolution Δx_c of the corresponding FV-method is close to 25 m. (The detailed mesh sizes are are listed in Table 3.1.) B_{coast} for the ADER-DG method is set to $-100\,\text{m}$.

In Figure 3.4 we show the resulting comparison for the water level, the momentum and the velocity at the coast. For water level and momentum all methods show an exact agreement with the analytic solution. The velocity is computed pointwise from momentum and water column in a post-processing step. We see that close to the shoreline values of the velocity highly differ from the analytic solution, which is caused by ill-condition of the velocity computation for water columns close to zero (see Section 2.5.2). In wet areas we get an accurate match for the velocity.

Figure 3.5 shows the recorded shoreline for all methods and compares them to the analytic solution. The stepwise pattern in the shorelines are a result of the mesh size and can be removed by further refining the coast. All methods match the reference equally well and capture the time of the largest inundation distance at 165 s as well as the trough at 220 s accurately. We can conclude, that the mechanisms for inundating coasts work equally well for all our numerical schemes.

3.4 The radial dam break scenario

To test and compare how well the Barth-Jespersen Limiter (Section 2.5.2) and the *a posteriori* FV sub-cell limiter (Section 2.6.2) resolve oscillations along shock-waves, we run the radial symmetric version of the one dimensional dam break problem. The one dimensional dam break problem is defined by an initial discontinuity in the water column and zero velocities.

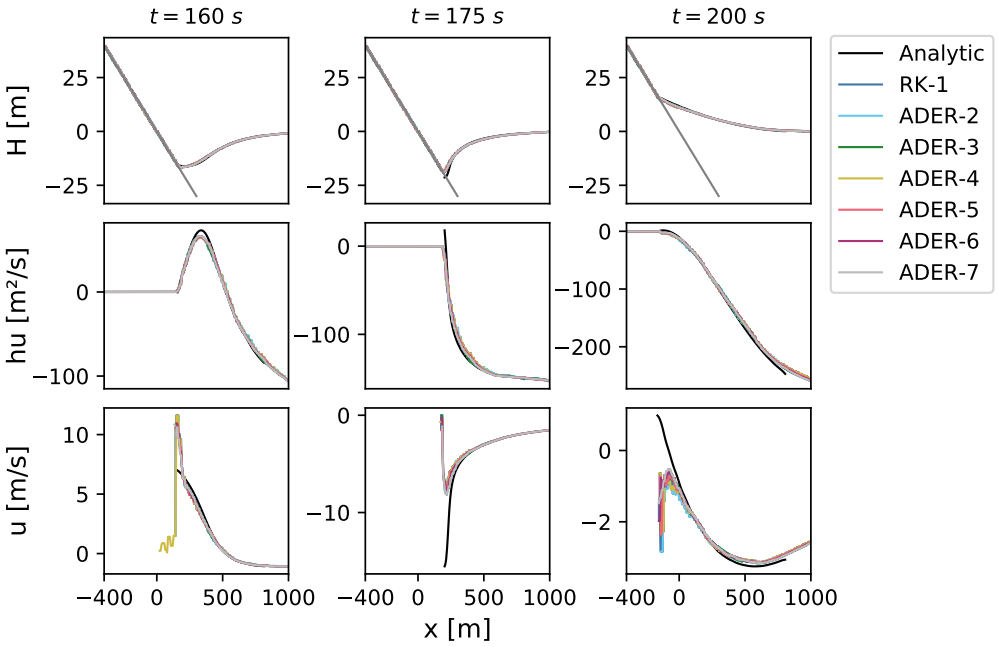


Figure 3.4.: Comparison of trajectories for the sloping beach benchmark.

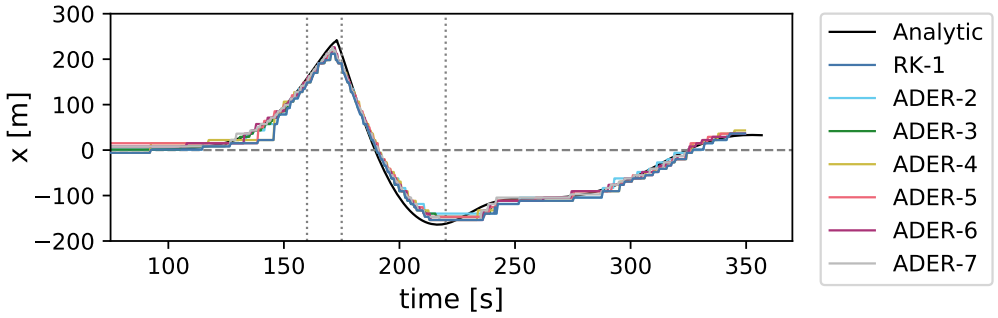


Figure 3.5.: Comparison of the shoreline for the sloping beach benchmark.

We use the values:

$$h_0(r) = \begin{cases} 4.0 & x < 0 \\ 7.0 & x > 0 \end{cases} \qquad (3.5)$$
$$hu_0(r) = 0.0.$$

Its analytic solution consists of three constant states, which are separated by a rarefaction and a shock wave.

To transform the initial condition to two dimensions, we consider it in polar coordinates and take Equation (3.5) along the radius, such that the jump is placed at $|r| = 5$. In its angle ϕ, we keep the condition constant,

$$q_0(r, \phi) = (h_0(r - 5), hu_0(r - 5)) . \qquad (3.6)$$

In the domain $\Omega = [-10, 10]^2$ we look at the numerical solution at $T = 0.3\,\mathrm{s}$, after which the solution has not yet reached the boundaries or the center of the domain.

As shown by Glaister [60], we cannot compare the two dimensional solution directly to the transformed one dimensional solution, as the change of variables introduces additional source terms. Instead we compare our results to a highly resolved reference simulation ($\Delta x = 0.005$), from a pure FV method on 2^{23} cells.

In Figure 3.6 we show a comparison of the water column and momentum along the line $\phi = 0$, $r \in [0, 10]$ at $T = 0.3\,\mathrm{s}$, of the FV reference (FV) against the second order RK method (RK-1) and the ADER-DG method for order 7. For the ADER-DG method we show the result of an unlimited simulation (ADER-U) and limited runs, where the DMP is checked only for the water-column (ADER-H) or for all quantities (ADER-Q).

Like the one dimensional analytic solution, the reference consists of a rarefaction wave (r) propagating towards $r = 0$ and a shock wave (s) propagating towards $r = 10$, which separate the constant states q_l, q_m, q_r. While q_l and q_r are the left and right initial value of the Riemann problem, the middle state q_m is the result of the conditions for the rarefaction wave and the Rankine-Hugoniot condition at the shock front [86]. In two dimensions the middle state becomes a linear sloping function connecting the rarefaction and shock wave.

The unlimited ADER-DG method ADER-U shows significant oscillations around the shock wave, as we would expected from Gibbs phenomenon. For RK-1 the Barth Jespersen limiter damps the oscillations, which is what we also see for both versions of the DMP in ADER-H and ADER-Q. The DMP

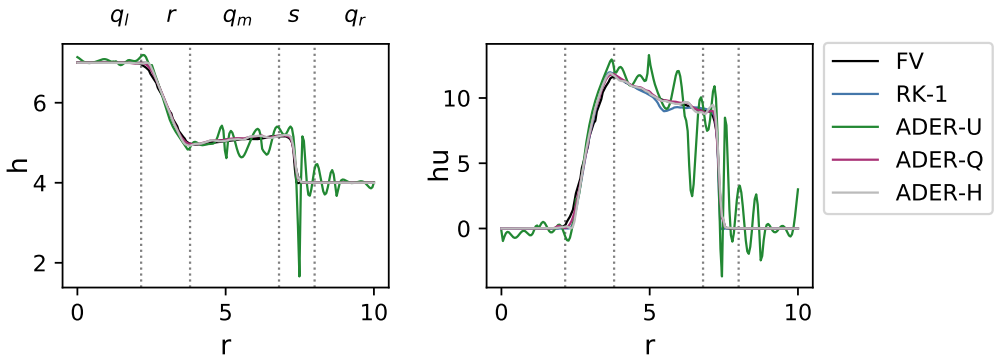


Figure 3.6.: Comparison of trajectories for the radial dam break problem.

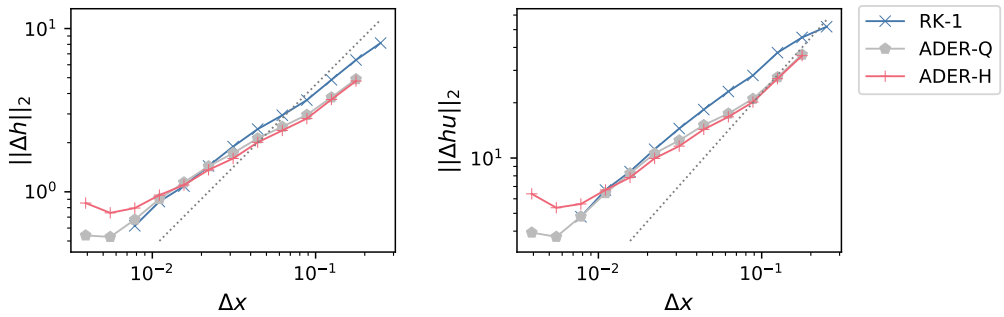


Figure 3.7.: Errors for the water height of limited and unlimited simulations of the radial dam break problem.

detects oscillations in both cases, such that the FV-method is able to repair them.

In order to quantify how well the limiters repair the convergence of the method, we show an error analysis for the water column and momentum for different mesh resolutions in Figure 3.7, for all three limiting approaches. For each method we plot the L_2-norm of the difference to the FV reference, against the cell resolution. We see that all methods approach the reference solution, until the error is bound by the accuracy of the numerical reference.

3.5 The oscillating lake scenario

The oscillating lake scenario is part of the periodic two dimensional analytic solutions found by Thacker [124]. It is characterized by a periodically moving water droplet, which propagates through a dry parabolic basin. Until this point we only looked at benchmarks where the shoreline is restricted to a known predefined area. In this benchmark the shoreline is the boundary of the droplet and moves constantly through the domain. For the ADER-DG method this implies that we cannot use a predefined region to constantly solve the shoreline with a FV method and that we have to rely on the DMP and the divergence of the Predictor to track the shoreline.

The solution to the test-case is given in closed analytic form, which allows us to use it to study the errors of our methods,

$$\begin{aligned} h(x,y,t) &= \max\left(0, 0.1 \cdot \left(x \cdot \cos\left(\omega t\right) + y \cdot \sin\left(\omega t\right)\right) - b(x,y)\right), \\ hu(x,y,t) &= \max\left(0, h(x,y,t)\right) \cdot \frac{\omega}{2}\left(-\sin\left(\omega t\right)\right), \\ hv(x,y,t) &= \max\left(0, h(x,y,t)\right) \cdot \frac{\omega}{2}\left(-\cos\left(\omega t\right)\right), \\ b(x,y) &= 0.1 \cdot \left(x^2 + y^2\right). \end{aligned} \tag{3.7}$$

The circular frequency $\omega = \sqrt{0.2\mathrm{g}}$ is set such that it takes a period of $T_P = 2\pi\omega^{-1} \approx 4.487\,\mathrm{s}$ for the droplet to propagate through the domain $\Omega = [-1,1]^2$.

In Figure 3.8 we show the water column of the droplet for a whole period, simulated with the ADER-DG method of order 2, on a uniform mesh of 2^{18} cells at $t = 0$, $1/3\,T_P$, $2/3\,T_P$ and T_P. The dry tolerance is set to $1.0 \times 10^{-6}\,\mathrm{m}$ and we let the DMP detect cells with a water column of less than $1.0 \times 10^{-2}\,\mathrm{m}$, such that the shoreline is constantly detected and limited (see the red contour in (a), (b), (c), (d)). Our scheme conserves radial symmetry and matches the position of the shoreline to the reference. However, the wetting and drying process introduces small perturbations to the solution which can be seen in the black contour lines in (c) and (d).

To study the convergence behavior, we show the L_2-errors of the water column and momentum on uniform meshes of 2^5 to 2^{18} cells in Figure 3.9, for the ADER-DG method form order 2 (ADER-2) to order 7 (ADER-7) and the Runge-Kutta (RK-1) method. The dashed lines hint linear and quadratic convergence orders. For the ADER-DG methods we can reach linear convergence, which is what we would expect from the Godunov scheme that we use in the limiting process. Higher orders reduce the error,

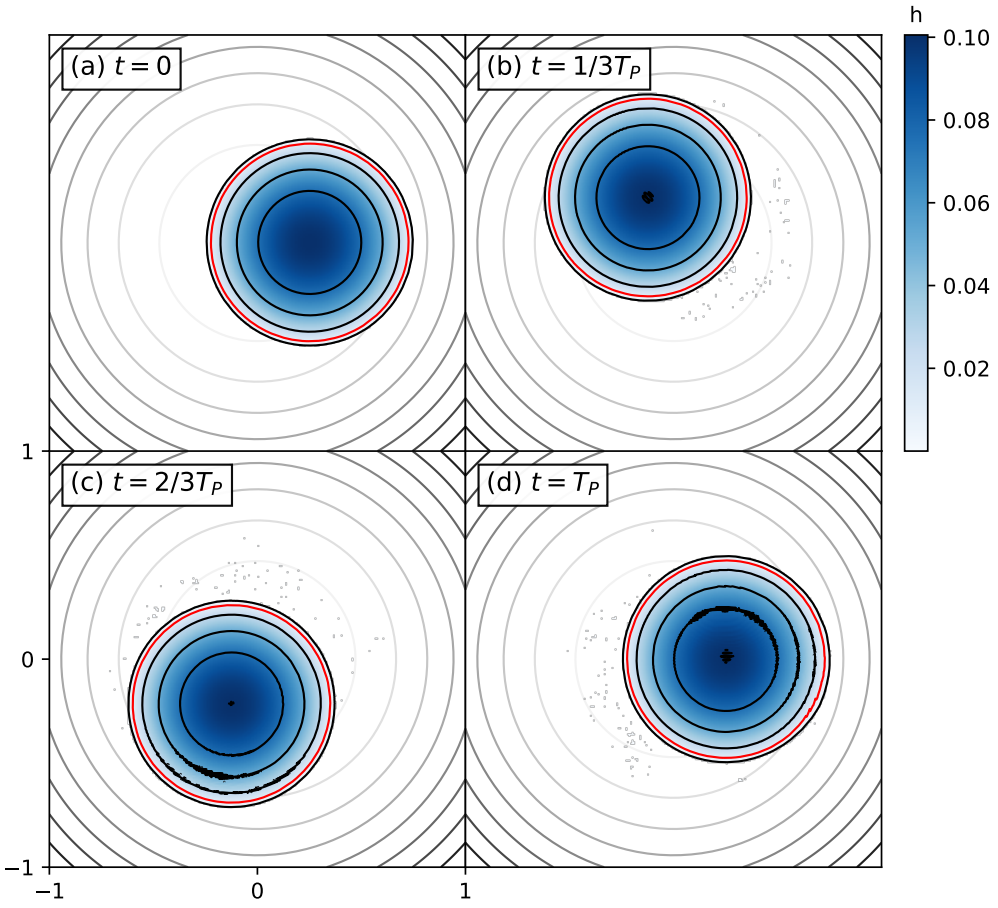


Figure 3.8.: Propagation of the oscillating lake after $t = 0$, $1/3T_P$, $2/3T_P$ and T_P. The bathymetry is hinted with black contour lines. The DMP activated for cells with a water column of less than 1.0×10^{-2} m is hinted by the red contour.

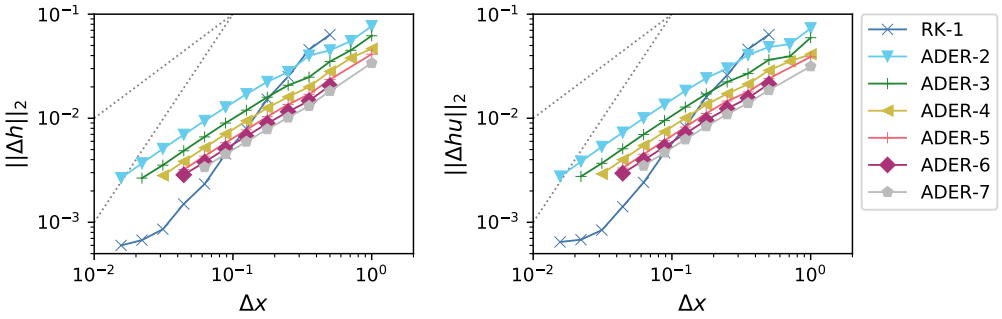


Figure 3.9.: L_2-errors of the water column and momentum of the oscillating lake for decreasing mesh sizes. The dashed lines hint the slope for linear and quadratic convergence rates.

which we can relate back to the lower sub-cell resolution in the FV-methods. The limiting of the RK method allows to reach second order convergence and shows a smaller error than the ADER-DG methods for a mesh of 2^{14} cells.

3.6 Okushiri: Wave on a complex bathymetry

The first benchmark with complex topography we look at is based on a physical experiment, reproducing the tsunami subsequent to the Hokkaido-Nansei-Oki earthquake in a small scale model. Matsuyama et al. modeled the bathymetry of the region in a flume and induced the tsunami by a wave generator on the left side of the model [93].

We use this benchmark to test how well our methods are able to resolve inundation in case of a complex topography.

Figure 3.10 (b) shows the set-up, which is at a scale of 1/400 of the original domain and the wave profile (a) that we set as boundary condition along the left side of the domain $x = 0$. As sam(oa)^2 is restricted to squared domains, we extend the rectangular domain of the set-up from $[0.0, 5.5] \times [0.0, 3.5]$ to $[0.0, 5.5] \times [-2.0, 3.5]$ and use constant extrapolation to define the bathymetry for $y < 0$. This allows us to reproduce the tsunami and its back propagation from the coast, reflections from the lower boundary of the fume are shifted in time.

The tsunami in the experiment was recorded at three gauges, placed at $(x, y) = (4.527, 1.196)$, $(4.521, 1.696)$ and $(4.521, 2.196)$ in front of the

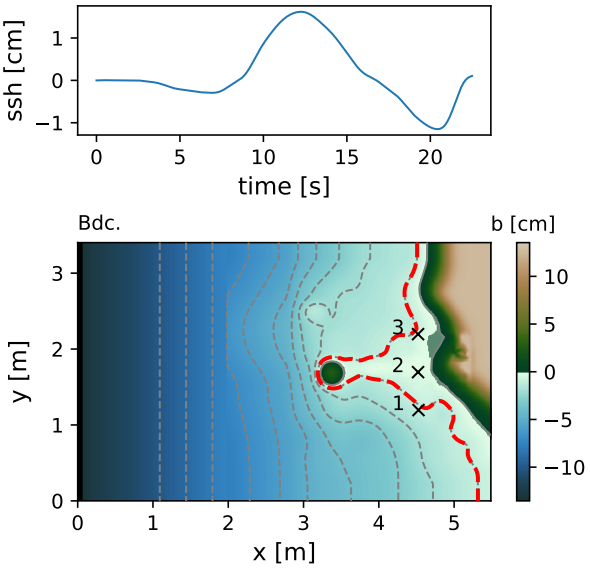


Figure 3.10.: Incoming wave (top) and bathymetry of the Okushiri benchmark (bottom). The incoming wave is set as boundary condition on the left of the domain. The three gauges, labeled 1,2 and 3 are placed in front of the coast. For the ADER-DG method the threshold B_{coast} from Equation (2.75) is hinted by the red dashed line.

coastline, which record the propagation of the tsunami towards the coast, as well as its propagation back, after it inundated the coast. Additionally the experimental shoreline is provided, which we manually reproduced from [136].

We use the ADER-DG method of order 7 (ADER-7) and the Runge-Kutta method (RK-1) and set a dry tolerance of 1.0×10^{-5} m on meshes with 2^{14} cells for ADER-7 (≈ 1.8 Mdofs) and 2^{20} for RK-1 (≈ 2.4 Mdofs). For ADER-7 we set B_{coast} to -1.0 m (see the red line in Figure 3.10). To capture the tsunami and some time after the inundation, we simulate the benchmark for the time of 40 s.

In Figure 3.11 we show the inundation of the coast for the RK-1 method after 15.0 s, 15.5 s, 16.0 s, 16.5 s and 17.0 s. The shoreline from the experiment is added as white line, with which the numeric results shows a good agreement. Between 15.5 s and 16.0 s the tsunami is reflected at the coast. The superposition of incoming and outgoing wave lead to an increased amplitude.

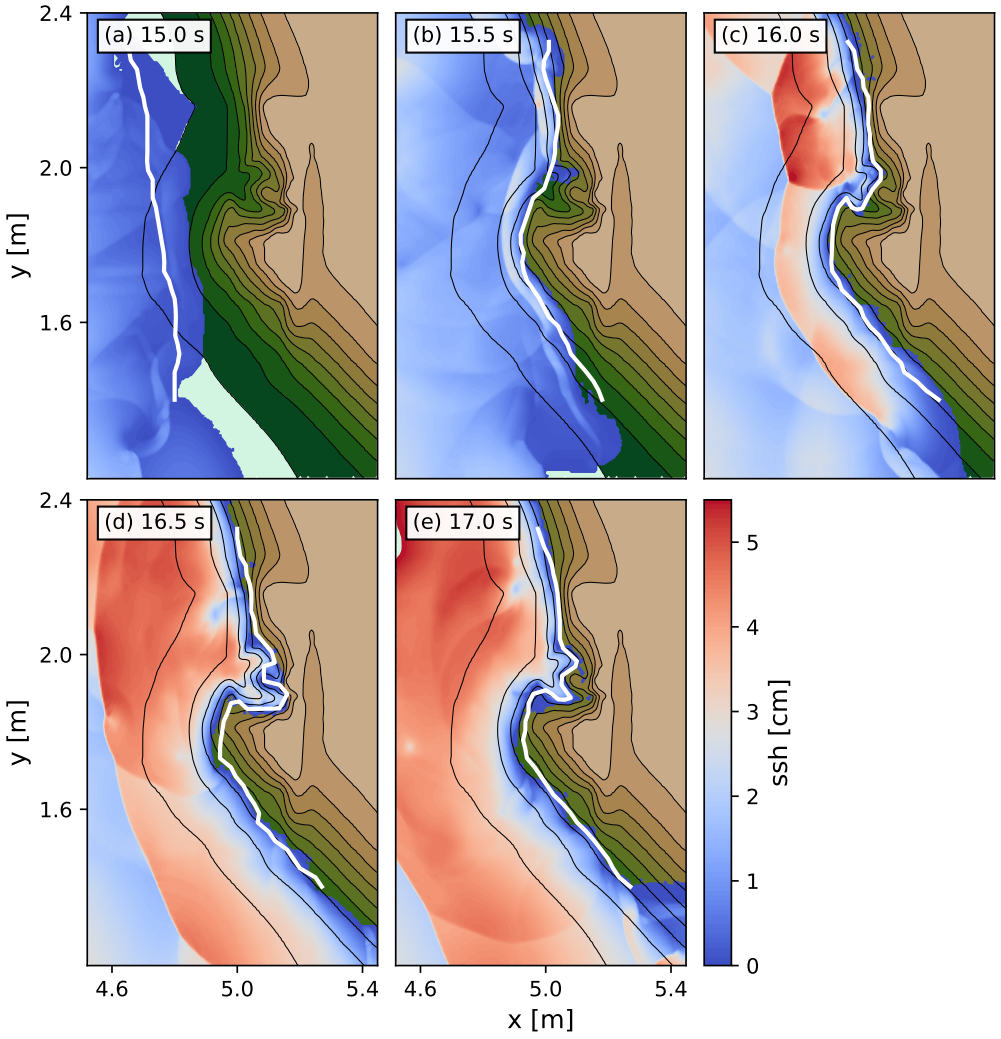


Figure 3.11.: Inundation of the coast at 15.0 s, 15.5 s, 16.0 s, 16.5 s and 17.0 s. The white line describes the experimental shoreline.

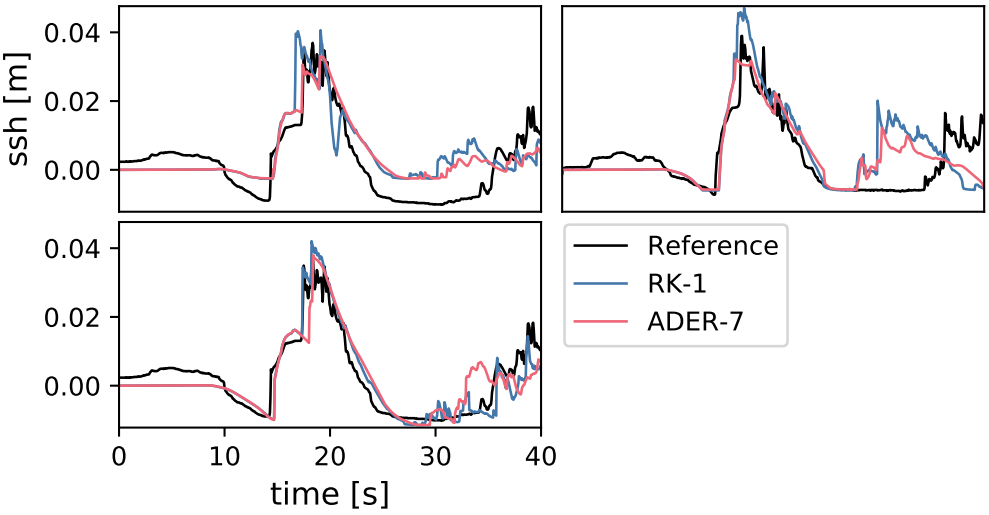


Figure 3.12.: Record of the wave for the three gauges placed at $(x, y) = (4.527, 1.196)$, $(4.521, 1.696)$ and $(4.521, 2.196)$.

This superposition is present in the record of three gauges as the highest amplitude, in Figure 3.12 after $\approx 17\,\mathrm{s}$. Naturally we see differences between experimental and numerical records, however the amplitude and phase of the main wave is captured accurately by both methods. After 35 s the numeric results differ from the record, which is caused by the extrapolation of the domain we performed for the set-up.

As large areas of the coast are inundated by the tsunami, we expect to see the same effects on the velocity we saw for the linear sloping beach in Section 3.3. We want to use this in order to compare both limiting approaches for their effect on the wave speeds ($u \pm \sqrt{gh}$ and $v \pm \sqrt{gh}$) and with that on the maximal admissible time-step size (see Section 2.4.2).

Figure 3.13 shows the recorded maximal wave speeds for both methods. Up to the first inundation at 17 s both cases show low maximal wave speed, close to $5\,\mathrm{m\,s^{-1}}$. After that, we see a significant increase of the maximal wave speed for ADER-7, which remain at a high level around $40\,\mathrm{m\,s^{-1}}$, with outliers of up to $300\,\mathrm{m\,s^{-1}}$. While the waves-speeds also increase for RK-1, they are significantly smaller after this point. ADER-7 is limited at the coast with no special mechanism to reduce wave speeds, RK-1 on the other hand runs with a dedicated limiter to reduce them.

The consequence can be seen in the number of time-steps required in both methods to reach 40 s. While ADER-7 has fewer degrees-of-freedom (≈ 1.8

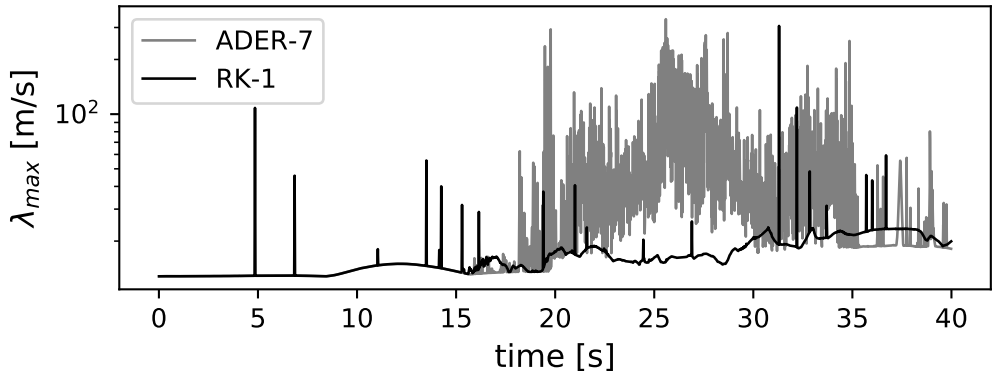


Figure 3.13.: Maximal waves speeds over time for the ADER-DG method of order 7 (ADER-7) and the second Runge-Kutta method (RK-1).

Mdofs) and requires 120 600 time-steps, RK-1 has more degrees-of-freedom and requires only 92 320 time-steps (≈ 2.4 Mdofs).

3.7 The Sumatra-Andaman tsunami

Finally, we look at the application of both methods to reproduce an actual tsunami event. The Sumatra-Andaman Earthquake occurred 2004 in the Indian ocean, causing a subsequent tsunami which killed around 283,000 people in the surrounding areas, including Indonesia, Sri Lanka and India. The epicenter of the earthquake was placed in the Sumatra subduction zone, close to the Nicobar islands and had a moment magnitude M_w of 9. The earthquake has been numerically reproduced several times, for example with the SeisSol software package [128, 132]. We use the source of Ulrich et al. which is based on weak sediments and has a maximal displacement of 33 m [128]. In order to incorporate the earthquake as initial condition for the tsunami, we postprocess the earthquake data with the coupling pipeline from Chapter 5. The tsunami is sourced time-independent, as introduced in Section 5.1. Hence, the initial condition of the test case is the post-processed displacement field of the earthquake simulation and applied as initial perturbation of the sea-surface, all velocities are set to zero (see Figure 3.14 a).

Figure 3.14 shows the evolution of the tsunami for an ADER-DG method of order 2 on a mesh of 2^{17} cells. B_{coast} is set to -500 m (see the blue contour in Figure 3.14 b), with a dry tolerance of 0.1 m.

Table 3.2.: Mesh configurations for the Sumatra-Andaman Tsunami, to reproduce the Jason-1 cross section reference with a relative error of $< 25\%$.

Method	Cells	Dofs		RK-1	2^{19}	12.5e6
ADER-2	2^{16}	3.1e6		ADER-5	2^{13}	1.4e6
ADER-3	2^{15}	2.6e6		ADER-6	2^{13}	1.8e6
ADER-4	2^{14}	1.95e6		ADER-7	2^{12}	1.2e6

The resulting tsunami arrives at the Indonesian coast after 15 min and reaches Sri-Lanka after 2 h (Figure 3.14 c). Close to the arrival in Sri-Lanka the Jason-1 Satellite recorded a cross section of the tsunami, which we use to verify our numerical results (see the black line in Figure 3.14 c). Because of the model-error and noise in the recorded data, we cannot expect a perfect match with our numerical result. Instead we compute a high resolved reference solution to the problem (RK-1 on 2^{22} cells) and evaluate the relative error between the cross section of the reference and our models.

The resulting relative errors ϵ_{rel} are plotted in Figure 3.15. We see that we lose the high order convergence of the error in all cases. However, for constant mesh size all ADER-DG methods show a smaller error than the RK-DG method and with increasing polynomial order the error decreases. The number of cells and the number of dofs with which we can achieve a relative error of $< 25\%$ are summarized in Table 3.2.

While RK-1 requires a mesh of 12.5×10^6 dofs to reach this relative error, ADER-7 requires around ten times fewer (1.2×10^6). For the chosen mesh configurations, Figure 3.16 shows a comparison of the cross sections against the Jason-1 record. All simulations are able to capture the location and amplitude of the two peaks at latitude $-5°$ and $-3°$ as well as the dip at $-2°$.

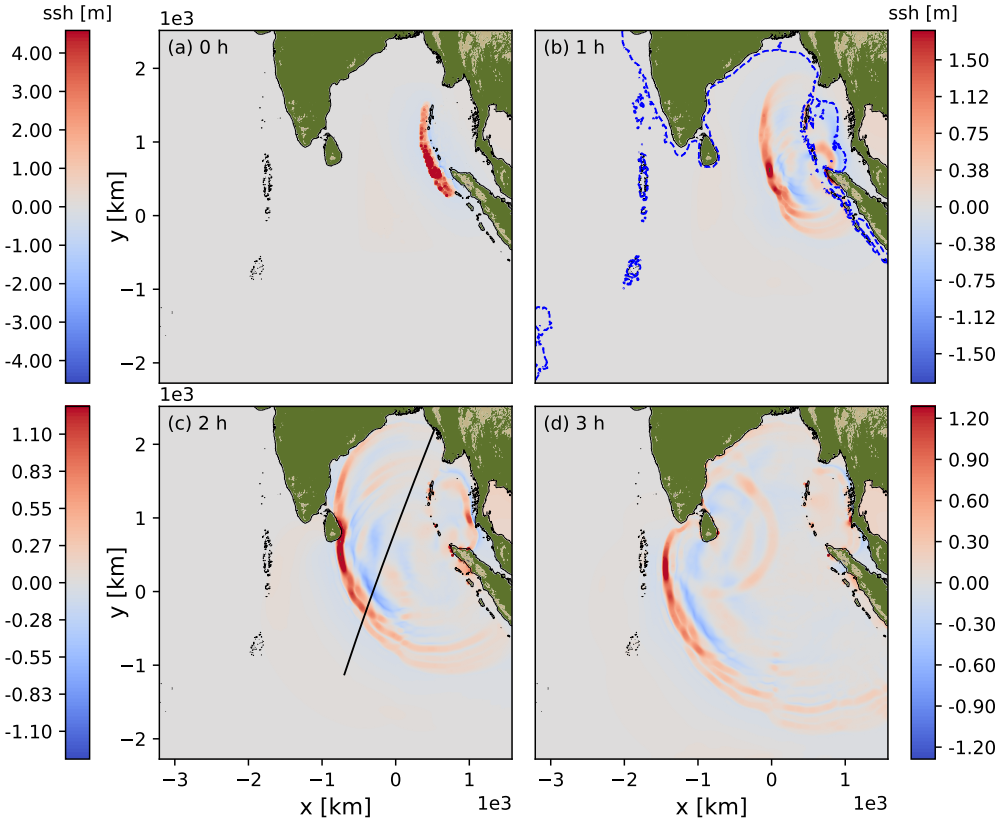


Figure 3.14.: Evolution of the Sumatra-Andaman tsunami in the Indian ocean for a simulation with the ADER-2 method. At initialization (a), after 1 h (b), 2 h (c) and 3 h (d). In (b) we mark the coastal-area, that is resolved with FV-cells (blue dashed contour). The trajectory of the Jason-1 Satellite is added in (c) (black curve).

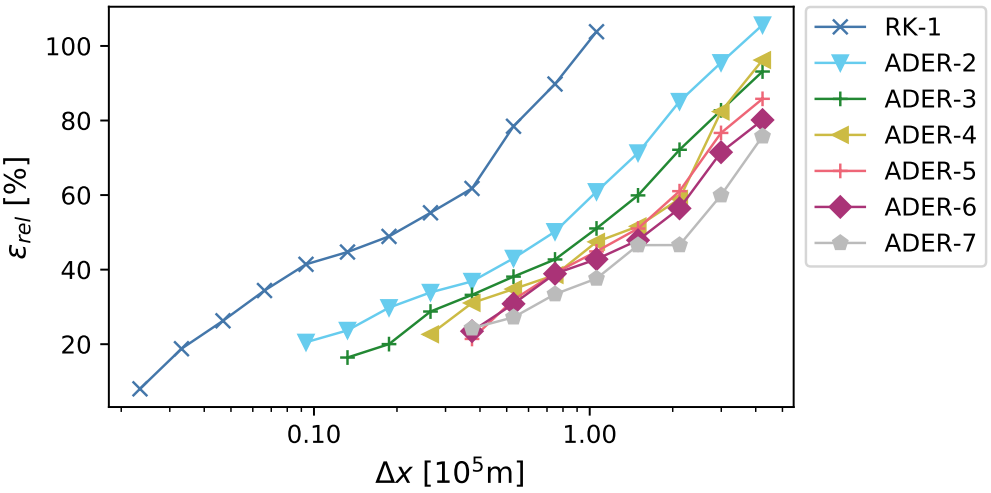


Figure 3.15.: Relative errors for decreasing mesh resolution, for the methods RK-1 and ADER-2 to ADER-7.

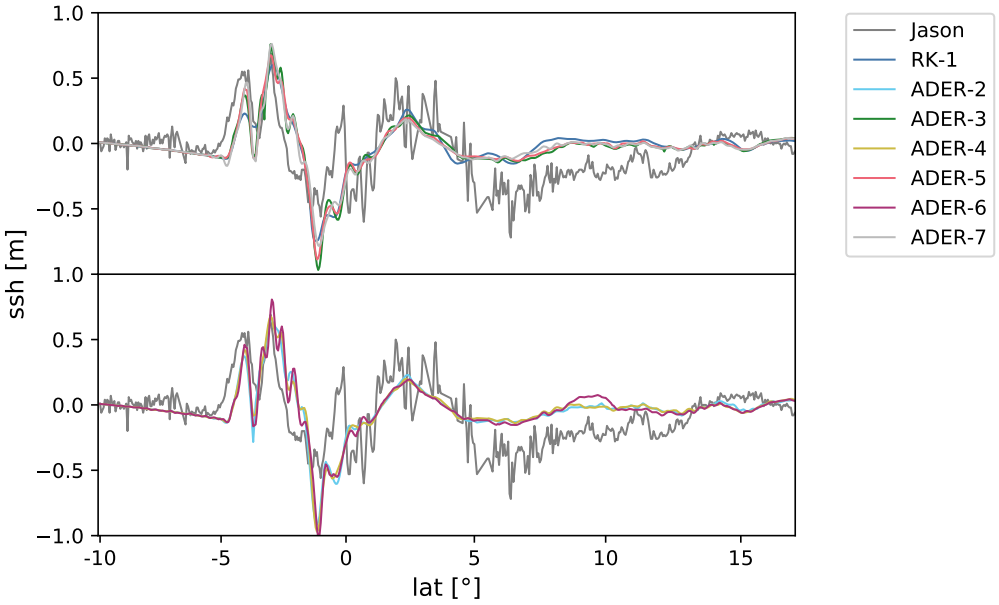


Figure 3.16.: Comparison of the cross sections, for the mesh configurations from Table 3.2.

CHAPTER

4

Comparing time-to-solution for Runge-Kutta and ADER-DG methods.

4.1 Introduction

In the last chapter we evaluated how well the RK-DG and ADER-DG methods solve typical test cases, which are crucial for the simulation of tsunamis, and finalized our analysis with the reproduction of the Sumatra-Andaman tsunami. In this chapter we add an analysis and comparison of the computational cost of both methods and test how well an implementation in the sam(oa)^2 framework can exploit modern CPU architectures. Tsunami simulations are two-dimensional and the size of a single simulation is small compared to the performance that is available on current supercomputing systems. However, in Probabilistic Tsunami Hazard Analysis (PTHA) [61] and Uncertainty Quantification [115] we often have to perform millions of distinct simulations, such that initially small performance issues can suddenly grow significant. In this context we want to compare time-to-solution for the second order Runge-Kutta method and the ADER-DG method.

Previous comparisons for time-to-solution have been performed for the linear ADER-DG method for seismic simulations [20] or for matrix-free DG methods for incompressible flows [6, 47]. While these papers restrict their evaluation of time-to-solution to one numerical method and vary its convergence order, we extent the analysis by a dimension and also compare between our two numerical methods.

In order to compare time-to-solution we look at the ratio of the accuracy, i.e. the inverse of some error norm, that a method can reach and the amount of time invested [47]. We split this ratio in a factor accounting for the accuracy of the spatiotemporal approximation and a factor accounting for the time our implementation of the model requires to evolve this approximation by one time-step

$$\frac{\text{accuracy}}{\text{time}} = \frac{\text{accuracy}}{\text{dofs} \cdot \text{time-steps}} \cdot \frac{\text{dofs} \cdot \text{time-steps}}{\text{time}}. \tag{4.1}$$

As we want to look at both factors separately, we assume that the implementation has no significant effect on the accuracy of the method. This means, we neglect small differences that might occur when we change the order of operations or use a vectorized fused multiply-add. While the first factor was the subject of the last chapter, we add an analysis of the second term in this chapter. Finally we use the results of this analysis and conclude on the time-to-solution for three representative benchmarks: We choose the smooth resting lake benchmark as we can use it as a baseline for all other test-cases. The second benchmark is the oscillating lake, as its moving shoreline is a challenging task for both limiting approaches. To assess the application on a real world example, we finally look at the simulation of the Sumatra tsunami.

The second term in Equation (4.1) is the time our implementation requires to propagate a degree of freedom by one time-step, which we call time-per-dof. In our particular case it depends on several factors:

- The bandwidth of the underlying sam(oa)^2 framework.
- The number of bytes each method requires to represent a dof.
- The number of mesh traversals each method requires to propagate by one time-step.
- The arithmetic intensity of the main kernels.
- The implementation and hardware optimizations we can perform for each method.
- Load-imbalance that can appear due to mesh refinement.
- In the case of the ADER-DG method load-imbalance also appears as the limiter is not activated in each cell and the number of iterations in the Picard loop is not fixed.

In order to reduce complexity and be able to focus on the comparison of the methods, we restrict our analysis in several points: We fix the used hardware resources to a single socket on the SuperMUC-NG cluster, such that we can avoid NUMA effects. As we explicitly want to include load-imbalance as characteristic of the numerical schemes, we deactivate all load-balancing mechanisms in sam(oa)^2. Finally we restrict meshes to the subset of uniform meshes, as we cannot be certain that we end up with the same dynamic mesh refinement for different methods using the same refinement strategy. The shape of the mesh has a high influence on the cost of the stack and stream approach which makes it challenging to compare the runtime of simulations performed on different meshes.

In order to abstract properties of the underlying sam(oa)^2 framework in our analysis, we develop a model to assess the bandwidth we can reach for each method in Section 4.2. As we want to consider the effect of the bandwidth of the sam(oa)^2 framework separately, we split the second term in Equation (4.1) into the time we require to process a byte in a single traversal, the number of traversals required for one time-step, and the number of bytes a method needs to represent a dof

$$\frac{\text{dofs} \cdot \text{time-steps}}{\text{time}} = \frac{\text{bytes} \cdot \text{traversals}}{\text{time}} \cdot \frac{\text{dofs}}{\text{bytes}} \cdot \frac{\text{time-steps}}{\text{traversals}}. \tag{4.2}$$

The resulting time per degree of freedom is assembled in Section 4.5.

In Section 4.4 we look at the arithmetic intensity and recapture the hardware-aware optimizations we performed for all methods and look at their effect. Finally we use all previous results and perform an overall comparison of time-to-solution in Section 4.6.

4.2 A memory-bandwidth model for sam(oa)^2

When we traverse the mesh in sam(oa)^2 additional costs are added to the pure memory transfers by the cell-stream and edge-stack we introduced in Section 2.2.1. In this section, we take a look at the bandwidth we can reach with this approach.

In our approach we follow the STREAM Triad benchmark [94]: The benchmark determines the maximum attainable bandwidth of a system by repeatedly performing the operation:

$$a_i = b_i + s \cdot c_i, \tag{4.3}$$

on three double-precision arrays a, b, and c and scalar s. The array sizes are chosen large enough such that all cache effects that might happen in the repetition are negligible. We know for this problem that the data transfers are the only important factor for the wall-time, as all floating point operations are overlapped by the dominant load and store instructions [120].

In order to test the available bandwidth in sam(oa)^2 we adapt the benchmark and persistently store three arrays on each cell in a uniform triangular mesh. Each array has a payload of d doubles, on which we perform the triad element-wise. We create a theoretical model to describe the memory bandwidth by splitting the wall-time in three components: With t_d we represent the time we require to perform the computation of the triad for a single index i. In t_d we consider the time spent for memory transfers as if a, b, and c where stored on a static array and ignore any effects of the cell stream or the underlying application. t_d is equivalent to the inverse of the available bandwidth B_{avail} [B s^{-1}], which we can reach with the original STREAM Triad benchmark

$$t_d = \frac{3 \cdot 8}{B_{avail}} \text{ [s]}. \tag{4.4}$$

To account for the cost of the traversal through the cell stream we define a time t_c required to load and store single cells without payload. In t_c we accumulate all latencies that occur for pointer resolutions in the SSFC traversal and for the memory transfer of metadata, like coordinates and transformations. Finally we add a constant time t_p that accumulates all operations in sam(oa)^2 that happen independent from the cell stream or its payload, as the initialization of data-structures or the synchronization of threads.

For a problem size of c cells, each with a payload of d doubles per array, we transfer $3 \cdot 8 \cdot c \cdot d$ bytes in every traversal. The required time in our model is $t_p + c \cdot t_c + c \cdot d \cdot t_d$. We get a theoretical bandwidth of

$$\text{bandwidth}(c, d) = \frac{3 \cdot 8 \cdot c \cdot d}{t_p + c \cdot t_c + c \cdot d \cdot t_d}. \tag{4.5}$$

We can see the most important characteristics of this model if we look at what happens for large numbers of cells $c \to \infty$ and a large payload $d \to \infty$,

$$\lim_{c \to \infty} \text{bandwidth}(c, d) = \frac{d \cdot 24}{t_c + d \cdot t_d} = B_{avail} \cdot \frac{d \cdot t_d}{t_c + d \cdot t_d}, \tag{4.6}$$

$$\lim_{d \to \infty} \text{bandwidth}(c, d) = \frac{24}{t_d} = B_{avail}. \tag{4.7}$$

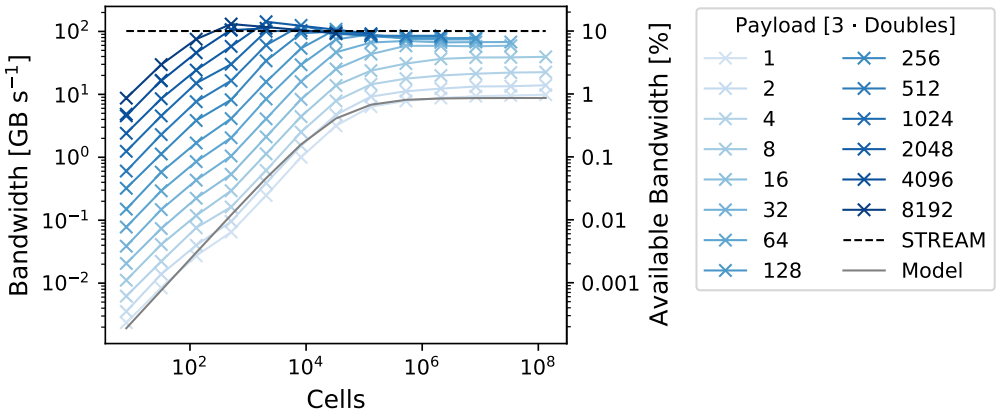


Figure 4.1.: Bandwidth for the STREAM triad in sam(oa)². We measure the bandwidth for increasing payload and cell number. As reference, we add the bandwidth for the theoretical model (Model) for a payload of $d = 1$ and the bandwidth reached by the original STREAM triad (STREAM).

For small payloads, we cannot reach the available bandwidth by scaling up the number of cells, as the constant factor accounting for the cell stream t_c remains. With larger payloads, the factor becomes small compared to the time required to process the doubles in each cell $d \cdot t_d$. The larger the payload in each cell, the closer the bandwidth we can reach is to the available bandwidth of the system. Finally the contribution of the constant factor t_d decreases faster for high payloads such that we can reach the available bandwidth with less cells than for lower payloads.

To verify the model we perform the benchmark on a single socket on SuperMUC-NG, with 24 OMP threads that are pinned to 24 cores. The available bandwidth we measure with the STREAM Triad is $101.5\,\mathrm{GB\,s^{-1}}$. We perform the benchmark with exponentially increasing payload $d = 2^0, 2^1, 2^2, 2^3, ..., 2^{15}$ on uniform meshes with exponentially increasing amount of cells $c = 2^3, 2^4, 2^5, ..., 2^{17}$. The resulting measured bandwidth for all settings is presented in Figure 4.1. For comparison we add the model Equation (4.5) for a payload of $d = 1$ which we fit with timings $t_c = 2.5\,\mathrm{ns}$ and $t_p = 10.0\,\mathrm{ms}$.

The results show a very good agreement with our model. Hence, we can increase the maximum attainable bandwidth with an increased payload. For high number of cells ($\geq 2^{19}$), the contribution of the cell-stream-independent parts t_p becomes negligible ($< 5\%$) for all payloads.

Payload on edges

Before we start to estimate the maximum attainable bandwidth for our numerical models, we have to include the influence of data transfers over edges to our consideration.

In order to solve Riemann problems between adjacent cells, we have to persistently store projections of their degrees-of-freedom and some additional information on the shared edge. Next we have to compute fluctuations, that we use to advance the dofs in each cell.

As not all edges in the mesh are stored on the edge stack, we cannot estimate a constant cost for edges as we did for cells in Equation (4.5). Instead, we measure the influence of edges and their payload by a series of benchmarks. Equivalently to the triad we performed for the cell stream, we persistently store three arrays a_i^e, b_i^e, and c_i^e, each with an edge payload d^e.

We emulate the projection from cell to edge, by broadcasting all cell arrays onto the edge.[1]

$$a_i^e = a^c_{(i \bmod d^e)}, \tag{4.8}$$

and do this equivalently for b_i^e and c_i^e.

Instead of solving a Riemann problem on the edge we simply exchange the three arrays from the left and right representation to set the arrays a_i^u, b_i^u and c_i^u for the updates of the left and right cell (denoted by the subscripts l and r):

$$_l a_i^u = {}_r a_i^e, \quad {}_r a_i^u = {}_l a_i^e. \tag{4.9}$$

We finally add the update to the arrays we stored in each cell:

$$a_i^c = a_i^c + a^u_{(i \bmod d^e)}. \tag{4.10}$$

The copy operations we perform are memory bound. (We see them as our version of the STREAM Move benchmark.)

In order to benchmark the cost of an edge payload d_e, we set fixed cell payloads d_c. We use $d_c = 6, 59$, and 348, which correspond to the payload on a cell for the methods RK-1, ADER-2 and ADER-7, as we see in the next section. Data transfers and computations over edges are additional effort we have to perform to advance the solution in a cell. As the payload on edges does not contribute to the number of dofs we can represented in a cell, we only consider the additional time required for the transfers and neglect the transferred bytes in the computation of the bandwidth.

[1]Note that the superscript c is introduced to mark arrays that belong to cells

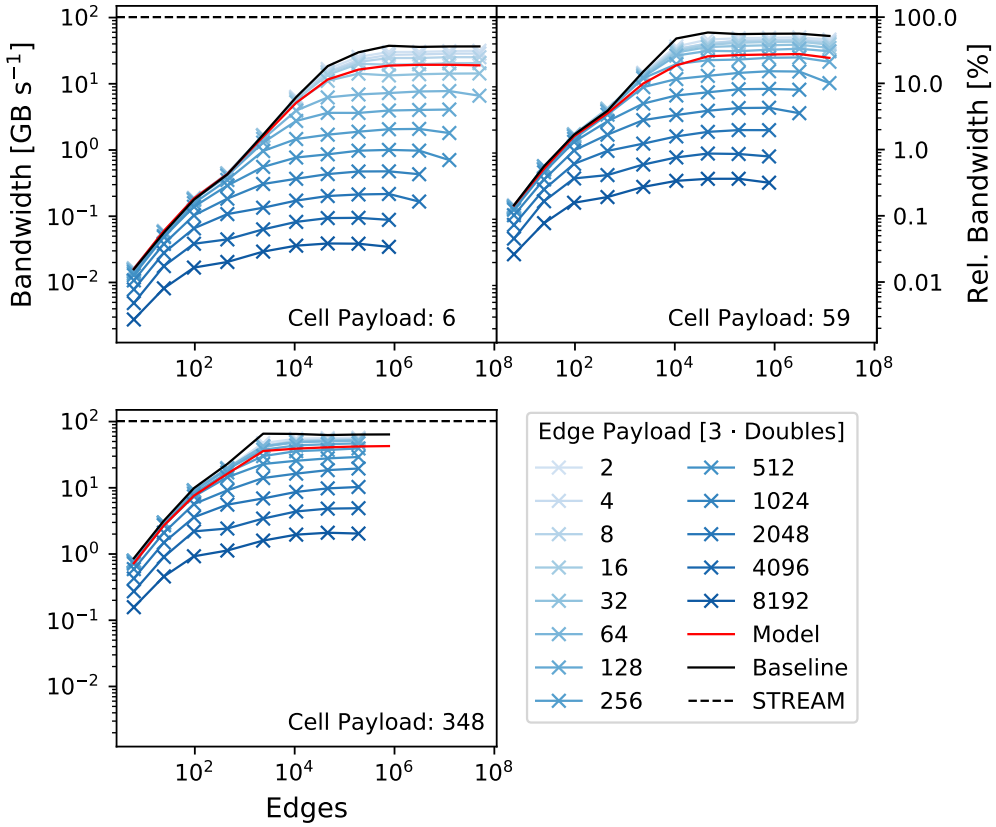


Figure 4.2.: Measured bandwidth for the edge STREAM benchmark in sam(oa)² for cell payloads of $d^c = 6, 59$, and 348. We increase the number of edges in the mesh and the edge payload. As reference we add the baseline for zero edge payload (Baseline) and the bandwidth of the original STREAM triad (Stream).

In Figure 4.2 we present the resulting bandwidth including the baseline for a cell-only benchmark as we performed it in the previous chapter. We see that for low edge payloads the bandwidth comes close to the bandwidth of a traversal that only considers cells. With increasing edge payload, the maximum attainable bandwidth decreases exponentially. The results show two significant effects for our analysis, opposing each other: A high order polynomial representation results in an increased payload on cells, for which we expect a higher bandwidth. However, higher order also leads to more bytes we have to store on each edge, which again decreases the maximum attainable bandwidth.

Table 4.1.: Number of bytes and dofs that are stored on each cell and edge for the methods RK-1 and ADER-2 to ADER-7 (left). Measured maximal bandwidth for the corresponding benchmarks STREAM-1 to STREAM-7 (right).

Method	dof/cell	B/cell	B/edge	Model	$B_{\max}$ [GByte/s]
RK-1	12	168	96	STREAM-1	22.9
ADER-2	24	1856	400	STREAM-2	37.9
ADER-3	40	3088	544	STREAM-3	38.4
ADER-4	60	4632	688	STREAM-4	39.7
ADER-5	84	6488	832	STREAM-5	40.9
ADER-6	112	8656	976	STREAM-6	42.0
ADER-7	144	11136	1120	STREAM-7	43.1

Bandwidth of the numerical models

In order to determine the bandwidth we can reach for each of our methods, we set the payload on cells and edges to the requirements of the numerical methods. This way we ensure that the benchmarks perform the same number of memory transfers in each iteration as their corresponding numerical method. The payloads for the methods RK-1 and ADER-2 to ADER-7 on cells and edges are summarized in Table 4.1. Additionally we show the number of dofs that are represented in each cell. We see that in case of the ADER-DG methods we store significantly more bytes to represent a dof (77 byte-per-dof) than for the RK method (14 byte-per-dof), for which the main reason is the state machine we developed in Section 2.6.2: All ADER-DG methods simultaneously store the DG and FV representations of the solution, as well as the time-averaged DG-Predictor. On edges, additional to the projected time-averaged DG-Predictor, we store the projected time-averaged fluxes, which also results in a significantly higher payload on edges (see Section 2.4.3).

For later use we call the corresponding benchmark models STREAM and annotate them with the polynomial order of their respective DG method.

In Figure 4.2 we mark the bandwidth of the models STREAM-1, STREAM-2 and STREAM-7 in red. The resulting maximal bandwidths show us that STREAM-1 can reach a maximal bandwidth of $22.9\,\mathrm{GB\,s^{-1}}$. The benchmarks that adapt the ADER-DG methods can reach up to twice the bandwidth (STREAM-2 $37.9\,\mathrm{GB\,s^{-1}}$, STREAM-7 $43.1\,\mathrm{GB\,s^{-1}}$). Between the ADER-DG methods, the maximum attainable bandwidth differs slightly

(5.2 GB s^{-1} difference for between ADER-7 and ADER-2). With increasing order, the maximum bandwidth also increases.

4.3 The roofline model for sam(oa)^2

Up to this point we resolved the dependency between the maximum attainable bandwidth in sam(oa)^2 and the numerical methods. To evaluate the cost of our methods, we look at the arithmetic intensity of their main kernels in this section.

The most common approach to model the performance of a method is the idealized roofline model [139]. Its basic assumption is that the scheme is either dominated by the memory transfers or by the computations that are performed:[2]

$$\mathrm{T} = \max\left(\frac{\#\mathrm{flops}}{\mathrm{P}_{max}}, \frac{\#\mathrm{bytes}}{\mathrm{B}_{max}}\right). \tag{4.11}$$

Here T denotes the time that is required to process a total of #bytes on which we perform #flops. P_{max} is the peak performance of a system, while B_{max} is the highest bandwidth we can reach. In our case we determined the maximum attainable bandwidth for each method in the previous chapter. The ratio of #flops and #bytes is the arithmetic intensity (AI) of the scheme, the ratio of B_{max} and P_{max} the machine balance (MB) and a property of the system we are using. In case the arithmetic intensity is larger than the inverse machine balance, $\#\mathrm{flops}/\#\mathrm{bytes} > \mathrm{P}_{max}/\mathrm{B}_{max}$, the time to perform all floating point operations dominates the memory transfers and we speak of a *compute bound* application. In the opposite case the application is dominated by the memory transfers and the application is *memory bound*.

The roofline model is built upon some radical simplifications, from which we want to enumerate a few, as they become important in our considerations later [120]:

1. All memory transfers and floating point operations overlap.

2. All memory latencies can be hidden by prefetching.

[2] Our version of the roofline model is derived from its standard formulation:

$$\mathrm{P} = \frac{\#\mathrm{flops}}{\mathrm{T}} = \min\left(\mathrm{P}_{max}, \frac{\mathrm{B}_{max}}{\frac{\#\mathrm{bytes}}{\#\mathrm{flops}}}\right)$$

Table 4.2.: Number of floating point operations #flops and transferred bytes #bytes for the main kernels of RK-1 and ADER-2 to ADER-7. Additionally we compute the arithmetic intensity (AI) and the individual inverse machine balance (IMB), based on the peak performance and the measured maximum attainable bandwidth from Table 4.1.

Method	#flops	#bytes	AI	IMB
RK-1	361	104	3.5	83.7
ADER-2	3551	567	6.2	50.7
ADER-3	12659	995	12.7	50.0
ADER-4	34320	1532	22.3	48.3
ADER-5	78330	2182	35.8	47.0
ADER-6	158592	2944	53.8	45.7
ADER-7	293832	3820	76.9	44.5

3. All operations can exploit the theoretical peak performance of the system.

4. The instruction pipeline is always fully utilized.

5. All cores on a CPU perform the same number of operations, hence load-imbalance is never an issue.

To see if our applications are memory or compute bound, we present the arithmetic intensity of the main kernels for both methods in Table 4.2. In case of the ADER-DG method the main kernel is the Picard loop, for RK-1 the volume integral is the kernel with the largest number of flops.

As in the previous chapter we perform our analysis on a single socket on SuperMUC-NG. With the original STREAM Triad benchmark we measure a bandwidth of $\mathrm{B}_{max}^{STREAM} = 101.5\,\mathrm{GB\,s^{-1}}$. The 24 cores on each socked each have two AVX-512 FMA vector units that run with a frequency of 2.5 GHz, which accumulates to a theoretical peak-performance of $\mathrm{P}_{max} = 1.92\,\mathrm{TFLOPS\,s^{-1}}$. The resulting inverse machine balance is ≈ 18.9.

As we can not reach B_{max} with our methods, we compute the inverse machine balance individually for each method (IMB), based on the maximum attainable bandwidth we measured in Table 4.1.

Comparing AI and the individual IMB reveals that only for ADER-6 and ADER-7 we can expect that the main kernels are compute bound when executed in sam(oa)^2 (ADER-6: AI 53.8 > IMB 45.7, ADER-7: AI 76.9

> IMB 44.5). In all other cases the kernel is bound by the maximum attainable bandwidth. This is especially significant for the RK-1 method, for which the individual inverse machine balance is increased to 83.7. In case of ADER-4 and ADER-5 the increased individual machine balance means that the stack and stream approach puts both methods from the compute bound to the memory bound regime (ADER-4: AI 22.3 < IMB 48.3, ADER-5: AI 35.8 < IMB 47.0).

Practically, factors that are neglected in the roofline model play a crucial role in the runtime of our codes: Not all functions can fully exploit the peak performance of the system. For example, the limiter of the RK-DG scheme performs a vast amount of integer operations as index sortings. The same applies to divisions, which are used several times in the computation of the velocity (for example in the flux function). Branching avoids that we fully utilize the instruction pipeline, which happens in the different treatment of wet and dry cells in both methods. Finally, load-imbalance occurs necessarily in the ADER-DG scheme, as we cannot know the number of limited cells a priori.

4.4 Node level optimizations

The implementations we presented in Section 2.5 and Section 2.6 leave room for several hardware optimizations we performed to improve the performance of both methods. Base for our optimizations is the Intel Skylake architecture used on SuperMUC-NG with the underlying AVX-512 instruction set and a vector-length of 64 byte. Compilation is performed with the 2019 version of the Intel Fortran and C++ compiler.

4.4.1 Optimizing the Runge-Kutta DG code

In order to avoid cache line splits, we align all arrays to Skylake's vector-length. The numerical flux is auto-vectorized by the compiler, which we guide with the OpenMP directive *!#omp simd*, similar to the work be Ferreira et al. [50]. To match with the length of the vector register, we use zero-padding of the quadrature nodes on edges to four in the Riemann solver.

Similarly we proceed with the volume kernels of the RK-1 method. We zero pad the volume quadrature nodes to a length of four, which guides the compiler to autovectorize Fortran's default *matmul* implementation.

The performance of the Barth-Jaspersen typed limiter Section 2.5.2 is

dominated by divisions and the sorting of water height and velocity in Equation (2.69) and Equation (2.72). As a vectorization of this loop is cumbersome and is most likely inefficient we decided to omit it at this point.

4.4.2 Optimizing the ADER-DG code

As for the RK-DG method we align all arrays with the AVX-512 vector-length. For the FV scheme used in the limiter from Section 2.6.2, we adapt the already provided vectorization by Ferreira et al. [50]. Projection and reconstruction (Section 2.6.4) are again auto-vectorized with OMP pragmas. As the Rusanov flux of the ADER-DG method only takes up a small fraction of the runtime, we skip its vectorization at this point.

YATeTo for the optimization of the Picard loop

The discretization of the predictor (see Equation (2.49)) leads to small dense tensor contractions that we perform repeatedly for a previously unknown number of iterations:

$$\begin{aligned} {}^{n+1}P_{lLq} = \frac{\mathrm{d}t}{\mathrm{d}x} \left({}^{t}\mathrm{K}^{T}{}^{t}\mathrm{M}^{-1}\right)_{JL} \left({}_{i}\mathrm{J}\right)_{mq} \mathrm{S}_{lJm} + \\ \left({}^{t}\mathrm{K}^{T}{}^{t}\mathrm{M}^{-1}\right)_{JL} \left({}^{s}\mathrm{M}^{-1}{}^{s}\mathrm{K}\left({}_{i}\mathrm{J}\right)^{-1}\right)_{lbj} \mathrm{F}_{jJbq} + \\ \left({}^{t}\mathrm{K}^{-1}{}^{t}\mathrm{K}_{0}^{T}\right)_{L} q_{lq}. \end{aligned} \tag{4.12}$$

Where S and F are the flux and source terms evaluated for the predictor at iteration n:

$$\mathrm{S}_{lJm} = \mathrm{S}\left({}^{n}P_{lJm}\right) \qquad \mathrm{F}_{iJbq} = \mathrm{F}\left({}^{n}P_{lJm}\right).$$

We use YATeTo to generate hardware optimized kernels for the Skylake architecture [131]. Based on a custom high-level language for the definition of tensor contractions, YATeTo generates assembly code to exploit the available AVX-512 vector registers. The framework performs several stages to optimize the finally compiled code, in our case these boil down to:

- *Strength reduction:* The sequence in which tensor contractions are evaluated is optimized for the lowest number of operations.

Table 4.3.: Key performance characteristics of the Picard loop. We present the floating point operations and transferred bytes per iteration. Additionally we show the resulting arithmetic intensity, the performance in $\mathrm{GB\,s^{-1}}$ and its fraction of the peak performance.

Order	Flops/It.	Bytes/It.	AI	Perf.	% Peakperf.
2	3551	567	6.2	516.7	13.4
3	12659	995	12.7	739.1	19.2
4	34320	1532	22.3	1091.4	28.4
5	78330	2182	35.8	1228.2	32.0
6	158592	2944	53.8	1522.3	39.5
7	293832	3820	76.9	1658.0	43.2

- *Data Layout:* Tensor contractions are evaluated as loop-over-GEMM (LoG). Whenever it is beneficial for vectorization indices are fused in a LoG.

- *Generated Assembler Code:* For the resulting GEMMs the libxsmm [69] library is called to generate assembler code. The code exploits the AVX-512 instruction-set to evaluate matrices in their outer-product formulation [63].

The key characteristics of the resulting kernels are presented in Table 4.3. By design the arithmetic intensity of the predictor increases with the order of the method. Orders 2 and 3 are memory bound, as we stay below the machine balance on SuperMUC-NG of 19.2. For higher orders we are in the compute bound regime.

In order to estimate the attainable peak performance of the predictor we run a test on 100 000 elements and 400 000 repetitions on 48 cores on a single node. The test shows that the performance of the predictor increases with the polynomial degree of the method. It can reach up to 1658 GFlops for ADER-7, which is 43.2 % of the theoretical peak performance and a factor of 3.2 times higher as for ADER-2 with 13.4 %.

4.4.3 Performance improvement

In order to quantify the effect of our custom optimizations, we want to calculate the speedup we get when we compare the time per degree-of-freedom

to runs with only auto-optimizations. For the baseline we leave out all custom optimizations and compile with most aggressive compiler optimization ('-O3'). The Picard iteration is directly mirrored from Equation (4.12) to Fortran code. Tensor contractions are resolved with loop-over-GEMM, where matrix products are computed with the default *matmul* implementation of Fortran. The reference does not consider zero padding and loops are only auto-vectorized.

In order to compare both implementations, we pick a mesh of 2^{19} cells for which we know from Section 4.2 that the stack and stream approach reaches the maximal bandwidth for all methods. As scenario we choose the Sumatra tsunami.

In Figure 4.3 we show a comparison for the time per degree-of-freedom per time-step for all seven methods for custom-optimized (O) and baseline (B) kernels. We show for each traversal the individual contribution in ns and for the optimized implementation the speedup, compared to the compiler optimized version.

As the optimizations of the adaption in both versions are the same, we see no difference for this traversal. We get the highest overall speedup (1.8×) for ADER-5. For all other ADER methods the speedup increases with the order of the method (from 1.1× for ADER-2 to 1.6× for ADER-7). The RK-1 method has a speedup of 1.2×.

In case of the ADER-DG methods the speedup is the result of the improved Picard loop, which we can accelerate by up to more than three times (3.2× for ADER-5). The Corrector shows a significant benefit only for orders above 5.

For RK-1 the padding of volume kernels and vectorization of the Riemann solver lead to a speedup of 1.3× in both stages. The limiter does not benefit from any of the optimizations we performed.

4.5 Degrees of freedom per second

In this chapter we compare the time T from Equation (4.11) between an implemented method and its corresponding STREAM benchmark. This way we get a description of the cost an implemented method raises, compared to its pure memory transfers.

Following the discussion of the roofline model, practically the measured difference has to be either related to flops that are not overlapping with memory accesses, or to one of the factors we mentioned at the end of Section 4.3.

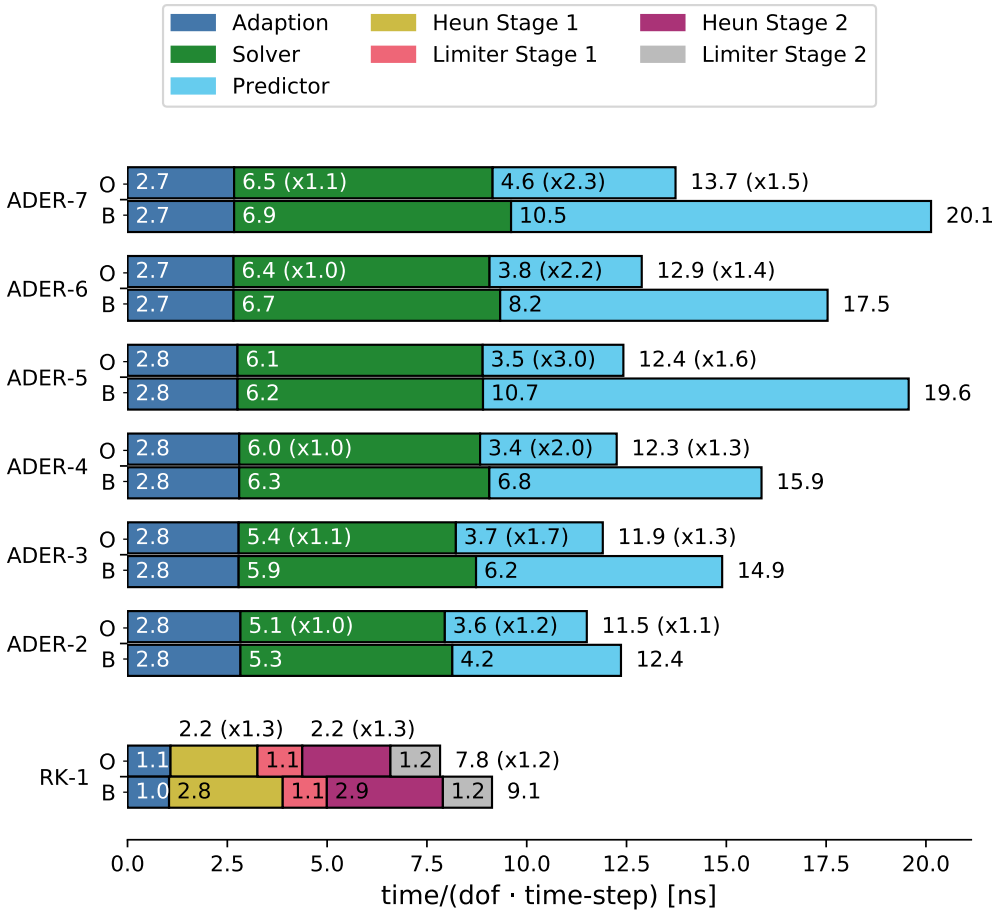


Figure 4.3.: Time-To-Solution for custom-optimized (O) and baseline (B) kernels for the RK-DG method and ADER-2 to ADER-7, split in the traversals required for a time-step. In case of the ADER-DG method the limiter is part of the Solver traversal.

We first look at the time-per-byte, which we get when we normalize the time T required for a simulation by its problem size.

Knowing the number of bytes we require to represent a dof, we can conclude on the time-per-dof:

$$\frac{\text{dofs} \cdot \text{time-steps}}{\text{time}} = \frac{\text{bytes} \cdot \text{traversals}}{\text{time}} \cdot \frac{\text{dofs}}{\text{bytes}} \cdot \frac{\text{time-steps}}{\text{traversals}}. \tag{4.2}$$

Time-per-byte

In Figure 4.4 we show a full comparison for the time-per-byte, between the benchmarks (STREAM-1 to STREAM-7) and their respective methods (RK-1, ADER-2 to ADER-7) for the resting lake benchmark. The number of cells are set such that all set-ups have close to 10^6 dofs, for which we know that all models run with the maximum attainable bandwidth (See Section 4.2). For each comparison we break down the analysis into the single stages of a time-step and estimate the cost of a method as the factor with which the time-per-byte increases between benchmark and model.

For the benchmarks (STREAM-1 to STEAM-7) the time-per-byte is the inverse of the measured maximum attainable bandwidth, and consequently decreases with the order. For ADER-2 to ADER-7 we see the opposite result, as the time-per-byte increases with the order and remains within the range $[49.2\,\mathrm{ps\,B^{-1}}, 59.1\,\mathrm{ps\,B^{-1}}]$. RK-1 is significantly slower compared to all ADER-DG methods with $98.1\,\mathrm{ps\,B^{-1}}$. The time-per-byte of all methods increases by a cost factor of $1.9\times$ to $2.6\times$ compared to the benchmarks and we measure the same factor for RK-1 and ADER-5 ($2.2\times$). As the data transfers of method and benchmark are equal and we do not perform any refinement operations, the adaption traversals agree in all comparisons. In case of the ADER-DG methods the additional effort comes mostly from the Solver traversals, which are slower than the benchmarks by a factor of $3.1\times$ to $4.8\times$. At first sight, this is a surprising result as we would expect that the Picard loop performed in the Predictor traversal to be the most expensive kernel. However, the Predictor traversals are only slower by a factor of $2.1\times$ to $3.4\times$ compared to the benchmark. We see two main reasons for this: The optimizations we perform for the Picard loop in Section 4.4 are very effective (We speed up the traversal by a factor of up to 3) and it becomes less significant. Moreover the Solver traversal contains OMP barriers which are caused by small load-imbalance in the mesh (see Figure 4.5). For RK-1 we see that the most effort is in the traversals that perform the element update (Heun), which are 3.5 to 4.0 times slower than the STREAM-1 benchmark. The Limiter traversals are less costly and $1.9\times$ to $2.0\times$ times slower than the benchmark.

We get a lower time-per-byte for the ADER-DG methods, as for the RK-1 method, which is caused by the two additional traversals we have to perform for each time-step for RK-1. For high-order ADER-DG methods we get a higher time-per-byte, than for low order methods, which corresponds to their higher computational cost (see Table 4.2). Parts of the higher computational effort of the high order methods can be hidden

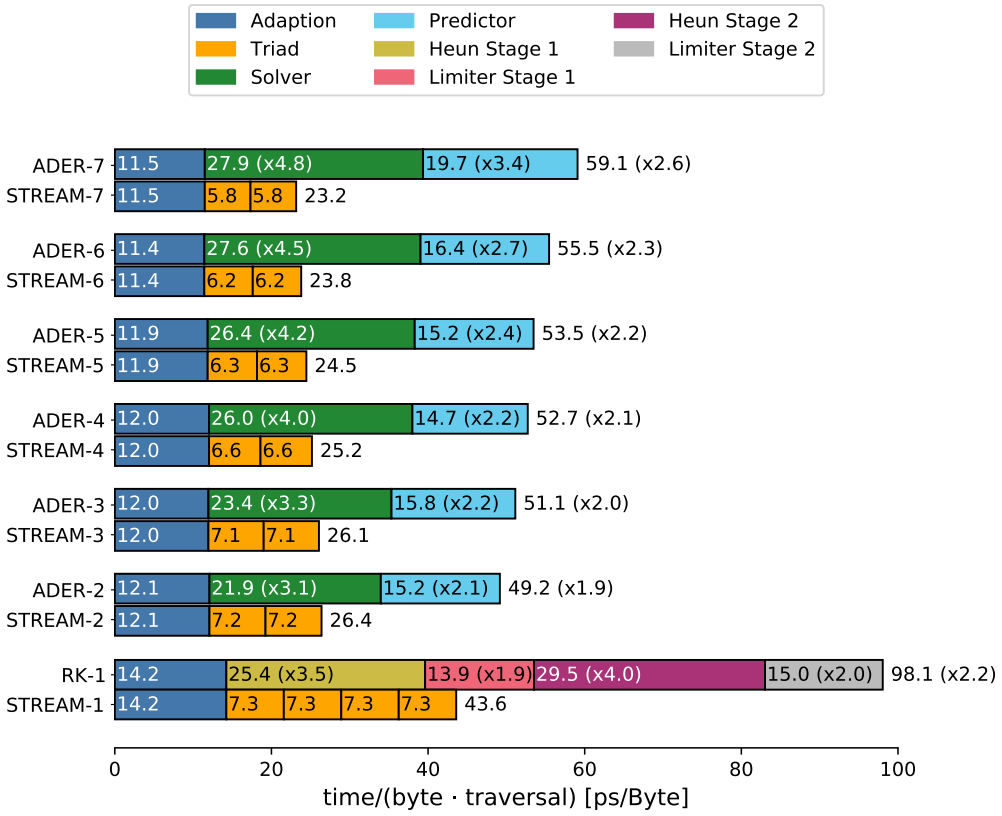


Figure 4.4.: Comparison of the time per byte between the bandwidth benchmarks STREAM-1 to STREAM-7 and the methods RK-1 and ADER-2 to ADER-7 in $\mathrm{ps\,B^{-1}}$. The overall time per byte is decomposed in the single traversals required to compute a single time-step. The factor with which the runtime increases compared to the corresponding bandwidth model is appended and marked with $\times$.

Table 4.4.: Time-per-dof for the resting lake, the oscillating lake and the Sumatra tsunami in ns for a mesh of around 10^6 dofs.

	resting lake	oscillating lake	Sumatra Tsunami
RK-1	7.84	6.98	7.4
ADER-2	9.22	6.08	10.9
ADER-3	9.34	5.68	11.6
ADER-4	9.37	6.02	12.1
ADER-5	9.57	6.18	12.9
ADER-6	9.63	6.24	13.9
ADER-7	10.5	6.66	15.8

behind their increased maximum attainable bandwidth. When we compare ADER-2 and ADER-7, we see that the additional effort for ADER-7 ($35.9\,\mathrm{ps\,B^{-1}}$ higher time-per-byte compared to STREAM-7) and for ADER-2 ($22.8\,\mathrm{ps\,B^{-1}}$ higher time-per-byte compared to STREAM-2) differs by $13.1\,\mathrm{ps\,B^{-1}}$. However the direct comparison of time-per-byte reveals that ADER-7 has $9.9\,\mathrm{ps\,B^{-1}}$ more effort than ADER-2. This shows us that the additional effort is reduced by around 25%, by memory transfers that are $3.2\,\mathrm{ps\,B^{-1}}$ faster for the order seven case.

Time-per-dof

Having derived a model for the time-per-byte we use it to conclude on the time-per-dof: In Table 4.1 we saw that the number of bytes we have to store in order to represent a dof and hold all required meta data, highly differs between RK-1 (bytes/dof = 14) and the ADER-DG method (bytes/dof $\approx$ 77).

This has a significant effect on how the methods compare in time-per-dof, which we present for all three set-ups in Table 4.4. While RK-1 is the method with the highest time-per-byte for the resting lake, its lower number of bytes-per-dof leads to the lowest time-per-dof (7.84 ns). In contrast all ADER-DG methods show higher time-per-dofs values, which again increase with the polynomial order (from 9.22 ns to 10.5 ns). We see the exact same relation for the Sumatra Tsunami, however the time-per-dof have significantly increased for the high order ADER-DG methods (ADER-7: resting lake 10.5 ns, Sumatra Tsunami: 15.8 ns). In case of the oscillating lake all ADER-DG methods are faster than RK-1 (RK-1 6.98 ns, ADER-DG from 6.08 ns to 6.66 ns).

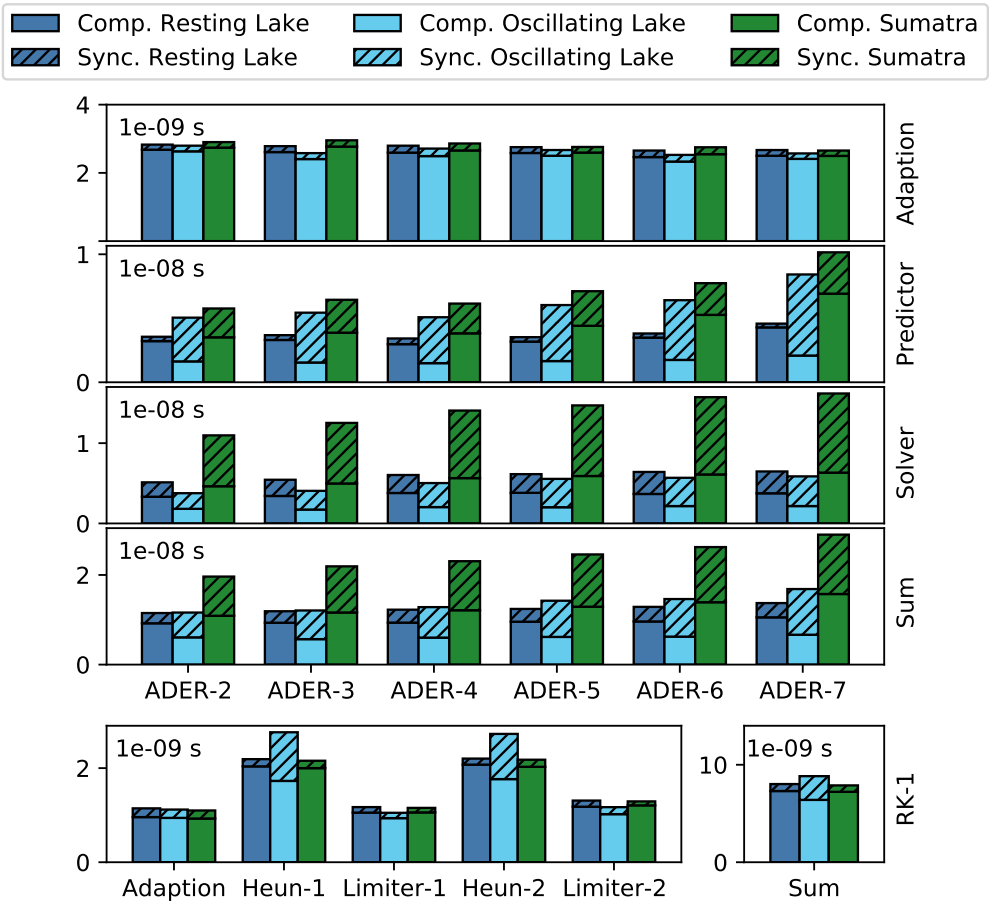


Figure 4.5.: Comparison of time-per-dof between three benchmarks.

To see in detail where these differences come from, we add a comparison for time-per-dof for all three scenarios in Figure 4.5 and look at the single traversals in detail. As they play a significant role we additionally split the time-per-dof into the time that is spent for each dof in computations or spent in OMP barriers. Data between OMP threads are exchanged in sam(oa)^2 at the end of each traversal after all threads finished computations for their part of the domain. Large synchronization barriers imply that threads have to wait for each other and indicate load-imbalance. As in our set-up all OMP threads hold the same number of cells, load-imbalance has to be a result of varying computational costs of cells.

In case of the RK-1 method the computational cost of cells only differs between the wet and dry case. On dry cells less computations are performed,

as all operations that require positive water columns (as the velocity) are skipped. In all other steps operations are performed equivalently on all cells.

In case of the ADER-DG method there are several aspects where the computational cost per cell differs. At first not all cells perform the same number of iterations of the Picard loop. When a cell contains a resting solution the loop usually converges after a single traversal, as the initial condition is already the right solution. Cells that hold a non-steady solution require more iterations of the loop, as the solution in the predictor has to evolve before it converges. If the water column in a cell is too small compared to the velocity the Picard loops can diverge as the flux is numerically not bound. In our implementation the divergence gets detected after a threshold of twenty iterations and is then treated by the limiting mechanism.

Another factor that leads to varying costs for cells in the ADER-DG scheme is given by the state-machine. Here cells that are captured and limited by the DMP (see Section 2.6.3) have a higher cost as unlimited cells, as they perform the corrector step and compute the FV Solution. The state machine also captures cells that are dry and are surrounded by dry cells at the beginning of each time-step. For these cells we can be certain that their state stays constant and skip the computation of their Picard loop and corrector.

The results of Figure 4.5 show us that the time-per-dof is similar for all scenarios simulated with the RK-1 method but vary for the ADER-DG methods

The only outstanding difference for RK-1 are in the OMP-Barriers and computations for the oscillating lake scenario. We can directly relate these difference back to the asymmetry of the solution: Large areas of the oscillating lake are dry, while a wet droplet propagates in circles through the domain. The asymmetry of the solution leads to unevenly distributed wet and dry cells and results in load-imbalance. The time for computations is reduced, compared to the resting lake and the Sumatra tsunami, as large parts of the domain are dry.

For the Predictor and Solver traversals of the ADER-DG method the load-imbalance for the oscillating lake are even more significant. As the state machine detects dry cells and skips their computation, the time in computations in both traversals is significantly reduced compared to the resting lake scenario. In case of the Predictor traversal, dry cells that do not perform the Picard loop, wet cells with a few iterations, and cells close to the wet/dry interface for which the Picard loop diverges are not evenly

distributed and lead to load-imbalance. We see the same behavior in the Solver iteration where dry, limited, and unlimited cells are unevenly distributed in the mesh.

For the Sumatra tsunami we do not see the same large load-imbalance for the Predictor in the Sumatra tsunami. Here the FV-threshold captures cells with a small water-column at the coast such that there is no cell for which the Picard loop diverges. However, the Solver traversal shows large load-imbalance that are partly caused by the asymmetric ribbon of static FV-cells we define around the coast. As the initial tsunami contains discontinuities, we also expect that several cells are detected by the DMP and have to be limited.

4.6 Time to solution

Having derived the single components of Equation (4.1), we can finally assemble them to a full comparison in time-to-solution

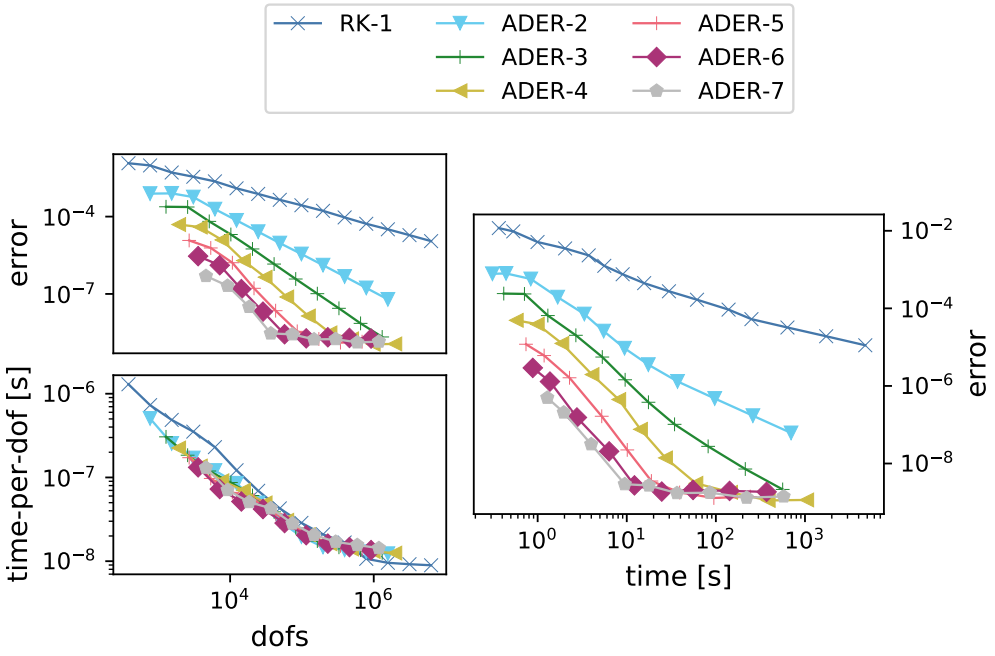
$$\frac{\text{accuracy}}{\text{time}} = \frac{\text{accuracy}}{\text{dofs}} \cdot \frac{\text{dofs} \cdot \text{time-steps}}{\text{time}} \cdot \frac{1}{\text{time-steps}}. \tag{4.1}$$

For our three scenarios we want to compute the error-per-time for increasing meshes of 2^4 to 2^{16} cells. For each mesh, we recapture the time-per-degree as measured in Section 4.5 and the error-per-dof which we derive from the results from Chapter 3. The full comparison is presented in Figure 4.6 for the resting lake (Figure 4.6a), the oscillating lake (Figure 4.6b) and the Sumatra tsunami (Figure 4.6c).

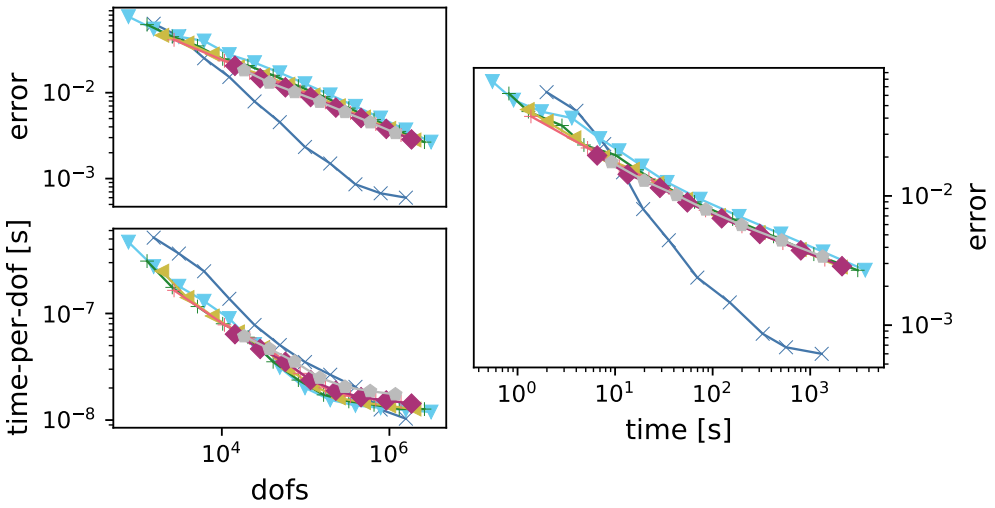
We first look at the smooth resting lake scenario in Figure 4.6a. As we already saw the representation of the smooth solution benefits from the high order representation such that the convergence rate and values of the error-per-dof are significantly better for the high order ADER methods than for the low order case. For ADER-2 as well as RK-1 this means that we cannot reach the convergence limit within our measured number of cells. We saw in Table 4.4 that the attainable time-per-dof is lower for the RK-1 method, however up to 2.0×10^5 dofs the time-per-dof is higher for RK-1 as for the ADER-DG methods. This is related to the later convergence of the bandwidth for RK-1 method that we observed for low payloads in Section 4.2. For all ADER-DG methods the time-per-dof increases with the polynomial degree. The best result for time-to-error is achieved with ADER-7 for a time of 9.5 s. With increasing order the double precision limit can be achieved within exponentially less invested time (ADER-6 12.2 s,

ADER-5 39.2 s ADER-4 146.2 s and ADER-3 561.9 s). We see in the smooth case that the better error-rate of high order methods pays off and can easily equalize the increased number of time-steps and the higher time-per-dof.

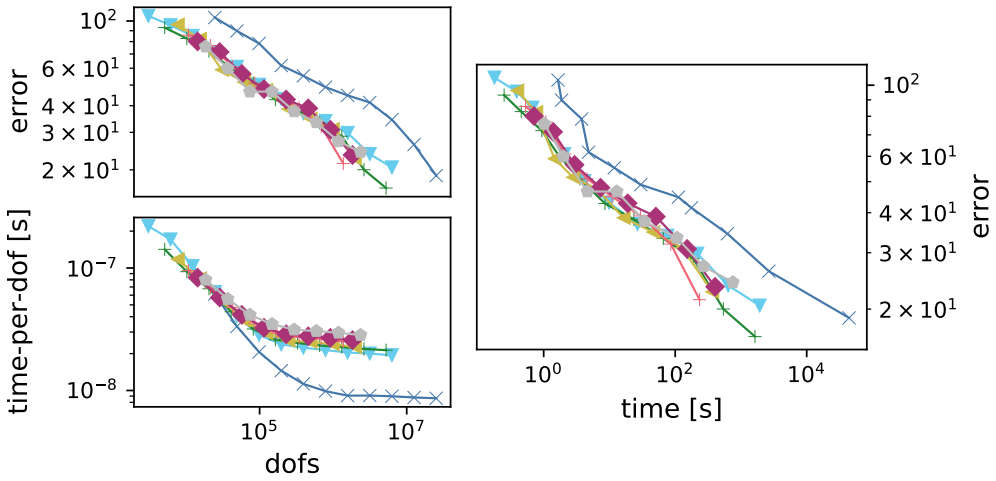
Next we look at the oscillating lake scenario. We observed that the Barth



(a) Time-to-solution for the resting lake.



(b) Time-to-solution for the oscillating lake.



(c) Time-to-solution for the Sumatra Tsunami.

Figure 4.6.: Time-to-solution comparison for the methods RK-1 and ADER-2 to ADER-7. We show the time required to advance a single dof for one time-step against the number of dofs, the error against the number of dofs and finally the time required to reach a certain error.

Jespersen limiter of the RK-1 method leads to an improved error-rate compared to the FV implementation we use in the limiting process of the ADER-DG method. The higher convergence rate of the RK-1 methods leads to significantly better error-per-dof compared to the ADER-DG methods (RK-1 0.0015 error for 1.97×10^6 dofs, ADER-7 0.006 for 3.0×10^6 dofs). Between ADER-DG methods high order still pays off as the error-per-dof decreases with the polynomial degree (For example ADER-2 0.007 for 4.0×10^6 dofs and ADER-3 0.006 for 3.2×10^6 dofs). In case of the ADER-DG methods, time-per-dof profits from the state-machine that detects and skips dry cells. The time-per-dof is better for all ADER-DG methods compared to RK-1 and increases with the polynomial degree. However, the differences in time-per-dof between all methods are small (around $\sim$ 10 % for the same number of dofs) such that the improved time-per-dof of the ADER-DG methods cannot equalize the slower error rate. For this reason the comparison in time-to-solution shows a very clear advantage for the RK-DG method. RK-1 outperforms all ADER methods, for simulations with a reasonable high number of dofs.

Finally we look at the Sumatra tsunami. Here we saw a lower error for

all ADER-DG methods compared to the RK-DG method, which is caused by the significantly lower initial error. As the convergence rate is equivalent between methods the error remains lower for all ADER-DG methods. As for the oscillating lake the error decreases slightly with the order.

The time-per-dof is lower for the RK-DG method, which is caused by the high load-imbalance we saw for the ADER-DG methods in Section 4.5 and increases with the order of the ADER-DG method. Combining all results reveals that the error-per time shows the best values for the ADER-DG methods. While a relative error of less than 25 % is reached with the RK-1 method in 2657 s, all ADER-DG methods require less than 640 s (ADER-2 639 s, ADER-3 540 s, ADER-4 365 s, ADER-5 235 s, ADER-6 404 s and ADER-7 268 s).

4.7 Conclusion

In this chapter we disassembled and compared time-to-solution for high-order ADER-DG methods and a second order Runge-Kutta method in the sam(oa)^2 framework. We showed in all cases that time-to-solution is better for the scheme that addresses a problem with the more complex method. In all cases the higher time-to-solution of the more complex scheme, is compensated for with better error behaviors.

In case of the ADER-DG methods, time-to-solution for the resting lake benefits from the high-order representation: The higher number of memory transfers and flops of the ADER-DG methods, compared to RK-1, lead to an higher time-per-dof. Better errors for the high-order representation of the smooth solution equalize these additional costs and lead to a better time-to-solution.

In case of the oscillating lake time-per-dof is lower for the ADER-DG methods than for RK-1. The higher time-per-dof for RK-1, is equalized by the better resolution of the inundation process by the limiting scheme.

In case of the simulation of the Sumatra tsunami we focused on a comparison for a trajectory of the water-level in the deep-ocean. We thus see the same better error behavior for ADER-DG, as we see it for the resting lake.

In conclusion, the error behavior of the polynomial representation results in better time-to-solution for solutions in deep-water, while for inundation processes the limiting scheme plays the fundamental role. A future comparison in time-to-solution should include RK methods of higher order and extend the limiting scheme of the ADER-DG method.

CHAPTER

5

Linking of earthquake and tsunami codes

Realistic sources and physically correct linking are the key to accurate predictions of the impact of tsunamis [15].

The common approach to reproduce earthquake events and model tsunami sources is based on the analytic solutions by Okada [15, 101]. Okada summarized closed solutions for the surface displacement field, in homogeneous half-spaces with singular point or finite rectangular sources [8]. While these solutions are still widely used to derive sources for tsunamis, they are restricted to flat topographies and homogeneous material parameters [146].

Advanced HPC codes from computational seismology allow us to model complex fault systems with realistic fracture mechanics. In the context of Probabilistic Tsunami Hazard Assessment (PTHA), where we want to estimate the long-term impact of tsunamis on exposed regions, we can easily vary uncertain parameters in the earthquake model, simulate the subsequent tsunamis and analyze their effects [61].

In this chapter we introduce a workflow for the one-way linking of earthquake simulations to discontinuous Galerkin codes for the simulation of tsunami models. The workflow has been applied to simulations in several publications [92, 140, 143].

The classical physical toolbox for linking of displacements from earthquake simulations, to source tsunamis is given by linear potential theory in inviscid fluids, which we use as foundation of our linking approach [121]. In linear potential theory the flow between sea floor and sea-surface is considered incompressible, all rotational forces are neglected. The displacement

of sea floor and sea-surface are boundary conditions of the water layer, where gravitation is considered the only acting force. To induce the displacement on the sea-surface a constant or time-dependent deformation of the sea-bottom is used.

Takahasi was the first to apply linear potential theory to find an analytical solution for the relation between a sea-bottom displacement on a flat bathymetry and the sea surface [121]. Kajiura generalizes Takahashi's approach and derives several analytical solutions for the vertical displacement of the sea-surface, caused by instantaneous sea floor deformations [75]. The Fourier analysis of Kajiura's solutions shows that in the transition from sea floor to sea-surface displacement wavenumbers which are larger than $1/H$ (where H is the sea-depth) are filtered in the water-layer. Today this result is still commonly used to convert sea floor displacements to tsunami sources and is widely known as Kajiura filter.

In Kajiura's theory the sea floor deformation happens instantaneously at $t = 0$, which does not allow to take the temporal evolution of the sea floor deformation into account. Saito extended the approach by modeling time-dependence of the deformation with a time-rate function. The rate-function leads to a uniform elevation of the sea floor displacement. After the source time σ_t the deformation remains permanent [108, 109]. Saito used his approach to show that depending on the size of the tsunami source σ_r, the filtering effects Kajiura observed are only significant for small source areas $\sigma_r < 13H$, or for long source process times $\sigma_t > \sigma_r/(8\sqrt{gH})$, with g as the gravitational acceleration [111].

To this point all presented approaches consider gravity as the single sourcing force of tsunamis. All kinetic energy in a tsunami evolves from the potential energy induced by the displacement of the sea-surface.

To include the kinetic energy, which is transmitted by horizontally moving slopes in the bathymetry, Song et al. propose to use an initial set horizontal momentum in the tsunami model [119]. With a fully coupled dynamic-rupture and tsunami model Lotto et al. study the effects of this transmitted initial momentum and show that it only effects ocean acoustic waves and is negligible for the tsunami [91].

In Lotto and Dunham's model the sea floor is modeled as boundary condition to couple seismic waves in a solid with acoustic waves in a compressible ocean layer under the influence of gravity. The sea-surface boundary condition is set to model tsunami gravity waves [80, 90]. Abrahms et al. use this model to conclude from source size and duration, on the effect of different coupling strategies [1].

The linking workflow we present in this chapter is based on the results

found by Kajiura [75], Saito [111] and Lotto [90]. We take the horizontal and vertical displacement fields from an earthquake simulation and use the well known Tanioka's method to compute a perturbation of the sea floor [122]. From Kajura's and Saito's work we know that we can use the perturbation of the sea floor to directly source the tsunami and do not have to consider any effects happening in the water layer. Lotto et al. tell us that this is the only source we require to properly source the tsunami.

We first show how we extend the numerical methods in sam(oa)^2 for time dependent and time independent tsunami sources in Section 5.1. To include horizontal displacements we revisit the well known Tanikoa's method and derive it for our particular application in Section 5.2. Seismic waves play a crucial role in our workflow, as in our sourcing they might generate spurious waves. As solution we introduce a novel Fourier based filtering approach in Section 5.3.

5.1 Sourcing tsunamis with the shallow water equations

We start with inclusion of the source in the shallow water equations (SWE) as we defined them in Section 2.3. The postprocessed results of an earthquake model lead to a perturbation of the sea floor $\Delta b(x, y, t)$ that we want to include in the tsunami model. The bathymetry becomes an initial constant sea floor $\hat{b}(x, y)$, modified by a time-dependent sea floor perturbation. After the last recorded time-step of the earthquake model T, we assume that the sea floor perturbation has converged and remains constant,

$$b(x, y, t) = \begin{cases} \hat{b}(x, y) + \Delta b(x, y, t) & \text{if } t \leq T \\ \hat{b}(x, y) + \Delta b(x, y, T) & \text{else.} \end{cases} \tag{5.1}$$

In the sea floor perturbation we accumulate all influences of the earthquake on the sea floor and follow the argumentation of Lotto, that these are the only effects we have to consider to properly initialize the tsunami [91]. Our linking approach is designated for large interplate events, where tsunami source size σ_r and duration σ_t allow us to directly translate the sea floor perturbation to the sea-surface and neglect all effects that might happen in the vertical water-flow ($\sigma_r \gg 13H$ or $\sigma_r/8\sqrt{gH} \gg \sigma_t$, where H is the sea-depth) [111].

We want to compare the sea floor perturbation as time-independent and time-dependent source. In case of time-independent sourcing, we take the

water column h_0 of the resting lake scenario based the initial constant sea floor $\hat{b}$ and add the sea floor perturbation Δb at the end of the earthquake,

$$h_0 = \max(-\hat{b}, 0), \qquad (hu)_0 = (hv)_0 = 0, \qquad b_0 = \hat{b} + \Delta b(x, y, \sigma_t). \tag{5.2}$$

For time-dependent sourcing we use the depth-averaged continuity from Equation (2.21) and replace the water column h with the difference of water level H and bathymetry b. We get the SWE in the form

$$\mathrm{Q}_t + \nabla \cdot \mathrm{F}\left(\mathrm{Q}\right) = \mathrm{S}\left(\mathrm{Q}\right), \tag{5.3}$$

with

$$\mathrm{Q} = \begin{pmatrix} H \\ hu \\ hv \end{pmatrix}, \nabla \cdot \mathrm{F}\left(\mathrm{Q}\right) = \begin{pmatrix} (hu)_x + (hv)_y \\ \left(\frac{1}{2}gh^2 + hu^2\right)_x + (huv)_y \\ (huv)_x + \left(\frac{1}{2}gh^2 + hv^2\right)_y \end{pmatrix}, \mathrm{S}\left(q\right) = \begin{pmatrix} b_t \\ -gb_x h \\ -gb_y h \end{pmatrix}. \tag{5.4}$$

Godunov splitting allows us to solve Equation (5.4) numerically [86]: We separate the source term in the part that affects the sea-surface and the part that adjusts the momenta,

$$\mathrm{S}_{surf}\left(\mathrm{Q}\right) = \begin{pmatrix} b_t \\ 0 \\ 0 \end{pmatrix}, \mathrm{S}_{mom}\left(\mathrm{Q}\right) = \begin{pmatrix} 0 \\ -gb_x h \\ -gb_y h \end{pmatrix}. \tag{5.5}$$

The splitting approximates the PDE in Equation (5.4) as PDE with subsequent ODE,

$$\text{PDE:} \qquad \mathrm{Q}^*_t + \nabla \cdot \mathrm{F}\left(\mathrm{Q}^*\right) = \mathrm{S}_{mom}\left(\mathrm{Q}^*\right) \tag{5.6}$$

$$\text{ODE:} \qquad \mathrm{Q}_t = \mathrm{S}_{surf}\left(\mathrm{Q}^*\right). \tag{5.7}$$

In order to use the unchanged DG methods from Section 2.5 and Section 2.6 to solve the PDE Equation (5.6), we have to remove the dependency on time in $\mathrm{S}_{mom}\left(\mathrm{Q}\right)$. We can do this by assuming that the gradient of the bathymetry hardly changes between two time-steps,

$$b_x(x, y, t_{n+1}) \approx b_x(x, y, t_n), \qquad b_y(x, y, t_{n+1}) \approx b_y(x, y, t_n). \tag{5.8}$$

The temporal evolution of the bathymetry is given from the earthquake model, such that we can solve the ODE in Equation (5.7) with explicit

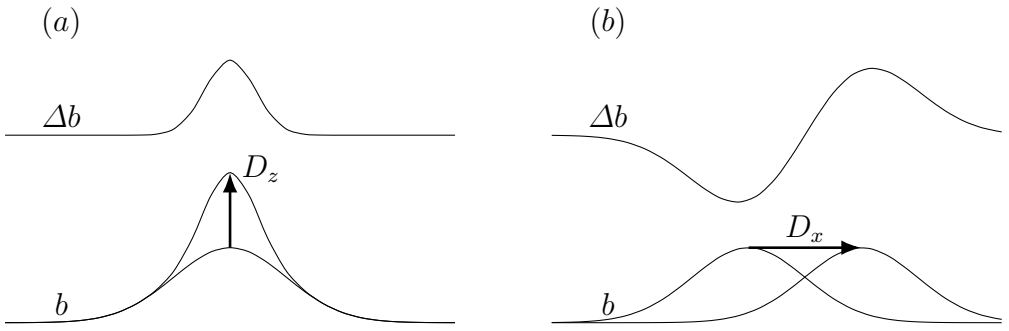


Figure 5.1.: Qualitative sketch of the sea floor perturbation Δb caused by vertical (a) and horizontal displacement (b) of the sea floor b.

integration in time,

$$H_{n+1} = H_n + \int_{t_n}^{t_{n+1}} b_t(t, x, y)\,\mathrm{d}t\,. \tag{5.9}$$

With the fundamental theorem of calculus we can show that the change of the bathymetry is given by the seafloor perturbation,

$$\begin{aligned} \int_{t_n}^{t_{n+1}} b_t\,\mathrm{d}t = b_{n+1} - b_n = \Delta b(x, y, t_{n+1}) - \Delta b(x, y, t_n) \overset{5.9}{\rightarrow} \\ H_{n+1} = H_n + \Delta b(x, y, t_{n+1}) - \Delta b(x, y, t_n). \end{aligned} \tag{5.10}$$

In the actual implementation we only have to store the water column and bathymetry explicitly. Equation (5.10) is implicitly realized, by updating the bathymetry with the change of the sea floor perturbation at the end of every time-step.

5.2 The method of Tanioka and Satake

To generate the sea floor perturbation we have to consider vertical and horizontal displacements of the sea floor. Figure 5.1 shows the two types of ground motion that change the vertical height of the bathymetry. Vertical displacements are directly translated to the bathymetry and cause uplift and subsidence (a). Horizontal displacements on the other hand lead to a shift of the bathymetry and also have to be reflected in the sea floor perturbation (b). Tanioka was the first to consider horizontal displacements in the tsunami generation. Here we translate his approach to time dependent

displacements [122].

We look at the Lagrangian description for the motion of a particle located at the surface of the bathymetry at $(x, y, b(x, y, t))$ during an earthquake. After a period of Δt the particle has been translated by $(\Delta x, \Delta y, \Delta z)$ and has to remain on the surface,

$$(x + \Delta x, y + \Delta y, b(x + \Delta x, y + \Delta y, t + \Delta t)) = (x, y, b(x, y, t)) + (\Delta x, \Delta y, \Delta z). \quad (5.11)$$

In order to get rid of the translations in the bathymetry at the left side of the equation, we evolve $b(x + \Delta x, y + \Delta y, t + \Delta t)$ in a Taylor series in x and y, and neglect the second order terms,

$$b(x + \Delta x, y + \Delta y, t + \Delta t) \approx b(x, y, t + \Delta t) + b_x(x, y, t + \Delta t)\Delta x + b_y(x, y, t + \Delta t)\Delta y. \quad (5.12)$$

As the magnitude of the displacement is small compared to the magnitude of the bathymetry, we assume that the gradient of the bathymetry hardly changes over time,

$$b_x(x, y, t + \Delta t) \approx b_x(x, y, 0), \qquad b_y(x, y, t + \Delta t) \approx b_y(x, y, 0). \quad (5.13)$$

Combining Equation (5.11), Equation (5.12) and Equation (5.13) leads to:

$$b(x, y, t + \Delta t) = b(x, y, t) + \Delta z - b(x, y, 0)_x \Delta x - b(x, y, 0)_y \Delta y. \quad (5.14)$$

Finally we have to determine the particle translations $(\Delta x, \Delta y, \Delta z)$ within the period Δt. We know, that at the beginning of the earthquake there is a particle $(\tilde{x}, \tilde{y}, b_0(\tilde{x}, \tilde{y}))$ on the sea floor, which is translated to $(x, y, b(x, y, t))$ by its displacements at time t,

$$(x, y, b(x, y, t)) = (\tilde{x}, \tilde{y}, b_0(\tilde{x}, \tilde{y})) + (D^x(\tilde{x}, \tilde{y}, t), D^y(\tilde{x}, \tilde{y}, t), D^z(\tilde{x}, \tilde{y}, t)) . \quad (5.15)$$

As we do not know the exact initial position $(\tilde{x}, \tilde{y}, b_0(\tilde{x}, \tilde{y}))$, we have to approximate it. Displacements are relatively small compared to the size of the domain, assuming that the field is locally smooth. Therefore, we approximate

$$D^*(\tilde{x}, \tilde{y}, t) \approx D^*(x, y, t). \quad (5.16)$$

Now we can use the difference of the displacements after Δt to compute the translation of the particle,

$$\begin{aligned}(\Delta x, \Delta y, \Delta z) = \\ (x + \Delta x, y + \Delta y, b(x + \Delta x, y + \Delta y, t + \Delta t)) - (\Delta x, \Delta y, \Delta z) \overset{(5.16)}{=} \\ (D^x(x, y, t + \Delta t) - D^x(x, y, t), \\ D^y(x, y, t + \Delta t) - D^y(x, y, t), \\ D^z(x, y, t + \Delta t) - D^z(x, y, t)).\end{aligned} \tag{5.17}$$

Equation (5.14) and Equation (5.17) give us a time dependent version of the method of Tanioka & Satake. The approximations made in Equation (5.12), Equation (5.13) and Equation (5.16) allow us to compute the sea floor perturbation locally in each coordinate,

$$\begin{aligned}\Delta b(x, y, t + \Delta t) = b(x, y, t) - b(x, y, 0) + (D^z(x, y, t + \Delta t) - D^z(x, y, t)) \\ - b(x, y, 0)_x(D^x(x, y, t + \Delta t) - D^x(x, y, t)) \\ - b(x, y, 0)_y(D^y(x, y, t + \Delta t) - D^y(x, y, t)).\end{aligned} \tag{5.18}$$

5.3 A Fourier filter to erase fast seismic waves

Simulating a three dimensional earthquake in high resolution is computationally expensive. Therefore the simulated time is usually chosen to only record the convergence of the permanent seafloor displacement reached after a couple of minutes. Following the method we presented in Section 5.1, the tsunami is sourced by the sea floor perturbation for the recorded time range. After this time, the sea floor perturbation is kept constant. As the permanent displacement has converged at this point, it is properly resolved in the tsunami model. Seismic waves on the other hand are abruptly stopped, as keeping the seafloor perturbation constant is equivalent to setting all wave velocities to zero. In the tsunami model these waves appear as permanent seafloor perturbation and generate spurious waves.

To avoid the false modeling of seismic waves, we choose to eliminate them from the displacement field. This is valid, as Saito showed with coupled earthquake tsunami models, that these waves have no influence on the tsunami, as they propagate out of the domain [110].

A simple method is to crop the field, such that seismic waves artificially propagate out of the domain in the recorded time-frame. This approach works whenever seismic waves and permanent displacement are not overlap-

ping and spatially separable. A more sophisticated approach is required, in case seismic waves overlay the permanent displacement in the last recorded time-step: In this section we look at a novel Fourier filtering approach, inspired by techniques used in computer vision [77]. Our filter works in the frequency-wavenumber (f-w) spectrum of a displacement field, to separate the content of fast seismic surface waves from the spectrum of the dominant slow lifting permanent displacement. We want to identify those f-w coefficients of the displacement field which distinctly belong to seismic surface waves and zero them out.

To give an idea on how the f-w representations of both types of displacement behave, we first look at the Fourier transformations of analytic examples. From these examples and their numerical counterparts we design a filtering kernel which damps fast seismic waves and keeps the permanent displacement. Finally we analyze the effects of our filtering approach.

5.3.1 Separation in Fourier space

Our filtering approach is designed on the underlying assumption that we can separate fast traveling seismic waves from a permanent displacement in their combined frequency-wavenumber representation. This assumption is strengthened by the observation that the analytic Fourier transformation of a traveling wave has its coefficients along the line $\omega - \frac{f}{v}$.

The main tool in this study is the two-dimensional Fourier transformation. For a function $g(t,x)$ in time t and space x, the Fourier transformation $\mathcal{F}_{t,x}(f,\omega)\left\{g\left(t,x\right)\right\}$ computes the frequency f and wavenumber ω with

$$\mathcal{F}_{t,x}(f,\omega)\left\{g\left(t,x\right)\right\} := \mathcal{F}_x(\omega)\left\{\mathcal{F}_t(f)\left\{g\left(t,x\right)\right\}\right\} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} g(t,x)\exp\left(-itf\right)\mathrm{d}t\exp\left(-ix\omega\right)\mathrm{d}x\,. \quad (5.19)$$

Here $\mathcal{F}_x(\omega)\left\{\cdot\right\}$ and $\mathcal{F}_t(f)\left\{\cdot\right\}$ denote the one dimensional Fourier transformations.

If we assume a wave traveling with velocity v, defined by a time-series $w(t):\mathbb{R}\to\mathbb{R}$, its Fourier transformation is

$$\mathcal{F}_{t,x}(f,\omega)\left\{w\left(t-\frac{x}{v}\right)\right\} = \mathcal{F}_x(\omega)\left\{\exp\left(-i\frac{x}{v}f\right)\mathcal{F}_t(f)\left\{w(t)\right\}\right\} =$$
$$\mathcal{F}_x(\omega)\left\{\exp\left(-i\frac{x}{v}\right)f\right\} * \mathcal{F}_t(f)\left\{w(t)\right\} = \delta\left(\omega-\frac{f}{v}\right) * \mathcal{F}_t(f)\left\{w(t)\right\}. \quad (5.20)$$

Where we used the convolution theorem:

$$\mathcal{F}_x(\omega)\left\{g(x)h(x)\right\} = \mathcal{F}_x(\omega)\left\{g(x)\right\} * \mathcal{F}_x(\omega)\left\{h(x)\right\} := \int_{-\infty}^{\infty} \mathcal{F}_x(\omega)\left\{g(x)\right\}(\eta)\,\mathcal{F}_x(\omega)\left\{h(x)\right\}(\omega-\eta)\,\mathrm{d}\eta \quad (5.21)$$

As a permanent displacement remains constant after a time-frame and is not highly oscillating but restricted to a fixed area, we expect to see large coefficients along the lines $f = 0$ and $\omega = 0$.

In order to confirm our assumption we look at two analytic examples: A slowly lifting displacement taken from Saito [109] and a constantly moving S-wave taken from Aki [3].

Fourier transformation of a permanent displacement

For the case of the slowly lifting displacement $D_p^z(x,t)$, we assume that it is defined by a final permanent displacement $d(x)$ and a time-rate function $\chi(t)$ [109]. The time-rate function $\chi(t)$ determines the rate that the displacement emerges with, starting at a flat seafloor,

$$D_p^z(x,t) = d(x)\int_{-\infty}^{t} \chi(T)\,\mathrm{d}T\,. \quad (5.22)$$

We chose a Gaussian hill as permanent displacement $d(x)$, with σ_r as the size of the source,

$$d(x) = \exp\left(-\frac{x^2}{2\cdot\sigma_r^2}\right). \quad (5.23)$$

For a source duration of σ_t and a maximum rate at t_0 we define the rate function by

$$\chi(t) = \frac{4}{\sigma_t\sqrt{\pi}}\exp\left(-\left(\frac{(t-t_0)\cdot 4}{\sigma_t}\right)^2\right), \quad (5.24)$$

such that $\int_{-\infty}^{\infty}\chi(t) = 1$.

The analytic fourier transformation for the permanent displacement is shown in Appendix A. We get the norm of the f-w coefficients

$$\left|\mathcal{F}_{t,x}(f,\omega)\left\{D_p^z\right\}\right| = \frac{\sqrt{2}\sigma_r}{f\sqrt{\pi}}\exp\left(-\left(\frac{\sigma_t}{8}\cdot f^2\right)\right)\exp\left(-\left(\frac{(\sigma_r\cdot\omega)^2}{2}\right)\right). \quad (5.25)$$

At $f = 0$ the coefficients have a singularity, as the integral of the permanent displacement diverges for $t \to \infty$. We see that the coefficients with the highest norm of Equation (5.25) are close to the lines $f = 0$ and $\omega = 0$.

Fourier transformation of a generic S-wave

For a point source with source-time function $X(t)$ placed at the origin in depth d, the vertical displacement for the emanating S-wave is given by Aki et al. [3],

$$D_s^z(x,t) = \frac{1}{4\pi\rho v^2}\left(1 - \frac{d^2}{r^2}\right) X\left(t - \frac{r}{v}\right), \tag{5.26}$$

where v is the velocity of the wave and $r = \sqrt{\|x\|_2^2 + d^2}$ the Euclidean distance to the point source location. In our example we remove all scalars of (5.27) and take a simple Gaussian impulse as point source,

$$D_s^z(x,t) = \frac{1}{v^2}\left(1 - \frac{d^2}{r^2}\right) \exp\left(-\left(t - \frac{r}{v}\right)^2\right). \tag{5.27}$$

In Appendix A we show that the norm of the f-w coefficients can be approximated by the equation

$$|\mathcal{F}_{t,x}(f,\omega)\{D_s^z\}| = \left|\delta(f)\,\delta(\omega)\,\frac{\sqrt{\pi}}{\sqrt{2}v^2} - \frac{\pi}{dv^2}\exp\left(-\frac{f^2}{4}\right)\exp\left(-d|\omega - \frac{f}{v}|\right)\right|. \tag{5.28}$$

At $(f,\omega) = (0,0)$ the norm of the f-w coefficients is equivalent to the wave's integral over $(t,x) \in \mathbb{R}^2$,

$$|\mathcal{F}_{t,x}(0,0)\{D_s^z\}| = \frac{\sqrt{\pi}}{\sqrt{2}v^2} + \frac{\pi}{dv^2}, \tag{5.29}$$

everywhere else it is

$$|\mathcal{F}_{t,x}(f,\omega)\{D_s^z\}| = \frac{\pi}{dv^2}\exp\left(-\frac{f^2}{4}\right)\exp\left(-d|\omega - \frac{f}{v}|\right). \tag{5.30}$$

While the first exponential factor shows us that the norm exponentially decays with f^2, the second exponential factor has its highest values along the lines $\frac{f}{\omega} = v$ and decreases towards the normals of the line. Compared to the trajectory of the S-wave ($x/t = v$) these characteristic lines have the

inverse slope ($\omega/f = v^{-1}$) in f-w representation.

5.3.2 Filter kernel design

We want to use the results from the analytic case, to remove fast traveling seismic waves from a discrete displacement field.

The Nyquist-Shannon sampling theorem tells us that for a discrete displacement field, with N_x samples in space and N_t samples in time, a resolution of Δx and Δt and a domain of $[0, W] \times [0, T]$, we can represent the wavenumbers $\omega_j = j \cdot 1/(2\Delta x)$ and frequencies $f_i = i \cdot 1/(2\Delta t)$, $\forall j \in [0, N_x]$, $\forall i \in [0, N_t]$. The discrete fourier transformation (DFT) of the displacement field gives us the matrix of complex coefficients D_{ji} for the represented wavenumbers and frequencies.

To erase the characteristic lines $\omega - f/v$ in the coefficient matrix we define a mask based on a cone that includes the minimal v_{min} and maximal velocity v_{max} of the waves we want to filter,

$$M^s_{ij} = \begin{cases} 0, & \text{if } v_{min} < |f_i|/|\omega_j| < v_{max} \\ 1, & \text{else.} \end{cases} \tag{5.31}$$

This cone also erases coefficients close to $f = 0$, $\omega = 0$ and affects the permanent displacement. To avoid this we add a buffer to the mask, to keep the parameters of the permanent displacement. Based on the results in Equation (5.25), we define the size of the buffer depending on the source size σ_r and source duration σ_t, with relaxation variables for frequency c^f_{rel} and wavenumber c^ω_{rel},

$$M^p_{ij} = \begin{cases} 1 & \text{if } w_j^2 \frac{\sigma_r^2}{2} < c^\omega_{rel} \\ 1 & \text{if } f_i^2 \frac{\sigma_t}{8} < c^f_{rel} \\ 0 & \text{else.} \end{cases} \tag{5.32}$$

The relaxation variables define how aggressive we filter the coefficients in the overlapping region of both waves. While high values conserve the permanent displacement, low values lead to higher cancellation of seismic waves.

The final filter kernel is the logical disjunction of both masks,

$$M_{ij} = M^s_{ij} \vee M^p_{ij}. \tag{5.33}$$

For which we get the DFT of the filtered displacement $\overline{D}_{ji}$ with the

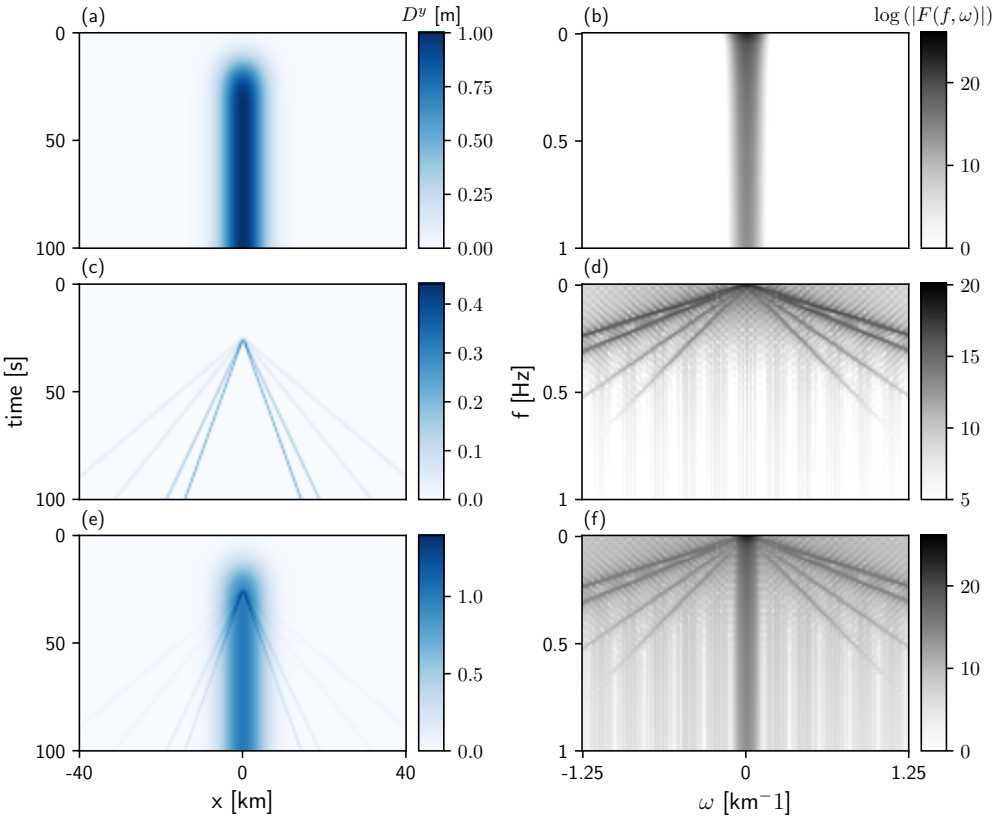


Figure 5.2.: Wavefield (a,c,e) and frequency-wavenumber representation (b,d,f) for a synthetic benchmark. We show the representations for a permanent displacement (a,b), several fast traveling S-waves (c,d) and the superposition of both (e,f).

Hadamard product of the unfiltered displacement and the filter kernel,

$$\overline{D} = M \circ D = (M_{ji} \cdot D_{ji})_{ji}. \tag{5.34}$$

Filtering a synthetic example

To test the filter, we look at a discrete synthetic example based on the definitions from Equation (5.27) and Equation (5.22). In Figure 5.2 we show the displacement field and the corresponding discrete fourier transformation (DFT) of a large slowly lifting displacement in the center of the domain (a,b), several fast S-waves (c,d) and the superposition of both (e,f).

We look at the displacement in the area $[-40, 40]$ km and for 100 s. The

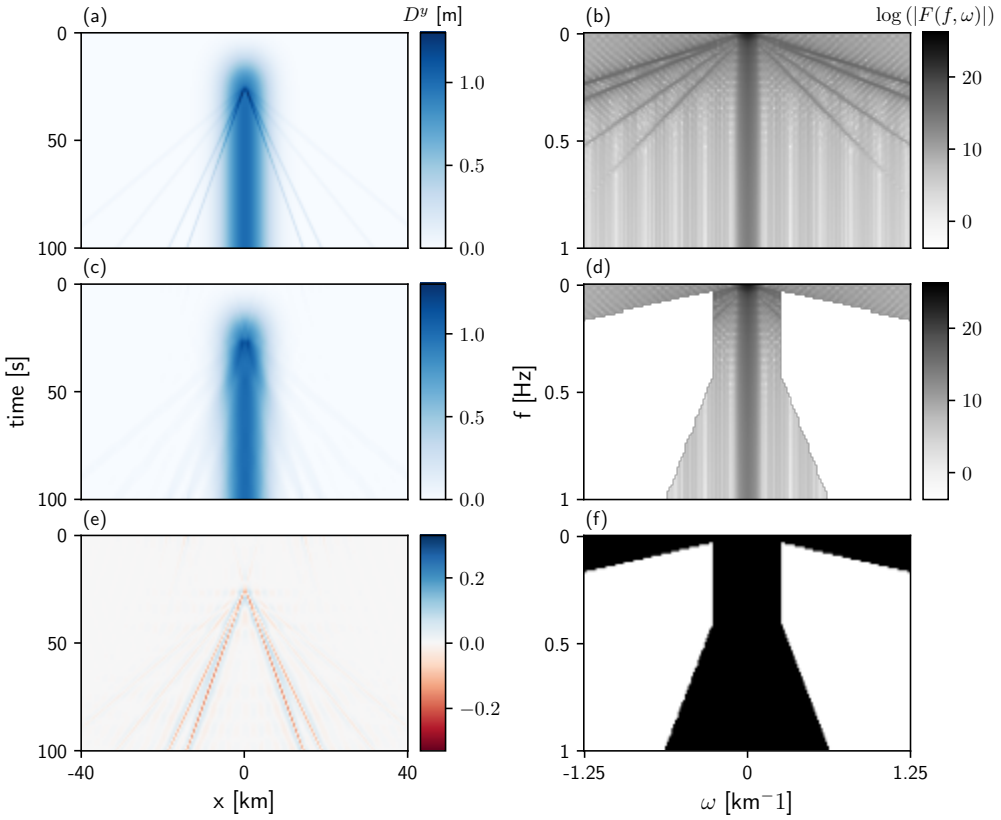


Figure 5.3.: Wavefield (a,c,e), f-w representation (b,d) and filter kernel (f) for the filtering process of the synthetic displacement. From the f-w representation of the unfiltered displacement (b), the filter kernel (f) erases characteristic lines of seismic waves (d). In the wavefield (a) this process removes seismic waves (c). The removed waves after the filtering are shown in (e).

permanent displacement is set for a source width of $L = 20\,\mathrm{km}$ and a duration of $\sigma_t = 40\,\mathrm{s}$ with the highest rate at $t_0 = 15\,\mathrm{s}$. S-waves are initiated at $t_0 = 25\,\mathrm{s}$, move with speeds $v \in [1.0, 3.0]\,\mathrm{km\,s^{-1}}$ and have amplitudes in the range of $[0.25, 0.4]\,\mathrm{m}$.

We record the displacement field for 100 s and a time-step of $\Delta t = 0.5\,\mathrm{s}$, the resulting highest represented frequency is $1\,\mathrm{s}^{-1}$. The spacial discretization is set on the interval $[-40, 40]\,\mathrm{km}$, with $\Delta x = 800\,\mathrm{m}$ and a represented wavenumber of $\omega = 1.25\,\mathrm{km}^{-1}$. These resolutions are chosen to resemble the typical output from an earthquake simulation.

Looking at the logarithmic norms ($\log|F(f,\omega)|$) of the DFTs and the

analytic f-w representation shows us the characteristic properties which were the motivation for the design of the filter kernel. The coefficients of the permanent displacement are dominant along the lines $f = 0$ and $\omega = 0$ (b) and those of the S-waves are characterized by the lines $\frac{f}{\omega} = v$ (d). Both share high norms close to the origin, which tells us that we are not able to separate them perfectly in a given combined displacement field (f), without knowing their exact distribution.

In order to filter the displacement, we set the filtered velocities from $v_{min} = 0.7\,\mathrm{km\,s^{-1}}$ to $v_{max} = 8\,\mathrm{km\,s^{-1}}$. Based on the analytically derived norm in Equation (5.25), we set $c^{\omega}_{rel} = 2.5 \times 10^{-8}$ and $c^{f}_{rel} = 2.5 \times 10^{-9}$ for the relaxation parameters. Figure 5.3 shows a comparison of the combined displacement (a) to the displacement after we applied the filter (c). The filter mask (f) erases the characteristic lines of the S-wave from the f-w coefficients, the coefficients around $f = 0$ and $\omega = 0$ are left untouched (b,d). In the unfiltered displacement the relative L_2 norm of the S-waves was 0.11. After we applied the filter the norm more than halved for the remaining artifacts to 0.052 (e). While we are not able to erase all artifacts of S-waves, we can significantly reduce their contribution to the filtered displacement.

5.3.3 Filtering seismic waves in discrete displacement fields

In Section 5.3.2 we can only compute the effect of the filter, because we know the analytical description of all waves in the superpositioned displacement field. In case of numerical results the only information we have, besides the displacement field, are the geometrical and material properties of the model.

To find the right relaxation parameters for the filter we propose to look at particular seismic waves, which can be clearly identified in the displacement field. Having found a seismic wave we gradually increase the strength we filter with (i.e. we decrease the values of the relaxation parameters), until the wave is removed from the displacement field. To see the effects of the relaxation parameters, we look at the displacement field of a one dimensional dynamic rupture simulation situated in a subduction zone [143]. Figure 5.4 shows the displacement fields before (a) and after (b) we filtered for velocities from $0.7\,\mathrm{km\,s^{-1}}$ to $8\,\mathrm{km\,s^{-1}}$ with relaxation parameters $c^{\omega}_{rel} = 1.0 \times 10^{-5}$ and $c^{f}_{rel} = 2.5 \times 10^{-7}$. The DFT of the field agrees with our analytical observations (d): Dominant coefficients of the permanent displacement are along the lines $\omega = 0$ and $f = 0$. The characteristic lines of seismic-waves also appear but show diffusion from the inhomogeneous material in the subduction zone. The difference of filtered and unfiltered

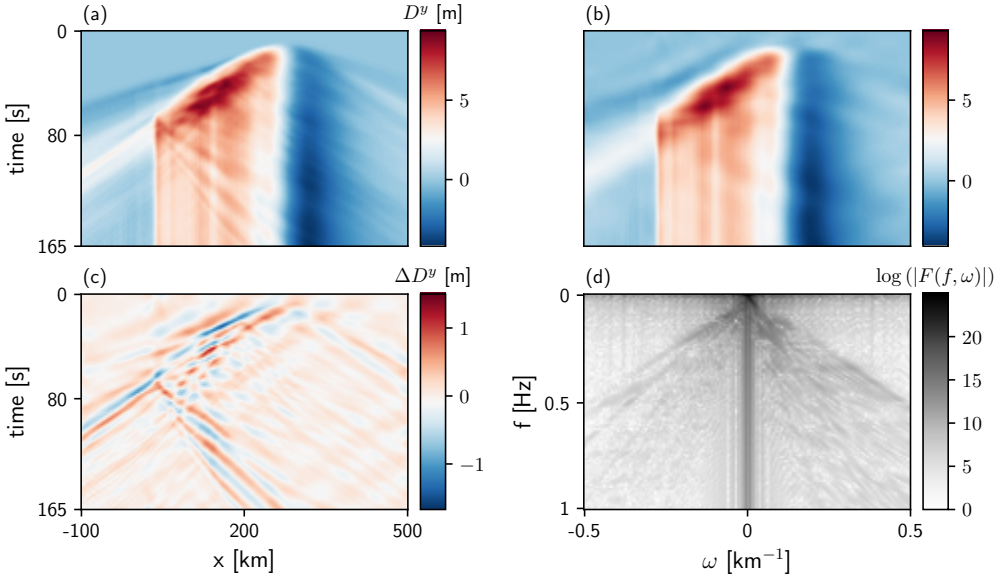


Figure 5.4.: Wavefield for the unfiltered (a) and filtered displacement in a subduction zone (b). The difference (c) shows that we can erase seismic waves from the field. The f-w representation of the field conforms with our analytical model (d).

displacements show that we erase a large part of the seismic waves for the displacement field (c).

To adjust the relaxation parameters, we use the fact that we know the speed of the P-wave, from the inhomogeneous material parameters of the earthquake simulation. Figure 5.5 shows seismograms for the particle velocity of the model for the range $[100, 300]$ km in a 10 km distance before (black) and after (red) we applied the filter (a). We see that initially the evolution of the permanent displacement (located between the two green lines) is interfered by several seismic-waves. In the unfiltered velocity field the P-wave moves with a speed of $4.429\,\mathrm{km\,s^{-1}}$. After applying the filter it is no longer present in the field (b, blue line). Accordingly, the seismic waves interfering the evolution of the dynamic displacement have reduced.

This is no longer the case, if we set the relaxation parameters to higher values and decrease the level of filtering. In Figure 5.6 we show the P-wave for the parameter sets increased by a factor of 10 ($c^{\omega}_{rel} = 1.0 \times 10^{-4}$, $c^{f}_{rel} = 2.5 \times 10^{-6}$) (a) and a factor of 100 ($c^{\omega}_{rel} = 1.0 \times 10^{-3}$, $c^{f}_{rel} = 2.5 \times 10^{-5}$) (b). We see that in both cases higher relaxation parameters lead to a reduced filtering of the P-wave and it remains present in the displacement field.

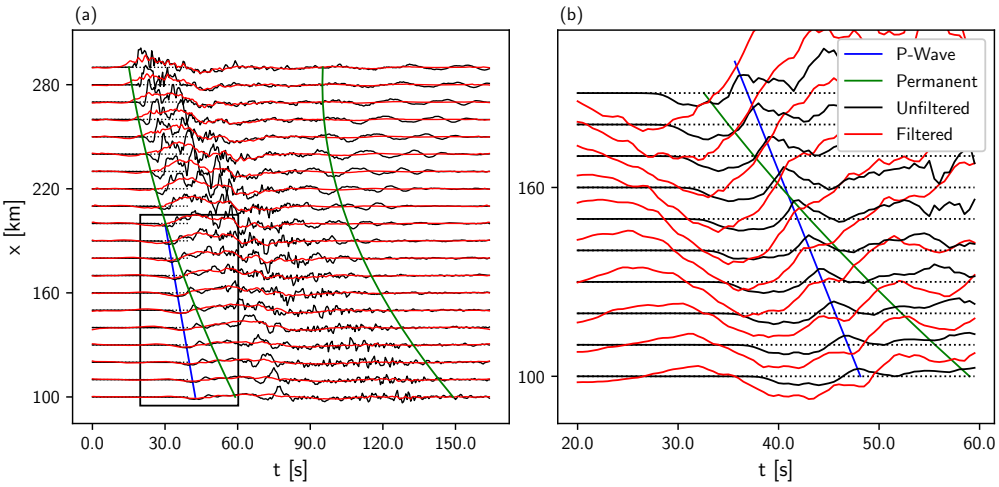


Figure 5.5.: Velocity components of seismograms for the unfiltered (black) and filtered displacement (red) in a subduction zone. The blue line indicates the trajectory of the peak of the P-wave, the green lines the area, the permanent displacement is generated in. (a) shows the seismograms in the domain $[100, 300]$ km up to 150 s. (b) the seismograms focusing on the P-wave. The filtering process erases the P-wave in the filtered displacement.

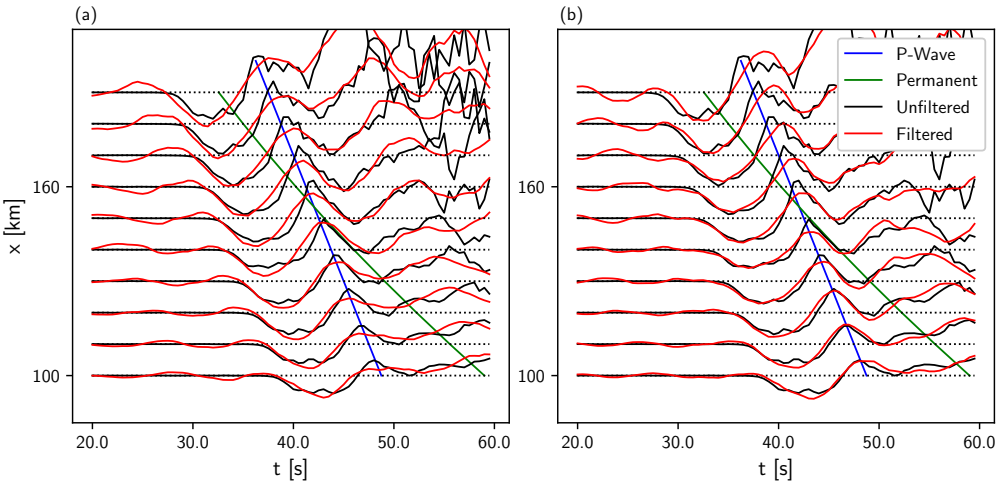


Figure 5.6.: Velocity components of seismograms focusing on the P-wave, for the unfiltered (black) and filtered displacement (red) in a subduction zone. Relaxation parameters are increased by a factor of 10 (a) and 100 (b). In both cases the P-wave is not fully filtered.

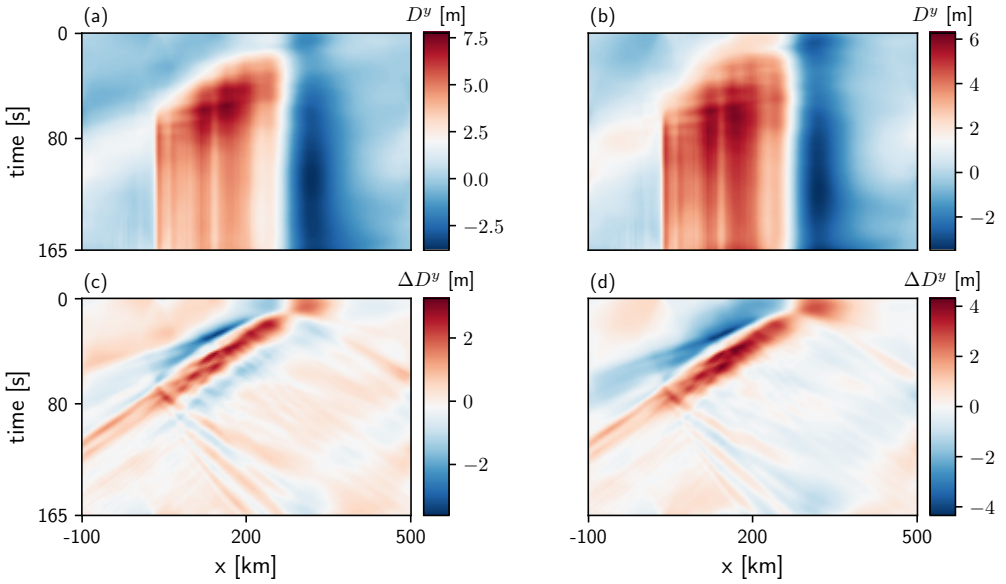


Figure 5.7.: Displacement field of an earthquake in a subduction zone. The relaxation parameters have been decreased by factors 10 (a) and 100 (b). The values lead to a high deformation of the permanent displacement (c,d).

If we increase the level of filtering with lower values for the relaxation parameters, we also increase the influence of the filter on the permanent displacement. Figure 5.7 shows the displacement field after we filtered with decreased parameters sets for division factors of 10 ($c^{\omega}_{rel} = 1.0 \times 10^{-6}$, $c^{f}_{rel} = 2.5 \times 10^{-8}$) (a) and 100 ($c^{\omega}_{rel} = 1.0 \times 10^{-7}$, $c^{f}_{rel} = 2.5 \times 10^{-9}$) (b). Both displacement fields are highly deformed and loose important characteristics as the maximal displacement (c,d). We also see that at the temporal and spatial boundaries of the displacements spurious waves are generated.

5.3.4 Going to two dimensions

When we move our concept to two spatial dimensions, the f-w spectrum of the displacement field is now defined in two wavenumbers ω_1 and ω_2 and the frequency f. Under the assumption that seismic waves and permanent displacement evolve approximately radial symmetric around a center, we can transform the observations we made in one dimension into two dimensions.

We take the analytic test setting from Section 5.3.1 and transform it to a radial symmetric solution in two dimensions, by taking it constant along

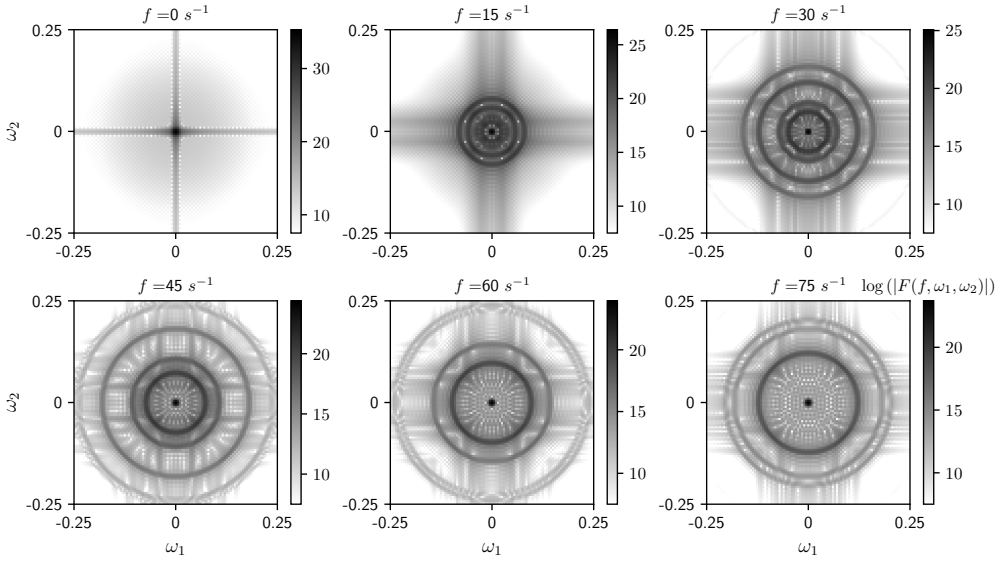


Figure 5.8.: f-w representation of the radial symmetric synthetic displacement field, for frequencies $f = 0, 15, 30, 45, 60$ and $75\,\mathrm{s}^{-1}$. For the displacement the significant coefficients are located along the axes ($f = 0, \omega_1 = 0$), ($f = 0, \omega_2 = 0$) and ($\omega_1 = 0, \omega_2 = 0$) in the f-w representation. The coefficients for seismic-waves appear as circles at $\omega_r = \frac{f}{v}$.

the angle in polar coordinates. The f-w representation of the example for selected frequencies is shown in Figure 5.8: The significant coefficients of the permanent displacement in the f-w spectrum are now along the axes ($f = 0, \omega_1 = 0$), ($f = 0, \omega_2 = 0$) and ($\omega_1 = 0, \omega_2 = 0$). Characteristic lines of seismic waves have turned into cones $f/\|\omega\|_2 = v$, where $\|\omega\|_2 = \sqrt{\omega_1^2 + \omega_2^2}$ is the spectral norm of the two wavenumbers.

We can use these observations to design the two dimensional filter kernel on the base of its one-dimensional counterpart. In Equation (5.31) we replace the wavenumber ω with the spectral norm of the wavenumbers,

$$M^s_{ijk} = \begin{cases} 0, & \text{if } v_{min} < |f_i|/|(\|\omega\|_2)_{jk}| < v_{max} \\ 1, & else. \end{cases} \tag{5.35}$$

For the permanent displacement we keep the three lines along the axes in

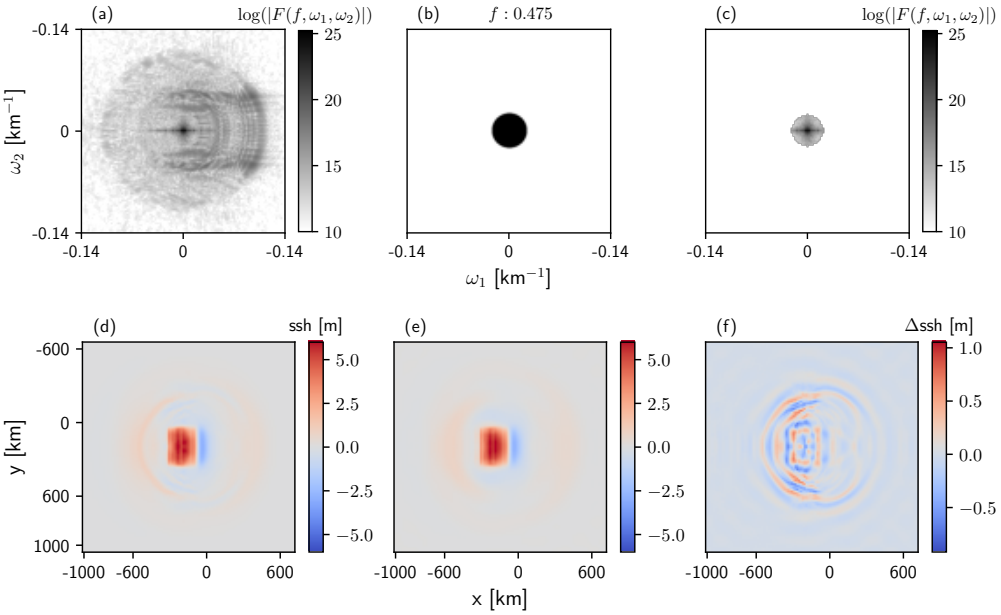


Figure 5.9.: Filtering process for the model set-5. From the f-w representation at $f = 0.457\,\mathrm{s}^{-1}$ (a) we erase the cone belonging to seismic waves with the two dimensional filter mask (b) to get the filtered f-w representation (c). The filter removes seismic waves from the displacement field at $t = 150\,\mathrm{s}$ (d) and damps imprinting waves in the filtered displacement field (e) of up to 1 m amplitude (f). Adapted from Figure S1 & S2 in [140].

the representation,

$$M^p_{ijk} = \begin{cases} 1, & \text{if } (w_1)^2_j \frac{L^2}{2} < c^\omega_{rel} \wedge (w_2)^2_k \frac{L^2}{2} < c^\omega_{rel} \\ 1, & \text{if } (w_1)^2_j \frac{L^2}{2} < c^\omega_{rel} \wedge f^2_i \frac{t_c}{8} < c^f_{rel} \\ 1, & \text{if } (w_2)^2_k \frac{L^2}{2} < c^\omega_{rel} \wedge f^2_i \frac{t_c}{8} < c^f_{rel} \\ 0, & \text{else.} \end{cases} \tag{5.36}$$

The final mask is build equivalently to Equation (5.33).

In Figure 5.9 we see the filtering process for the displacement field of model set-5, that we introduce in Chapter 6 We filter the velocities $v_{min} = 0.7\,\mathrm{km\,s}^{-1}$ and $v_{max} = 8.0\,\mathrm{km\,s}^{-1}$, the relaxation parameters are set to $c^w_{rel} = 5.0 \times 10^{-7}$ and $c^f_{rel} = 1.25 \times 10^{-9}$. (a) shows the f-w representation of the displacement field at the frequency $f = 0.475\,\mathrm{s}^{-1}$. Clearly visible as circles

are the conic sections in (a) with the coefficients belonging to seismic waves. At the origin we see the line ($\omega_1 = 0, \omega_2 = 0$) of the permanent displacement.

We follow the one dimensional workflow and apply the mask, for every sampled frequency (b) to the f-w representation to get the filtered f-w representation (c). From the unfiltered displacement field (d), seismic waves have been damped in the filter displacement field (e). The difference shows us that we removed waves up to 1 m amplitude and especially waves that where imprinting the permanent displacement. Remaining seismic waves in the filtered displacement can simply be erased by cropping the displacement field.

5.4 Conclusion and discussion

The Fourier filter we presented in this chapter is build under several mathematical approximations:

1. The permanent displacement is close to the axes in f-w-representation.
2. The fast seismic waves we want to filter are characterized by cones $f/\|\omega\|_2 = v$ (and lines in one dimension).
3. The displacement field is approximately radial symmetric.

From these assumptions we derived a filter kernel based on parameters c^f_{rel} and c^ω_{rel}. These parameters balance between the factor with which seismic waves are damped and with which the permanent displacement is kept. In case of displacement fields that were generated by a seismic code, we do not know suiting values a priori. Instead we have to test multiple sets, until we found a proper configuration.

The need to use the filter can be seen in a comparison of wavefields, for the set-3 benchmark that we introduce in Chapter 6. In Figure 5.10 we show the tsunami at the end of the source after 200 s, at 1400 s and 2000 s for filtered and cropped, unfiltered and cropped and unfiltered displacement fields.

In case of the filtered displacement we used the parameters $c^w_{rel} = 5.0 \times 10^{-7}$ and $c^f_{rel} = 1.25 \times 10^{-9}$ for the filtering mask. For the cropped displacement field, we take the unfiltered displacement in the area $\Omega = [0, 450] \times [-300, 300]$ and set all values outside to zero.

We see significant differences for the tsunamis of the three sources. After the source finished at 200 s, the tsunami to the unfiltered source contains seismic waves, which erroneously become tsunami waves at 1400 s. In the

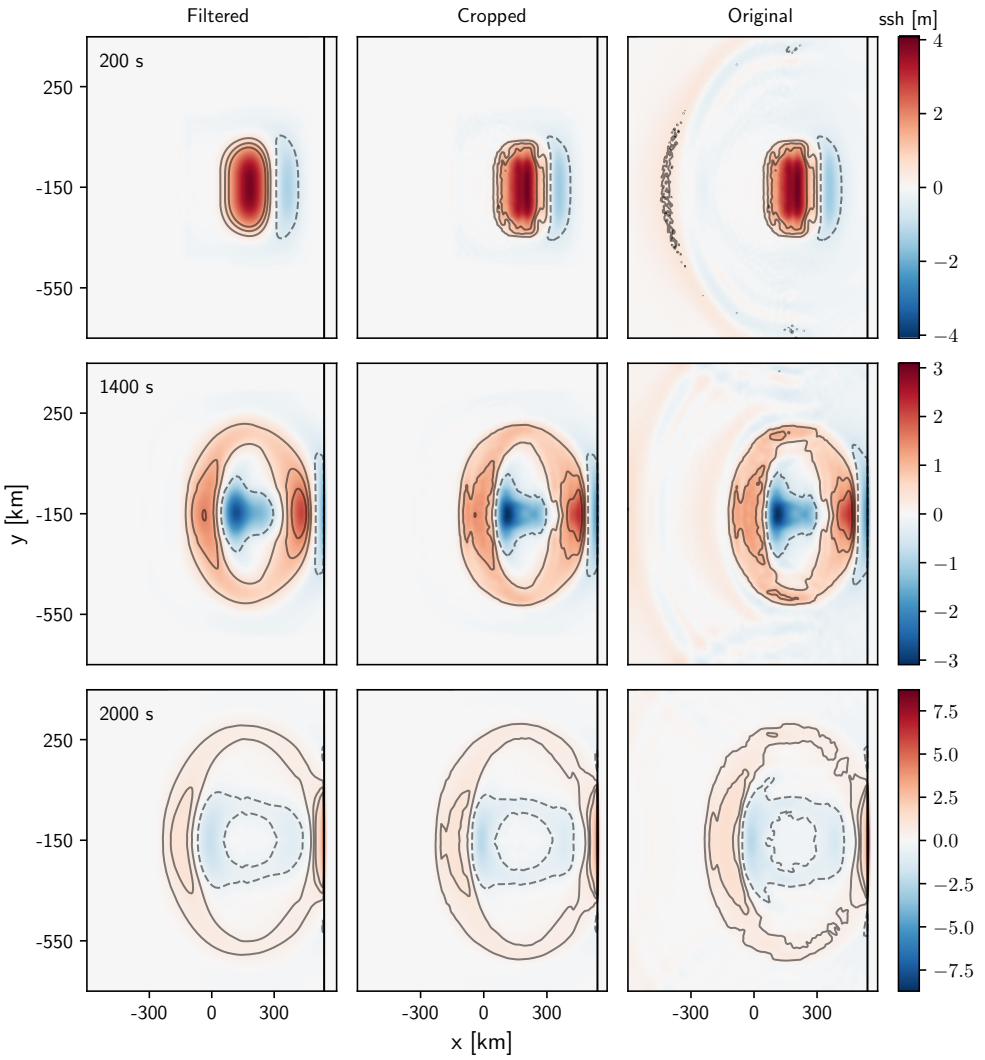


Figure 5.10.: Wavefields for set-3 after 200 s, 1400 s and 2000 s for filtered, cropped and unfiltered displacement fields.

cropped displacement field, the fast traveling seismic waves that do not imprint the permanent displacement have been erased. However we still see the effect of seismic waves that imprint the permanent displacement on the tsunami 1400 s, compared to the tsunami of the filtered displacement. After 2000 s we can still see a significant influence of the fast seismic waves on the tsunami to the unfiltered source. Cropped and filtered displacement show slight differences.

CHAPTER

6

The ASCETE framework

6.1 Introduction

The linking methods we presented in Chapter 5 are part of the subduction earthquake tsunami linking pipeline that was presented by Madden et al. [92] and used for an extended parameter study by Wirp et al. [140]. In both publications simulations of the movement of continental plates in subduction zones, of dynamic rupture earthquake events and of tsunamis are linked and used to isolate the effects of parameters in earthquake models on the subsequent tsunamis. In this chapter we revisit the most important findings and refer readers to the corresponding publications for details in Section 6.4 and Section 6.5. The core metrics of the considered models are accumulated for the seafloor-perturbation, the tsunami and the inundation process in Table 6.1, Table 6.2 and Table 6.3.

The work that we present in this chapter has been created in a joint effort. Results of models for subduction zones and for dynamic rupture earthquakes were performed by the first authors or other co-authors of the respective papers. All contributions that are tsunami related, as the generation of the sea floor perturbation, the application of the Fourier filter and the simulation and evaluation of tsunamis models were created as part of this dissertation. In order to show the full picture, we give an overview of all findings.

Finally, we use the models to compare differences between time dependent and time independent sourcing in Section 6.6.

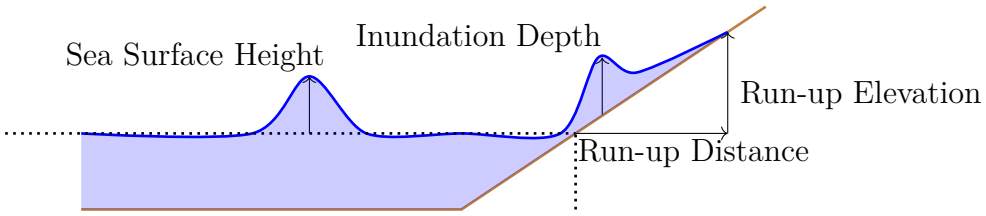


Figure 6.1.: Sketch of the most important definitions to describe a tsunami and the inundation of the coast.

6.2 Metrics and nomenclature

Before we look at studies of linked earthquake tsunami models, we define a common nomenclature and look at the most important metrics for the analysis of tsunamis.

Nomenclature and tsunami metrics

Figure 6.1 shows a sketch of the definitions we use to describe a tsunami and its impact on the coast [129]. We call the height of the tsunami relative to the flat surface of an ocean *sea-surface height* (ssh). The sea-surface height is the difference of the water column h of a tsunami to the water-column in the resting case $max(0, -b)$, where b is the bathymetry.

On all points on the coast and all areas that were initially dry (usually where $b > 0$), we call the height of the water column *inundation depth*. At each point on the coast, we will look at its maximal value over all time-steps, called the *highest inundation depth*. The first time a location on the coast is inundated is the first time-step where the inundation depth is higher than zero, called the *inundation time*. With the highest inundation depth and the inundation time of a coast we get a temporal and spatial description for the impact of the tsunami on the coast. The first time any point at the coast is inundated is the *arrival time* of the tsunami. To see how deep a wave propagates on land, we compute the *run-up distance*. Along each normal of the coastline it is the distance of the deepest point the land was inundated. The *run-up elevation* is the value of the topography at that point. In case of analytic topographies, the run-up elevation can directly be derived from the run-up distance.

Metrics for the seafloor-perturbation

The simplest way to characterize a sea floor perturbation is given by its geometrical properties. We will look at the duration σ_t, the size[1] σ_r and the range $[\min \Delta b, \max \Delta b]$ of the potentially time dependent source.

For a water depth $H = -b$, Abrahams et al. introduce the measure [1]

$$\sigma_p = \frac{\sigma_r}{\sigma_t \sqrt{gH}}, \tag{6.1}$$

which gives us a relation between the size of the source and the distance the tsunami travels in the duration of the source. For $\sigma_p \gg 1$ it is assumed that time independent sourcing is enough to initialize the model.

As measure how much energy is added by the source $\Delta b(x, y, t)$, we compute its potential energy, also called tsunami potential (TP) [97],

$$\mathrm{TP}\left(\Delta b(x, y, t)\right) = \int_\Omega \frac{\rho \cdot g}{2} \Delta b(x, y, t)^2 dx dy, \tag{6.2}$$

where $g > 0$ is the gravitational acceleration and $\rho > 0$ the density of seawater. From Equation (5.2) we know that in case of time independent linking, the tsunami potential at the end of the earthquake is the initial potential energy of the tsunami. As we initialize with zero velocities, it is also equal to the constant total energy of the system.

Energy metrics

To quantify the wave fields over the whole domain we look at their total energy (TE), potential energy (PE) and kinetic energy (KE). In our linking approaches the only potential energy that can be converted to kinetic energy is given by the change in the sea-surface. We thus always compute the potential energy in relation to the resting lake. The TE, PE and KE can be computed as

$$\mathrm{PE}\left(q\left(\vec{x,t}\right)\right) = \int_\Omega \frac{\rho g}{2} H^2 d\Omega \tag{6.3}$$

$$\mathrm{KE}\left(q\left(\vec{x,t}\right)\right) = \int_\Omega \frac{\rho}{2} h \left(u^2 + v^2\right) d\Omega \tag{6.4}$$

$$\mathrm{TE} = \mathrm{PE} + \mathrm{KE}, \tag{6.5}$$

[1] Here we use the diameter of the circumcircle around the source.

Several studies suggest that the total energy and the energy distribution are indicators for the maximal inundation depth and the run-up distance [64, 109, 145].

6.3 Set-up for the tsunami models

For all linked earthquake-tsunami models of the ASCETE framework, we use the same base definition for the tsunami model. The set-up incorporates a flat seafloor at 2 km depth, with a linearly sloping beach. The position of the slope is given by an intersect with the flat seafloor at the beach toe x_0 and an ascent of 0.05,

$$b(x,y) = \begin{cases} 0.05(x - x_0) - 2000 & \text{for } x > x_0 \\ -2000 & \text{else.} \end{cases} \tag{6.6}$$

The coastline at x_{coast} is located at 40 km away from x_0. Initially water-height and velocities are set to the sea at rest. For 2 km water depth the analytic tsunami velocity is $141.5\,\mathrm{m\,s^{-1}}$. This relatively simple bathymetry is adapted from the single wave on a sloping beach benchmark (see Section 3.3), to reduce the influence of the bathymetry to a minimum.

In all simulations we initialize the seafloor perturbation field according the time dependent method from Section 5.1. Displacement fields are filtered with the Fourier filter we introduced in Section 5.3.

6.4 Earthquake-tsunami models

We first look at two synthetic linked earthquake tsunami models (denoted by et-a and et-b)[92]. Both earthquake models are placed in a layer of homogeneous material, to resemble the subduction zone in an oceanic crust. Faults are planar, 200 km long, 35 km deep and have a dip of 16°.

The strength of both faults is determined by a linear slip-weakening friction law, which we describe in more detail in Section 7.2.1. Linear slip-weakening models the weakening of a fault as linear relation between slip and fault strength: Initially a predefined *static strength* is used to model the strength of the fault, for shear stresses below no slip is generated. As soon as stresses exceed this threshold, the fault starts to move and slip and seismic waves are initiated. From an initial strength, defined by the *static friction coefficient*, the fault strength linearly decreases with the slip. As

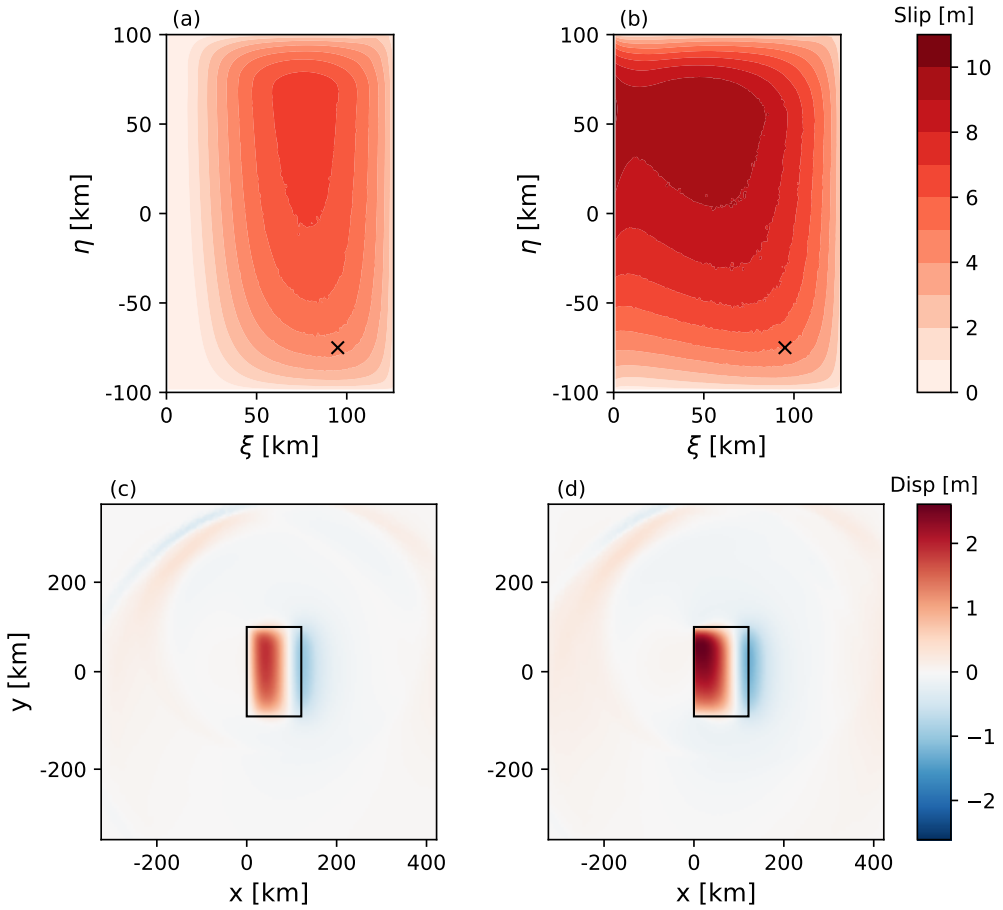


Figure 6.2.: On fault accumulated slip (a,b) (η: along-strike and ξ: down-dip) and unfiltered final displacement fields (c,d) for et-a (a,c) and et-b (b,d). Adapted from Figure 3 in [92].

the total slip reaches the *critical distance*, the fault strength remains in a constant state defined by the *dynamic friction coefficient*.

In model et-b the static fault strength linearly increases with the depth of the fault. For model et-a, the same setting is used at 15 km and below. At positions above the static fault strength is smoothly increased up to 8 MPa at the surface (compared to 0.4 MPa for model et-b). The resulting accumulated slip on the fault and the final vertical displacement field for both models are presented in Figure 6.2.

Nucleation is initiated in the south-east corner of the fault at 26 km depth, from this point the rupture propagates unidirectional towards north. For

Table 6.1.: Maximal and minimal uplift ($\max \Delta b$, $\min \Delta b$), final uplift ($\max \Delta b(T_{max})$, $\min \Delta b(T_{max})$) and maximal ($\max \mathrm{TP}(T_{max})$) and final ($\mathrm{TP}(T_{max})$) tsunami potential for the earthquakes to all five models.

	et-a	et-b	set-3B	set-5	set-6
$\max \Delta b$ [m]	2.34	3.10	5.88	8.60	8.21
$\min \Delta b$ [m]	−1.03	−1.27	−2.15	−3.16	−3.22
$\max \Delta b(T_{max})$ [m]	2.13	2.56	4.55	5.93	5.49
$\min \Delta b(T_{max})$ [m]	−0.99	−1.27	−2.03	−3.03	−3.03
$\max \mathrm{TP}(T_{max})$ [PJ]	0.19	0.37	4.77	9.96	9.34
$\mathrm{TP}(T_{max})$ [PJ]	0.16	0.32	3.13	6.95	6.18

et-a the higher fault strength stops slipping close to the surface ($\eta = 0$ in Figure 6.2 a) and leads to a blind rupture (the seafloor remains intact in Figure 6.2 c). The lower fault strength of et-b, produces high slip which propagates to the surface ($\eta = 0$ in Figure 6.2 b) and leads to a breaching of the surface (Figure 6.2 d). The main characteristics of the displacement fields, which are important for the subsequent tsunami, are presented in Table 6.1. The range of the uplift is larger for et-b, with a minimum of $-1.27\,\mathrm{m}$ and a maximum of $3.1\,\mathrm{m}$, compared to $-1.03\,\mathrm{m}$ and $2.34\,\mathrm{m}$ for et-a. We see the same result at the end of the earthquake. Uplifts have a wider range for the surface breaching rupture ($[-1.27\,\mathrm{m},\ 2.56\,\mathrm{m}]$), as for the blind case ($[-0.99\,\mathrm{m},\ 2.13\,\mathrm{m}]$). Those differences are also reflected in the maximal and final tsunami potential, which are both higher ($0.37\,\mathrm{PJ}$ and $0.32\,\mathrm{PJ}$) in et-b than in et-a ($0.19\,\mathrm{PJ}$ and $0.16\,\mathrm{PJ}$).

Tsunami Results

For both models we use the resulting displacement fields to generate the sea floor perturbation with Tanioka's method, as introduced in Section 5.2. In order to remove waves that imprint the displacement, results are filtered with the Fourier filter, from Section 5.3 for parameters $c^w_{rel} = 5.0 \times 10^{-7}$ and $c^f_{rel} = 1.25 \times 10^{-9}$. The toe of the beach is placed at $x_0 = 200\,\mathrm{km}$, such that the coastline is located at $x_{coast} = 240\mathrm{km}$. The domain is set to $\Omega = [-400\,\mathrm{km}, 400\,\mathrm{km}]^2$, which is sufficient to avoid any boundary effects. Highest mesh resolution is at the coasts with a mesh size of $12.2\,\mathrm{m}$. We source the tsunami time-dependent, following the method presented in Section 5.1. Figure 6.3 shows the wavefield for the tsunami to the blind

Table 6.2.: Maximal water height (max ssh), velocity (vel), average total ($\overline{\text{TE}}$), potential ($\overline{\text{PE}}$) end kinetic energy ($\overline{\text{KE}}$) of the tsunamis to all five models.

	et-a	et-b	set-3B	set-5	set-6
max ssh [m]	2.049	2.544	4.308	5.745	5.209
vel [m s^{-1}]	156.25	140.62	146.48	165.39	163.53
$\overline{\text{TE}}$ [PJ]	0.162	0.302	3.183	6.795	6.080
$\overline{\text{PE}}$ [PJ]	0.077	0.148	1.463	3.200	2.840
$\overline{\text{KE}}$ [PJ]	0.086	0.155	1.720	3.595	3.239

rupture (a) and the tsunami to the surface breaching rupture (b). We show time-steps during the initialization at 100 s, around the time of first inundation at 1200 s and close to the highest inundation at 1600 s. We see that the blind rupture leads to a smooth wave, while the surface-breaching rupture produces a jump after the tsunami initialization. Around the displaced seafloor, waves evolve in elliptic shape. We can see asymmetries in the wave height, caused by the unidirectionality of the ruptures. After the wave arrives at the coast after 1210 s (et-a) and 1220 s (et-b), the amplitude of the wave amplifies (1600 s).

The most important metrics of both wavefields are summarized in Table 6.2. For both tsunamis the maximal sea surface height agrees with the highest value of the seafloor perturbation. The peak of the waves for the blind rupture propagates faster with 156 m s^{-1} then the peak of the wave for the surface-breaching rupture with 140 m s^{-1}. The average total energy is higher for the wave of the surface breaching than for the blind rupture (0.162 PJ, 0.302 PJ), which is also true for the kinetic and potential energy.

To look at the waves in more detail, we show cross sections for the same events as the wavefield in Figure 6.4 (a,b,c) along the lines $y = 0$, $y = -150$ and $y = 150$ km. Additionally we show a time series of the inundation depth for points placed 10 m inland on the coast (d).

At $t = 120$ s we again see that the blind rupture leads to a smooth wave, while the surface-breaching rupture incorporates a jump (Figure 6.4 a). The tsunami of the surface-breaching rupture has lower minimal and higher maximal sea surface heights [−1.05 m, 2.4 m], then in the case of the blind rupture [−0.75 m, 1.6 m], which agrees with the ranges we observed for the displacement fields. At 1210 s the wave from the blind rupture arrives at the coast, 10 s later the wave of the surface-breaching rupture (Figure 6.4 b,d). The extrema of both waves ([−0.85 m, 0.85 m] for et-a and [−0.95 m, 1.1 m]

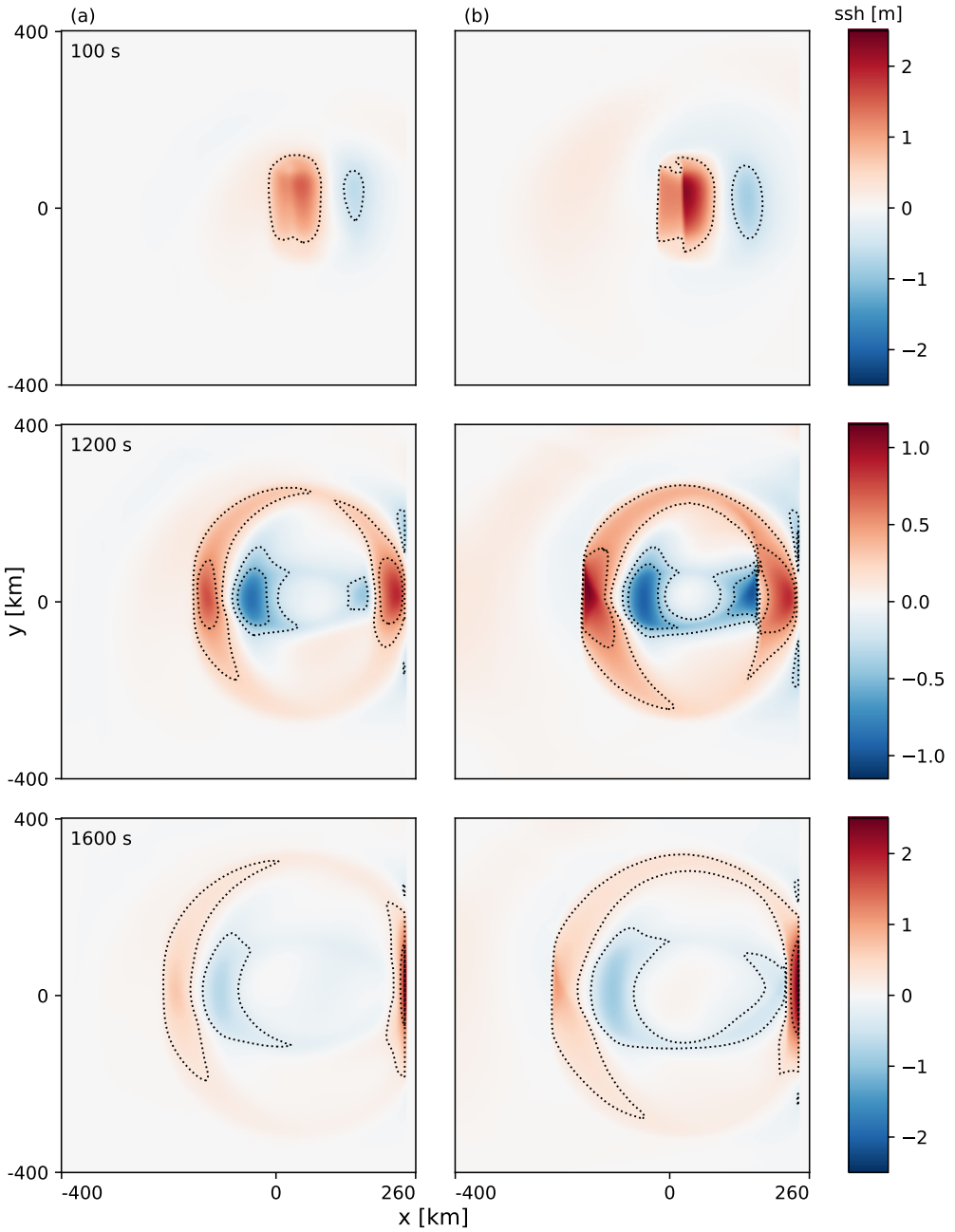


Figure 6.3.: Wavefield of the tsunamis to model et-a and et-b during the sourcing at 100 s, the arrival on the coast at 1200 s and close to the highest inundation at 1600 s. Adapted from Figure 5 in [92].

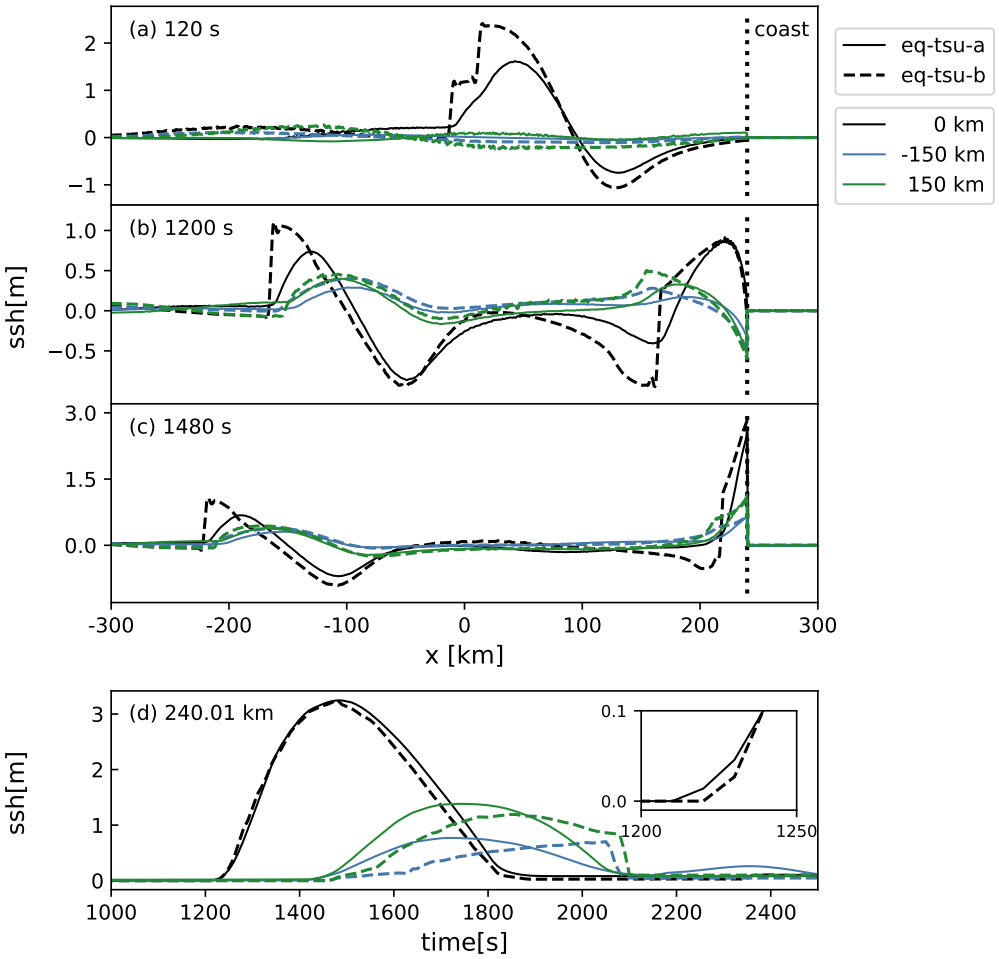


Figure 6.4.: Cross sections of the wavefield to the tsunamis in models et-a and et-b at 120 s (a), 1200 s (b) and 1480 s (c) along the lines y=0 km, 150 km and 150 km. Time series of the inundation depth 10 km on land (d). Adapted from Figure 6 in [92].

for et-b) differ less at this point. The peak of inundation height is reached at around 1480 s (Figure 6.4 c,d) in both cases. At the coast, the wave for the blind rupture produces a lower peak, than the wave for the surface-breaching rupture ([−0.7 m, 2.5 m] for et-a and [−0.9 m, 2.8 m] for et-b).

To look at the inundation process, we show the inundation time (a,b,c) and the highest inundation depth (d,e,f) in Figure 6.5. Additionally the most important metrics are summarized in Table 6.3. Both models show similar timings for the inundation, as the wave arrives first in the center of

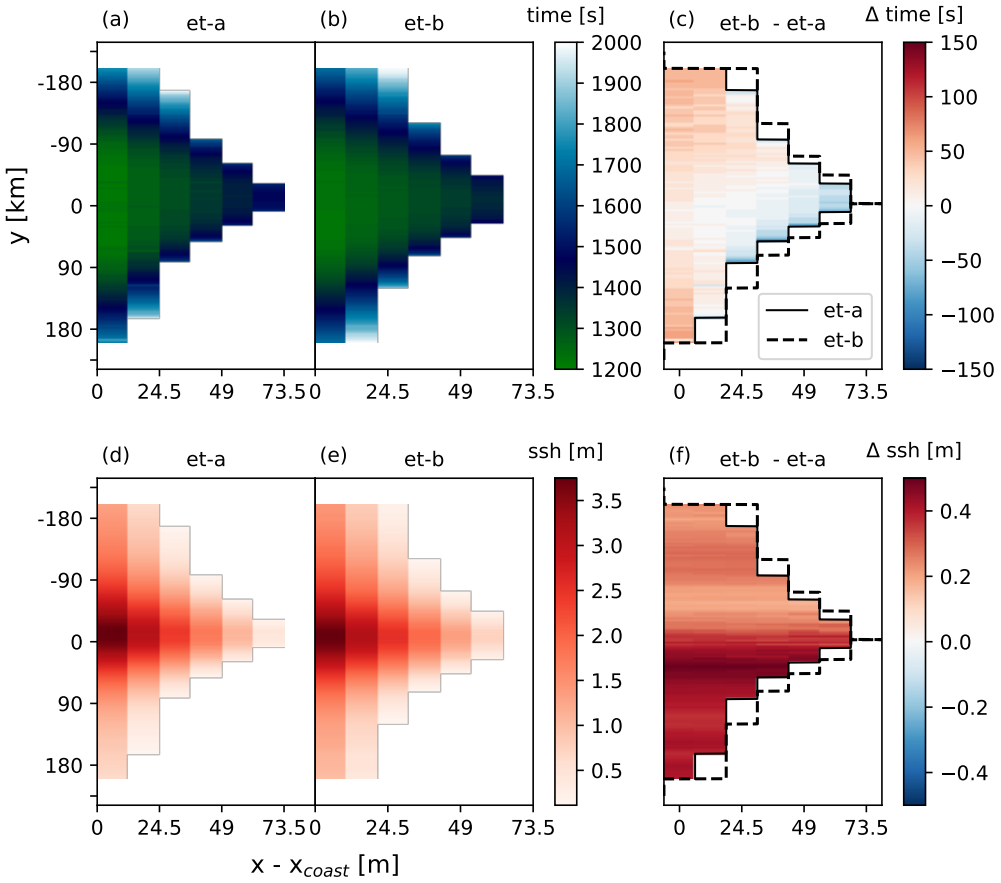


Figure 6.5.: Inundation time (a,b,c) and maximal inundation depth (d,e,f) for the tsunamis of the models et-a (a,d) and et-b (d,e) and their difference et-b - et-a (c,d). The stepwise pattern is the result of the mesh size at the coast. Adapted from Figure 7 in [92].

the coast and later in outer regions (a,b). By looking at the difference in timing (c), we see that areas at the coastline are inundated earlier by the wave of model et-b, positions further in land by the wave of model et-a.

For the inundation depth, both models again show similar patterns. The peak is at the center of the coast and decreases towards its boundaries (d,e). Inland the maximal inundation depth becomes smaller, until the run-up depth is reached. The difference (f) shows that the surface-breaching rupture produces higher waves at the coast (up to 3.75 m) compared to the blind rupture (up to 3.4 m). This also explains the higher run-up depth

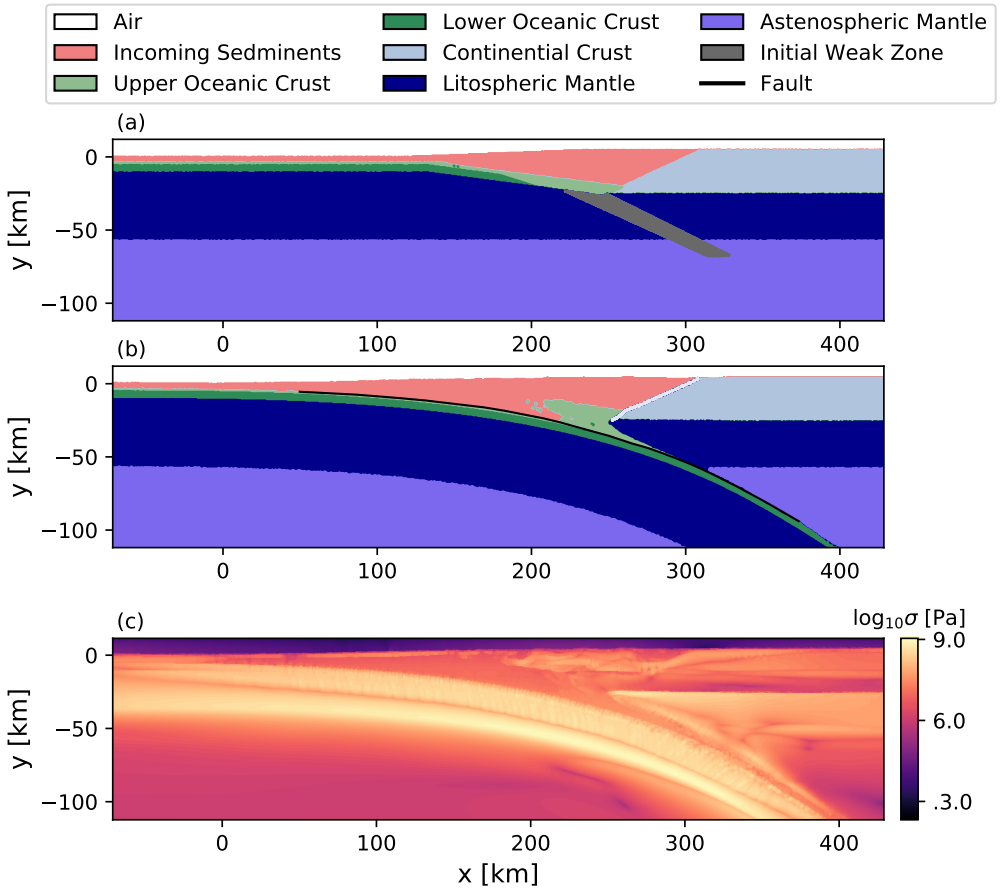


Figure 6.6.: Material layers and second invariant of the stress tensor for the subduction zone model. An initial set weak zone in the set-up (a) is displaced by the propagating oceanic plate and causes its subduction of the continental plate (b). After 3.6 Myr the second invariant of the stress tensor (c) causes materials to fail. Adapted from Figure A2 in [92].

and inundated area (73 m and 18.32 km^2) in model et-b, than in model et-a (61 m and 15.82 km^2) (c,f).

6.5 Subduction-earthquake-tsunami models

Instead of using approximated parameters for the earthquake model as we did in the last section, the simulation of dynamics in an oceanic subduction

zone allows us to generate more realistic parameters. We use the geodynamic seismic cycle model (SC), based on the work by van Zelts et al. [134]. The model depicts the south Chilean subduction zone, in which an oceanic plate subducts a continental plate (Figure 6.6 a).

By solving thermomechanical conservation equations their model can simulate the long-term evolution of the plates and their influence on the Earth's upper mantle (consisting of the Lithosphere and Asthenosphere) over a span of millions of years. The oceanic plate propagates with small velocity towards the continental plate. An initially set weak zone is displaced by the moving oceanic plate and causes it to subduct the continental plate. After 3.6 Myr the subduction model has reached a steady-state, for which the subsequent seismic cycle is initiated and resolved with a reduced time-step (Figure 6.6 b). In the seismic cycle earthquakes occur as spontaneous frictional instabilities (so called slip events). Through the constantly moving oceanic plate, stresses continue to build up in the model and more material fails (Figure 6.6 c). The result are multiple consecutive slip events from which we pick one as input to the earthquake model. Several adjustments have to be made to link the basic axioms of both models, which are in detail described by Wirp et al. [140]:

Earthquake results

The subduction model is the foundation of the parameter study by Wirp et al. [4]. The authors conclude from the effects of the location of the hypocenter, varied material and rupture parameters on the subsequent tsunami [140]. In this chapter we compare three exemplary models, for which the tsunami models where created as part of this work and repeat their core findings: The reference model (set-3B) shares the location of the nucleation with the SC model (40 km depth, at the center of the fault at $y = -150$ km). Materials are modeled as Poisson solid ($\nu = 0.25$), hence the first Lameé parameter λ and the shear modulus μ share the same value in the whole domain (an introduction to both parameters is given in Section 7.2).

In model set-5 the fracture energy, which represents the amount of energy that has to be expended to start and uphold the fracture, is tripled compared to the reference model. This is done by tripling the critical slip weakening distance in the linear-slip weakening law (compare to Section 6.4).

To see the effects of material properties Poisson's ratio is increased compared to model set-3B (to $\nu = 0.30$), in model set-6, leading to an increased P-wave speed.

The resulting final slip and displacement fields are presented in Figure 6.7.

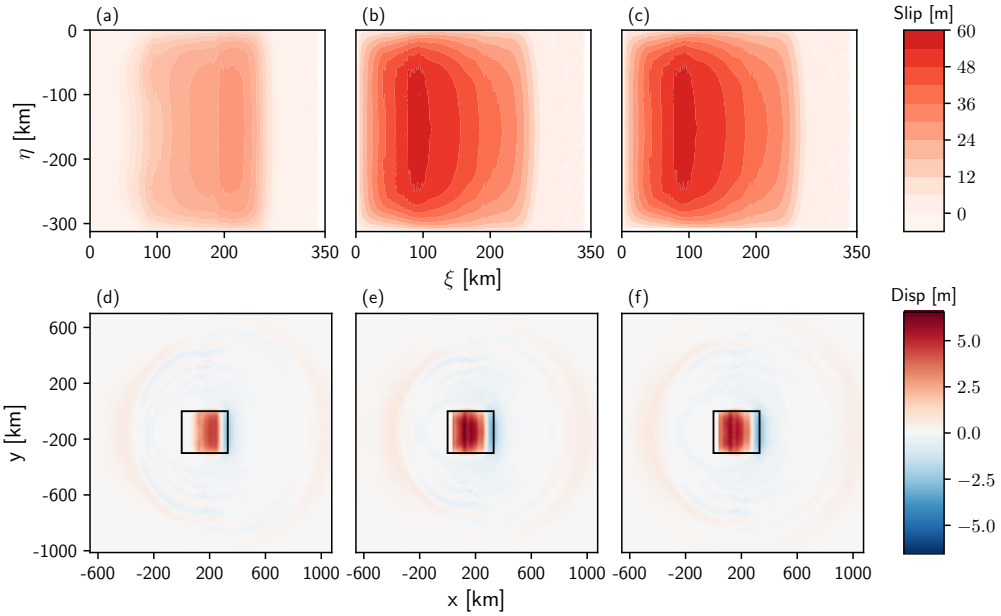


Figure 6.7.: On-fault accumulated slip (a,b,c) (η: along-strike and ξ: down-dip) and unfiltered final displacement fields (d,e,f) for the models set-3B (a,d), set-5 (b,e) and set-6 (c,f). Adapted from Figure 6 & 9 in [140].

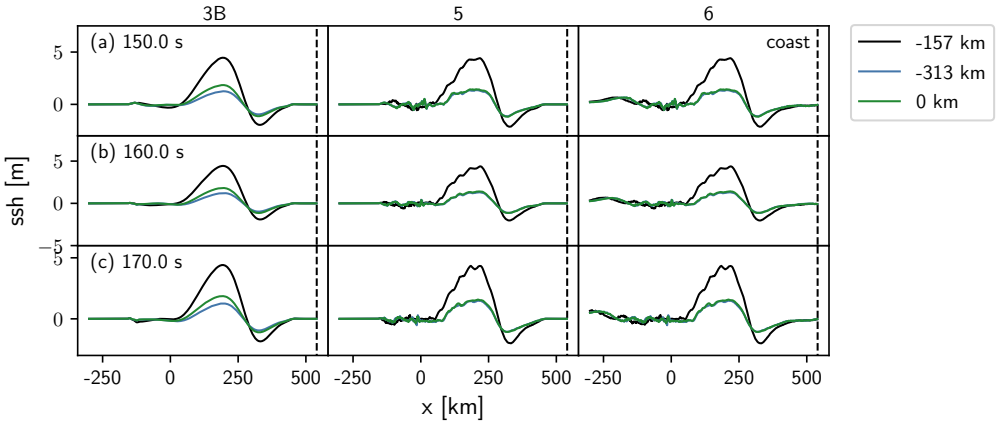


Figure 6.8.: Cross sections of the tsunamis to model set-3B, set-5 and set-6 after 200 s, 1400 s an 2000 s. Along the lines $y = -157$ km, in the center of the domain and the parallels $y = 0$ km and $y = -313$ km. Adapted from Figure S5 & 10 in [140].

Table 6.3.: Maximal run-up (max Ru), highest inundation depth max Id, arrival time (min It) and inundated area (area) of the tsunamis for the earthquake tsunami models eq-a and eq-b and the subduction zone earthquake tsunami models set-3B, set-5, set-6.

	et-a	et-b	set-3B	set-5	set-6
max Ru [s]	61.04	73.24	87.89	161.13	124.51
max Id [m]	3.40	3.75	5.42	7.53	6.90
min It [s]	1210.00	1220.00	2010.00	2090.00	2070.00
area [km^2]	15.82	18.32	52.18	67.44	59.77

In all three models slip evolves circularly away from the position of the nucleation, which leads to a bilateral evolution of the rupture. The final accumulated slips (a,b,c) show us that the ruptures are stopped close to the surface, the seafloor, is breached in no case. Higher fracture energy leads to twice as high final accumulated slip (a,b) compared to the reference, the same effect happens for when we increase Poisson's ratio (c).

The final tsunami potential of model set-3B is 3.13 PJ for set-5 and set-6 it is twice as high (6.95 PJ, 6.18 PJ in Table 6.1). Maximal uplift of set-5 (8.6 m) and set-6 (8.31 m) are also significantly increased compared to the reference model (5.88 m). Uplift and TP show us, that we can expect the tsunamis of model set-5 and set-6 to be of higher magnitude than the one in set-3B.

The permanent displacement in all three fields is imprinted by trapped waves (d,e,f), which we have to remove with our filtering approach from Section 5.3. The exact filtering process including a suiting parameter set for set-5 is explained in Section 5.3.4, the same set-up was used to filter set-3B and set-6.

Tsunami results

We again apply the postprocessing pipeline from Section 5.2 and Section 5.3. The displacement field are filtered with parameters $c^w_{rel} = 5.0 \times 10^{-7}$ and $c^f_{rel} = 1.25 \times 10^{-9}$. The resulting seafloor perturbation is applied as time-dependent source. In the bathymetry of the tsunami model we set $x_0 = 500\,\text{km}$, such that the beach is located at $x_{coast} = 540\,\text{km}$. For the sake of boundary effects the domain is set to $\Omega = [-600, 600] \times [-750, 450]\,\text{km}$.

The cross sections at the lines $y = 0\,\text{km}$, $y = -157\,\text{km}$ and $y = -313\,\text{km}$ are shown in Figure 6.8. The seafloor perturbations converge after 200 s.

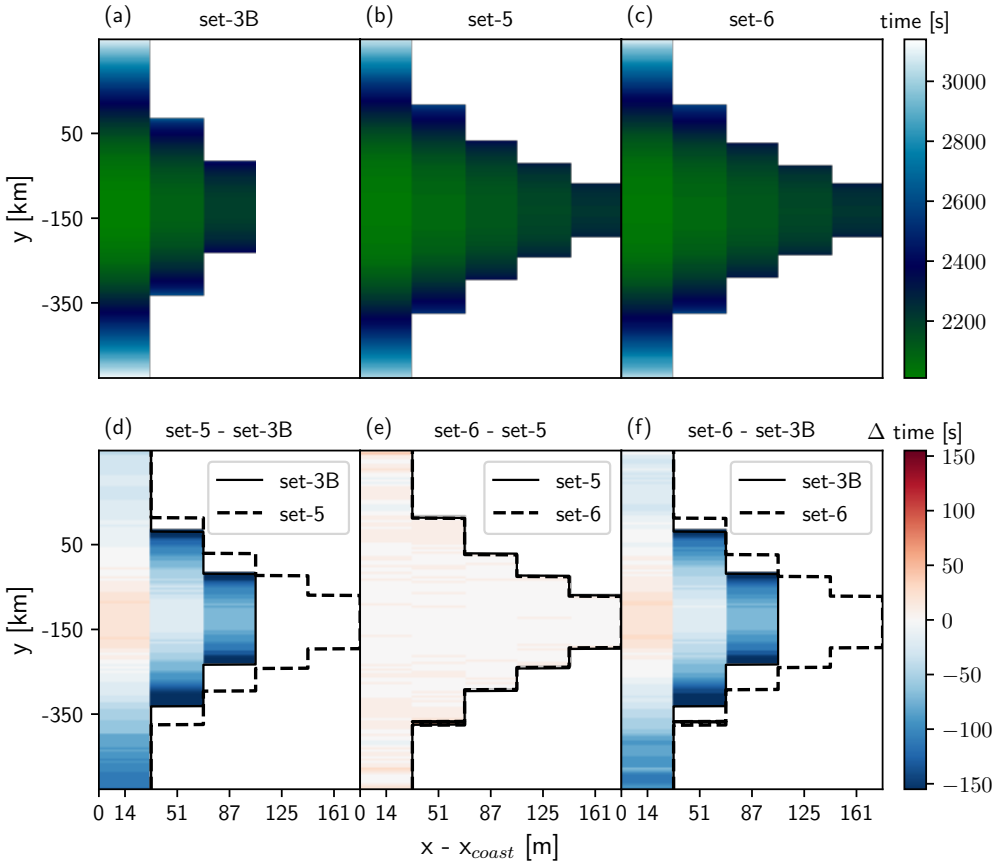


Figure 6.9.: Inundation time for the models set-3B (a), set-5 (b) and set-6 (c). The difference in timing are plotted in (d,e,f) The run-up depth for all three models is added to the difference plots. The stepwise pattern is a result of the mesh resolution at the coast. Adapted from Figure 8 & 11 in [140].

At this point we see that the displaced volume is larger for set-5 and set-6 than set-3B. Both models also show higher peaks of the wave compared to the reference (set-3B: 4.3 m, set-5: 5.7 m, set-6: 5.2 m) which corresponds to the observed differences in the tsunami potentials of the seafloor perturbation and is reflected in the resulting total energy of the tsunamis(set-3B: 3.183 PJ, set-5: 6.795 PJ, set-6: 6.080 PJ in Table 6.2).

The waves evolve symmetric to the center of the domain $y = -150$ km, which is caused by the central position of the nucleation patch and the bilateral rupture in all models. After 1400 s we see that the peaks of the

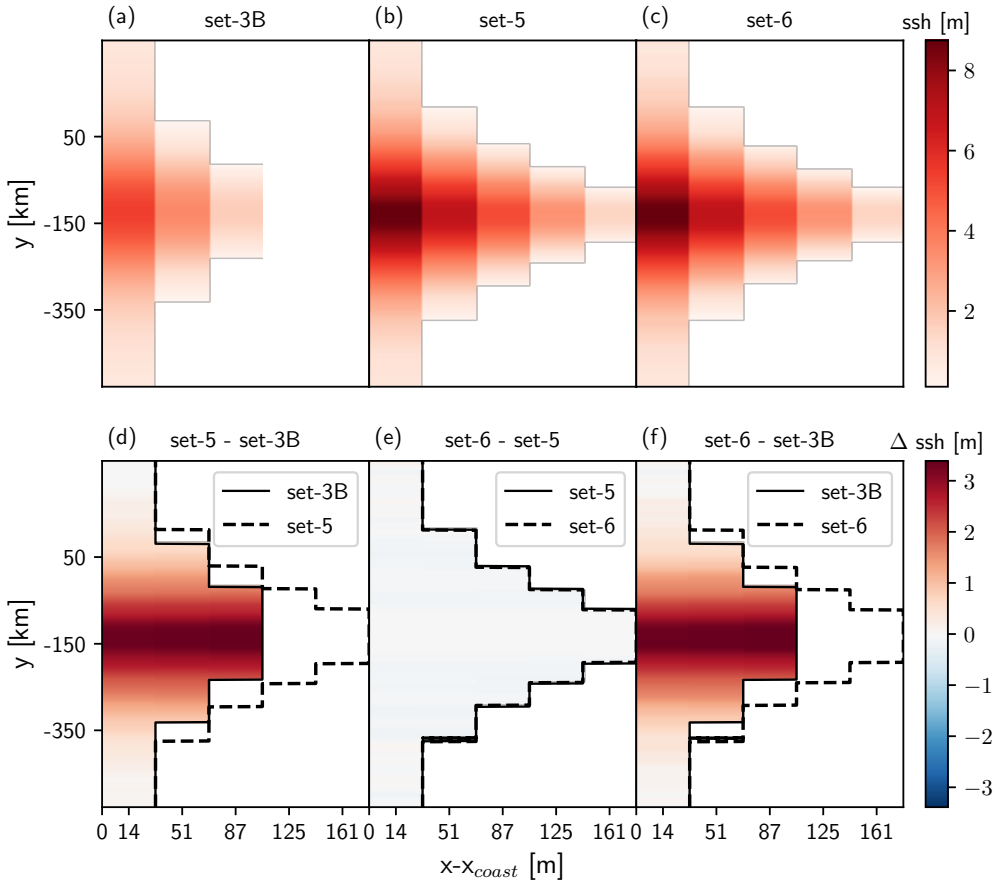


Figure 6.10.: Maximal inundation depth for the models set-3B (a), set-5 (b) and set-6 (c). Additionally we show the differences between the models in (d,e,d). set-5 shows the highest maximal inundation depth (e) followed by set-6 (f) and set-3B (d). The run-up depth for all three models is added to the difference plots. The stepwise pattern is a result of the mesh resolution at the coast. Adapted from Figure 8 & 11 in [140].

wave in the tsunami to the increased Poisson's ratio and increased fracture energy are higher than in the reference model. At 2000 s the tsunamis have reached the coast and amplify due to the slope of the bathymetry.

Inundation time for all models is shown in Figure 6.9, maximal inundation in Figure 6.10. The tsunami of model set-3B arrives first at the coast after 2010 s, for set-5 80 s and set-6 60 s later. Areas further inland are inundated most early by the wave to set-6 and latest by the wave to set-3B (Figure 6.9

d,e,f). The highest maximal inundation depth in all three models is in the center of the coast, where set-5 has the highest depth, followed by set-6 and set-3B (set-3B: 5.3 m, set-5: 7.5 m, set-6: 6.9 m). We can directly relate these observations back to the difference in the wave peaks before the inundation. On the whole coast the maximal inundation depth is higher in model set-5 than in set-3B (up to 2.2 m difference). Compared to set-6 the difference is smaller (up to is 0.7 m), but especially in the center set-5 has a higher maximal inundation depth. The same relations hold for the run-up depth (set-3B: 88 m, set-5: 161 m, set-6: 124 m) and the inundated area (set-3B: 52 km^2, set-5: 67.5 km^2, set-6: 60 km^2).

6.6 Time dependent vs. time independent source

We perform a meta-analysis using the models eq-a and eq-b from Section 6.4 and set-3B, set-5 and set-6 from Section 6.5, to compare the difference between time dependent and time independent tsunami sourcing (see Section 5.1).

We base our analysis on the work of Abrahams et al. [1], who claim that for a large source size σ_r relative to a short source duration σ_t and tsunami wave speed $\sqrt{gH}$, the difference of both approaches is negligible.

The relation is expressed by the measure we introduced in Equation (6.1):

$$\sigma_p = \frac{\sigma_r}{\sigma_t \sqrt{gH}}. \tag{6.1}$$

For $\sigma_p \gg 1$ the size of earthquake source is significantly larger than the distance the tsunami propagates during the earthquake evolution. As the tsunami hardly evolves during the earthquake, the effects of time dependent sourcing should be negligible compared to time independent sourcing. For all five models the source duration is in the range of minutes, the source size in hundreds of kilometers ($\sigma_t \in [68, 198]$ s and $\sigma_r \in [176, 540]$ km). The resulting values for σ_p are in the range $\sigma_p \in [13.06, 19.27]$, we thus expect little difference between both sourcing approaches (Table 6.4).

In order to compare the tsunamis for both sourcing approaches, we rerun the simulations using time independent sourcing, with the same settings as in Section 6.4 and Section 6.5. In case of time independent sourcing, the initial time is not clearly defined, we choose to synchronize the tsunamis of both sourcing approaches by their arrival time at the coast.

We first look at the peak of the tsunamis and the mean of the energy metrics for the wavefields, as defined in Section 6.2 in Table 6.5. In the table we

Table 6.4.: Source duration σ_t, source size σ_r, the metric σ_r (from Equation (6.1)) and the tsunami potential energy (TP) of the seafloor perturbations to the models et-a, et-b, set-3B, set-5 and set-6.

	et-a	et-b	set-3B	set-5	set-6
σ_t [s]	68	98	198	198	198
σ_r [km]	176	181	395	531	540
σ_p	18.30	13.06	14.11	18.96	19.27
TP(T_{max}) [PJ]	0.16	0.32	3.13	6.95	6.18

present all values for time dependent sourcing (d), time independent sourcing (s) and their relative difference ($\Delta = 100\cdot$(d-s)/d). A negative sign in the relative difference tells us a higher value for time independent sourcing. For the peak of the wave (max ssh) we see that in all cases, except set-3B, time dependent sourcing leads to a smaller value than time independent sourcing. The differences are in a range of 0.3 to 2.4 %. The only noticeable difference in the total mean energy ($\overline{\text{TE}}$) is in model et-b with 1.6 %, for all other cases it is less than 0.64 %. Looking at the distributions of mean potential $\overline{\text{PE}}$ and mean kinetic energy $\overline{\text{KE}}$ we see differences between 1.16 % and 5.4 %. For the earthquake tsunami models eq-a and eq-b time dependent sourcing produces more kinetic and less potential energy than time independent sourcing, for the subduction earthquake tsunami models set-3B, set-5 and set-6 we observe the opposite.

On the coast we do the same comparison for the run-up distance (max Ru), the maximal inundation depth (max ID) and the inundated area (area) in Table 6.6. For the run-up distance we see zero difference between both approaches. The highest maximal inundation depth shows differences up to 2.5 %. For all models except set-5 it is larger for time dependent displacement. The total inundated area shows difference in the same magnitude, with up to 2.2 %. In all cases the time independent sourcing leads to a larger area.

In order to compare the inundation process on the coast, we compute the mean and the standard deviation of the difference in inundation time and maximal inundation depth between both sourcing approaches. The mean difference of the inundation time depends on our manual synchronization of arrival times and cannot be used to express a difference (one can synchronize both results such that the mean difference is zero and not by the arrival time). However the standard deviation gives us a summarizing measure

Table 6.5.: Maximal water height max ssh, mean total $\overline{\text{TE}}$, mean potential $\overline{\text{PE}}$ and mean kinetic energy $\overline{\text{KE}}$ of tsunamis for models et-a, et-b, set-3B, set-5 and set-6, for time dependent (d) and time independent (s) sourcing, and their relative difference (Δ = 100·(d-s)/d).

		et-a	et-b	set-3B	set-5	set-6
max ssh [m]	d	2.049	2.544	4.308	5.745	5.209
	s	2.097	2.564	4.293	5.874	5.295
[%]	Δ	−2.380	−0.789	0.338	−2.253	−1.636
$\overline{\text{TE}}$ [PJ]	d	0.162	0.302	3.183	6.795	6.080
	s	0.162	0.297	3.183	6.791	6.041
[%]	Δ	0.216	1.671	0.006	0.050	0.632
$\overline{\text{PE}}$ [PJ]	d	0.077	0.148	1.463	3.200	2.840
	s	0.079	0.151	1.443	3.108	2.760
[%]	Δ	−2.445	−2.270	1.380	2.853	2.821
$\overline{\text{KE}}$ [PJ]	d	0.086	0.155	1.720	3.595	3.239
	s	0.083	0.146	1.740	3.683	3.281
[%]	Δ	2.602	5.438	−1.164	−2.444	−1.288

Table 6.6.: Comparison of run-up max Ru, maximal inundation depth max Id and inundation area area of tsunamis for models et-a, et-b, set-3B, set-5 and set-6, for time dependent (d) and time independent (s) sourcing, and their relative difference (Δ = 100·(d-s)/d).

		et-a	et-b	set-3B	set-5	set-6
max Ru [m]	d	61.04	73.24	87.89	161.13	124.51
	s	61.04	73.24	87.89	161.13	124.51
[%]	Δ	0.00	0.00	0.00	0.00	0.00
max Id [m]	d	3.40	3.75	5.42	7.53	6.90
	s	3.49	3.81	5.54	7.48	6.96
[%]	Δ	−2.64	−1.71	−2.25	0.75	−0.90
area [km^2]	d	15.82	18.32	52.18	67.44	59.77
	s	16.17	18.72	52.22	68.15	60.11
[%]	Δ	−2.22	−2.19	−0.08	−1.05	−0.56

Table 6.7.: Mean and standard deviation of the inundation time ($\overline{\Delta \mathrm{It}}$,$\sigma\Delta\mathrm{It}$) and inundation depth ($\overline{\Delta \mathrm{Id}}$,$\sigma\Delta\mathrm{Id}$) for time dependent and time independent sourced tsunamis for models et-a, et-b, set-3B, set-5 and set-6.

Model	et-a	et-b	set-3B	set-5	set-6
$\overline{\Delta \mathrm{It}}$ [m]	53.93	40.14	−1.55	8.54	−3.14
$\sigma\Delta\mathrm{It}$ [m]	51.33	40.09	5.09	24.11	20.46
$\overline{\Delta \mathrm{Id}}$ [m]	-0.03	−0.02	−0.03	0.01	0.02
$\sigma\Delta\mathrm{Id}$ [m]	0.07	0.08	0.06	0.09	0.09

for the difference in inundation time in all locations, independent from the mean of both approaches. We see values ranging from 5 s in model set-3B up to 51 s for et-a, showing a significant difference.

The mean difference in inundation depth is in the range of centimeters ($[-0.03, 0.02]$ cm) its standard deviation close to 10 cm.

In conclusion, we can see differences in all five models between both sourcing approaches. The differences for the wavefields in energy and water height are relatively small, where the highest difference is 5.5 % more kinetic energy for model et-b in time dependent sourcing compared to time independent sourcing.

The impact on the coast is represented with both sourcing approaches equally well for all models, with small differences up to 3.0 %. Especially the run-up distance is equal for both sourcing approaches. The average inundation depth differs up to 3 cm, with a relatively high standard deviation of up to 9 cm. We see the most significant difference of both sourcing approaches in the inundation time. In case of time-independent sourcing, we neglect the temporal variation of the source, which also has an influence on the temporal variation of the inundation. Overall our results agree with the assumption by Abrahams et al. [1].

CHAPTER

7

ExaSeis: A curvilinear ADER-DG method for the simulation of dynamic rupture scenarios

7.1 Introduction

In this chapter, we introduce the components of the ExaSeis collection of applications for the ExaHyPE-Engine. For a detailed introduction to the ExaHyPE-Engine we refer the reader to the work by Reinarz et al. [104]. The core concept of the ExaHyPE-Engine for hyperbolic PDEs is adapted from game engines: The engine provides a finished implementation of solvers, mesh infrastructure, hardware optimizations and parallelization strategies for hyperbolic partial differential equations and gives users a clean C++ interface to implement terms of a specific PDE. The MPI and TBB parallelization is based on the Peano framework on Cartesian meshes [138]. As numerical solver, the linear and non-linear ADER-DG method with a posteriori finite volume sub-cell limiting is used [33, 36]. The implementation of the linear ADER-DG method, also called the Cauchy-Kovalevskaya (CK) method [126], was realized as part of this thesis and is introduced in Section 7.4. Hardware optimized kernels for the CK methods are provided by a code generation approach, that relies on the libxsmm library [53, 69]. The CK method has been optimized with a sum factorization approach by Gallard et al. reaching 22.5 % of the peak performance on a current Skylake node on SuperMUC-NG [54]. To perform load-balancing, Samfass et al. implemented a novel work-stealing approach [113]. The performance

of the Engine has been assessed by Charrier et al. and is not part of this discussion [26].

The ExaHyPE-Engine comes with the promise to provide a scalable implementation of dynamic adaptive meshes. In the context of dynamic rupture earthquake simulations this allows to address one of the main problems of this field: In order to simulate the propagating rupture accurately, the rupture front has to be resolved in the range of centimeters, while the simulated domain usually is in the range of hundreds of kilometers. This large difference of the required length-scales makes a simulation with static grids infeasible, as the resulting number of unknowns can not yet be solved computationally [79]. As solution to this problem dynamic adaptive mesh refinement (AMR) allows us to track the rupture front with a highly resolved grid, while that parts of the fault where the rupture has already finished or has no effect, can be simulated with a more coarse resolution.

In this chapter, we present the methods that make ExaSeis ready for such a simulation. We focus on earthquakes in domains with complex topographies and fault structures, that we represent on curvilinear meshes. In the context of uncertainty quantification and probabilistic seismic hazard analysis, we enable the automated set-up of simulations, by introducing an automated inductive meshing approach.

In the first section, we look at the elastic wave equation in their first order formulation and incorporate earthquakes as point sources and interface conditions [3, 39]. With perfectly matched layers we remove erroneous fluctuations coming from the truncated boundaries of a simulated domain in Section 7.3 [42]. In Section 7.4 we introduce the linear Cauchy Kovalevskaya scheme for curvilinear meshes, for which we require numerical fluctuations as presented in Section 7.5. In order to incorporate complex topographies and structures, we present a novel meshing approach in Section 7.6.

The results presented in Section 7.3, Section 7.4 and Section 7.5 were created in joint work and published in [41–43]. The automated mesher, presented in Section 7.6 was created as part of this dissertation.

7.2 Elastodynamics in first order formulation

Before we go into the details of the ExaSeis framework, we want to give the reader an impression of the underlying physics. For a more detailed introduction we refer to [3, 86]. This chapter is inspired by the work of LeVeque et al. from which we adapt the notation [86].

The objective is the modeling of particles as continuum in an elastic

medium. Waves that propagate through the continuum follow the first order stress velocity formulation, defined by:

$$\rho u_t - \sigma_x^{xx} - \sigma_y^{xy} - \sigma_y^{xz} = f_v^x \tag{7.1}$$
$$\rho v_t - \sigma_x^{xy} - \sigma_y^{yy} - \sigma_y^{yz} = f_v^y \tag{7.2}$$
$$\rho w_t - \sigma_x^{zx} - \sigma_y^{yz} - \sigma_y^{zz} = f_v^z \tag{7.3}$$
$$\sigma_t^{xx} - (\lambda + 2\mu)u_x - \lambda v_y - \lambda w_z = f_\sigma^{xx} \tag{7.4}$$
$$\sigma_t^{yy} - \lambda u_x - (\lambda + 2\mu)v_y - \lambda w_z = f_\sigma^{yy} \tag{7.5}$$
$$\sigma_t^{zz} - \lambda u_x - \lambda v_y - (\lambda + 2\mu)w_z = f_\sigma^{zz} \tag{7.6}$$
$$\sigma_t^{xy} - \mu(v_x + u_y) = f_\sigma^{xy} \tag{7.7}$$
$$\sigma_t^{xz} - \mu(u_z + w_x) = f_\sigma^{xz} \tag{7.8}$$
$$\sigma_t^{yz} - \mu(v_z + w_y) = f_\sigma^{yz}. \tag{7.9}$$

Here

$$\sigma = \begin{pmatrix} \sigma^{xx} & \sigma^{xy} & \sigma^{xz} \\ \sigma^{xy} & \sigma^{yy} & \sigma^{yz} \\ \sigma^{xz} & \sigma^{yz} & \sigma^{zz} \end{pmatrix}, \tag{7.10}$$

defines the symmetric stress tensor, while u, v and w are the particle velocities in x, y and z direction.

In this work we only consider isotropic materials, which are described by the density ρ and the Lamé parameters λ and μ. Unlike anisotropic materials, as crystals or wood, isotropic materials share identical properties in all directions.

Displacement and strain

Not apparent in the first order formulation, but crucial for their understanding are the displacement field $\vec{U} \in \mathbb{R}^3$ and the strain tensor $\epsilon \in \mathbb{R}^{3\times3}$. When a material is deformed the particles in its interior are displaced. The continuously differentiable field $\vec{U}$ represents all particle displacements in the continuum. Its temporal derivative are the particle velocities:

$$\vec{U}_t = \vec{u} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}. \tag{7.11}$$

We can consider a material as elastic and particles as a continuum, when deformations are *infinitesimal*, in the sense that the spatial gradient of the displacements is small compared to the extent of the domain $\|\nabla U\| \ll 1$.

With strain we describe the relative change of a body between its original state and after a deformation happened. If we compare two separate points P and Q, with connecting vector $\Delta x = Q - P$, that are displaced after the deformation to the points $\mathbf{P} = P + U(P)$ and $\mathbf{Q} = Q + U(Q) = Q + U(P + \Delta x)$ with connecting vector $\Delta \mathbf{x}$. The change of the vector Δx in the i-th dimension is computed by:

$$\Delta \mathbf{x}_i - \Delta x_i = U_i(P + \Delta x) - U_i(P). \tag{7.12}$$

As we only consider very small Δx, we can linearize the displacement field to approximate $U(Q)$:

$$U(P + \Delta x) \approx U(P) + \nabla U\,(P) \cdot \Delta x. \tag{7.13}$$

Rigid rotations in the displacement do not change the relative position of two points and thus have no influence on strain. In order to remove them from the equation, we evolve $\nabla U\,(P)$ into a symmetric and an antisymmetric part. The antisymmetric part represents the rigid rotations:[1]

$$\nabla U = \frac{1}{2}\left(\nabla U + \nabla U^T\right) + \frac{1}{2}\left(\nabla U - \nabla U^T\right). \tag{7.14}$$

Equation (7.12) becomes:

$$\Delta \mathbf{x}_i - \Delta x_i \approx \left(\epsilon \cdot \Delta x\right)_i, \tag{7.15}$$

where the matrix

$$\epsilon^{ij} = \frac{1}{2}\left(\nabla U + \nabla U^T\right)_{ij} \tag{7.16}$$

is called the strain tensor.

Using Equation (7.11), we can compute the temporal derivative of the strain tensor, depending on particle velocities:

$$\epsilon_t^{ij} = \frac{1}{2}\left(\nabla \vec{u} + \nabla \vec{u}^T\right)_{ij}. \tag{7.17}$$

A visual interpretation for the strain tensor is given by the deformation of a body that initially has the shape of the unit sphere. After the deformation, the unit sphere becomes an (potentially shifted) ellipsoid in which the principle semi-axes are changed by the eigenvalues ϵ^1, ϵ^2 and ϵ^3 of the

[1] Compare to curl(U)

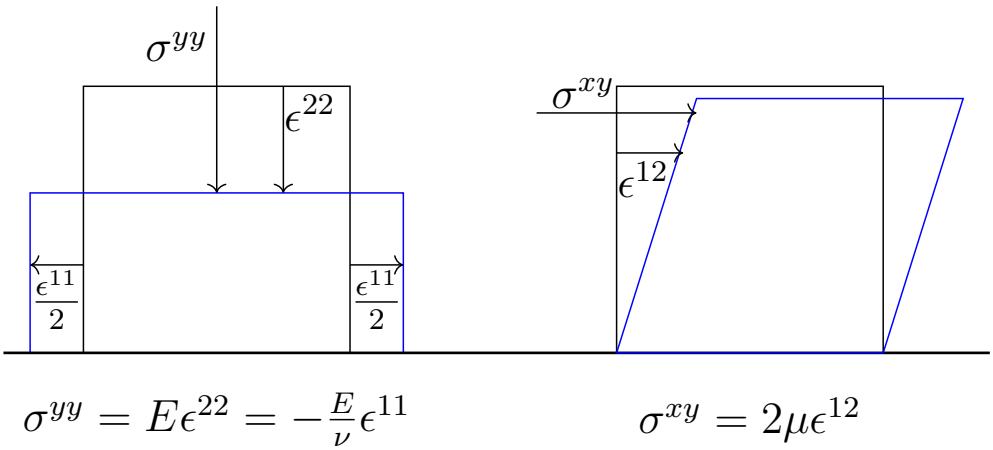


Figure 7.1.: Relation between normal and shear stress and strains.

strain tensor:

$$\frac{x^2}{(1+\epsilon^1)^2} + \frac{y^2}{(1+\epsilon^2)^2} + \frac{z^2}{(1+\epsilon^3)^2} = 1. \tag{7.18}$$

Stresses and Hook's law

If we consider a point on a plane in the continuum with normal $\vec{n}$, the corresponding traction vector $\vec{t}$ describes the force in that point applied from one side of the plane to the other. For a point that is the intersection of three planes with normals $\vec{e_1}$, $\vec{e_2}$ and $\vec{e_3}$, we can summarize those tractions with the stress tensor σ. Vice versa having the stress tensor, we can derive the traction for an arbitrary plane with normal $\vec{n}$:

$$\vec{t} = \sigma \cdot \vec{n}. \tag{7.19}$$

Hook's law tells us the relation between stress and strain. If we consider a cuboid body of isotropic material in the continuum and apply a normal force on one of its faces at the x-axis, the relative deformation of the body in x is proportional to the stress (Compare to Figure 7.1 left):

$$\sigma_{xx} = E\epsilon^{11}. \tag{7.20}$$

The proportion factor E is called Young's modulus. In the other directions the cuboid is also stretched proportional to the stress and equally in both

dimensions, as the material is isotropic

$$-\nu\epsilon^{11} = \epsilon^{22} = \epsilon^{33}. \tag{7.21}$$

ν is called Poisson's ratio. If we apply stresses to all faces of the cuboid, strains accumulate and we end up with the relation:

$$\begin{aligned}
\epsilon^{11} = &\quad \frac{1}{E}\sigma_{xx} - \frac{\nu}{E}\sigma_{yy} - \frac{\nu}{E}\sigma_{zz}, \\
\epsilon^{22} = &-\frac{\nu}{E}\sigma_{xx} + \frac{1}{E}\sigma_{yy} - \frac{\nu}{E}\sigma_{zz}, \\
\epsilon^{33} = &-\frac{\nu}{E}\sigma_{xx} - \frac{\nu}{E}\sigma_{yy} + \frac{1}{E}\sigma_{zz}.
\end{aligned} \tag{7.22}$$

One can show that shear strain and stress are proportional:

$$\epsilon^{12} = 2\mu\sigma^{xy}, \qquad \epsilon^{13} = 2\mu\sigma^{xz}, \qquad \epsilon^{23} = 2\mu\sigma^{yz}, \tag{7.23}$$

with the shear modulus μ as proportion factor (Compare to Figure 7.1 right). If we invert Equation (7.23) and Equation (7.22) and combine the result with the temporal derivative of the strain tensor from Equation (7.17), we end up with Equation (7.5) to Equation (7.9) in the elastic wave equation. The Lamé parameter λ and μ can be derived from Young's modulus and Poisson's ratio:

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \qquad \mu = \frac{E}{2(1+\nu)}. \tag{7.24}$$

Newton's second law

Equation (7.2) - Equation (7.4) are derived from Newton's second law, which states that the change of impulse $\mathbf{p}$ in a point has to be equal to the acting forces:

$$\vec{F} = \frac{\partial}{\partial t}\mathbf{p} = \frac{\partial}{\partial t} m\vec{v}. \tag{7.25}$$

If we consider Equation (7.25) for an arbitrary cuboid reference element V we get:

$$\frac{\partial}{\partial t}\int_V \rho\vec{v}\,\mathrm{d}V = \int_V \vec{f}\,\mathrm{d}V + \int_{\delta V} \vec{t}\,\mathrm{d}\delta V\,, \tag{7.26}$$

where we split the acting forces in body forces f and tractions on the surface $\vec{t}$. We know from Equation (7.19), that the traction on a plane with normal $\vec{n}$ can be described as $\vec{t} = \sigma \cdot \vec{n}$. Applying the divergence theorem leads us

to:

$$\frac{\partial}{\partial t}\int_V \rho\vec{v}\,\mathrm{d}V = \int_V \vec{f} + \nabla\cdot\sigma\,\mathrm{d}V\,. \tag{7.27}$$

For an infinitesimal small volume V, we finally end up with Equation (7.2) to Equation (7.4). In this work we consider body forces that are applied at a single location x_c of the form:

$$\vec{f_v}(t) = S(t)M\cdot\nabla\delta(x-x_c), \tag{7.28}$$

for the symmetric moment tensor:

$$M = \begin{pmatrix} M^{xx} & M^{xy} & M^{xz} \\ M^{xy} & M^{yy} & M^{yz} \\ M^{xz} & M^{yz} & M^{zz} \end{pmatrix} \tag{7.29}$$

and a moment time history $S(t)$.

We can move body forces of this type from Newton's second law (Equations (7.2) - (7.4)) to Equations (7.5) - (7.9), by setting the body forces of the stress tensor to:

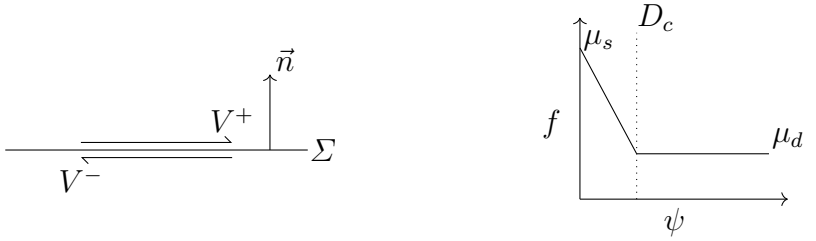
$$f_\sigma(t) = \frac{\partial}{\partial t}S(t)M\delta(x-x_c). \tag{7.30}$$

As the spatial derivative of the Dirac delta is not present in Equation (7.30), this form is numerically more convenient.

7.2.1 Sourcing earthquakes

While seismic waves can be initialized by external forces as for example volcanoes, landslides or the movement of elephant herds [3, 99], we focus solely on sources originating in the interior of the earth. In general we consider fault systems, as in the subduction zone example we showed in Section 6.5.

Geometrically a fault is modeled as a surface Σ with normals $\vec{n}$ in the earths interior (Figure 7.2a). Due to tectonic movements, stresses are accumulated on the fault until the material of the surface cannot uphold its locked state. The fault ruptures and slipping is initiated. The two adjacent sides of the fault Σ^- and Σ^+ are displaced in opposite directions by displacements V^+ and V^-, for which the difference is called slip $[[V]] = V^+ - V^-$ and its derivative in time the slip-rate $[[v]] = v^+ - v^-$. Because of the discontinuity in the displacement, linear elastics cannot be used to model the



(a) A simplified fault. **(b)** Linear slip-weakening friction law.

propagation of waves directly on the fault. However, in the adjacent volumes separated by the fault the equations are still valid and initiate seismic waves.

There are two established methods to include this sourcing in the elastic wave equation. The first accumulates slip in a single equivalent point dislocation, which is used as body force in Equations (7.2) - (7.4). The second directly models slip and strength on the fault by a friction law and sources the earthquake by enforcing traction components on the fault.

Pointsources

It can be shown, that the evolution of the displacement field in a volume only depends on the initial velocity and stresses, and forces applied at the inside of the medium [3][2]. In case the represented fault Σ is significantly smaller than the wave length of the occurring seismic waves, we can accumulate the effects of the displacement on the fault, to an equivalent body force in a single point. These body forces are represented in the form [3]:

$$\vec{f_v} = \vec{S}(t) M \nabla \delta(x - x_c), \tag{7.31}$$

where $M \in \mathbb{R}^{3\times3}$ is the moment tensor component and $S(t) : \mathbb{R} \to \mathbb{R}$ the moment time history.

In Equation (7.30) we saw that with some adjustments we can apply this body force to the stress components of the equation:

$$\vec{f_\sigma}(t) = \frac{\partial}{\partial t} S(t) M \delta(x - x_c). \tag{7.32}$$

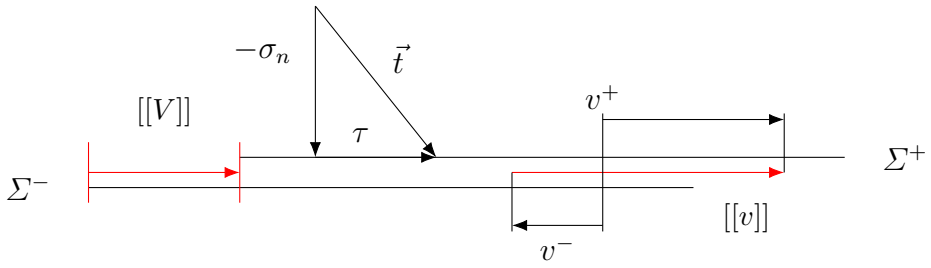


Figure 7.3.: Dynamic rupture mechanism on a fault. After the shear traction τ is equal to the fault strength τ_s, slip $[[s]]$ is initiated.

Dynamic rupture

If we want to model the rupture directly on the fault, we decompose the traction $\vec{t}$ into the components normal and tangential to the fault (Figure 7.3):

$$\vec{t} = \sigma \cdot \vec{n}, \qquad t_n = t \cdot \vec{n}, \qquad \tau = \vec{t_n} - t_n \vec{n}. \tag{7.33}$$

Equivalently we get the normal $[[v_n]]$ and tangential parts $s = [[v]] - [[v_n]]\vec{n}$ of the slip-rate. As we do not allow opening or interpenetration of two sides of the fault Σ^- and Σ^+, the normal slip rate has to be zero $[[v_n]] = 0$.

The shear traction τ models the force acting in the tangential direction of the fault. The strength of the fault τ_s sets the maximal shear traction it can uphold. As the shear traction reaches the fault strength, the fault breaks and slip is initiated in the same direction as the shear traction. Hence, the shear traction may never exceed the fault strength:

$$\|\tau\| \leq \tau_s. \tag{7.34}$$

In case the shear traction is less than the fault strength, the fault remains locked and no slip is generated:

$$\|\tau\| < \tau_s \implies s = 0. \tag{7.35}$$

We can summarize all three conditions in the equation:

$$\frac{s}{V} = \frac{\tau}{\tau_s}, \tag{7.36}$$

[2] As we do not consider surface forces in this work.

where $V = \|s\|$ is the absolute slip-rate.

As the fault breaks it is further weakened and the fault strength is reduced. In order to model this behavior, we set fault strength and the path of the slip $\psi(t) = \int_0^t \|s(\theta)\| d\theta$ in direct relation by a friction law $f(\psi)$:

$$\tau_s = c - \sigma_n f(\psi). \tag{7.37}$$

Here, $\sigma_n := \max(0, -\sigma \cdot \vec{n})$ is the normal stress[3] and the cohesion c summarizes that part of the fault strength that is independent from friction.

Generally speaking, the friction law models to what extent compression forces contribute to the fault strength. We use linear slip weakening as introduced by Ida [72]:

$$f(\psi) = \mu_s - (\mu_s - \mu_d) \frac{\min(\psi, D_c)}{D_c}. \tag{7.38}$$

For zero slip the friction law has the value of the static friction coefficient μ_s, as soon as the slip-path exceeds the critical distance D_c it becomes the dynamic friction coefficient μ_d (Compare Figure 7.2b right). In between, the coefficient linearly decreases with ψ.

All three components of the friction law have an influence on the energy that is present in the final system [100] and with that on the magnitude of the earthquake. Is is shown that the fracture energy G (which can be seen, as the energy required to break the connection between both sides of the fault) is part of the energy balance. For linear slip weakening it is:

$$G = \frac{1}{2} (\mu_s - \mu_d) D_c \sigma_n. \tag{7.39}$$

7.3 Perfectly Matched Layers in a nutshell

The commonly used artificial absorbing boundary conditions, used to emulate the exit of waves that propagate out of a simulated domain, are not perfect and lead to reflections that spuriously appear in seismograms and wavefields [45]. A simple solution for this is the extension of the domain, such that waves never reach the boundary and no reflections can be generated. With this extension, additional cells are introduced and the problem size increased, which can be reduced by coarsening cells towards the boundary of the domain. However, a working and scalable implementa-

[3] As convention it is negative for compressing stresses

tion for statically refined meshes is required for this solution. Additionally, the coarsening of cells can again introduce reflections, as the accuracy of the representation of a wave changes between elements. Alternatively, our approach follows the method of perfectly matched layers (PML) first introduced by Berenger et al. [17] and tailored for our scheme by Duru et al. [42].

The approach is based on the *complex coordinate stretching* [27] technique. For a quick illustration, we assume a solution composed by plane waves in the form [74]:

$$u(x,t) = \sum_{m=0}^{M} \sum_{n=0}^{N} \hat{u}_{nm} e^{i(k_m x - \omega_n t)}, \tag{7.40}$$

with wave vector and frequency $k_m, \omega_n \geq 0$, and coefficients $\hat{u}_{nm}$. If we change the coordinate x from the real valued line x to the complex valued curve $\tilde{x} = x + i\theta(x)x$, we get:[4]

$$\begin{aligned} u(\tilde{x},t) &= \sum_{m=0}^{M} \sum_{n=0}^{N} \hat{u}_{nm} e^{i(k_m(x+i\theta(x)x) - \omega_n t)} \\ &= \sum_{m=0}^{M} \sum_{n=0}^{N} \hat{u}_{nm} e^{i(k_m x - \omega_n t)} e^{-\theta(x) x k_m}. \end{aligned} \tag{7.41}$$

We see that in the region $x < 0$ the solution is equal to $u(x,t)$, for $x > 0$ the plane waves are damped by $e^{-k_m x}$ and exponentially decay in x and in the wave vector k_m. The concept of complex coordinate stretching is equivalent to this example: In order to damp the solution in a designated layer, a suiting change of the variable to the complex space x is introduced.

In order to apply PML to the elastic wave equation, the PDE is first transformed with the Laplace transformation $\mathcal{L}[*]$, as temporal derivatives disappear in complex space after the transformation:

$$\mathcal{L}[t]\left(\frac{\partial}{\partial t} u(x,t)\right) = s\tilde{u}(x,s), \tag{7.42}$$

where $\tilde{u}(x,s)$ is the Laplace transformation of $u(x,t)$ and s its Laplace dual time variable.[5] With this property we can transform Equation (7.2) to

[4] With $\theta(x)$ we denote the Heaviside step function

[5] Here we used the assumption the PML layer is initially empty.

Equation (7.9) into a system of ordinary differential equations:

$$s\tilde{u}(x,s) = \sum_{i=1,3} A_i \frac{\partial}{\partial x_i}\tilde{u}(x,s). \tag{7.43}$$

To add PML to the equation, we perform a change of variables in each spatial dimension x_i, such that:

$$\frac{\partial}{\partial x_i} \to \frac{1}{S_i(x,s)}\frac{\partial}{\partial x_i}, \qquad \text{for } S_i(x_i,s) = 1 + \frac{d_i(x_i)}{s+\alpha_i(x_i)}. \tag{7.44}$$

Changing the variables in Equation (7.43) leads to:

$$s\tilde{u}(x,s) = \sum_{i=1,3} A_i \frac{1}{S_i(x_i,s)}\frac{\partial}{\partial x_i}\tilde{u}(x,s). \tag{7.45}$$

In order to improve the stability of the method the change of variables was chosen by the approach of Appelö et al. [5]. Here $d_i(x) \geq 0$ denotes the *PML damping functions* and $\alpha_i(x) \geq 0$ the *complex frequency shift*.

The term $S_i(x_i,s)^{-1}\partial/\partial_{x_i}\tilde{u}(x,s)$ does not allow us to analytically transform Equation (7.45) back to a PDE. Alternatively, we introduce helper variables:

$$\tilde{w}_i(x,s) = \frac{1}{(s+\alpha_i)\,S_i} A_i \frac{\partial}{\partial x_i}\tilde{u}(x,s), \tag{7.46}$$

which leads to a system of ODEs:[6]

$$\begin{aligned} s\tilde{u}(x,s) &= \sum_{i=1,3} A_i \frac{\partial}{\partial x_i}\tilde{u}(x,s) - d_i(x)\tilde{w}(x,s) \\ s\tilde{w}_i(x,s) &= \qquad A_i \frac{\partial}{\partial x_i}\tilde{u}(x,s) - (\alpha_i(x) + d_i(x))\tilde{w}_i(x,s). \end{aligned} \tag{7.47}$$

We can analytically compute the inverse Laplace transformation for Equation (7.47) and retrieve the elastic wave equation with perfectly matched layers:

$$\begin{aligned} \frac{\partial}{\partial t}u(x,t) &= \sum_{i=1,3} A_i \frac{\partial}{\partial x_i}u(x,t) - d_i(x)w(x,t) \\ \frac{\partial}{\partial t}w_i(x,t) &= \qquad A_i \frac{\partial}{\partial x_i}u(x,t) - (\alpha_i(x) + d_i(x))w_i(x,t). \end{aligned} \tag{7.48}$$

[6]Here we used $\frac{1}{S_x} = 1 - \frac{d(x)}{(s+\alpha)S_x}$.

The added helper variables w_i blow up the hyperbolic PDE from 9 to $9 + 3 \cdot 9 = 36$ unknowns. The main contribution of the work by Duru et al. [42] lies in the development of a provably stable discontinuous Galerkin method to discretize Equation (7.47) [42]. Additionally to the new terms in Equation (7.47) Duru et al. introduce stable numerical fluctuations, incorporating PML on element and boundary interfaces. We do not revisit the fluctuations at this point, but refer the reader to [42] for a detailed presentation and stability proof.

7.3.1 The choice of PML-Parameters

In order to damp the solution we have to find a proper parameterization of $d_i(x)$ and $\alpha_i(x)$. The PML damping function $d_i(x)$ defines the degree, with which waves are damped in the spatial dimension x_i. For $d_i(x) = 0$ waves are unchanged and we return at the elastic wave equation, which also implies that the helper variables we introduced in Equation (7.46) are zero. The complex frequency shift $\alpha_i(x)$ is an additional stabilizer of the equation, that was first introduced for Maxwell's equations by Kuzuoglu et al. [84].

The choice of parameters is based on the work by Appelö et al. [5]: For a cubic domain $\bigotimes_{i=1}^{3} \left[x_i^l, x_i^r\right]$ we define an enclosing cube $\bigotimes_{i=1}^{3} \left[\bar{x}_i^l, \bar{x}_i^r\right]$, such that the emerging layer has a constant width of $\Delta x = \bar{x}_i^l - x_i^l$ and $\Delta x = x_i^r - \bar{x}_i^r$ in all dimensions. We set the damping functions such that they increase within the layer towards the outer boundary of the PML layer:

$$d_i(x_i) = \begin{cases} 0, & \text{if } x_i \in [x_i^l, x_i^r] \\ d_{max}, \left(\frac{x_i^r - x_i}{\Delta x}\right)^n & \text{if } x_i \in [x_i^r, \bar{x}_i^r] \\ d_{max}, \left(\frac{x_i - x_i^l}{\Delta x}\right)^n & \text{if } x_i \in [\bar{x}_i^l, x_i^l]. \end{cases} \tag{7.49}$$

As function we chose a polynomial of degree n that increases from 0 to d_{max} within the layer. The complex frequency shift $\alpha_i(x)$ is chosen as function depending on $d_i(x)$

$$\alpha_i(x) = \alpha^c + \alpha^s d_i(x). \tag{7.50}$$

Appelö shows that the stability of the numerical solution is sensible to the choice of the parameters Δx, d_{max}, n, α^c, α^s. Unfortunately, no general rule for suiting parameter sets has been found up to this date. Alternatively, a few rules of thumb can be derived [5]: In order to filter all reflections in the layer Δx the maximal damping d_{max} and the polynomial degree n must be

Table 7.1.: PML parameters for the benchmarks of Chapter 8.

Benchmark	Domain	Δx	n	d_{max}	α^s	α^c
Zugspitze	$[0\,\text{km}, 90\,\text{km}]^3$	1.8 km	1	20.0	0.0	1.5
TPV 5/26/28/29	$[0\,\text{km}, 51\,\text{km}]^3$	1.9 km	1	20.0	0.0	1.5
HHS1/LOH1	$[0\,\text{km}, 17\,\text{km}]^3$	2.0 km	3	18.0	0.0	1.5

chosen large enough. Too aggressive damping leads to instabilities, which can be stabilized for sufficiently small $d_i(x)$ by increasing $\alpha_i(x)$. As the parameters do not depend on the resolution of the mesh, we suggest a series of low resolved test simulations to empirically determine the right of choice of parameters for a final large production run. In case a coarse solution contains reflections the damping parameters have to be increased, in case of instabilities, we have to increase α or decrease $d_i(x)$. As orientation we summarize the choice of parameters for all benchmarks that are object of Chapter 8 in Table 7.1. We see that in all benchmarks parameters are chosen in the same magnitude and especially that a constant representation of α was sufficient to ensure stability.

7.3.2 An analytic example: PML for the one dimensional advection equation

In order to give an intuitive idea how the PML layer works, we want to look at an analytic example for the linear scalar advection equation in one dimension:

$$u_t(x,t) = a u_x(x,t), \tag{7.51}$$

with initial condition $u_0(x)$ and solution $u(x,t) = u_0(at + x)$. We assume $\alpha = 0$, such that the advection equation with perfectly matched layers (analogous to Equation (7.47)) collapses to a single equation:

$$U_t(x,t) = a U_x(x,t) - d(x) U(x,t). \tag{7.52}$$

The solution to Equation (7.52) is given by the function:[7]

$$\tilde{U}(x,t) = u(x,t) \cdot \exp\left(\int_{-\infty}^{x} \frac{d(x')}{a} dx' \right) =: u(x,t) f(x). \tag{7.53}$$

[7]Which we can trivially prove by evaluating Equation (7.52) in Equation (7.53).

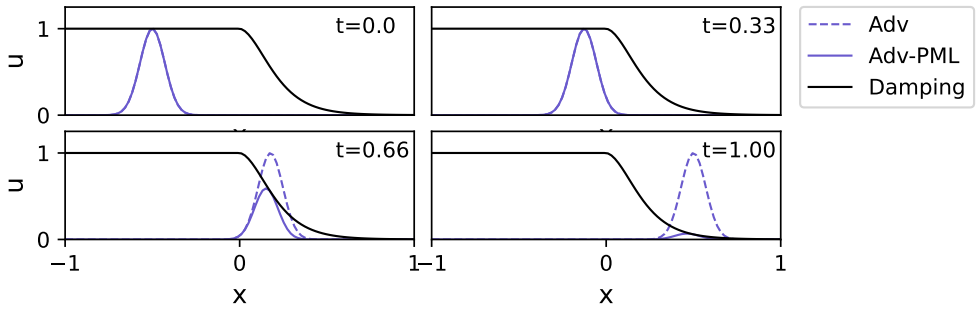


Figure 7.4.: Solutions for the advection equation without (Adv) and with Perfectly Matched Layers (Adv-PML). The damping function exponentially decreases the amplitude of the solution.

The solution for the advection equation with PML is the solution to the advection equation scaled by a spatial damping factor. With $d(x)$, we can define the position at which we start damping the wave and the rate with which the damping increases for higher x. In order to ensure that the wave is damped to 0, we have to define $d(x)$ such that $d(x) < 0, \forall x > x'$ for some x'. In Figure 7.4 we show the evolution of a Gaussian wave for the damping function $d(x) = -0.5x \cdot \theta(x)$. For $x < 0$ both solutions are identical, while for $x > 0$ the amplitude decreases, until it is close to 0.

7.4 Cauchy-Kovalevskaya scheme on curvilinear meshes

In order to numerically solve the elastic wave equation, we use the linear Cauchy-Kovalevskaya method [126]. We render the method for curvilinear meshes based on Cartesian reference elements as introduced by Duru et al. in [43].

Starting point is a hyperbolic PDE on curvilinear coordinates (ξ, η, ν) of the form

$$\frac{\partial}{\partial t}\mathrm{Q} + \tilde{\nabla} \cdot \mathrm{F}\left(\mathrm{Q}\right) + \mathrm{B}\left(\mathrm{Q}\right) \cdot \tilde{\nabla}\mathrm{Q} = \mathrm{S}\left(\mathrm{Q}\right). \tag{7.54}$$

For the elastic wave equation fluxes and source terms are linear in Q and only depend on temporally constant material parameters P. Flux F, non-conservative product B, and source term S turn into matrix-vector multi-

plications. In Einstein notation Equation (7.54) becomes

$$\frac{\partial}{\partial t}\mathrm{Q}_n + \tilde{\nabla}_d \left(\mathrm{F}_{dnm}\left(\mathrm{P}\right)\mathrm{Q}_m\right) + \mathrm{B}_{dnm}\left(\mathrm{P}\right)\left(\tilde{\nabla}_d \mathrm{Q}_m\right) = \mathrm{S}_{nm}\left(\mathrm{P}\right)\mathrm{Q}_m. \tag{7.55}$$

We transformed the divergence operator to a tensor contraction over the dimensional index $d = 1, ..., 3$, corresponding to the three spatial dimensions ξ,η and ν. n and $m \in 1, ..., M$ are the indices for the M physical quantities. The elastic wave equation in Section 7.2 and the elastic wave equation with perfectly matched layers in Section 7.3 are of the form in Equation (7.55), for different numbers of physical quantities. For the sake of brevity we drop the notation for material parameters at this point.

7.4.1 Transforming between computational and physical mesh

Our approach is build on the goal to create a DG scheme on curvilinear elements under the constraint that the meshing infrastructure is restricted to Cartesian elements, as it the case for the underlying Peano framework. From this point on we talk about a *physical mesh* of curvilinear elements $\mathcal{C}^{\boldsymbol{k}}$ and a uniform *computational mesh* of cubes $\Omega^{\boldsymbol{k}}$. We restrict the derivation to uniform meshes, as the meshing infrastructure reduces element updates in arbitrary statically and dynamically refined Cartesian meshes back to the uniform case.

In order to conform with the interface of the ExaHyPE-Engine, we have to derive the scheme such that all components of the physical mesh become part of the flux and source terms. The final kernels have to be computed on the reference element of the computational mesh $\Omega = [0, 1]^3$. For this approach we require element-wise bijective transformations $\tau^{\boldsymbol{k}}$ from each element in the computational mesh to a curvilinear element in the physical mesh. The construction of such a transformation for boundary fitting meshes is discussed in Section 7.6. The transformation has to keep the neighbor relations between adjacent elements.

With the multi-index $\boldsymbol{k} = (k_1, k_2, k_3) : k_i \in \{0, N_e\}$, we describe the position of element $\Omega^{\boldsymbol{k}}$ in a uniform mesh that has N_e elements in each dimension. The six neighbors of $\Omega^{\boldsymbol{k}}$ (and $\mathcal{C}^{\boldsymbol{k}}$) are denoted with $\Omega^{\boldsymbol{k}+\boldsymbol{o}}$ (and $\mathcal{C}^{\boldsymbol{k}+\boldsymbol{o}}$) and share the six faces $\Gamma^{(\boldsymbol{o})}$ (and $\mathcal{E}^{(\boldsymbol{k},\boldsymbol{o})}$).[8] With the multi-index $\boldsymbol{o} \in O = \{(-1, 0, 0), (1, 0, 0), (0, -1, 0), (0, 1, 0), (0, 0, -1), (0, 0, 1)\}$, we encode the relative position of the neighboring elements in all three dimensions

[8] This notation is well defined, if we assume a layer of boundary elements around the domain to model boundary conditions.

$((-1,0,0)$ is the left neighbor, $(1,0,0)$ the right neighbor, ...).

Additionally, we know that in the computational mesh there exists a transformation from the reference element $\Omega = [0,1]^3$ to every Cartesian element $\Omega^{\boldsymbol{k}}$. The transformation consist of a transposition of the element and a scaling of every edge with Δx. We denote this transformation with $\gamma^{\boldsymbol{k}}$, which has a derivative of $D\gamma^{\boldsymbol{k}} = \Delta x \mathbb{I}_3$.[9]

By composing both transformations, we get a bijective transformation from the reference element to each curvilinear element $\mathcal{C}^{\boldsymbol{k}}$ in the physical mesh,

$$\tau^{\boldsymbol{k}}(x,y,z) : \tau^{\boldsymbol{k}}(\Omega^{\boldsymbol{k}}) = \mathcal{C}^{\boldsymbol{k}} \text{ and } \gamma^{\boldsymbol{k}}(x,y,z) : \gamma^{\boldsymbol{k}}(\Omega) = \Omega^{\boldsymbol{k}} \\ \implies \tau^{\boldsymbol{k}}(\gamma^{\boldsymbol{k}}(x,y,z)) : \tau^{\boldsymbol{k}}(\gamma^{\boldsymbol{k}}(\Omega)) = \mathcal{C}^{\boldsymbol{k}}. \quad (7.56)$$

In order to transform the derivative to the reference element, we have to perform a change of variables from curvilinear ($\tilde{\nabla}_d$) to Cartesian coordinates (∇_b). To ensure numerical stability of the discretized scheme, we use two analytically equivalent versions of the chain rule [43],

$$B_{dnm}\tilde{\nabla}_d = B_{dnm} D((\gamma^{-1})^{\boldsymbol{k}} \circ (\tau^{-1})^{\boldsymbol{k}})_{db} \nabla_b = \frac{1}{\Delta x} \tilde{B}^{\boldsymbol{k}}_{bnm} \nabla_b, \\ \text{with } \tilde{B}^{\boldsymbol{k}}_{bnm} := B_{dnm} D(\tau^{-1})^{\boldsymbol{k}}_{db}. \quad (7.57)$$

$$\tilde{\nabla}_d F_{dnm} = \frac{1}{J^{\boldsymbol{k}}} \nabla_b \left(J^{\boldsymbol{k}} D((\gamma^{-1})^{\boldsymbol{k}} \circ (\tau^{-1})^{\boldsymbol{k}}) F_{dnm} \right) = \frac{1}{\Delta x J^{\boldsymbol{k}}} \nabla_b \tilde{F}^{\boldsymbol{k}}_{bnm}, \\ \text{with } \tilde{F}^{\boldsymbol{k}}_{bnm} := J^{\boldsymbol{k}} D(\tau^{-1})^{\boldsymbol{k}}_{db} F_b. \quad (7.58)$$

Here $D(\tau^{-1})^{\boldsymbol{k}}_{db}$ is the *Jacobian* of the inverse to $\tau^{\boldsymbol{k}}_{db}$,

$$D(\tau^{-1})^{\boldsymbol{k}}_{db} = \begin{pmatrix} \frac{\partial}{\partial\xi}(\tau_x^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\eta}(\tau_x^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\nu}(\tau_x^{-1})^{\boldsymbol{k}} \\ \frac{\partial}{\partial\xi}(\tau_y^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\eta}(\tau_y^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\nu}(\tau_y^{-1})^{\boldsymbol{k}} \\ \frac{\partial}{\partial\xi}(\tau_z^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\eta}(\tau_z^{-1})^{\boldsymbol{k}} & \frac{\partial}{\partial\nu}(\tau_z^{-1})^{\boldsymbol{k}} \end{pmatrix}, \quad (7.59)$$

$J^{\boldsymbol{k}} = \det\left\{ D(\tau^{-1})^{\boldsymbol{k}}_{db} \right\}$ its determinant, and the index $b = 1, ..., 3$ corresponds to the three Cartesian dimensions x, y and z.

We end up with the locally transformed representation of Equation (7.55) in Cartesian coordinates,

$$\frac{\partial}{\partial t} \mathrm{Q}_n + \frac{1}{\Delta x J^{\boldsymbol{k}}} \nabla_b \left(\tilde{F}^{\boldsymbol{k}}_{bnm} \mathrm{Q}_m \right) + \frac{1}{\Delta x} \tilde{B}^{\boldsymbol{k}}_{bnm} \left(\nabla_b \mathrm{Q}_m \right) = \mathrm{S}_{nm} \mathrm{Q}_m. \quad (7.60)$$

[9] $\mathbb{I}_3$ denotes the 3×3 identity matrix.

7.4.2 Interpolation and quadrature rule

To construct an element local polynomial approximation of Q, we start with the definition of the tensor basis on the unit cube:

$$\Phi_{\boldsymbol{i}}(x,y,z) = \phi_{i_1}(x)\phi_{i_2}(y)\phi_{i_3}(z). \tag{7.61}$$

Where $\phi_i(x)$ are the one dimensional Lagrange polynomials to a set of $N+1$ quadrature nodes x_k, and multi-index $\boldsymbol{i} = (i_1, i_2, i_3)$. While in general the basis can be build with an arbitrary set of quadrature nodes, we choose the Gauss Lobatto Legendre nodes (GLL), as they have nodes placed on the element interfaces ($x_0 = 0$, $x_N = 1$) and the corresponding quadrature rule is $2N-1$ accurate [70]. We can construct an interpolatory basis to the quadrature nodes

$$(\xi^{\boldsymbol{k}}_{i_1}, \eta^{\boldsymbol{k}}_{i_2}, \nu^{\boldsymbol{k}}_{i_3}) = \tau^{\boldsymbol{k}}(\gamma^{\boldsymbol{k}}(x_{i_1}, x_{i_2}, x_{i_3})), \tag{7.62}$$

on the curvilinear element by composing Equation (7.61) with the inverse of the transformation:

$$\Phi^{\boldsymbol{k}}_{\boldsymbol{i}}(\xi,\eta,\nu) := \Phi_{\boldsymbol{i}}\left((\gamma^{-1})^{\boldsymbol{k}}((\tau^{-1})^{\boldsymbol{k}}(\xi,\eta,\nu))\right). \tag{7.63}$$

In the curvilinear element $\mathcal{C}^{\boldsymbol{k}}$ we get the approximation

$$\mathrm{Q}(\xi,\eta,\nu,t)|_{\mathcal{C}^{\boldsymbol{k}}} \approx \sum_{\boldsymbol{i}=(0,0,0)}^{(N,N,N)} \Phi^{\boldsymbol{k}}_{\boldsymbol{i}}(\xi,\eta,\nu)\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{i}}(t) =: \Phi^{\boldsymbol{k}}_{\boldsymbol{i}}(\xi,\eta,\nu)\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{i}}(t), \tag{7.64}$$

where the definition on the right introduces the Einstein-notation on multi-indices. Equivalently we approximate the flux

$$\tilde{\mathrm{F}}^{\boldsymbol{k}}_{bnm}(\mathrm{P})\,\mathrm{Q}_m(\xi,\eta,\nu,t) \approx \Phi^{\boldsymbol{k}}_{\boldsymbol{i}}(\xi,\eta,\nu)\tilde{\mathrm{f}}^{\boldsymbol{k}}_{bin}(t). \tag{7.65}$$

In order to numerically solve volume integrals on the reference element and face integrals on its boundaries, we apply the quadrature rule that we used to build the interpolatory polynomial basis. To numerically solve volume integrals, we span the one-dimensional quadrature rule to three dimensions and get quadrature nodes $\mathbf{x}_{\boldsymbol{l}} = (x_{l_1}, x_{l_2}, x_{l_3})$ with quadrature weights $\mathbf{w}_{\boldsymbol{l}} = \omega_{l_1}\omega_{l_2}\omega_{l_3}$. With this choice the evaluation of the basis in

integrals becomes trivial,

$$\Phi_{\boldsymbol{i}}\left(\mathbf{x}_{\boldsymbol{l}}\right) = \delta_{\boldsymbol{il}} = \begin{cases} 1, & \text{if } (i_1 = l_1) \wedge (i_2 = l_2) \wedge (i_3 = l_3) \\ 0, & \text{else.} \end{cases} \tag{7.66}$$

$$\Phi_{\boldsymbol{j}}\left(\mathbf{x}_{\boldsymbol{l}}\right) \Phi_{\boldsymbol{i}}\left(\mathbf{x}_{\boldsymbol{l}}\right) = \delta_{\boldsymbol{jk}} \cdot \delta_{\boldsymbol{ik}}. \tag{7.67}$$

In order to get a quadrature rule for the faces $\Gamma^{(\boldsymbol{o})}$, we pick the indices of that nodes of the three dimensional rule, that lie on the face. For example, on face $\Gamma^{((-1,0,0))}$ we use the quadrature nodes (x_0, x_{l_2}, x_{l_3}) with weights $\omega_{l_2}\omega_{l_3}$. We generalize this constructive approach to all faces, with the set of multi-indices $\hat{\boldsymbol{L}}(\boldsymbol{o})$ located on face $\Gamma^{(\boldsymbol{o})}$,

$$\hat{\boldsymbol{L}}(\boldsymbol{o}) = \left\{ \left(\hat{l}_1(o_1), \hat{l}_2(o_2), \hat{l}_3(o_1) \right) \right\} \text{ , such that:} \tag{7.68}$$

$$\forall s \in \{1,2,3\} : \hat{l}_s(o_s) = \begin{cases} 0, & \text{for } o_s = -1 \\ \hat{l}_s, & \text{for } o_s = \quad 0 \\ N, & \text{for } o_s = \quad 1 \end{cases} , \hat{l}_s \in 1, ..., N. \tag{7.69}$$

and quadrature weights on o:

$$\omega(\boldsymbol{o})_{\hat{\boldsymbol{l}}} := \omega_{\hat{l}_1}^{1-|o_1|} \omega_{\hat{l}_2}^{1-|o_2|} \omega_{\hat{l}_3}^{1-|o_3|}. \tag{7.70}$$

For the set of multi-indices $\hat{\boldsymbol{L}}(\boldsymbol{o})$, we additionally introduce the characteristic function:

$$\chi\left[\boldsymbol{j} \in \hat{\boldsymbol{L}}(\boldsymbol{o})\right] = \begin{cases} 1, & \text{if } \boldsymbol{j} \in \hat{\boldsymbol{L}}(\boldsymbol{o}) \\ 0, & else, \end{cases} \tag{7.71}$$

which tells us if the node to a multi-index $\boldsymbol{j}$ is located on the face $\Gamma^{(\boldsymbol{o})}$. Note that the quadrature rule on the reference element and the face inherit the $2N-1$ accuracy from the one-dimensional rule.

7.4.3 The semidiscrete scheme

As we have now prepared all tools, we next construct the semi-discrete scheme following the road map from Section 2.4: We first apply the DG ansatz (See Section 2.4.1) on each curvilinear element $\mathcal{C}^{\boldsymbol{k}}$ with test functions

$\Phi_{\boldsymbol{j}}^{\boldsymbol{k}}$. Under the assumption that $\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\mathrm{q}_{im}^{\boldsymbol{k}}$ is globally differentiable we get

$$\forall \boldsymbol{j} : \int_{\mathcal{C}^{\boldsymbol{k}}} \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\frac{\partial}{\partial t}\mathrm{q}_{\boldsymbol{i}}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\left(\frac{1}{\Delta x J^{\boldsymbol{k}}}\nabla_b\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\right)\tilde{\mathrm{f}}_{b\boldsymbol{i}n}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\frac{1}{\Delta x}\tilde{\mathrm{B}}_{bnm}\left(\nabla_b\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\right) - \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\mathrm{S}_{nm}\Phi_{\boldsymbol{i}}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\,\mathrm{d}\mathcal{C}^{\boldsymbol{k}} = 0. \quad (7.72)$$

In order to treat discontinuities on the six element interfaces $\mathcal{E}^{(\boldsymbol{k},\boldsymbol{o})}$, we introduce numerical fluctuations $\mathcal{F}\left(\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\mathrm{q}_{in}^{\boldsymbol{k}}, \Phi_{\boldsymbol{i}}^{\boldsymbol{k}+\boldsymbol{o}}\mathrm{q}_{in}^{\boldsymbol{k}+\boldsymbol{o}}\right)\cdot\vec{n}$ (See Section 2.4.1)

$$\forall \boldsymbol{j} : \int_{\mathcal{C}^{\boldsymbol{k}}} \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\frac{\partial}{\partial t}\mathrm{q}_{\boldsymbol{i}}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\left(\frac{1}{\Delta x J^{\boldsymbol{k}}}\nabla_b\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\right)\tilde{\mathrm{f}}_{b\boldsymbol{i}n}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\frac{1}{\Delta x}\tilde{\mathrm{B}}_{bnm}\left(\nabla_b\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\right) - \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\mathrm{S}_{nm}\Phi_{\boldsymbol{i}}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\,\mathrm{d}\mathcal{C}^{\boldsymbol{k}} + \sum_{\boldsymbol{o}\in O}\int_{\mathcal{E}^{(\boldsymbol{k},\boldsymbol{o})}} \Phi_{\boldsymbol{j}}^{\boldsymbol{k}}\mathcal{F}\left(\Phi_{\boldsymbol{i}}^{\boldsymbol{k}}\mathrm{q}_{in}^{\boldsymbol{k}}, \Phi_{\boldsymbol{i}}^{\boldsymbol{k}+\boldsymbol{o}}\mathrm{q}_{in}^{\boldsymbol{k}+\boldsymbol{o}}\right)\cdot\vec{n}\,\mathrm{d}\mathcal{E}^{(\boldsymbol{k},\boldsymbol{o})} = 0. \quad (7.73)$$

An overview of numerical fluctuations in ExaSeis is given in Section 7.5.

We can solve the integral on the reference element Ω, by using integration by substitution based on the transformations from Equation (7.56)

$$\forall \boldsymbol{j} : \Delta x^3\int_{\Omega} J^{\boldsymbol{k}}\left(\Phi_{\boldsymbol{j}}\Phi_{\boldsymbol{i}}\frac{\partial}{\partial t}\mathrm{q}_{in}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}\left(\frac{1}{\Delta x J^{\boldsymbol{k}}}\nabla_b\Phi_{\boldsymbol{i}}\right)\tilde{\mathrm{f}}_{b\boldsymbol{i}n}^{\boldsymbol{k}}(t) + \Phi_{\boldsymbol{j}}\frac{1}{\Delta x}\tilde{\mathrm{B}}_{bnm}^{\boldsymbol{k}}\left(\nabla_b\Phi_{\boldsymbol{i}}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\right) - \Phi_{\boldsymbol{j}}\Phi_{\boldsymbol{i}}\mathrm{S}_{nm}\mathrm{q}_{im}^{\boldsymbol{k}}(t)\right)\mathrm{d}\Omega + \Delta x^2\sum_{\boldsymbol{o}\in O}\int_{\Gamma^{(\boldsymbol{o})}} J^{\boldsymbol{k}}\Phi_{\boldsymbol{j}}\mathcal{F}\left(\Phi_{\boldsymbol{i}}\mathrm{q}_{in}^{\boldsymbol{k}}, \Phi_{\boldsymbol{i}}^{\boldsymbol{o}}\mathrm{q}_{in}^{\boldsymbol{k}+\boldsymbol{o}}\right)\cdot\vec{n}\,\mathrm{d}\Gamma^{(\boldsymbol{o})} = 0. \quad (7.74)$$

In order to solve Equation (7.74), we use the previously introduced quadrature rules in volumes and on faces. By evaluating all basis functions according to Equation (7.66) we get

$$\forall \boldsymbol{j} : \frac{\partial}{\partial t}\mathrm{q}_{\boldsymbol{j}n}^{\boldsymbol{k}}(t) + \left(\frac{1}{\Delta x J^{\boldsymbol{k}}(x_{\boldsymbol{j}})}\left(\nabla_b\Phi_{\boldsymbol{i}}\right)(x_{\boldsymbol{j}})\right)\tilde{\mathrm{f}}_{b\boldsymbol{i}n}^{\boldsymbol{k}}(t) + \frac{1}{\Delta x}\tilde{\mathrm{B}}_{bnm}(x_{\boldsymbol{j}})\left(\left(\nabla_b\Phi_{\boldsymbol{i}}\right)(x_{\boldsymbol{j}})\mathrm{q}_{im}^{\boldsymbol{k}}(t)\right) - \mathrm{S}_{nm}(x_{\boldsymbol{j}})\,\mathrm{q}_{\boldsymbol{j}}^{\boldsymbol{k}}(t) + \frac{1}{\Delta x}\sum_{\boldsymbol{o}\in O}\chi\left[\boldsymbol{j}\in\hat{\boldsymbol{L}}(\boldsymbol{o})\right]\frac{\omega_{\boldsymbol{j}}(\boldsymbol{o})}{\omega_{\boldsymbol{j}}}\mathcal{F}\left(\mathrm{q}_{\boldsymbol{j}n}^{\boldsymbol{k}}, \mathrm{q}_{\tilde{\boldsymbol{j}}n}^{\boldsymbol{k}+\boldsymbol{o}}\right)\cdot\vec{n} = 0. \quad (7.75)$$

With the index $\tilde{\boldsymbol{j}} = (\hat{j}_1 - |\boldsymbol{o}_1|N, \hat{j}_2 - |\boldsymbol{o}_1|N, \hat{j}_3 - |\boldsymbol{o}_1|N)$, we denote those indices of the neighboring element that lie on face $\Gamma^{(\boldsymbol{o})}$.

In order to resolve the analytic definition of the divergence operator, we introduce the discrete derivative operator[10]

$$\bar{\nabla}_{\boldsymbol{bji}} := \left(\nabla_b \Phi_{\boldsymbol{i}}\right)(x_{\boldsymbol{j}}). \tag{7.76}$$

After we factorized and simplified the equation we get the semidiscrete scheme

$$\begin{aligned}\forall \boldsymbol{j} : \frac{\partial}{\partial t}\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{j}n}(t) + \frac{1}{\Delta x J^{\boldsymbol{k}}_{\boldsymbol{j}}}\bar{\nabla}_{\boldsymbol{bji}}\tilde{\mathrm{f}}^{\boldsymbol{k}}_{bin}(t) + \frac{1}{\Delta x}\tilde{\mathrm{B}}^{\boldsymbol{k}}_{\boldsymbol{bj}nm}\left(\bar{\nabla}_{\boldsymbol{bji}}\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{i}m}(t)\right) - \\ \mathrm{S}_{nm}\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{j}m}(t) + \frac{1}{\Delta x}\sum_{\boldsymbol{o}\in O}\chi\left[\boldsymbol{j} \in \hat{\boldsymbol{L}}(\boldsymbol{o})\right]\frac{\omega_{\boldsymbol{j},\boldsymbol{o}}}{\omega_{\boldsymbol{j}}}\mathcal{F}\left(\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{j}n}, \mathrm{q}^{\boldsymbol{k}+\boldsymbol{o}}_{\tilde{\boldsymbol{j}}n}\right)\cdot\vec{n} = 0,\end{aligned} \tag{7.77}$$

where we denote the non-conservative product and Jacobian evaluated at $x_{\boldsymbol{j}}$ with

$$\tilde{\mathrm{B}}^{\boldsymbol{k}}_{\boldsymbol{bj}nm} := \tilde{\mathrm{B}}^{\boldsymbol{k}}_{bnm}(x_{\boldsymbol{j}}) \qquad\qquad J^{\boldsymbol{k}}_{\boldsymbol{j}} := J(x_{\boldsymbol{j}})^{\boldsymbol{k}}. \tag{7.78}$$

There are two remarkable characteristics of Equation (7.77), that we want to highlight: The scheme contains no mass or stiffness matrices, the only operator remaining is the discrete derivative and numerical fluctuations only change the solution in nodes that directly lie on one of the interfaces.

We split all terms into linear volume and boundary operators, which we require for the integration in time

$$\frac{\partial}{\partial t}\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{j}n}(t) = \frac{1}{\Delta x}\left(\mathrm{Vol}^{\boldsymbol{k}}_{\boldsymbol{j}ni\boldsymbol{m}}\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{i}m}(t) + \sum_{\boldsymbol{o}\in O}\mathrm{Bnd}^{\boldsymbol{k}}(\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{j}n}, \mathrm{q}^{\boldsymbol{k}+\boldsymbol{o}}_{\tilde{\boldsymbol{j}}n})\right). \tag{7.79}$$

7.4.4 The curvilinear transformation as material parameter

Before we show the integration in time of Equation (7.79), we want to recapture what components of the curvilinear transformation $\tau^{\boldsymbol{k}}$ have to be computed in order to represent a boundary fitting mesh. We saw in Equation (7.57) that the Jacobian of the inverse to the curvilinear transformation $D(\tau^{-1})^{\boldsymbol{k}}$ has been moved to the redefined flux and non-conservative product. The resulting terms contain the Jacobian and its determinant. In

[10] The construction of the tensor basis allows us to simplify Equation (7.76) to a single matrix-multiplication.

the interpolation of the flux term in Equation (7.65) and the application of the quadrature rule in Equation (7.75) the Jacobian an its determinant are evaluated at the nodes $x_{\boldsymbol{j}}$. The only components of the curvilinear transformation we require are the Jacobian and its determinant evaluated on the quadrature nodes,

$$D(\tau^{-1})^{\boldsymbol{k}}(x_{\boldsymbol{j}}) \text{ and } J^{\boldsymbol{k}}(x_{\boldsymbol{j}}) \ \forall \boldsymbol{j}. \tag{7.80}$$

As we redefined the flux and non-conservative product, we also have to adjust the computation of their eigenvalues. One can show that for Equation (7.60), we get the new eigenvalues

$$\tilde{\lambda}_{i1} = c_p \|J_i^{\boldsymbol{k}}\|_2, \qquad \tilde{\lambda}_{i2} = c_s \|J_i^{\boldsymbol{k}}\|_2, \qquad \tilde{\lambda}_{i3} = c_s \|J_i^{\boldsymbol{k}}\|_2, \tag{7.81}$$

where $J_i^{\boldsymbol{k}}$ is the Euclidean norm of the i-th row of the Jacobian,

$$\|J_i^{\boldsymbol{k}}\|_2 := \sqrt{(D(\tau^{-1})_{i1}^{\boldsymbol{k}})^2 + (D(\tau^{-1})_{i2}^{\boldsymbol{k}})^2 + (D(\tau^{-1})_{i3}^{\boldsymbol{k}})^2}. \tag{7.82}$$

As the time-step size depends on the eigenvalues of the PDE (compare to Section 2.4.2) the transformation directly influences the admissible time-step size.

7.4.5 Time integration

In order to advance $\mathrm{q}_{\boldsymbol{i}}$ from t_n to t_{n+1}, we use the assumption that for small enough $\Delta t = t_{n+1} - t_n$ the temporal derivative can be substituted with the volume operator we derived in Equation (7.79) [126]:

$$\frac{\partial}{\partial t} \approx \mathrm{Vol}_{\boldsymbol{jnim}}^{\boldsymbol{k}}. \tag{7.83}$$

We integrate Equation (7.79) in time and get

$$\begin{aligned} &\mathrm{q}_{\boldsymbol{jn}}^{\boldsymbol{k}}(t_{n+1}) = \mathrm{q}_{\boldsymbol{jn}}^{\boldsymbol{k}}(t_n) + \int_{t_n}^{t_{n+1}} \frac{\partial}{\partial t} \mathrm{q}_{\boldsymbol{jn}}^{\boldsymbol{k}}(t)\,\mathrm{d}t \approx \\ &\mathrm{q}_{\boldsymbol{jn}}^{\boldsymbol{k}}(t_n) + \frac{1}{\Delta x} \int_{t_n}^{t_{n+1}} \mathrm{Vol}_{\boldsymbol{jnim}}^{\boldsymbol{k}} \mathrm{q}_{\boldsymbol{im}}^{\boldsymbol{k}}(t)\,\mathrm{d}t + \sum_{\boldsymbol{o}\in O} \frac{1}{\Delta x} \int_{t_n}^{t_{n+1}} \mathrm{Bnd}^{\boldsymbol{k}}(\mathrm{q}_{\boldsymbol{jn}}^{\boldsymbol{k}}, \mathrm{q}_{\tilde{\boldsymbol{jn}}}^{\boldsymbol{k}+\boldsymbol{o}})\,\mathrm{d}t\,. \end{aligned} \tag{7.84}$$

Next $\mathrm{q}_j(t)$ is approximated with a Taylor series of order N in t_n

$$\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t) \approx \sum_{l=0}^{N} \frac{(t-t_n)^l}{l!} \frac{\partial}{\partial t^l} \mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t_n) \overset{(7.83)}{\approx} \sum_{l=0}^{N} \frac{(t-t_n)^l}{l!} \left(\mathrm{Vol}^{\boldsymbol{k}}\right)^l_{\boldsymbol{jnim}} \mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{im}}(t_n), \tag{7.85}$$

where we can compute the single temporal derivatives iteratively

$$\left(\mathrm{Vol}^{\boldsymbol{k}}\right)^{l+1}_{\boldsymbol{jnim}} \mathrm{q}_{\boldsymbol{im}}(t_n) = \mathrm{Vol}^{\boldsymbol{k}}_{\boldsymbol{jnkl}} \left(\mathrm{Vol}^{\boldsymbol{k}}\right)^{l}_{\boldsymbol{klim}} \mathrm{q}_{\boldsymbol{im}}(t_n). \tag{7.86}$$

We use the series to approximate the integral $\mathrm{p}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t_n)$ of the solution $\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t)$:

$$\int_{t_n}^{t_{n+1}} \mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t)\,\mathrm{d}t \approx \int_{t_n}^{t_{n+1}} \sum_{l=0}^{N} \frac{(t-t_n)^l}{l!} \left(\mathrm{Vol}^{\boldsymbol{k}}\right)^l_{\boldsymbol{jnim}} \mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{im}}(t_n)\,\mathrm{d}t = \sum_{l=0}^{N} \frac{(\Delta t)^{l+1}}{(l+1)!} \left(\mathrm{Vol}^{\boldsymbol{k}}\right)^l_{\boldsymbol{jnim}} \mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{im}}(t_n) =: \mathrm{p}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t_n).$$

With which we can finally evolve Equation (7.84) to a time-stepping scheme

$$\mathrm{q}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t_{n+1}) = \frac{1}{\Delta x} \mathrm{Vol}^{\boldsymbol{k}}_{\boldsymbol{jnim}} \mathrm{p}^{\boldsymbol{k}}_{\boldsymbol{im}}(t_n) + \sum_{\boldsymbol{o}\in O} \frac{1}{\Delta x} \mathrm{Bnd}^{\boldsymbol{k}}(\mathrm{p}^{\boldsymbol{k}}_{\boldsymbol{jn}}(t_n), \mathrm{p}^{\boldsymbol{k}+\boldsymbol{o}}_{\tilde{\boldsymbol{jn}}}(t_n)). \tag{7.87}$$

7.5 A physically motivated numerical flux

Numerical fluctuations play the fundamental role for the inclusion of dynamic rupture in the numerical scheme. In this chapter we do not revisit the whole scheme, development, stability and convergence proves of the solver, but look at its core properties. For details of the one dimensional Riemann solver, we refer the reader to the work by Duru et al. [41]. The three dimensional solver extends the results of the one dimensional case and is described in detail by Duru et al. [43]. Dynamic rupture is included equivalent to the approach by Duru et al. [39]. From all three publications we adapt notation and nomenclature in this subsection.

A common solution to compute numerical fluctuations for problems in linear elasticity is the Godunov flux. Numerical fluctuations are computed by solving the exact Riemann problem between DG elements to find the state on the interface of two elements. Using the Rankine-Hugoniot jump

condition numerical fluctuations can be computed from the boundary state of an element and the derived state on the interface. In case of internal or external boundary conditions, an initial boundary value problem is solved: To a set of physical constraints, states on the interface are constructed, from which energy-stable numerical fluctuations are computed.

The Riemann solver we use in ExaSeis is defined one abstraction level above the upper approach: Numerical fluctuations are constructed in a *reverse engineering* approach, to obey numerical stability and consistency. Numerical stability is ensured by the energy method by Gustafsson et al. [66].

We start with the one dimensional case and use it to conclude on the numerical fluctuations in three dimensions. In contrast to other approaches, all element interfaces in a mesh are attributed with a frictional strength α. Classical DG interfaces have an infinite frictional strength $\alpha \to \infty$. In case we model a friction law on an interface, α is finite. On the interface we want to impose left and right states $\hat{\sigma}^-$, $\hat{\sigma}^+$, $\hat{v}^-$ and $\hat{v}^+$ such that they hold the force balance and friction law from Equation (7.37)

$$\text{Force balance: } \sigma^- = \sigma^+ = \sigma \tag{7.88}$$

$$\text{Friction Law: } \quad \sigma = \alpha[[v]], \qquad \alpha = \sigma_n \frac{f(\psi)}{V}. \tag{7.89}$$

Note that $V = \|[[v]]\|$ and $f(\psi)$ is the friction coefficient. In order to develop a provably stable convergent flux, we have to look at the characteristics in the eigenspace of the flux matrix of the one dimensional elastic wave equation

$$\begin{pmatrix} v \\ \sigma \end{pmatrix}_t = \begin{pmatrix} 0 & \rho^{-1} \\ \mu & 0 \end{pmatrix} \begin{pmatrix} v \\ \sigma \end{pmatrix}_x, \tag{7.90}$$

where velocity and stress tensor reduce to single values. We get the characteristics by transforming σ and v to the eigenspace of the flux matrix

$$p(\sigma, v) = \frac{1}{2}(Zv + \sigma) \qquad q(\sigma, v) = \frac{1}{2}(Zv - \sigma), \tag{7.91}$$

where we introduced the impedance $Z = \sqrt{\mu\rho}$.

For two neighboring projected solutions (σ^-, v^-) and (σ^+, v^+) the characteristics are

$$\text{outgoing:} \qquad p(\sigma^-, v^-) \qquad q(\sigma^+, v^+), \text{ or} \tag{7.92}$$

$$\text{incoming:} \qquad p(\sigma^+, v^+) \qquad q(\sigma^-, v^-). \tag{7.93}$$

In order to solve for well-posed boundary or interface conditions the amplitude of outgoing characteristics have to be preserved by the imposed states on the boundary [66]

$$p(\sigma^-, v^-) = p(\hat{\sigma}^-, \hat{v}^-) \qquad q(\sigma^+, v^+) = q(\hat{\sigma}^+, \hat{v}^+). \tag{7.94}$$

Duru et al. [41] show that the physical constraints in Equation (7.88) and the preservation of amplitudes in Equation (7.94) are solved by defining the interface variables as:

$$\begin{aligned} \hat{\sigma}^+ = \hat{\sigma}^- = \frac{\alpha}{\eta+\alpha}\Phi \qquad [[\hat{v}]] = \frac{1}{\eta+\alpha}\Phi \qquad \alpha = \sigma_n \frac{f(\psi)}{V} \\ \hat{v}^- = \frac{1}{Z^+}\left(2p\left(\sigma^+, v^+\right) - \hat{\sigma}^+\right) - [[\hat{v}]] \\ \hat{v}^+ = \frac{1}{Z^-}\left(2p\left(\sigma^-, v^-\right) + \hat{\sigma}^-\right) + [[\hat{v}]], \end{aligned} \tag{7.95}$$

where

$$\Phi = \eta\left(\frac{2}{Z^+}p(\sigma^+, v^+) - \frac{2}{Z^-}q(\sigma^-, v^-)\right) \text{ and } \eta = \frac{Z^+Z^-}{Z^+ - Z^-}. \tag{7.96}$$

Having found the states in Equation (7.94) we can introduce the flux fluctuations by penalizing data against incoming characteristics:

$$\begin{aligned} F^-(t) &= q(\sigma^-, v^-) - q(\hat{\sigma}^-, \hat{v}^-) \\ F^+(t) &= p(\sigma^+, v^+) - p(\hat{\sigma}^+, \hat{v}^+). \end{aligned} \tag{7.97}$$

Finally numerical fluctuations are constructed by penalizing Equation (7.97), such that physical dimensions match:

$$\mathcal{F}^-(t) = \begin{pmatrix} 1 \\ -Z^{-1} \end{pmatrix} F^-(t), \qquad \mathcal{F}^+(t) = \begin{pmatrix} 1 \\ -Z^{-1} \end{pmatrix} F^+(t). \tag{7.98}$$

For the choice of Equation (7.98) Duru et al. show stability and provide an error estimate proving the convergence of the scheme [41].

When we move to curvilinear coordinates in three dimensions we consider the problem on a curvilinear interface between two elements. We solve for numerical fluctuations in single quadrature nodes (See Equation (7.54)) on which we impose a local basis $\{\vec{n}, \vec{m}, \vec{l}\}$, where $\vec{n}$ is the normal of the face and $\vec{m}$ and $\vec{l}$ can be generated with the Gram-Schmidt process. In each quadrature node we can compute the corresponding rotated tractions t_n,

t_m, t_l and velocities v_n, v_m, v_l.

In three dimensions the physical constraints on the interface from Equation (7.88) become

$$\begin{aligned} &\text{Force balance: } t_\xi^- = t_\xi^+ = t_\xi, \xi \in \{n, m, l\} \\ &\text{No opening: } \quad [[v_n]] = 0 \\ &\text{Friction Law: } \quad t_\nu = \sigma_n \frac{f(\psi)}{V}, \nu \in \{m, l\}. \end{aligned} \tag{7.99}$$

Equivalently to the one dimensional case we create variables $\hat{v}_\xi^+$, $\hat{v}_\xi^-$, $\hat{t}_\xi^+$ and $\hat{t}_\xi^-$ in all three directions $\xi \in \{n, m, l\}$, such that they keep the constraints in Equation (7.99)

$$\begin{aligned} [[\hat{v}_n]] &= 0 & \hat{t}_n &= \Phi_n & \\ [[\hat{v}_\nu]] &= \frac{1}{\eta_\nu + \alpha} \Phi_\nu & \hat{t}_\nu &= \frac{\alpha}{\eta_\nu + \alpha} \Phi_\nu & \nu \in \{m, l\} \end{aligned} \tag{7.100}$$

$$\begin{aligned} \hat{v}_\xi^- &= \frac{1}{Z_\xi^+} \left(2p\left(t_\xi, v_\xi^+\right) - \hat{t}_\eta \right) - [[\hat{v}_\xi]] \\ \hat{v}_\xi^+ &= \frac{1}{Z_\xi^-} \left(2p\left(t_\xi, v_\xi^-\right) + \hat{t}_\eta \right) + [[\hat{v}_\xi]]. \end{aligned}$$

The final numerical flux is constructed equivalent to the one dimensional case in Equation (7.98) and can be found in [43]

7.6 Curvi: An automated mesh generator

As we described in Section 7.4, we can represent curvilinear meshes in ExaHyPE by mapping each single element from the computational mesh to an element in the physical mesh. The transformation has to represent structures and simultaneously meet constraints given by the meshing infrastructure of the computational mesh:

- At the end of Section 7.4, we showed that in order to incorporate the curvilinear mesh into the numerical scheme, we require the metric derivative and its determinant of the element-wise transformation on all quadrature nodes. To generate these parameters it is sufficient to represent the element wise transformation as an N-th order polynomial in $\mathbb{P}^N(\mathbb{R}^3)$.

- In the resulting transformation the topography, faults and material interfaces are thus approximated by two-dimensional N-th order splines, which correspond to the projected faces of computational elements.

- We have to keep the same neighbor relations in the computational and the physical mesh. We thus have to enforce that the shared faces of neighboring elements in the computational mesh are mapped to the same face in the physical mesh. This implies that shared vertices and edges are also mapped to the same vertices and curves. The transformation has to be continuous between elements.

Following the first property, we write the transformation $\tau^{\boldsymbol{k}}(x, y, z)$ of element $\Omega^{\boldsymbol{k}}$ to $\mathcal{C}^{\boldsymbol{k}}$ in terms of the spatial approximation from Equation (7.63)

$$\tau^{\boldsymbol{k}}(x, y, z) = \sum_{\boldsymbol{i}=(0,0,0)}^{(N,N,N)} \Phi_{\boldsymbol{i}}^{\boldsymbol{k}}(x, y, z) \left(\xi_{i_1}^{\boldsymbol{k}}, \eta_{i_2}^{\boldsymbol{k}}, \nu_{i_3}^{\boldsymbol{k}}\right). \tag{7.101}$$

This representation allows us to reduce the problem to finding the $(N+1)^3$ transformed coordinates $\left(\xi_{i_1}^{\boldsymbol{k}}, \eta_{i_2}^{\boldsymbol{k}}, \nu_{i_3}^{\boldsymbol{k}}\right)$, to the interpolation points $(x_{i_1}^{\boldsymbol{k}}, y_{i_2}^{\boldsymbol{k}}, z_{i_3}^{\boldsymbol{k}})$ in element $\Omega^{\boldsymbol{k}}$. From Equation (7.101) we can then compute the metric derivatives with the discrete derivative operator $\bar{\nabla}$ and derive its determinant.

In order to conserve the neighbor relation between computational and physical mesh, the transformed coordinates on faces of adjacent elements have to be equal. For a mesh of N_e elements in each dimension, this leaves us with $(NN_e + 1)^3$ transformed coordinates, that we can use to define the transformation.

For arbitrary cuboid domains we can always find a bijective projection to the unit-cube, which allows us to simplify the generation of the transformation and only consider it for the unit-cube $[0, 1]^3$. To construct the transformation for the computational mesh, we first find a suiting transformation $\boldsymbol{\tau}$ for the nodes of a uniform grid of $N_g = (NN_e + 1)$ equidistant nodes in each dimension

$$\boldsymbol{\tau}\left(\frac{l_1}{N_g}, \frac{l_2}{N_g}, \frac{l_3}{N_g}\right) = \left(\boldsymbol{\xi}_{l_1}, \boldsymbol{\eta}_{l_2}, \boldsymbol{\nu}_{l_3}\right), \tag{7.102}$$

where $l_1, l_2, l_3 \in \{0, ..., N_g\}$.

Having the transformed coordinates of all nodes in the uniform grid, we can reconstruct the transformation of the nodes in the computational mesh

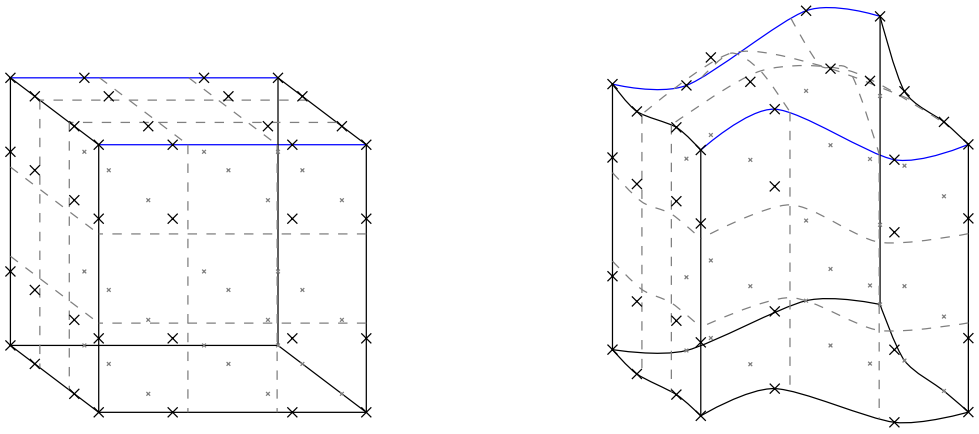


Figure 7.5.: Transformation of a Cartesian element (left) to a curvilinear element (right). We generate the transformation for a uniform mesh (dashed lines). With interpolation we can reproduce the transformation for the quadrature nodes of the DG-scheme (marked as crosses) and ensure that adjacent elements share the same transformation of their interface.

with a simple interpolation (See Figure 7.5)

$$\left(\xi^{\boldsymbol{k}}_{i_1}, \eta^{\boldsymbol{k}}_{i_2}, \nu^{\boldsymbol{k}}_{i_3}\right) = \sum_{l_3=Nk_3}^{N(k_3+1)} \sum_{l_2=Nk_2}^{N(k_2+1)} \sum_{l_1=Nk_1}^{N(k_1+1)} \boldsymbol{\Phi}^{\boldsymbol{k}}_{\boldsymbol{l}} \left(x^{\boldsymbol{k}}_{i_1}, y^{\boldsymbol{k}}_{i_2}, z^{\boldsymbol{k}}_{i_3}\right) \left(\boldsymbol{\xi}_{l_1}, \boldsymbol{\eta}_{l_2}, \boldsymbol{\nu}_{l_3}\right), \tag{7.103}$$

where $\boldsymbol{\Phi_l}$ are the interpolation polynomials to the uniform nodes

$$\boldsymbol{\Phi}^{\boldsymbol{k}}_{\boldsymbol{l}}(x_{\boldsymbol{i}}) = \delta_{\boldsymbol{li}}, \text{ for } \boldsymbol{l}, \boldsymbol{i} : \; l_s, i_s \in \{Nk_s, ..., N(k_s+1)\}, s = 1, ..., 3. \tag{7.104}$$

Equation (7.103) ensures that nodes on the shared face of two adjacent elements in the computational mesh, share the same transformation. As consequence we can be certain that their interfaces in computational space are projected to the same interface in physical space. The method we use to construct a transformation, that represents topographies, faults and sharp material interfaces, is the object of the next sections.

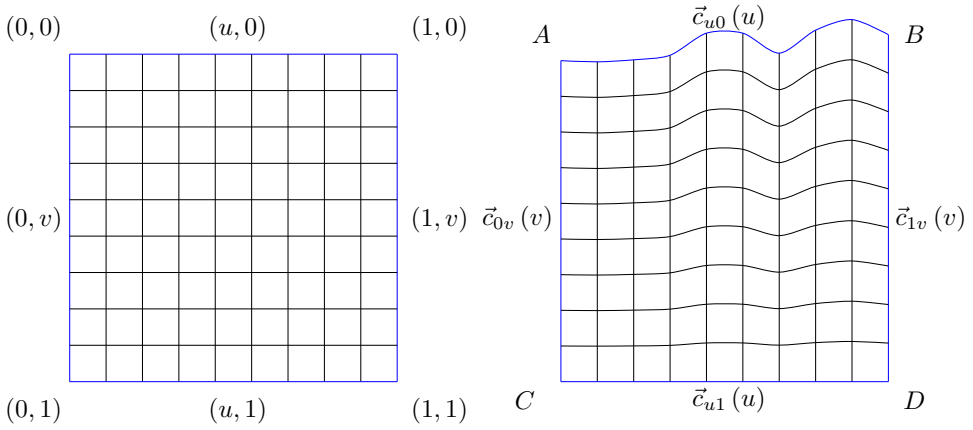


Figure 7.6.: Sketch of the two dimensional transfinite interpolation. The conforming boundary curves $\vec{c}_{u0}(u), \vec{c}_{0v}(v), \vec{c}_{u1}(u), \vec{c}_{1v}(v)$ define a surface (right, blue), for which the transfinite interpolation returns a diffeomorphic transformation from the unit square (left). The isolines are evenly distributed over the domain (black lines).

7.6.1 The transfinite interpolation

The core of our automatic meshing approach is the transfinite interpolation. The transfinite interpolation describes a locally-invertible (diffeomorphic) transformation from Cartesian coordinates to curvilinear coordinates (also called computational and physical domain) for a given set of boundary curves in two and boundary surfaces in three dimensions. For the sake of brevity we only recapture the formulation of the two dimensional interpolation $\vec{\mathcal{C}}^2[\lambda^n](u,v) : [0,1]^2 \to \mathbb{R}^n$ at this point and refer the reader for the three dimensional case to Appendix B

$$\begin{aligned} \vec{\mathcal{C}}^2[\lambda^n](u,v) = \\ (1-u)\vec{c}_{0v}(v) + u\vec{c}_{1v}(v) + (1-v)\vec{c}_{u0}(u) + v\vec{c}_{u1}(u) - \\ \left((1-v)(1-u)\cdot\vec{A} + (1-v)u\cdot\vec{B} + v(1-u)\cdot\vec{C} + vu\cdot\vec{D}\right). \end{aligned} \tag{7.105}$$

$\vec{\mathcal{C}}^2[\lambda^n](x,y)$ is the transfinite interpolation to the ordered set of boundary curves

$$\lambda^n = (\vec{c}_{u0}(u), \vec{c}_{u1}(u), \vec{c}_{0v}(v), \vec{c}_{1v}(v)), \tag{7.106}$$

which are each functions $[0,1] \to \mathbb{R}^n$.

The set λ^n holds the left, right, top and bottom curves that the transfinite interpolation returns along the lines $(u,0),(u,1),(0,v),(1,v)$ for $u,v \in [0,1]$. The curves have to be conforming, i.e. they have to be parameterized such that their intersection points are placed at the vertices of the domain of definition $(u,v) \in \{(0,0),(1,0),(0,1),(1,1)\}$

$$\begin{aligned} \vec{A} &= \vec{c}_{0v}(0) = \vec{c}_{u0}(0), & \vec{B} &= \vec{c}_{u0}(1) = \vec{c}_{1v}(0), \\ \vec{C} &= \vec{c}_{u1}(0) = \vec{c}_{0v}(1), & \vec{D} &= \vec{c}_{1v}(1) = \vec{c}_{u1}(1). \end{aligned} \tag{7.107}$$

In Figure 7.6 we show the transformation of the uniform square to a curvilinear surface with prescribed topography $\vec{c}_{u0}(u) = (u, \mathcal{T}(u))$, while all other boundary curves are modeled as straight lines. We see that the projected isolines $v_j = \frac{j}{9}$, $j = 0,...,9$ in the uniform square are evenly distributed over the domain and gradually match the straight line at the bottom and the topography at the top. For the uniform nodes $(u_i, v_j) = \left(\frac{i}{9}, \frac{j}{9}\right)$, $i,j = 0,...,9$ the transformation gives us corresponding projected nodes $\vec{\mathcal{C}}^2[\lambda^2](u_i, v_j)$ for a curvilinear mesh.

In three dimensions the transfinite interpolation becomes a function $\vec{\mathcal{C}}^3[\Lambda^n](u,v,w) : [0,1]^3 \to \mathbb{R}^n$. It is now based on a set of conforming boundary surfaces containing the left, right, top, bottom, front and back surface

$$\begin{aligned} \Lambda^n = \Big(& \vec{F}_{0vw}(v,w), \vec{F}_{1vw}(v,w), \vec{F}_{u0w}(u,w), \\ & \vec{F}_{u1w}(u,w), \vec{F}_{uv0}(u,v), \vec{F}_{uv1}(u,v) \Big). \end{aligned} \tag{7.108}$$

All surfaces are functions $[0,1]^2 \to \mathbb{R}^n$. Analogous to boundary curves in two dimensions, boundary surfaces define the images of the transfinite interpolation for the six faces of the unit-cube $[0,1]^3$

$$\begin{aligned} \vec{\mathcal{C}}^3[\Lambda^n](u,v,0) &= \vec{F}_{uv0}(u,v), & \vec{\mathcal{C}}^3[\Lambda^n](u,v,1) &= \vec{F}_{uv1}(u,v), \\ \vec{\mathcal{C}}^3[\Lambda^n](u,0,w) &= \vec{F}_{u0w}(u,w), & \vec{\mathcal{C}}^3[\Lambda^n](u,1,w) &= \vec{F}_{u1w}(u,w), \\ \vec{\mathcal{C}}^3[\Lambda^n](0,v,w) &= \vec{F}_{0vw}(v,w), & \vec{\mathcal{C}}^3[\Lambda^n](1,v,w) &= \vec{F}_{1vw}(v,w). \end{aligned} \tag{7.109}$$

In order to be conforming with the transfinite interpolation, the parameterization of the boundary surfaces has to be matching at the twelve intersection

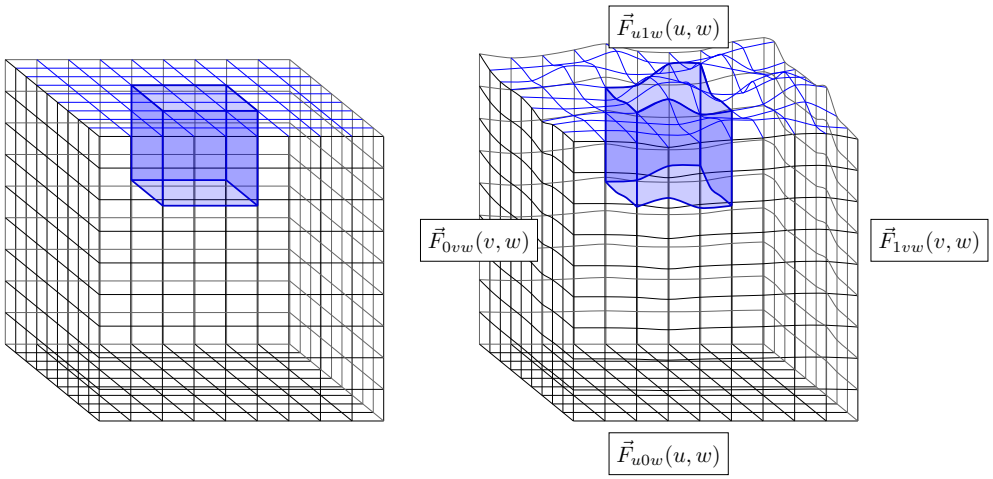


Figure 7.7.: Transfinite interpolation to a set of conforming boundary surfaces (black isolines), modeling a topography on top (blue isolines). Marked is the transformation for the element from Figure 7.5 (blue volume).

paths:

$$\begin{aligned}
\vec{F}_{uv0}(u,0) &= \vec{F}_{u0w}(u,0), & \vec{F}_{uv1}(u,0) &= \vec{F}_{u0w}(u,1),\\
\vec{F}_{uv0}(u,1) &= \vec{F}_{u1w}(u,0), & \vec{F}_{uv1}(u,1) &= \vec{F}_{u1w}(u,1),\\
\vec{F}_{uv0}(0,v) &= \vec{F}_{0vw}(v,0), & \vec{F}_{uv1}(0,v) &= \vec{F}_{0vw}(v,1),\\
\vec{F}_{uv0}(1,v) &= \vec{F}_{1vw}(v,0), & \vec{F}_{uv1}(1,v) &= \vec{F}_{1vw}(v,1),\\
\vec{F}_{u0w}(0,w) &= \vec{F}_{0vw}(0,w), & \vec{F}_{u1w}(0,w) &= \vec{F}_{0vw}(1,w),\\
\vec{F}_{u0w}(1,w) &= \vec{F}_{1vw}(0,w), & \vec{F}_{u1w}(1,w) &= \vec{F}_{1vw}(1,w),
\end{aligned} \tag{7.110}$$

for $u, v, w \in [0,1]$. A set of conforming boundary surfaces in three dimensions (and boundary curves in two dimensions) can be interpreted as the definition of empty hulls. The transfinite interpolation is the transformation from the unit cube onto a boundary fitting volume for that hull. Having the transformation to a set of boundary surfaces, we can generate transformed coordinates for the uniform mesh from Equation (7.102), that fit with predefined surfaces (See Figure 7.7).

7.6.2 Topography fitting curvilinear meshes

The entry point of our meshing approach is the generation of conforming boundary surfaces for a prescribed region with topography. We want to represent the topography of the domain as one of the boundary surfaces and model with the domain, the interior of the Earth down to a preset depth.

The method we present here is invariant under all permutations of the coordinates x, y and z. We construct the example from Figure 7.7 and assume w.l.o.g. that we model the region $[\tilde{x}_0, \tilde{x}_1] \times [\tilde{z}_0, \tilde{z}_1]$ down to a depth of $\tilde{y}_1$. The topography is given as scalar field $\mathcal{T} : \mathbb{R}^2 \to \mathbb{R}$ in the coordinates x and z. In order to construct a conforming set of boundary surfaces that represent the topography in the defined region, we map the top and bottom faces of the unit cube to the surfaces given by the topography and the depth of the domain. For $(u, w) \in [0, 1]^2$

$$\begin{aligned} \vec{F}_{u0w}(u, w) &:= (\tilde{x}(u), \mathcal{T}(\tilde{x}(u), \tilde{z}(w)), \tilde{z}(w)) \\ \vec{F}_{u1w}(u, w) &:= (\tilde{x}(u), \tilde{y}_1, \tilde{z}(w)), \end{aligned} \tag{7.111}$$

where $\tilde{x}(u) = u \cdot (\tilde{x}_1 - \tilde{x}_0) + \tilde{x}_0$ and $\tilde{z}(w) = w \cdot (\tilde{z}_1 - \tilde{z}_0) + \tilde{z}_0$. In order to conform with the edges of the domain, we generate the remaining boundary surfaces using the two dimensional curvilinear interpolation. Our constructive approach is shown in Figure 7.8: As the generated surfaces have to be conforming, the top and bottom edges of all missing faces are determined by the edges of the top and bottom surface. For the front surface $\vec{F}_{uv0}(u, v)$, we construct the left and right edge by connecting the vertices of the edges with a straight line (left, red lines)

$$\begin{aligned} \vec{c}_{u0}(u) &:= \vec{F}_{u0w}(u, 0), \\ \vec{c}_{u1}(u) &:= \vec{F}_{u1w}(u, 0), \\ \vec{c}_{0v}(v) &:= v\left(\vec{F}_{u1w}(0, 0) - \vec{F}_{u0w}(0, 0)\right) + \vec{F}_{u0w}(0, 0), \\ \vec{c}_{1v}(v) &:= v\left(\vec{F}_{u1w}(1, 0) - \vec{F}_{u0w}(1, 0)\right) + \vec{F}_{u0w}(1, 0). \end{aligned} \tag{7.112}$$

Having the set of all four boundary curves λ^3_{uv0} we can represent the front surface with the two dimensional transfinite interpolation

$$\vec{F}_{uv0}(u, v) = \vec{\mathcal{C}}^2\left[\lambda^3_{uv0}\right](u, v). \tag{7.113}$$

We get the remaining three boundary surfaces $\vec{F}_{uv1}(u, v)$, $\vec{F}_{0vw}(v, w)$ and

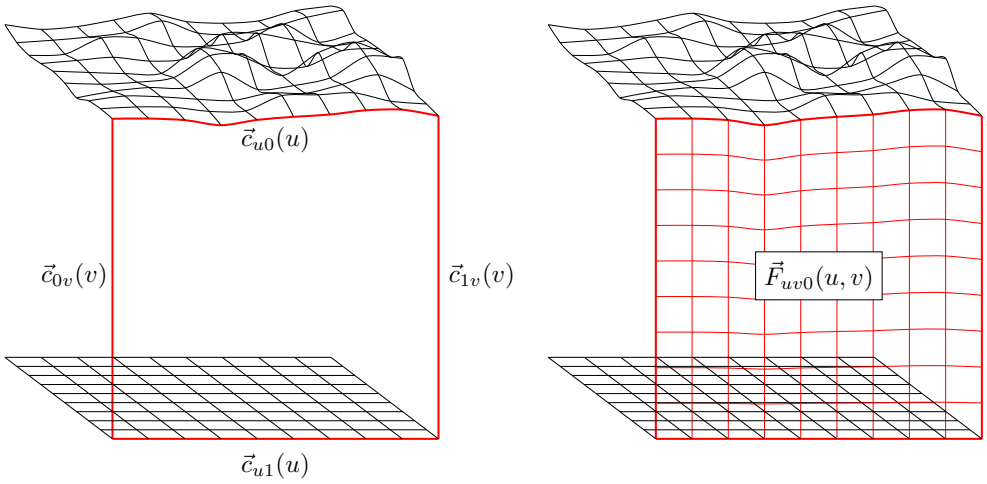


Figure 7.8.: Construction of the front face for a given top and bottom face. The top surface represents some topography, while the bottom is predefined by the depth. By taking the two edges of the top and bottom face and constructing the two missing edges (left, red lines), we get the front face by evaluating the two dimensional transfinite interpolation (right red).

$\vec{F}_{1vw}(v,w)$ analogously and end up with a set of conforming boundary surfaces Λ^3. The corresponding transfinite interpolation gives us the transformation from the unit-cube to a topography fitting volume

$$\tau(u,v,w) = \vec{\mathfrak{C}}^3\left[\Lambda^3\right](u,v,w)\,. \tag{7.114}$$

In case we require a mesh only to represent the topography (as for the Zugspitze scenario in Section 8.4) and no structures in its interior, we can generate the transformation for the uniform grid in Equation (7.102) and are finished with meshing at this point.

7.6.3 Imposing fault structures

Next we want to impose structures in the physical domain we constructed in the last section. Examples for these structures can be sharp material parameter interfaces, as we encounter them in the Loh1 scenario (Section 8.2.2) or fault structures, as they appear for the TPV benchmarks (Section 8.3). For each structure we want to incorporate, we define a corresponding face

h_i inside the unit-cube, parallel to the u-v, u-w or v-w-plane, and a surface S_i in the physical domain. The final transformation has to project each face onto its corresponding surface,

$$\tau(h_i) \stackrel{!}{=} S_i,\ \forall (h_i, S_i) \in \mathcal{G}. \tag{7.115}$$

Here we introduced the set of *constraints* $\mathcal{G}$ to summarize all pairs of faces and surfaces we want to incorporate in the final transformation

$$\mathcal{G} = \{(h_i, S_i), i \in I\}, \tag{7.116}$$

with a set of indices I. In order to define the transformation, we use the faces h_i to inductively divide the unit-cube into cuboids

$$Q_j = [\bar{u}_j, \bar{u}_{j+1}] \times [\bar{v}_j, \bar{v}_{j+1}] \times [\bar{w}_j, \bar{w}_{j+1}]. \tag{7.117}$$

For each cuboid Q_j we construct corresponding (conforming) boundary surfaces Λ_j^3. Pairs of cuboids and boundary surfaces, describe a simultaneous *sub-division* of the computational and physical domain, which we summarize with the set $\mathcal{B}$:

$$\mathcal{B} = \left\{ \left(Q_j, \Lambda_j^3 \right), j \in J \right\}, \tag{7.118}$$

for a set J of indices. The final transformation for the unit-cube is defined as piecewise transfinite interpolations for each pair in $\mathcal{B}$

$$\tau(u, v, w) := \vec{\mathcal{C}}^3 \left[\Lambda_j^3 \right] \left(\frac{u - \bar{u}_j}{\bar{u}_{j+1} - \bar{u}_j}, \frac{v - \bar{v}_j}{\bar{v}_{j+1} - \bar{v}_j}, \frac{w - \bar{w}_j}{\bar{w}_{j+1} - \bar{w}_j} \right), \tag{7.119}$$

for $(u, v, w) \in Q_j$ and $j \in J$.

In the next sections of this chapter we introduce an inductive approach that generates a sub-division (Equation (7.118)) of the unit-cube and the boundary faces we introduced in Section 7.6.2, such that a set of constraints as in Equation (7.116) are fulfilled in the final transformation in Equation (7.119).

As approach and proof become very technical, we want to outline the single steps up front. Figure 7.9 illustrates the approach:

1. We start with the unit-cube $[0, 1]^3$ and its corresponding set of boundary faces Λ from Section 7.6.2, that represent a topography. Additionally, we consider the first constraint (h_0, S_0) from $\mathcal{G}$.

2. We split the unit cube into two cuboids Q_1, Q_2 and the set of boundary

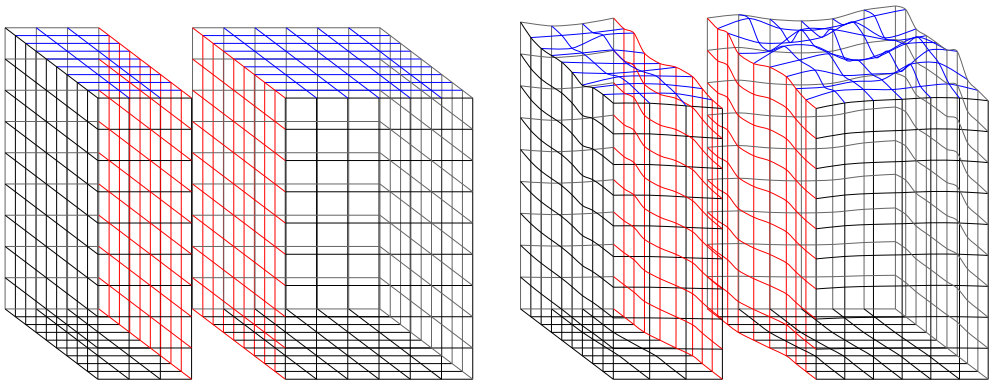


Figure 7.9.: Sub-division of the unit cube and the conforming set boundary surfaces based on a condition (red). The new sets of boundary faces (right) represent the structure at their shared interface (red), while the previously introduced topography remains in the representation.

faces into two new sets ${}^1\Lambda$, ${}^2\Lambda$. The cuboids share the face h_0 at their interface. The new sets of boundary faces share the surface at their interface.

Splitting the unit cube is trivial: For example a face $h_0 = 0.5 \times [0,1] \times [0,1]$, gives us cuboids $Q_1 = [0, 0.5] \times [0,1] \times [0,1]$ and $Q_2 = [0.5, 1] \times [0,1] \times [0,1]$. To split the set of boundary faces with the surface we have to perform a series of intermediate steps:

- We have to find those boundary faces $F \in \Lambda$ that intersect with S_i.
- For these faces we find the path along which they intersect with S_i.
- With the intersection path, we can split the faces into two new faces ${}^1F$ and ${}^2F$.
- Having split all faces we can construct new conforming sets of boundary faces ${}^1\Lambda$, ${}^2\Lambda$, that share the structure S_i at their interface.

3. For the next structure (h_{n+1}, S_{n+1}), we return to the first step and perform the following steps for all pairs of cuboids that intersect with the face h_{n+1}.

Inductive construction of sub-divisions

In order to simplify the notation, we define the domain $\mathcal{D}(\hat{\Lambda}^3)$ of boundary surfaces $\hat{\Lambda}^3$, as the image of the unit-cube to their corresponding three dimensional transfinite interpolation,

$$\mathcal{D}(\hat{\Lambda}^3) = \left\{ \vec{\mathcal{C}}^3 \left[\hat{\Lambda}^3 \right] (u,v,w) \,, (u,v,w) \in [0,1]^3 \right\}. \tag{7.120}$$

By definition we get the equation

$$\forall j \in J : \tau\left(Q_j\right) = \mathcal{D}(\Lambda_j^3). \tag{7.121}$$

The global domain from Section 7.6.2 is denoted with $\mathcal{D}(\Lambda^3)$.

As the transformation has to be well defined, the cuboids have to cover the whole unit-cube. We want keep the representation of the topography in the global domain as image of the unit cube, thus the union of the images of the cuboids has to be $\mathcal{D}(\Lambda^3)$:

$$\bigcup_{j\in J} Q_j \stackrel{!}{=} [0,1]^3, \qquad \bigcup_{j\in J} \tau\left(Q_j\right) = \bigcup_{j\in J} \mathcal{D}(\Lambda_j^3) \stackrel{!}{=} \mathcal{D}(\Lambda^3). \tag{7.122}$$

With the definition of the transformation in Equation (7.119), we can rewrite Equation (7.115) as condition of the sub-division $\mathcal{B}$. For every constraint $(h_i, S_i) \in \mathcal{G}$, we require that there exist two subsets of the sub-division such that the face is represented by the interface of the two subset unions of cuboids and the surface by the interface of the two subset unions of the domains. $\forall (h_i, S_i) \in \mathcal{G}$, $\exists \mathcal{B}_1, \mathcal{B}_2 \subset \mathcal{B}$:

$$\left(\bigcup_{(Q_{j_1}, \Lambda_{j_1}^3) \in \mathcal{B}_1} Q_{j_1} \right) \cap \left(\bigcup_{(Q_{j_2}, \Lambda_{j_2}^3) \in \mathcal{B}_2} Q_{j_2} \right) = h_i \tag{7.123}$$

$$\left(\bigcup_{(Q_{j_1}, \Lambda_{j_1}^3) \in \mathcal{B}_1} \mathcal{D}(\Lambda_{j_1}^3) \right) \cap \left(\bigcup_{(Q_{j_2}, \Lambda_{j_2}^3) \in \mathcal{B}_2} \mathcal{D}(\Lambda_{j_2}^3) \right) = S_i. \tag{7.124}$$

We provide an inductive approach to construct a sub-division $\mathcal{B}$, that fulfills an arbitrary set of constraints $\mathcal{G}$, as defined in Equation (7.122), Equation (7.123) and Equation (7.124).

As initial case we choose the result from Section 7.6.2, with $\mathcal{B}_0 := \{[0,1]^3, \Lambda^3\}$, that trivially meets Equation (7.123), Equation (7.124), and Equation (7.122) as $\mathcal{G}_0 = \emptyset$. For the induction step we assume an existing

sub-division $\mathcal{B}_n$, that obeys all n constraints in

$$\mathcal{G}_n := \mathcal{G}_{n-1} \cup \{(h_n, S_n)\}. \tag{7.125}$$

To add a new surface S_{n+1} as the image of a face h_{n+1} to the sub-division, we pick those pairs in $\mathcal{B}_n$ where the cuboid intersects with the face

$$\underline{\mathcal{B}}_n = \left\{(Q_j, \Lambda_j^3) : Q_j \cap h_{n+1} \neq \emptyset, (Q_j, \Lambda_j^3) \in \mathcal{B}_n\right\}. \tag{7.126}$$

For the definition of the new surface S_{n+1} we require, that it fully lies in the unified domain of the corresponding boundary surfaces,[11]

$$S_{n+1} \cap \left(\bigcup_{(Q_j, \Lambda_j^3) \in \underline{\mathcal{B}}_n} \mathcal{D}(\Lambda_j^3) \right) = S_{n+1}. \tag{7.127}$$

We divide each cuboid Q_j and its corresponding boundary surfaces Λ_j^3 into two new pairs, to the left $({}^1Q_j,\ {}^1\Lambda_j^3)$ and to the right $({}^2Q_j, {}^2\Lambda_j^3)$ of the face and surface.[12] The new cuboids share h_{n+1} as interface and the fraction of S_{n+1} that lies inside the initial domain $\mathcal{D}(\Lambda_j^3)$, is the interface of the domains to the divided boundary surfaces

$${}^1Q_j \cup {}^2Q_j = Q_j, {}^1Q_j \cap {}^2Q_j = h_{\bar{c}} \cap Q_j \tag{7.128}$$

$$\mathcal{D}({}^1\Lambda_j^3) \cup \mathcal{D}({}^2\Lambda_j^3) = \mathcal{D}(\Lambda_j^3), \tag{7.129}$$

$$\mathcal{D}({}^1\Lambda_j^3) \cap \mathcal{D}({}^2\Lambda_j^3) = S_{n+1} \cap \mathcal{D}(\Lambda_j^3). \tag{7.130}$$

We sort the pairs into the ones located on the left of the face $\overline{\mathcal{B}}_n^1$ and on the right $\overline{\mathcal{B}}_n^2$. In order to construct a new sub-division $\mathcal{B}_{n+1}$ we replace each original pair $\left(Q_j, \Lambda_j^3\right)$ in $\mathcal{B}_n$ with the newly constructed pairs

$$\mathcal{B}_{n+1} = (\mathcal{B}_n / \underline{\mathcal{B}}_n) \cup \overline{\mathcal{B}}_n^1 \cup \overline{\mathcal{B}}_n^2.$$

The unions in Equation (7.128) make it trivial to prove that the new sub-division fulfills Equation (7.123) and Equation (7.124). We can also use them to easily prove that Equation (7.122) holds for all constraints $(h_i, S_i) \in \mathcal{G}_n$.

[11] This constraint is naturally met for realistic mesh geometries.

[12] We define that the normals to the faces h_i are oriented in positive direction, which gives us definition of left and right.

In case of (h_{n+1}, S_{n+1}), we take the new defined sets $\overline{\mathcal{B}}_n^1$ and $\overline{\mathcal{B}}_n^2$ and get

$$\left(\bigcup_{({}^1Q_j, {}^1\Lambda_j^3) \in \overline{\mathcal{B}}_n^1} \mathcal{D}({}^1\Lambda_j^3) \right) \cap \left(\bigcup_{({}^2Q_j, {}^2\Lambda_j^3) \in \overline{\mathcal{B}}_n^2} \mathcal{D}({}^2\Lambda_j^3) \right) =$$

$$\left(\bigcup_{({}^1Q_j, {}^1\Lambda_j^3) \in \overline{\mathcal{B}}_n^1} \bigcup_{({}^2Q_j, {}^2\Lambda_j^3) \in \overline{\mathcal{B}}_n^2} \mathcal{D}({}^1\Lambda_j^3) \cap \mathcal{D}({}^2\Lambda_j^3) \right) = \tag{7.131}$$

$$\left(\bigcup_{(Q_j, \Lambda_j^3) \in \underline{\mathcal{B}}_n} \mathcal{D}(\Lambda_j^3) \cap S_{n+1} \right) \overset{(7.128)}{=} S_{n+1} \cap \left(\bigcup_{j \in \underline{J}} \mathcal{D}(\Lambda_j^3) \right) \overset{(7.127)}{=} S_{n+1}.$$

The inductive approach gives us a method to add structures bit by bit to an existing transformation. The last tool that is missing are the operators to split cuboids by faces and boundary surfaces by structures that meet Equation (7.128).

Splitting a cuboid by a face is trivial, the definition of an operator to split a set of conforming boundary surfaces by a surface is the object of the next paragraphs.

Splitting conforming sets of boundary surfaces

We have to construct a method to split a set of boundary surfaces Λ_j^3 into two sets ${}^1\Lambda_j^3$ and ${}^2\Lambda_j^3$, that obey Equation (7.129). In order to simplify our approach, we require that every surface S_i has to be represented by an associated scalar field $\mathcal{S}_i : \mathbb{R}^2 \to \mathbb{R}$, that is evaluated for a set of curvilinear coordinates:

$$S_i(u, v) = (\tilde{u}, \tilde{v}, \mathcal{S}(\tilde{u}, \tilde{v})), \text{ for } (\tilde{u}, \tilde{v}) = \vec{\mathcal{C}}^2 \left[\lambda_i^2 \right] (u, v) \text{ or} \tag{7.132}$$

$$S_i(u, w) = (\tilde{u}, \mathcal{S}(\tilde{u}, \tilde{v}), \tilde{w}), \text{ for } (\tilde{u}, \tilde{w}) = \vec{\mathcal{C}}^2 \left[\lambda_i^2 \right] (u, w) \text{ or} \tag{7.133}$$

$$S_i(v, w) = (\mathcal{S}(\tilde{v}, \tilde{w}), \tilde{v}, \tilde{w}), \text{ for } (\tilde{v}, \tilde{w}) = \vec{\mathcal{C}}^2 \left[\lambda_i^2 \right] (v, w). \tag{7.134}$$

for $u, v, v \in [0, 1]$ and $\hat{u}, \hat{v}, \hat{w} \in \{0, 1\}$. λ_i^2 are the corresponding sets of conforming boundary curves,

$$\lambda_i^2 = (\vec{c}_{u0}(u), \vec{c}_{u1}(u), \vec{c}_{0v}(v), \vec{c}_{1v}(v)). \tag{7.135}$$

This choice for the representation of surfaces allows us to interpret every surface as the cutaway of the graph to a scalar-field. The cutting lines are

the boundary curves of λ_i^2. In order to split a surface we can redefine the boundary curves to match with the intersection.

W.l.o.g. we present the method for the division by a surface in v-w, as defined in Equation (7.134). With simple permutations the method can be equivalently used for structures defined in u-v (Equation (7.132)) and u-w (Equation (7.133)). Figure 7.9 illustrates the operator for this example.

In order to find two new conforming sets of boundary faces ${}^1\Lambda_j^3$ and ${}^2\Lambda_j^3$, we have to split all surfaces in Λ_j^3 that intersect with S_i. For our example these are the top, bottom, front and back surface in Λ_j^3. By denoting the surface-surface splitting operators with $\bullet/^1\bullet$ and $\bullet/^2\bullet$, we get ${}^1\Lambda_j^3$ and ${}^2\Lambda^3$ with

$$
\begin{aligned}
{}^1\Lambda_j^3 = \Big(& \vec{F}_{0vw}(v,w)\,,\ \hat{S}_i(v,w), \vec{F}_{u0w}(u,w)\,/^1S_i, \\
& \vec{F}_{u1w}(u,w)\,/^1S_i,\ \vec{F}_{uv0}(u,v)\,/^1S_i, \vec{F}_{uv1}(u,v)\,/^1S_i \Big)
\end{aligned} \tag{7.136}
$$

$$
\begin{aligned}
{}^2\Lambda_j^3 = \Big(& \hat{S}_i(v,w),\ \vec{F}_{v1w}(v,w)\,, \vec{F}_{u0w}(u,w)\,/^2S_i, \\
& \vec{F}_{u1w}(u,w)\,/^2S_i,\ \vec{F}_{uv0}(u,v)\,/^2S_i, \vec{F}_{uv1}(u,v)\,/^2S_i \Big).
\end{aligned} \tag{7.137}
$$

The surface-surface splitting operators return the left and right parts of a boundary surface, divided along the intersection path with S_i, as shown in Figure 7.10. The new sets of boundary faces share the reparametrized surface $\hat{S}_i$ at their interface and by design meet the constraints in Equation (7.124).

In order to define the surface-surface splitting operators, we again look at an example for a fixed set of coordinates. With permutations, the method can be translated to all cases. We look at the intersections $F_1 := \vec{F}_{u0w}(u,w)\,/^1S_i$ and $F_2 := \vec{F}_{u0w}(u,w)\,/^2S_i$. By the restriction in Equation (7.133), we can represent $\vec{F}_{u0w}(u,w)$ with

$$
\vec{F}_{u0w}(u,w) := (\tilde{u}, \mathcal{F}(\tilde{u},\tilde{w}), \tilde{w}),\ \text{for}(\tilde{u},\tilde{w}) = \vec{\mathcal{C}}^2\left[\mu^2\right](u,w) \tag{7.138}
$$

for some scalar field $\mathcal{F}$ and boundary curves

$$
\mu^2 = (\vec{C}_{u0}(u)\,, \vec{C}_{u1}(u)\,, \vec{C}_{0w}(w)\,, \vec{C}_{1w}(w)). \tag{7.139}
$$

We have to find the intersection path $\vec{I}(w)$ of $\vec{F}_{u0w}(u,w)$ and S_i along

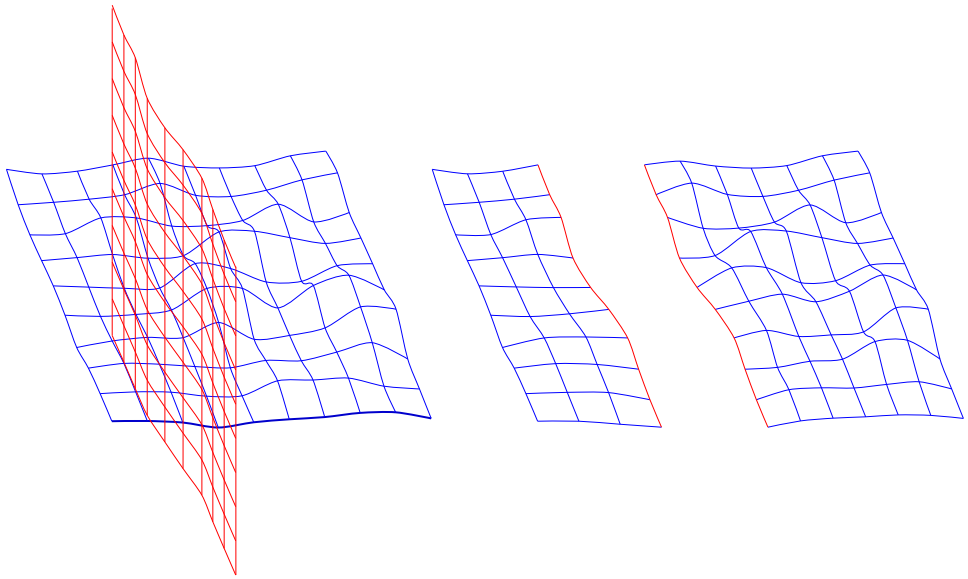


Figure 7.10.: Surface-surface splitting operator: We find the intersection path of two surfaces (left) to split the first surface (right blue) along the intersection path (right red). The two resulting new surfaces are reparametrized.

their shared coordinate w,

$$\begin{aligned} \vec{I}(w) &= (I_x(w), I_y(w), I_z(w)) : \\ (I_x(w), \mathcal{F}(I_x(w), I_z(w)), I_z(w)) &= (\mathcal{S}(I_x(w), I_y(w)), I_y(w), I_z(w)) \,, \end{aligned} \tag{7.140}$$

for $w \in [0, 1]$.

Having the intersection path $I(w)$, we can next define two new sets of conforming boundary curves μ_1^2 and μ_2^2 and split the surface into two reparametrized surfaces F_1 and F_2,

$$\begin{aligned} F_1(u, w) =& (\tilde{u}, F(\tilde{u}, \tilde{w}), \tilde{w}), \text{ for } (\tilde{u}, \tilde{w}) = \vec{\mathcal{C}}^2 \left[\mu_1^2\right](u, w), \\ F_2(u, w) =& (\tilde{u}, F(\tilde{u}, \tilde{w}), \tilde{w}), \text{ for } (\tilde{u}, \tilde{w}) = \vec{\mathcal{C}}^2 \left[\mu_2^2\right](u, w). \end{aligned} \tag{7.141}$$

We take the boundary curves from Equation (7.139) and split the top and

bottom curve with the intersection path

$$\begin{aligned} {}^1\vec{C}_{u0}(u) &= \vec{C}_{u0}(u \cdot \bar{u}_0), \quad & {}^2\vec{C}_{u0}(u) &= \vec{C}_{u0}(u \cdot (1-\bar{u}_0) + \bar{u}_0), \\ {}^1\vec{C}_{u1}(u) &= \vec{C}_{u1}(u \cdot \bar{u}_1), \quad & {}^2\vec{C}_{u1}(u) &= \vec{C}_{u1}(u \cdot (1-\bar{u}_1) + \bar{u}_1), \\ & & \underline{\vec{I}}(w) &= \vec{I}(w \cdot (\bar{w}_1 - \bar{w}_0) + \bar{w}_0), \end{aligned} \tag{7.142}$$

where $\bar{u}_0$,$\bar{u}_1$,$\bar{w}_0$ and $\bar{w}_1$ are the points where the curves intersect with the intersection path

$$\vec{C}_{u0}(\bar{u}_0) = I(\bar{w}_0), \qquad \vec{C}_{u1}(\bar{u}_1) = I(\bar{w}_1). \tag{7.143}$$

With the split reparametrized curves and the intersection path, we can generate the two new conforming sets of boundary curves,

$$\mu_1^2 = ({}^1\vec{C}_{u0}(u), {}^1\vec{C}_{u1}(u), \vec{C}_{0w}(w), \underline{\vec{I}}(w)), \tag{7.144}$$

$$\mu_2^2 = ({}^2\vec{C}_{u0}(u), {}^2\vec{C}_{u1}(u), \underline{\vec{I}}(w), \vec{C}_{1w}(w)). \tag{7.145}$$

After splitting all faces in Equation (7.136) and Equation (7.137), we find a conforming reparameterization of the surface S_i, by taking the four intersection paths as boundary curves

$$\hat{\lambda}^2 = \left(\underline{\vec{I}}_{uv0}(v), \underline{\vec{I}}_{uv1}(v), \underline{\vec{I}}_{u0w}(w), \underline{\vec{I}}_{u1w}(w)\right) \tag{7.146}$$

and define then new boundary face with:

$$\hat{S}_i(v,w) = (\mathcal{S}(\tilde{v},\tilde{w}), \tilde{v}, \tilde{w}), \text{ for } (\tilde{v},\tilde{w}) = \vec{\mathcal{C}}^2\left[\hat{\lambda}^2\right](v,w). \tag{7.147}$$

Approximating intersections

The last missing pieces are the intersection path in Equation (7.140) and the intersections in Equation (7.143). First we numerically approximate Equation (7.140) to the problem of finding intersections on isolines. We take the N isolines of both surfaces along z for equidistant values $z_k \in [z_{min}, z_{max}]$, $z_{k+1} = \Delta z + z_k$, and z_{min} and z_{max} are the minimal an maximal z values in $S_i([0,1]^2)$ and $\vec{F}_{u0w}([0,1]^2)$. By interpolating the isolines with linear splines on N subintervals, defined by $x_{i+1} = x_i + \Delta x$, and $y_{j+1} = y_j + \Delta y$, we can reduce the problem to a linear equation system

$$\begin{aligned} &(x, \mathcal{F}(x, z_k), z_k) \approx \\ &(x_i, \mathcal{F}(x_i, z_k), z_k) + \lambda_i(x)\,(\Delta x, \mathcal{F}(x_{i+1}, z_k) - \mathcal{F}(x_i, z_k), 0) \end{aligned} \tag{7.148}$$

$$
\begin{aligned}
&(\mathcal{S}(y, z_k), y, z_k) \approx \\
&(\mathcal{S}(y_j, z_k), y_j, z_k) + \mu_j(y) \left(\mathcal{S}(y_{j+1}, z_k) - \mathcal{S}(y_j, z_k), \Delta y, 0\right), \qquad (7.149)\\
&\text{where } \lambda_i(x) = (x - x_i)\Delta x^{-1} \text{ and } \mu_j(y) = (y - y_j)\Delta y^{-1}, \qquad (7.150)
\end{aligned}
$$

for $x \in [x_i, x_{i+1}]$, $y \in [y_i, y_{i+1}]$. Along the isolines z_k we are left with the problem

$$
\begin{aligned}
\mathcal{F}(y_j, z_k) + \mu_j(y) \cdot (\mathcal{F}(y_{j+1}, z_k) - \mathcal{F}(y_j, z_k)) &= x_i + \lambda_i(x) \cdot \Delta x, \qquad (7.151)\\
\mathcal{S}(x_i, z_k) + \lambda_i(x) \cdot (\mathcal{S}(x_{i+1}, z_k) - \mathcal{S}(x_i, z_k)) &= y_j + \mu_j(y) \cdot \Delta y, \qquad (7.152)
\end{aligned}
$$

which we can easily solve for λ_i or μ_j. With λ_i and μ_j, we can compute the intersection points $\hat{x}_k$ and $\hat{y}_k$, along the isolines z_k. With all N intersection points $(\hat{x}_n, \hat{y}_n, z_n)$, we reconstruct the intersection path with linear splines,

$$
\vec{I}(w) = (w \cdot N - n)\left(\hat{x}_{n+1} - \hat{x}_n, \hat{y}_{n+1} - \hat{y}_n, \Delta z\right) + (\hat{x}_n, \hat{y}_n, z_n), \qquad (7.153)
$$

for $w \in [n/N, (n+1)/N]$.

With the approach that we used to intersect isolines, we can find the intersection points of two curves as required in Equation (7.143).

TPV24: Recursive boundary splitting

The resulting sets of boundary faces are both conforming and their elements all agree with the assumptions we made. This means, we can impose multiple structures, by recursively splitting pairs of sub-cuboids and corresponding conforming sets of boundary faces. In order to illustrate this approach, we look at the example of the TPV24 dynamic rupture benchmark, which is defined by a main fault and a branch fault [67]. The domain represents a flat topography and is defined in the cube $[-14, 28] \times [0, 28] \times [0, 28]$ km. We can define the initial domain as in Section 7.6.2 and receive a conforming set of boundary faces Λ^3.

The main fault of the benchmark is a planar fault in y and z at $x = 0$, this can be defined by the surface:

$$
S_1 = \{(0, y, z),\ (y, z) \in [0, 28] \times [0, 28]\}. \qquad (7.154)
$$

The branch fault has an angle of 30 degrees to the main fault, and is only

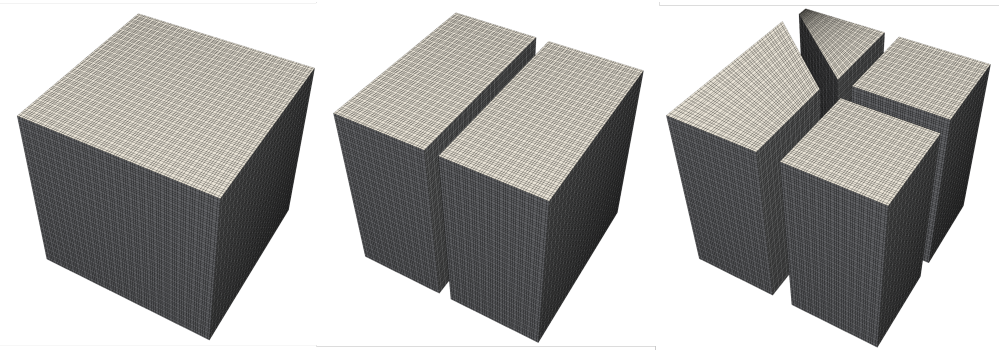


Figure 7.11.: Generation of the conforming boundary sets for TPV24.

defined in the area $x > 0$. We can incorporate it with the surface:

$$S_2 = \{(x, y, 14),\, (x, y) \in [-14, 0] \times [0, 28]\} \cup \\ \{(x, y, 14 + \sqrt{3}/2 \cdot x),\, (x, y) \in [0, 14] \times [0, 28]\}. \quad (7.155)$$

In order to generate suiting constraints, we define one face in the cuboid's center, parallel to the y-z plane $h_1 = \{0\} \times [0, 28] \times [0, 28]$ for the main fault and a second face for the branch fault $h_2 = [-14, 14] \times [0, 28] \times \{14\}$. Finally we split the boundary faces of the initial domain with the first structure:

$$\left({}^1\Lambda^3, {}^2\Lambda^3\right) = \Lambda^3 // S_1 \quad (7.156)$$

and then both resulting boundary faces with the second structure:

$$\left({}^{11}\Lambda^3, {}^{12}\Lambda^3\right) = {}^1\Lambda^3 // S_2, \qquad \left({}^{21}\Lambda^3, {}^{22}\Lambda^3\right) = {}^2\Lambda^3 // S_2 \quad (7.157)$$

For the resulting four sets of boundary faces:

$${}^{11}\Lambda^3, {}^{12}\Lambda^3, {}^{21}\Lambda^3, {}^{22}\Lambda^3, \quad (7.158)$$

we end up with four corresponding cuboids:

$$Q_{11} = [-14, 0] \times [0, 28] \times [0, 28],\; Q_{21} = [-14, 14] \times [0, 28] \times [0, 28], \\ Q_{12} = [-14, 0] \times [0, 28] \times [0, 28],\; Q_{22} = [-14, 24] \times [0, 28] \times [0, 28]. \quad (7.159)$$

The resulting subdivision defines the transformation for TPV24, as illustrated in Figure 7.11.

CHAPTER

8

Verification and applications of ExaSeis

8.1 Introduction

Before we start with the simulation of earthquake events, we verify our numerical method against a series of community benchmarks. A full verification for kinematic point sources, is given in the work by Duru et al. [42, 43]. In this chapter we summarize the core findings of both papers and extend the verification by benchmarks for dynamic earthquake rupture. First, we check for benchmarks with simple kinematic point sources (Homogeneous half-space: Section 8.2.1) and material layers (Layer over half-space: Section 8.2.2), whether the code is able to resolve seismic waves in inhomogenous material. The (semi-)analytic references for these benchmarks are provided by the *Seismic Modeling Web Interface*[1] [98]. Simultaneously, we verify that the perfectly matched layers scheme from Section 7.3 removes reflections coming from the boundary. In order to do this, we reduce the extent of the simulated domain and compare simulations with PML against simulations with the common absorbing boundary conditions.

After the verification for point sources and homogeneous material parameters, we focus on the verification of our method against dynamic rupture benchmarks. Definitions and numeric references for the benchmarks are provided by the *SCEC Spontaneous Rupture Code Verification Project*[2] [68].

[1] Accessed 18.08.2021: `http://www.sismowine.org/`
[2] Accessed 18.08.2021: `https://strike.scec.org/cvws/`

We use tests with increasing difficulty to verify the mechanisms of our method required for the simulation of real spontaneous dynamic rupture events. Finally, we show that out approach works on dynamically refined meshes and rerun one of the benchmarks.

Having verified the method, we present earthquake events simulated with ExaSeis: In the Zugspitze model in Section 8.4, we analyze the effects of a complex topography on scattering, amplification and deamplification. Finally, we simulate a dynamic rupture event for the Húsavík fault in the north Iceland region in Section 8.5.

8.2 Kinematic point sources

8.2.1 The homogeneous half-Space benchmark

As most basic benchmark we look at the homogeneous half-space problem (HHS1) [30]. The domain of the benchmark is a half-space with free-surface placed at $y = 0$. The homogeneous, isotropic, elastic material is described by parameters

$$\rho = 2.7\,\mathrm{t\,m^{-3}}, \qquad c_p = 6.0\,\mathrm{km\,s^{-1}}, \qquad c_s = 3.464\,\mathrm{km\,s^{-1}}. \tag{8.1}$$

A point source is placed 0.693 km deep at $x_0 = (0, 0.693, 0)$ and initialized with a moment-rate time history (compare to Section 7.2.1):

$$f(t) = M_0 \left(\frac{t}{T^2} \exp\left(-\frac{t}{T} \right) \right). \tag{8.2}$$

The moment tensor of the source is zero everywhere, except the shear component xy, where $M_{xy} = M_0 = 1000.0\,\mathrm{PN\,m}$. M_0 is the moment magnitude of the source and $T = 0.1\,\mathrm{s}$ the point of its highest rate. At the free surface nine receivers are placed to record the evolution of seismic waves (Table 8.1).

In order to avoid significant reflections from the absorbing boundary conditions, the original benchmark is defined within the domain $[-26, 32] \times [0, 34] \times [-26, 32]$ km. As the PML layer allows us to restrict the domain even further, we only have to simulate that part of the domain that contains seismograms and the point source $[-2.287, 14.046] \times [0, 16.333] \times [-2.287, 14.046]$ km, which is less than 3 % of the original domain. The PML parameters we use in the layer are summarized in Table 7.1.

With this benchmark we can test how well our numerical schemes represents body waves (P- and S-waves) in the domain, as well as surface-waves

Table 8.1.: Positions of the receivers for the HHS1 benchmark.

Receiver	1	2	3	4
x [km]	0	0	0	0.490
z [km]	0.636	5.542	10.392	0.490
5	6	7	8	9
3.919	7.348	0.577	4.612	8.647
3.919	7.348	0.384	3.075	5.764

(as Love and Rayleigh-waves). To see the effects of the PML layer we additionally run the simulation in the same reduced domain with absorbing boundary conditions. In Figure 8.1 we show the seismograms at the 7th, 8th and 9th receiver for an order 7 scheme with absorbing boundary conditions (ADER-7-ABC) and with PML at the boundaries of the domain (ADER-7-PML) and compare against the reference. Each dimension is resolved by 25 elements, leading to a resolution of $\Delta x = 1.04\,\mathrm{km}$ and a sub-cell resolution of $\Delta x_c = 0.13\,\mathrm{km}$. Waves are recorded for 5 s. We see that both schemes are able to resolve the seismograms sufficiently. Receivers close to the boundary (8 and 9) show spurious reflections that appear in the solution of ADER-7-ABC, for ADER-7-PML these reflections are damped.

To determine the accuracy level of the method we compute the envelop misfit (EM) and the phase misfit (PM) between the reference and our measured receivers [82]. We skip the detailed introduction to envelop-phase misfits and interpret EM as a measure for the relative difference of the amplitudes and PM for the relative difference of phases for two signals. Codes that can reach EM and PM values of less than 5 % are considered to be of the highest accuracy level [98]. In Table 8.2 we show EM and PM for ADER-7-PML for all nine receivers. We see that in all cases misfit values are below the required limit.

8.2.2 The layer over half-space benchmark

The layer over homgeneous half-space (Loh1) benchmark is an extension of the just presented HHS1 [30]. In Loh1 we model a half-space that is covered by a layer of material with less density and lower P- and S-wave speed:

$$\begin{aligned} y > 1.0: &\quad \rho = 2.6\,\mathrm{t\,m^{-3}}, \quad c_p = 4.0\,\mathrm{km\,s^{-1}}, \quad c_s = 2.0\,\mathrm{km\,s^{-1}} \\ y < 1.0: &\quad \rho = 2.7\,\mathrm{t\,m^{-3}}, \quad c_p = 6.0\,\mathrm{km\,s^{-1}}, \quad c_s = 3.464\,\mathrm{km\,s^{-1}}. \end{aligned} \tag{8.3}$$

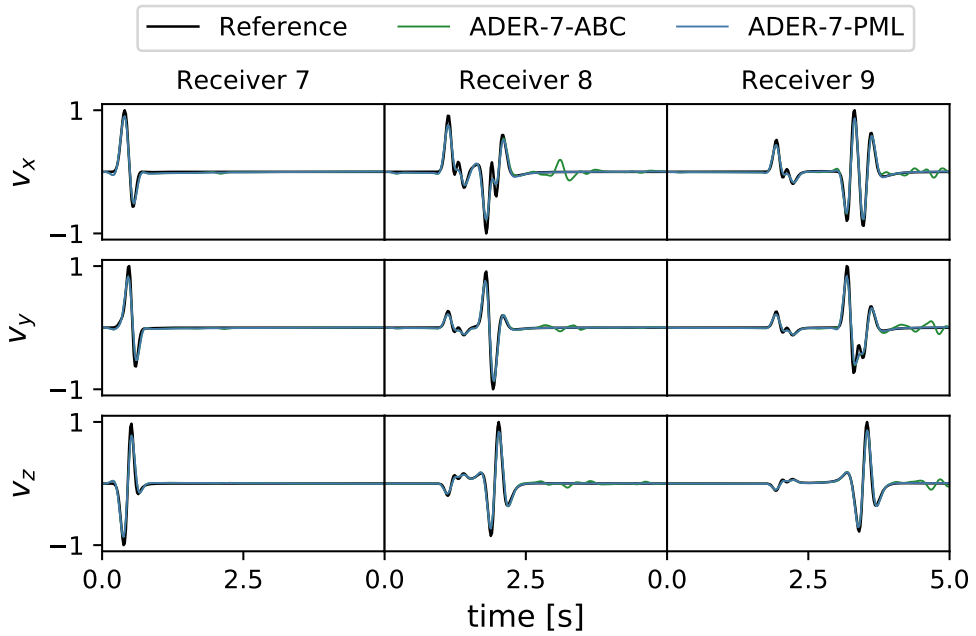


Figure 8.1.: Comparison of semi-analytic and numerical results for the velocity components at the 7th 8th and 9th receiver for the HHS1 benchmark. Numerical results are computed with the ADER-DG method of order 7, with absorbing boundary condition (ADER-7-ABC) and perfectly matched layers (ADER-7-PML).

Table 8.2.: Envelope (EM) and Phase misfit (PM) analysis for the numerical solution of the ADER-7-PML method for the HHS1 benchmark at all nine receivers. All misfits are below the required limit of 5%.

Receiver	1	2	3	4	5	6	7	8	9
EM(u) [%]	2.56	1.16	1.98	1.02	1.39	1.30	1.38	0.73	1.93
PM(u) [%]	0.95	0.61	0.69	0.43	0.43	0.37	0.46	0.35	0.63
EM(v) [%]	0.05	0.02	0.40	1.02	1.39	1.30	1.47	1.58	1.79
PM(v) [%]	0.00	0.00	0.00	0.43	0.43	0.37	0.56	0.64	0.54
EM(w) [%]	0.05	0.00	0.08	2.39	2.44	2.74	2.13	1.71	2.72
PM(w) [%]	0.00	0.00	0.00	0.66	0.69	0.76	0.65	0.49	0.76

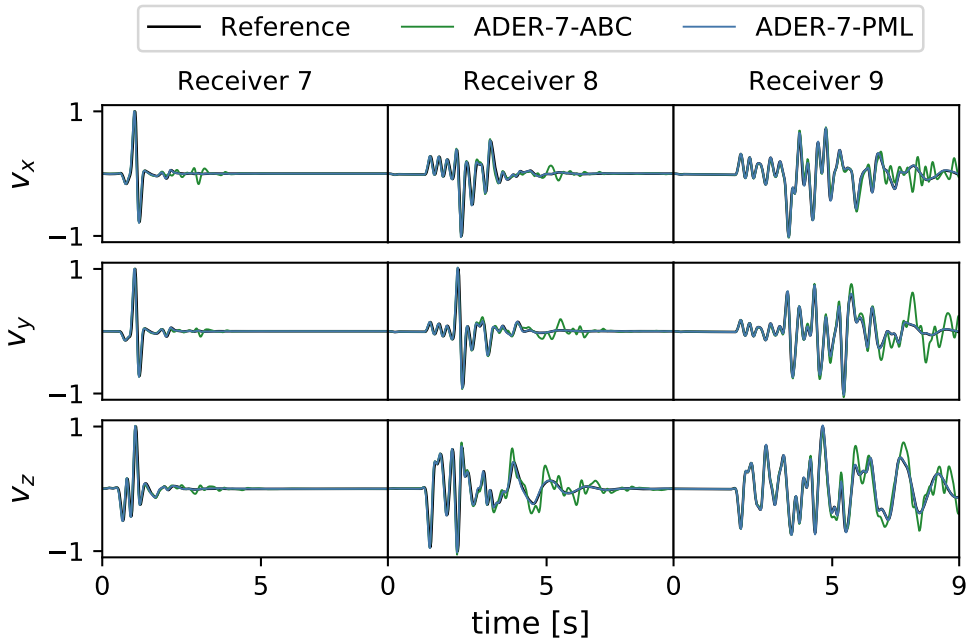


Figure 8.2.: Velocity components at the 7th, 8th and 9th receiver for the Loh1 benchmark. The results are generated with ADER-DG methods of order 7, with absorbing boundary conditions and perfectly matched layers.

The point source has the moment-rate history from Equation (8.2) and is located 2 km deep at $x_0 = (0.0, 2.0, 0.0)$. Again nine recievers are placed at the surface positions according to Table 8.1. Additionaly to the waves that are part of the solution of HHS1, the sharp interface between the two layers introduces interface and guided waves.

We again restrict the simulated domain to $[-2.287, 14.046] \times [0, 16.333] \times [-2.287, 14.046]$ km and use a mesh of 25 elements in each direction. In order to represent the sharp material interface with the geometry of the mesh, we use the curvilinear mesher (See Section 7.6), to place a sharp element interface at $y = 1$.

Figure 8.2 shows the comparison of ADER-7-PML against ADER-7-ABC and the analytic reference. As for HHS both schemes can represent the main characteristic waves sufficiently. For ADER-7-ABC receivers close to the boundary are significantly disturbed by the reflections. In all cases EM and PM (Table 8.3) are below the required limit for ADER-7-PML.

Table 8.3.: Envelope (EM) and Phase misfit (PM) analysis for the numerical solution of the ADER-7-PML method for the LOH1 benchmark at all nine receivers. All misfits are below the required limit of 5%.

Receiver	1	2	3	4	5	6	7	8	9
EM(u) [%]	1.51	1.77	2.23	1.39	1.72	0.95	0.83	1.56	1.47
PM(u) [%]	3.93	2.88	2.59	3.91	3.14	1.70	2.34	2.75	3.11
EM(v) [%]	0.29	0.54	2.58	1.39	1.72	0.95	1.41	1.66	1.89
PM(v) [%]	0.00	0.00	0.00	3.91	3.14	1.70	3.91	3.03	3.82
EM(w) [%]	0.55	0.98	2.65	0.78	0.98	1.28	0.60	0.75	1.44
PM(w) [%]	0.00	0.00	0.00	1.51	1.98	2.11	1.13	1.55	2.85

8.3 Dynamic rupture

After we verified our code for benchmarks with simple point sources, we next compare our implementation to dynamic rupture problems from the SCEC Spontaneous Rupture Code Verification Project [67, 68]. The project provides a set of benchmarks for dynamic rupture simulations, with incremental difficulty and publishes a collection of numeric results from various codes.

The provided sets of results consist of off-fault seismograms placed in the interior of the domain and on its surface, on-fault seismograms placed directly on the fault and the contour of the rupture-process. In off-fault seismograms we track all three components of the particle velocity (v_x,v_y,v_z). On-fault seismograms record the slip-rate, slip and shear stresses in horizontal and vertical direction. The rupture contour records the first time the absolution slip rate on the fault exceeds $1\,\mathrm{mm\,s^{-1}}$ and describes the temporal evolution of the slip. We compare our results to uploaded numerical results generated with the high order finite difference code WaveQLab [40] and the low order finite element code FaultMod [9].

In all benchmarks we simulate vertical strike-slip faults with homogeneous material:

$$\rho = 2.670\,\mathrm{t\,m^{-3}}, \qquad c_p = 6.0\,\mathrm{km\,s^{-1}}, \qquad c_s = 3.464\,\mathrm{km\,s^{-1}}. \tag{8.4}$$

At the borders of the domain reflections are damped with perfectly matched layers (for which parameters are presented in Table 7.1), at $y = 0$ a free surface boundary condition is set.

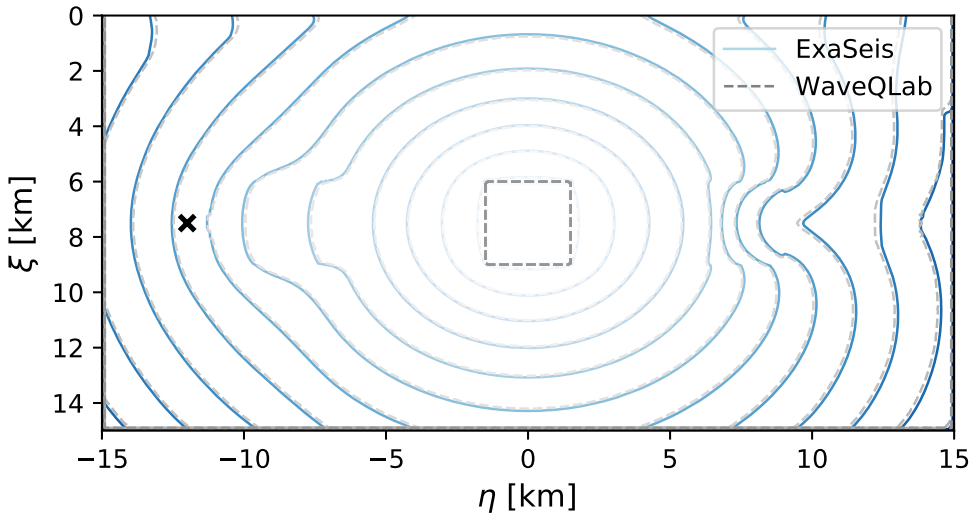


Figure 8.3.: Contour plot for the TPV5 benchmark. Contours are placed at $t = 0, 0.5, ..., 5.5\,\text{s}$. The station from Figure 8.4 is marked.

8.3.1 TPV5: A plain fault

The first benchmark is the TPV5 benchmark, which models spontaneous rupture in a homogeneous halfspace. We consider the domain $\Omega = [-19.75, 19.75] \times [0, 39.5] \times [-19.75, 19.75]$ km, where the rupture happens on the planar fault $(x, y, z) \in \{0\} \times [0, 15] \times [-15, 15]$ km. The fault strength is defined with zero cohesion and linear friction parameters $\mu_s = 0.667$, $\mu_d = 0.525$ and $D_c = 0.4$. Initially stresses are set zero everywhere, except $\sigma_{xx} = -120$ and $\sigma_{xz} = 70$ MPa. The rupture is initiated by a 3 km wide nucleation patch centered around $(\xi_0, \eta_0) = (7.5, 0.0)$, where the initial shear traction τ is set higher than the fault strength τ_s. Here, we introduced the coordinates on the fault ξ and η, corresponding to the horizontal and vertical direction. To the left at $(\xi, \eta) = (7.5, -7.5)$ and to the right at $(\xi, \eta) = (7.5, 7.5)$ of the nucleation patch, two additional patches with same extent and reduced and increased initial stress are placed. The resulting initial shear stress yz is described by

$$\sigma_{yz} = \begin{cases} 81.6, & \text{if } (\xi, \eta) \in [6.0, 9.0] \times [-1.5, 1.5] \\ 78.0, & \text{if } (\xi, \eta) \in [6.0, 9.0] \times [-9.0, -6.0] \\ 62.0, & \text{if } (\xi, \eta) \in [6.0, 9.0] \times [6.0, 9.0] \\ 70.0, & \text{else.} \end{cases} \tag{8.5}$$

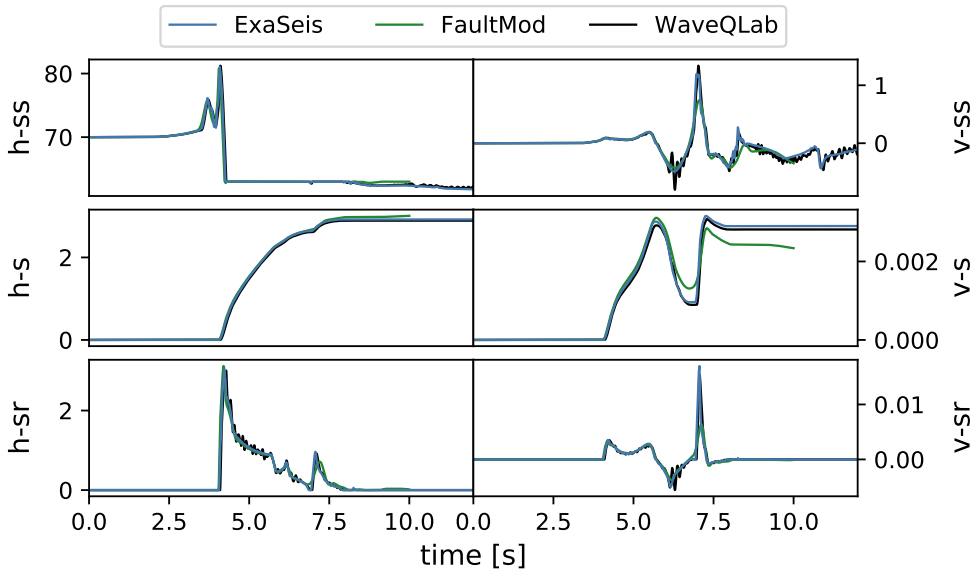


Figure 8.4.: TPV5: On-fault station at $(\eta, \xi) = (-12.0, 7.5)$ km. We show the shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction.

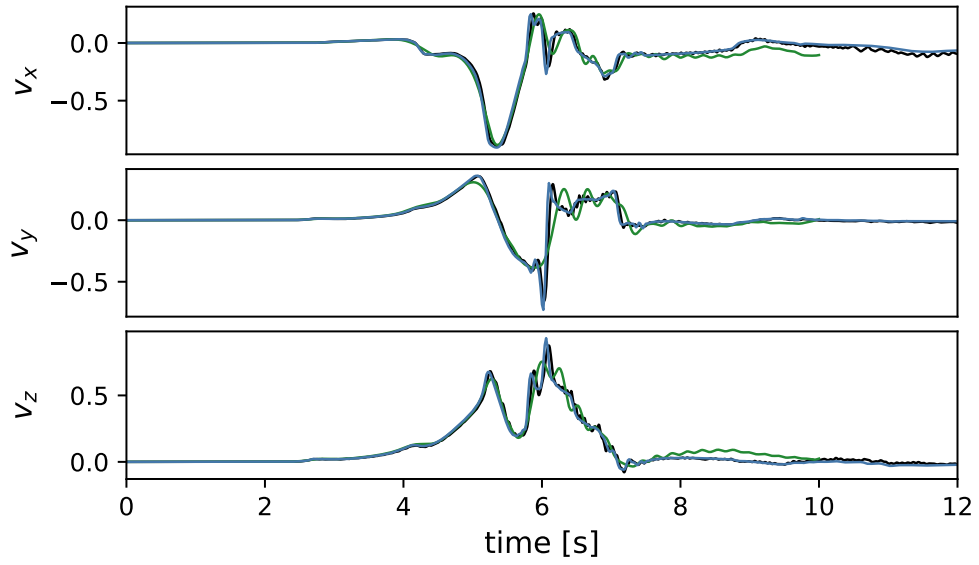


Figure 8.5.: TPV5: Off-fault station at the surface at $(x, y, z) = (3.0, 0.0, -12.0)$ km. We show the three components of the velocity vector (v_x, v_y, v_z).

This benchmark is the simplest case we consider and allows us to verify our implementation of the rupture process in its most basic form.

We run the simulation with polynomial order 7 and on a mesh of 79 elements in all dimensions, such that the domain is resolved with a cell resolution of $\Delta x = 0.5\,\text{km}$ and a sub-cell resolution of $\Delta x_s = 65.5\,\text{m}$. With this set-up, computational mesh and physical mesh are identical as the fault is identical to an element interface of the computational mesh.

In Figure 8.3, we present a comparison of the rupture contour for our simulation against the WaveQLab result. We see that the contours of both codes match perfectly. A comparison for on-fault and off-fault stations is given in Figure 8.4 and Figure 8.5. We see a perfect match between ExaSeis and WaveQLab, while Fault-Mod shows very small differences to the other two codes for the on-fault station.

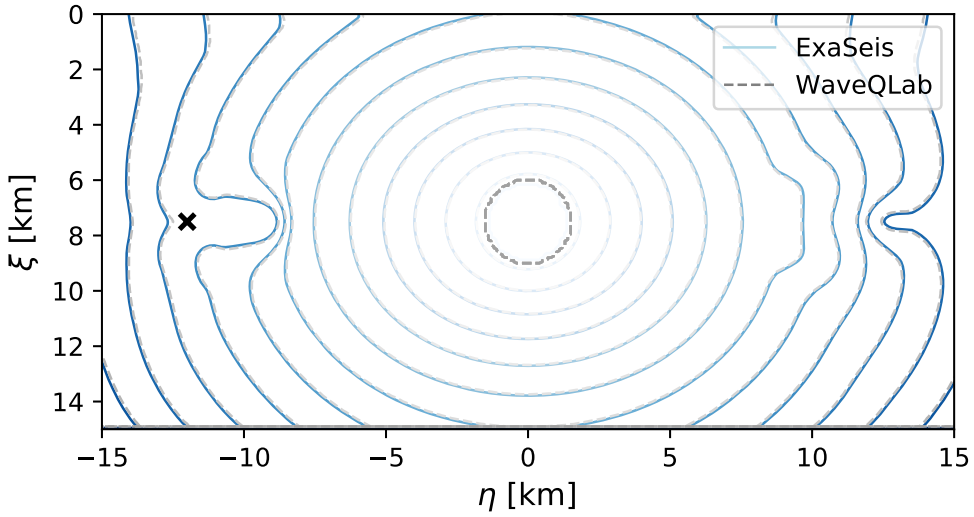


Figure 8.6.: Contour plot for the TPV28 benchmark. Contour lines are placed at $t = 0, 0.5, ..., 15$ s. The fault station from Figure 8.7 is marked.

8.3.2 TPV28: A structured fault

To test if we can resolve geometrically complex structures on a fault, we next look at the TPV28 benchmark. In TPV28 two hills with 10.5 km horizontal distance to the hypocenter are added to the planar fault. The structure of the fault is defined by

$$\begin{aligned} r_1 &= \|(\xi,\eta) - (-10.5, 7.5)\|_2 \\ r_2 &= \|(\xi,\eta) - (\ \ 10.5, 7.5)\|_2 \\ t(\xi,\eta) &= \begin{cases} -0.3\left(1+\cos\left(\pi r_1\right)\right), & \text{if } r_1 < 3.0 \\ -0.3\left(1+\cos\left(\pi r_2\right)\right), & \text{if } r_2 < 3.0 \\ 0, & \text{else,} \end{cases} \end{aligned} \tag{8.6}$$

which we can model with our meshing approach.

Initial stresses are zero everywhere except $\sigma_{xx} = -60\,\text{MPa}$, $\sigma_{zz} = -60\,\text{MPa}$ and $\sigma_{xz} = 29.38\,\text{MPa}$. As in TPV5 the rupture is initialized in the nucleation point $(\xi_0, \eta_0) = (7.5, 0)$ by a nucleation patch,

$$\begin{aligned} r_n =& \|(\xi,\eta) - (7.5, 0.0)\|_2 \\ \sigma_{yz} =& \begin{cases} 11.60, & \text{if } r <_n 1.4 \\ 5.8\left(1+\cos\left(\pi\dfrac{(r_n - 1.4)}{0.6}\right)\right), & \text{if } 1.4 < r_n < 2.0 \\ 0.0 & \text{else.} \end{cases} \end{aligned} \tag{8.7}$$

The parameters for linear slip weakening are set to $\mu_s = 0.677$, $\mu_d = 0.373$, $S_c = 0.4\,\text{m}$ and zero cohesion.

Figure 8.6 shows a comparison of the rupture contour for TPV28 between ExaSeis and WaveQLab. We again see that both codes match perfectly. Around the centers of the two hills ($(7.5, -10.5)$ and $(7.5, 10.5)$), we can clearly see their effect on the rupture. For the on-fault and off-fault stations in Figure 8.7 and Figure 8.8, we also get a perfect match between ExaSeis and the two reference codes.

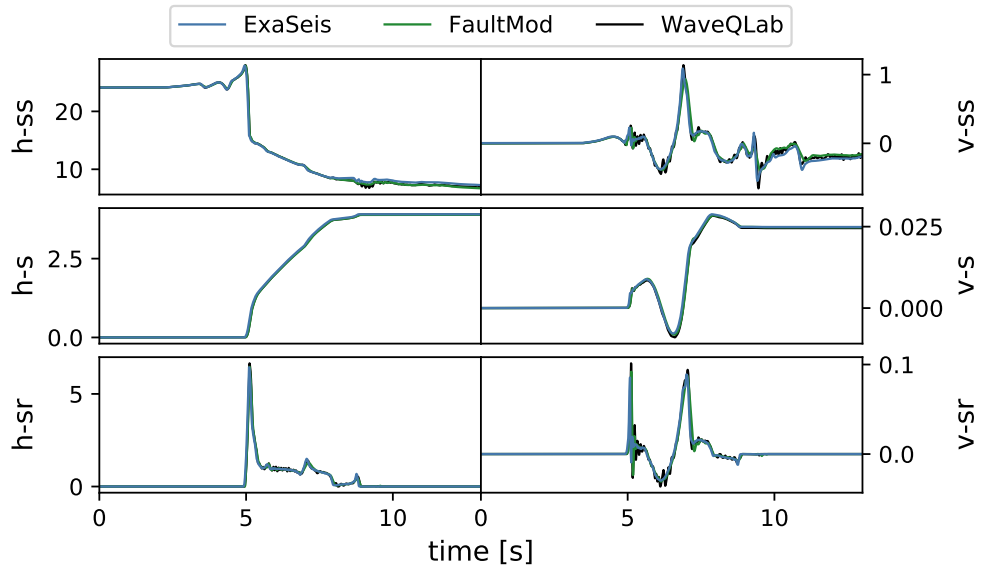


Figure 8.7.: TPV28: On-fault station at $(\xi, \eta) = (-12.0, 7.5)$ km. We present the shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction.

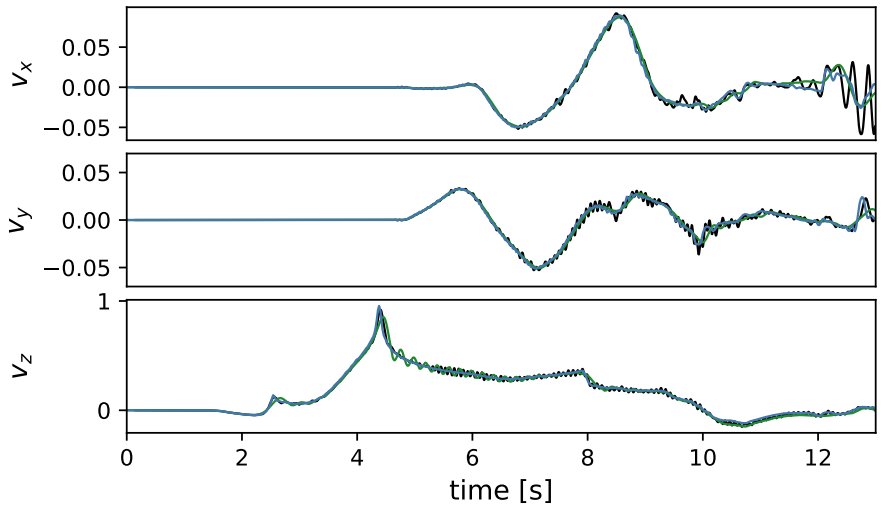


Figure 8.8.: TPV28: Off-fault station at surface at $(x, y, z) = (3.0, 0.0, 15.0)$ km. We show the three components of the velocity vector (u,v,w).

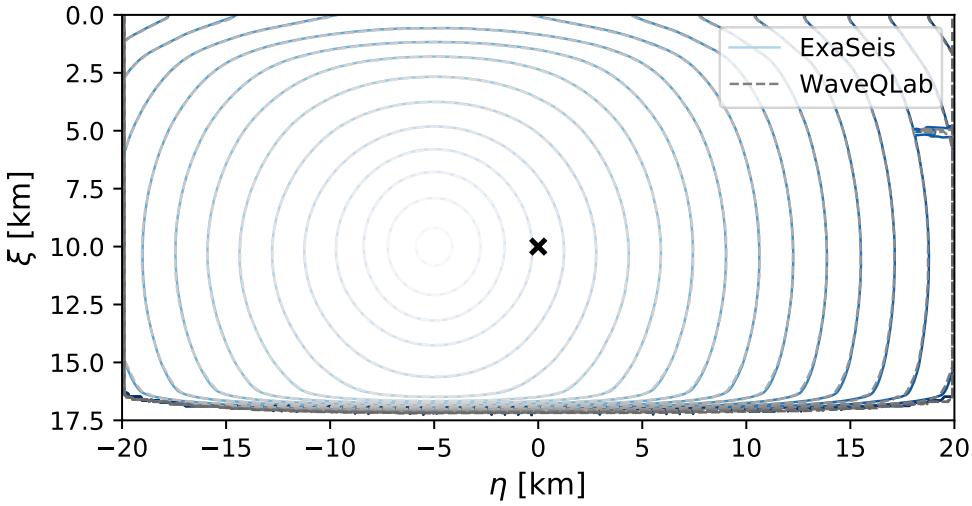


Figure 8.9.: Contour plot for the TPV26 benchmark. Contour lines are placed at $t = 0, 0.5, ..., 15$ s. The fault stations from Figure 8.10 is marked.

8.3.3 TPV26: Forcing rupture

Up to this point we initialized the rupture artificially within a predefined nucleation patch, where the initial shear traction exceeds the fault strength. This approach leads to a sudden initialization of slip within the nucleation patch. In order to induce the rupture smoothly a second approach is introduced for TPV26: Instead of setting the shear traction to exceed the fault-strength, the fault-strength is gradually weakened in a circle around the hypocenter (ξ_0, η_0) with radius r_{crit}, until it falls below the initial shear traction [67]. We implement this approach by adding the parameter $f_T(\xi, \eta, t) \in [0, 1]$ to the linear slip weakening law from Equation (7.38),

$$f(\psi) = \mu_s - (\mu_s - \mu_d) \max\left(f_T(\xi, \eta), \frac{\min(\psi, D_c)}{D_c}\right). \tag{8.8}$$

The parameter linearly increases from 0 to 1 in a time-frame of fixed size t_0. Within the time-frame the length of the slip path is forced to the critical distance D_c. The starting point $T_f(r)$ of the time-frame depends on the distance from the hypocenter $r(\xi, \eta) = \|(\xi, \eta) - (\xi_0, \eta_0)\|_2$,

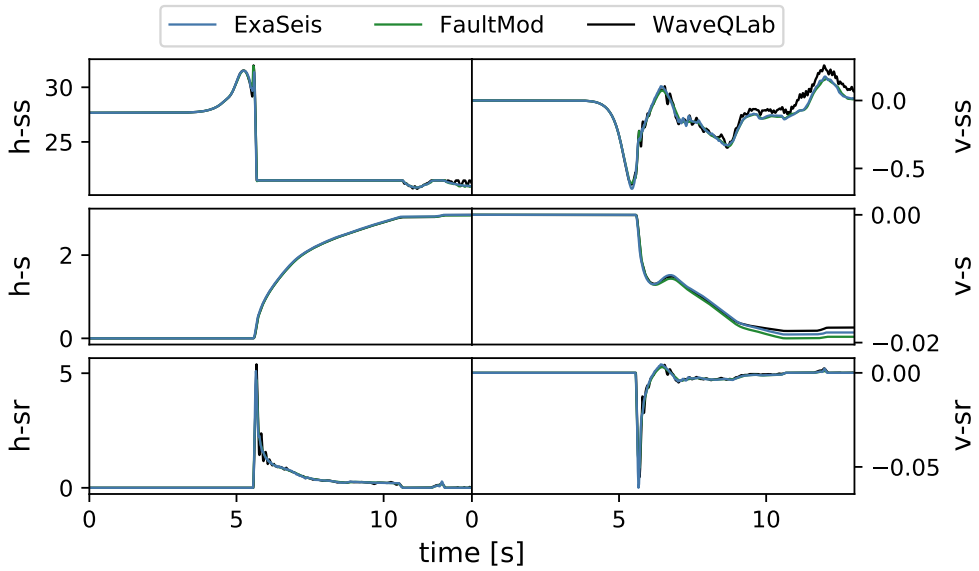


Figure 8.10.: TPV26: On-fault station at $(\xi, \eta) = (10.0, 10.0)$ km. We present the shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction.

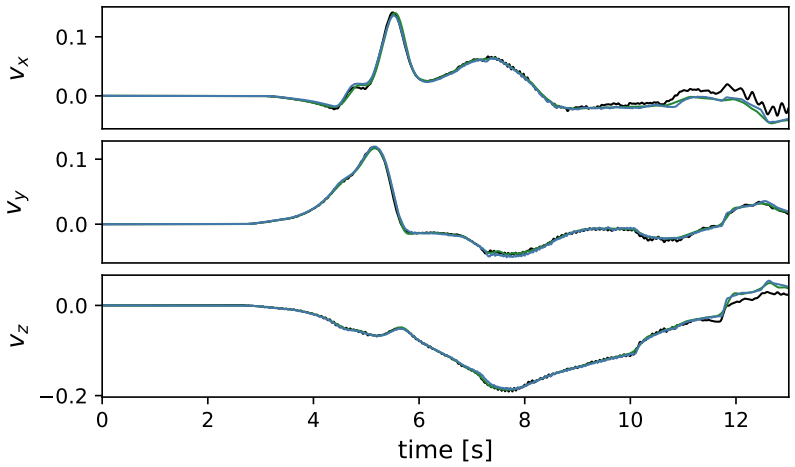


Figure 8.11.: TPV26: Off-fault station at surface at $(x, y, z) = (-3.0, 0.0, 5.0)$ km. We show the three components of the velocity vector (u,v,w).

$$f_T(\xi, \eta, t) = \begin{cases} 0, & \text{for } t < T_f(r) \\ (t - T_f(r))/t_0, & \text{for } T_f(r) \le t \le T_f(r) + t_0 \\ 1, & \text{for } T_f(r) + t_0 \le t. \end{cases} \tag{8.9}$$

The starting point $T_f(r)$ is chosen such that the rupture expands with variable velocity $\partial F(r)/\partial r$,

$$T_f(r) = \begin{cases} \dfrac{r}{0.7V_s} + \dfrac{0.081 r_{crit}}{0.7V_s}\left(\dfrac{1}{1-(r/r_{crit})^2} - 1\right) & \text{, for } r < r_{crit} \\ 1.0 \times 10^9 \text{ else.} \end{cases} \tag{8.10}$$

At the hypocenter $r = 0$ the rupture expands with $0.7V_s$ at r_{crit} the velocity is zero. For TPV26 the parameters for the forced nucleation are set to

$$\begin{aligned} V_s &= 3.464 \text{ km s}^{-1}, & t_0 &= 0.5 \text{ s}, \\ r_{crit} &= 4.0 \text{ km}, & (\eta_0, \nu_0) &= (-5, 10) \text{ km}. \end{aligned} \tag{8.11}$$

Additionally, TPV26 introduces depth-variable cohesion and initial stress tensor [67]: In order to avoid interaction with the surface of the domain, cohesion linearly decreases in the top 5 km from 4.0 GPa to 0.4 GPa,

$$c(\eta) = 0.40 + 0.72 \cdot \max\left(5.0 - \eta, 0\right) \text{ GPa}. \tag{8.12}$$

The initial stress tensor is chosen to increase with the depth,

$$\begin{aligned} P_f &= 1.0 \cdot 9.8 \cdot y \\ \hat{\sigma}_{yy} &= -2.670 \cdot 9.8 \cdot y \\ \sigma_{xx} &= \Omega\left(b_{xx}\left(\hat{\sigma}_{yy} + P_f\right) - P_f\right) + \hat{\sigma}_{yy}(1-\Omega) + P_f \\ \sigma_{xz} &= \Omega\left(b_{xz}\left(\hat{\sigma}_{yy} + P_f\right)\right) \\ \sigma_{zz} &= \Omega\left(b_{zz}\left(\hat{\sigma}_{yy} + P_f\right) - P_f\right) + \hat{\sigma}_{yy}(1-\Omega) + P_f \\ \sigma_{yy} &= \hat{\sigma}_{yy} + P_f \qquad \sigma_{xy} = 0.0 \qquad \sigma_{yz} = 0.0. \end{aligned} \tag{8.13}$$

Where the parameters for TPV26 are set to

$$\Omega = \begin{cases} 1.0 & \text{, for } y \le 15.0 \\ 0.2 \cdot (20.0 - y) & \text{, for } y \le 20.0 \\ 0.0 & \text{else} \end{cases} \tag{8.14}$$

$$b_{xx} = 1.073206, \qquad b_{xz} = -0.169029, \qquad b_{zz} = 0.926793.$$

We simulate the problem in the domain $\Omega = [-25.675, 25.675] \times [0, 51.35] \times [-25.675, 25.675]$ km with an order 7 method, which leads to a resolution of

$\Delta x = 650\,\text{m}$ sub-cell resolution of $\Delta x_c = 81.25\,\text{m}$. We again get a perfectly matching rupture contour between ExaSeis and the reference WaveQLab in Figure 8.9. The stations of all three references also match perfectly in Figure 8.10 and Figure 8.11.

8.3.4 TPV29: A rough fault

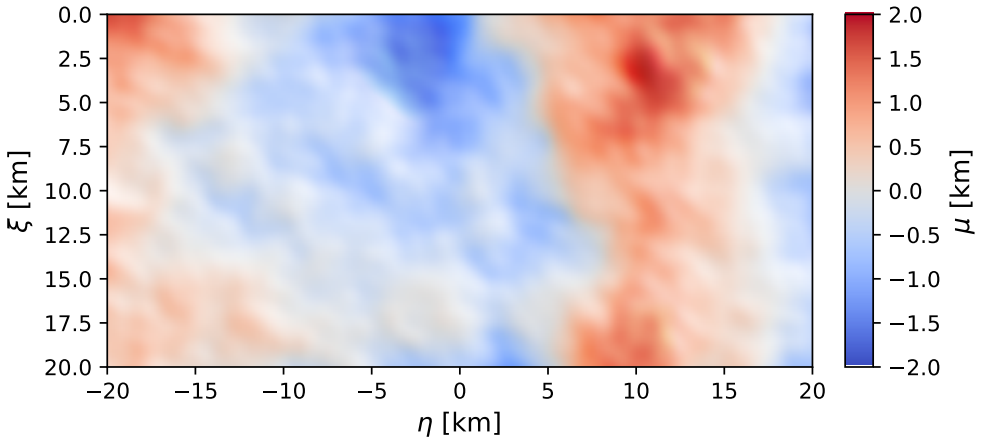
The last benchmark we verify our code against is TPV29. TPV29 is defined similar to TPV26, but introduces a rough fault based on a randomly generated data set (see Figure 8.12a). For TPV29 the mechanisms to initialize the rupture are adapted from TPV26, with a changed set of parameters: Cohesion is decreased in the upper 4 km layer from 1.2 GPa to 0.4 GPa,

$$c(y) = 0.40 + 0.2 \cdot \max\left(4.0 - y, 0\right) \ \text{GPa}. \tag{8.15}$$

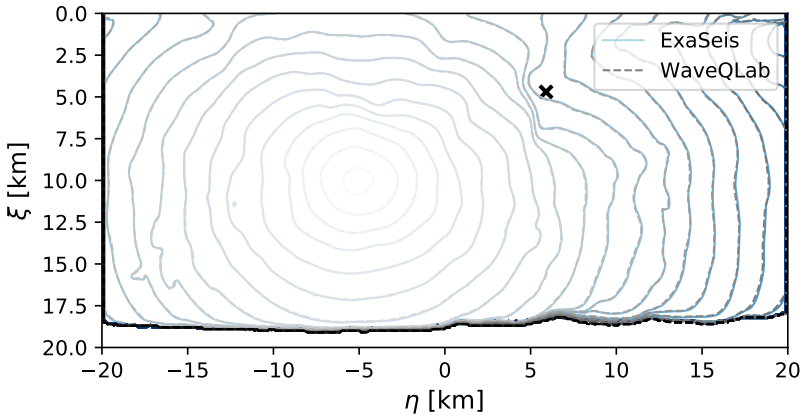
The parameters for the initial stresses are also adjusted,

$$\Omega = \begin{cases} 1.0 & \text{, for } y \leq 17.0 \\ 0.2 \cdot (22.0 - y) & \text{, for } y \leq 22.0 \\ 0.0 & \text{else} \end{cases} \tag{8.16}$$

$$b_{xx} = 0.974162, \qquad b_{xz} = -0.158649, \qquad b_{zz} = 1.025837.$$



(a) Fault structure of the TPV29 benchmark.



(b) Contour plot for the TPV29 benchmark

Figure 8.12.: Fault structure and rupture contour plot for TPV29. The contour lines are placed at $t = 0, 0.5, ..., 15$ s. The fault station from Figure 8.13 is marked.

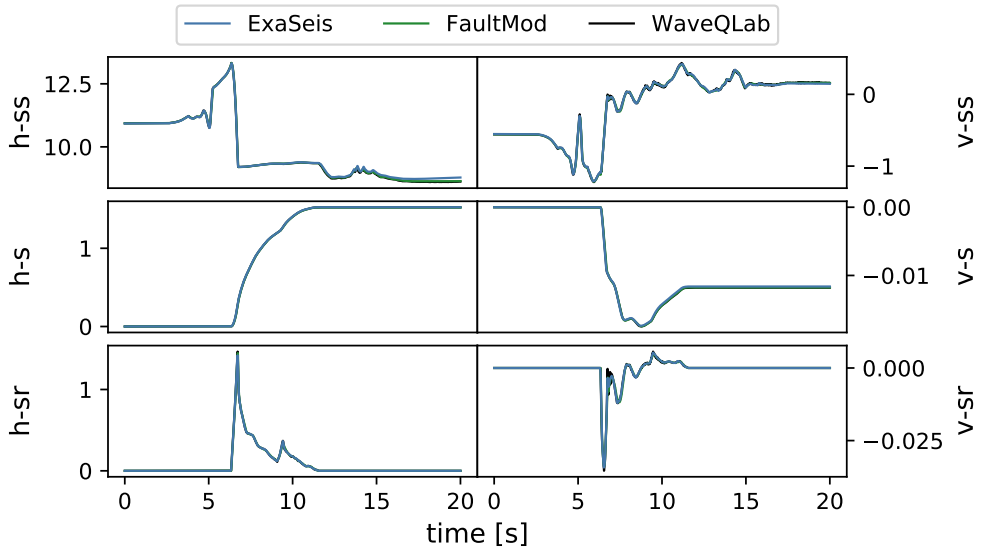


Figure 8.13.: TPV29: On-fault station at $(\xi, \eta) = (5.9, 4.7)$ km. We present the shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction.

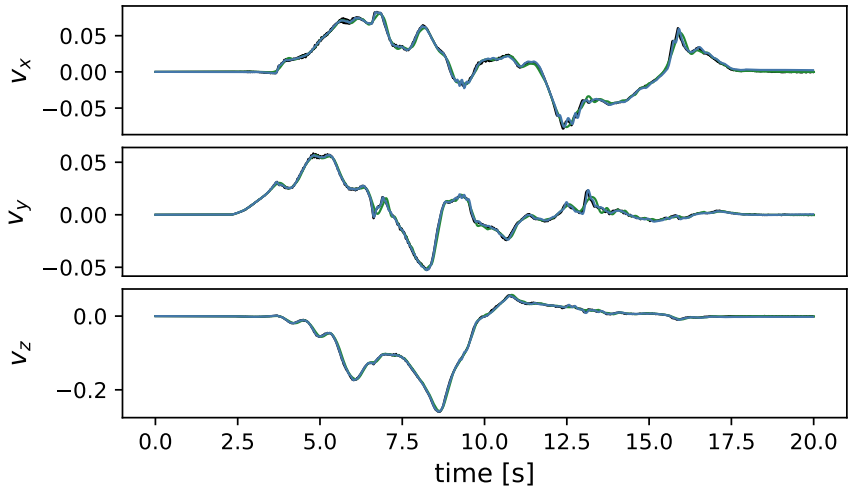


Figure 8.14.: TPV29: Off-fault station at surface at $(x, y, z) = (2.0, 0.0, 0.0)$ km. We show the three components of the velocity vector (u,v,w).

We simulate the problem in the same domain as TPV26:
$\Omega = [-25.675, 25.675] \times [0, 51.35] \times [-25.675, 25.675]$ km and again use an order 7 method.

In Figure 8.12b, we can see the significant effect of the rough fault on the propagation of the rupture front. Again ExaSeis and WaveQLab show perfectly matching results. The comparison of on-fault and off-fault stations in Figure 8.13 and Figure 8.14 shows a perfect match.

8.3.5 Dynamic adaptive mesh refinement

Finally, we want to verify our code for dynamic rupture simulations on dynamically refinement meshes. As example we choose the TPV5 benchmark from Section 8.3.1. Following the mesh infrastructure, cells are refined by splitting single elements into patches of $3\times3\times3$ elements [138]. Patches that were refined, can be coarsened back at a later point in time. The solution is projected between refinement levels with L2-projections [26]. Refinement and coarsening of cells is determined by a refinement criterion. In order to track the rupture front, we provide a criterion based on the maximal norm

of the velocity vector in all nodes i within a cell,

$$\|v\|_2 = \max_i \sqrt{(v_x)_i^2 + (v_y)_i^2 + (v_z)_i^2}. \tag{8.17}$$

Cells that contain a maximal velocity norm above v_{refine} should be refined, below $v_{coarsen}$ coarsened,

$$\text{Criterion}(\|v\|_2) = \begin{cases} \text{Refine}, & \text{if } \|v\|_2 > v_{refine} \\ \text{Coarsen}, & \text{if } \|v\|_2 < v_{coarsen} \\ \text{Keep}, & \text{else.} \end{cases} \tag{8.18}$$

In theory the mesh infrastructure of the ExaHyPE-Engine provides arbitrary levels of refinement and coarsening. At the time of this work several issues in the implementation and with the performance of the engine, only allowed us to run the simulation with one level of refinement, without coarsening and only with an ADER-DG method of order up to 5 until an end-time of $T = 8.5\,\text{s}$. In our test we set the refinement velocity to $v_{refine} = 0.5\,\text{m}\,\text{s}^{-1}$ and deactivate coarsening by setting $v_{coarsen} < 0.0\,\text{m}\,\text{s}^{-1}$. We use an initial mesh of 25 elements in each dimension such that the resolution is $\Delta x = 0.5\,\text{km}$ and the sub-cell resolution $\Delta x_s = 0.083\,\text{km}$. For refined cells this implies a resolution of $\Delta x_r = 0.166\,\text{km}$ and the sub-cell resolution $\Delta x_{rs} = 0.027\,\text{km}$. To highly resolve the nucleation we initially refine a patch of $3 \times 3 \times 3$ elements around the hypocenter.

In Figure 8.16 we show an illustration of the evolution of the mesh and the velocity norm. The mesh tracks the wavefield velocity, after 5 s most parts of the domain are fully refined. In Figure 8.17, we compare the refined mesh on the fault against the measured slip rate. At $t = 0.0\,\text{s}$, we see the initial patch defined around the hypocenter. For later time-steps at $t = 1.0\,\text{s}, 2.0\,\text{s}$ and $4.0\,\text{s}$ the rupture-front is fully resolved by the refined mesh. In order to test if the rupture is tracked sufficiently, we compare the on-fault station from Section 8.3.1 in Figure 8.15 for a simulation with dynamic AMR against the solution we generated on a uniform mesh. We see all components match between the simulation on a uniform mesh and with dynamic AMR, small differences in the vertical components are negligible.

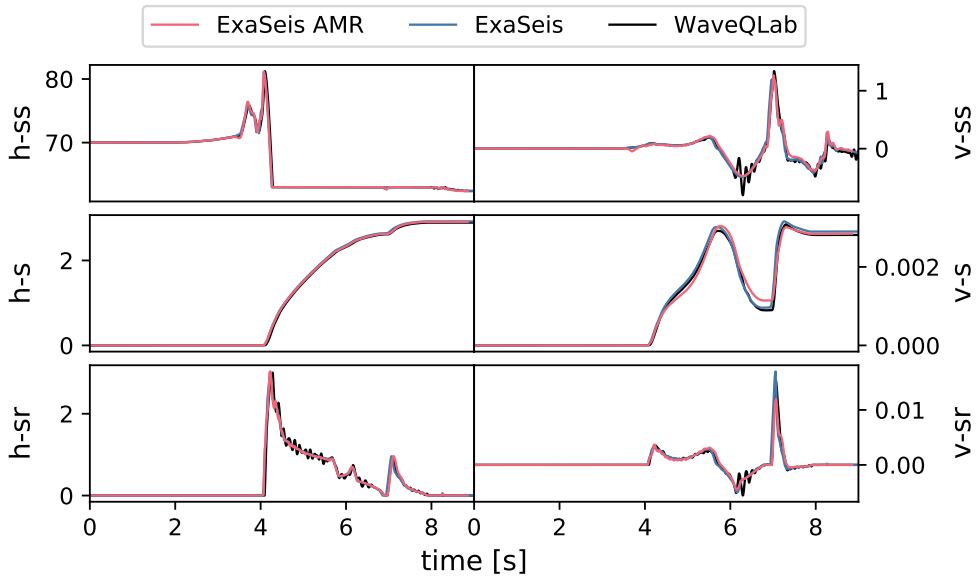


Figure 8.15.: Comparison of a simulation with dynamic AMR against a simulation on a uniform mesh, for the on-fault station at $(\eta, \xi) = (-12.0, 7.5)$ km.

8.4 The Zugspitze scenario

The first application of our method is the Zugspitze model. In order to quantify the influence of a complex topography on seismic waves, we created a benchmark based on a single kinematic point source in the northern region of the Alps [43, 89]. The domain is chosen in an 80×80 km square, centering the Zugspitze in the Wettersteingebirge. The topography of the area shows a high variability and large deviations from a planar surface (See Figure 8.19). From a complex surface we expect amplification, deamplification scattering and channeling effects on the ground motion [10, 56].

The moment-rate time history of the point source, to initialize the earthquake is adapted from Equation (8.2) and placed 10 km deep at $x_0 = (10.0, 10.0, 10.0)$ km. In order to focus on the effects of the topography, we pick homogeneous material parameters,

$$\rho = 2.670\,\mathrm{t\,m^{-3}}, \qquad c_p = 6.0\,\mathrm{km\,s^{-1}}, \qquad c_s = 3464\,\mathrm{km\,s^{-1}}. \tag{8.19}$$

We use the scenario to analyze two main aspects: We want to test how well the curvilinear mesh can represent topographies and if the meshing ap-

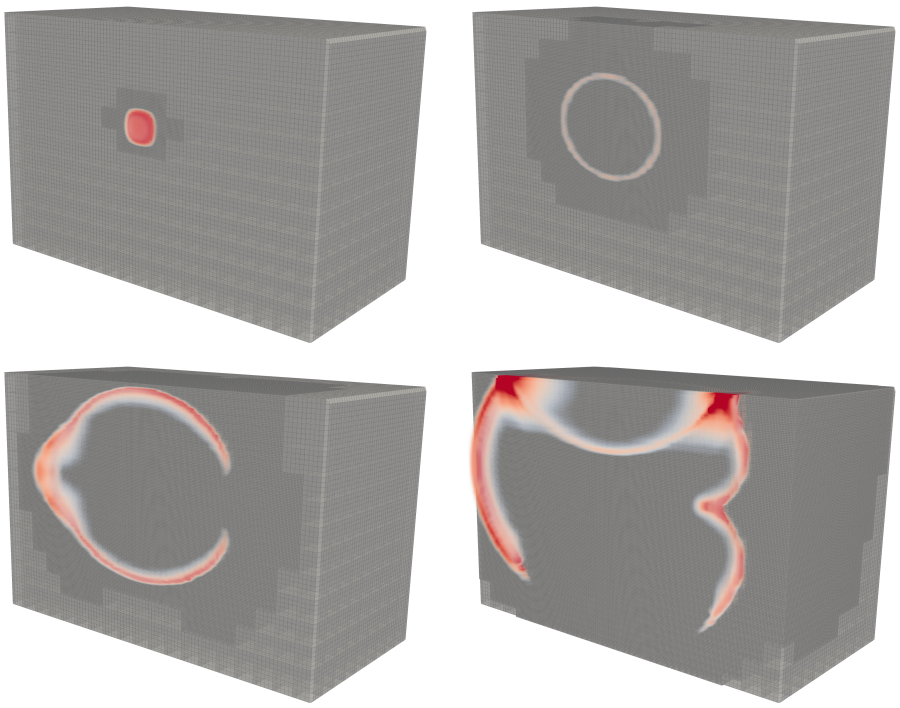


Figure 8.16.: Illustration of the wavefield and dynamically refined mesh for TPV5 at t=0.5 s, t=2.0 s, t=3.5 s and t=5.0 s.

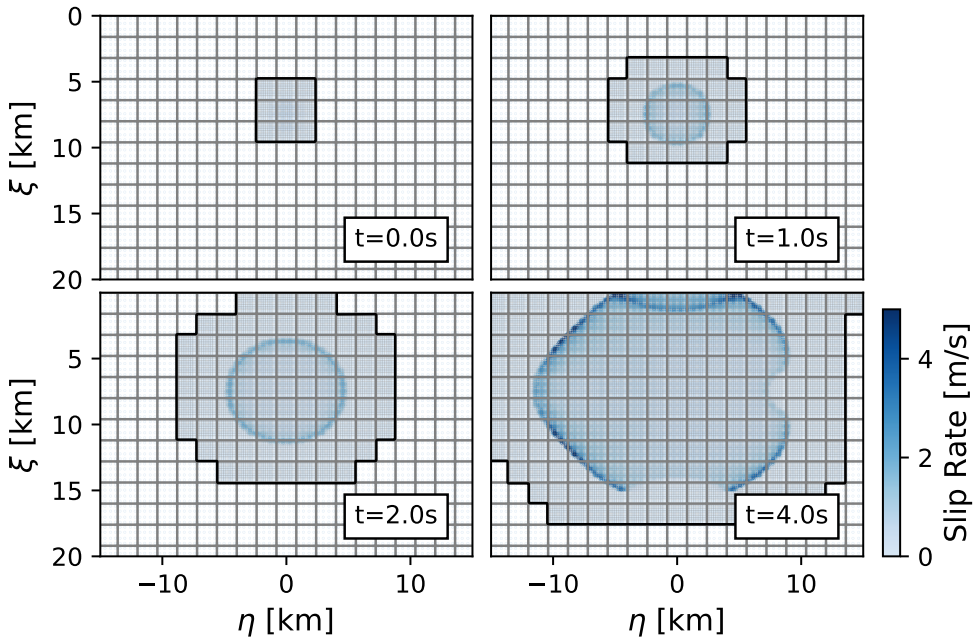


Figure 8.17.: Slip rate and mesh for the TPV5 benchmark with dynamically refined meshes.

proach from Section 7.6 gives us valid representations for complex surfaces. Second, we analyze the effects of the complex topography on characteristic numbers of the resulting earthquake.

The model is simulated on a mesh of 241 cells in each dimension with an order 3 ADER-DG method and a 5 km wide PML-layer around the domain, resulting in a resolution of $\Delta x = 0.4$ km and s sub-cell resolution of $\Delta x_c = 0.1$ km. Wavepropagation is simulated for 30 s. Receivers are placed on the surface equidistantly along the diagonal $(x_i, z_i) = (i + 1) \cdot (10, 10)$, for $i = 1, ..., 4$. The topography of the scenario and the resulting wavefield is illustrated in Figure 8.18 after 5, 10, 15, and 20 s.

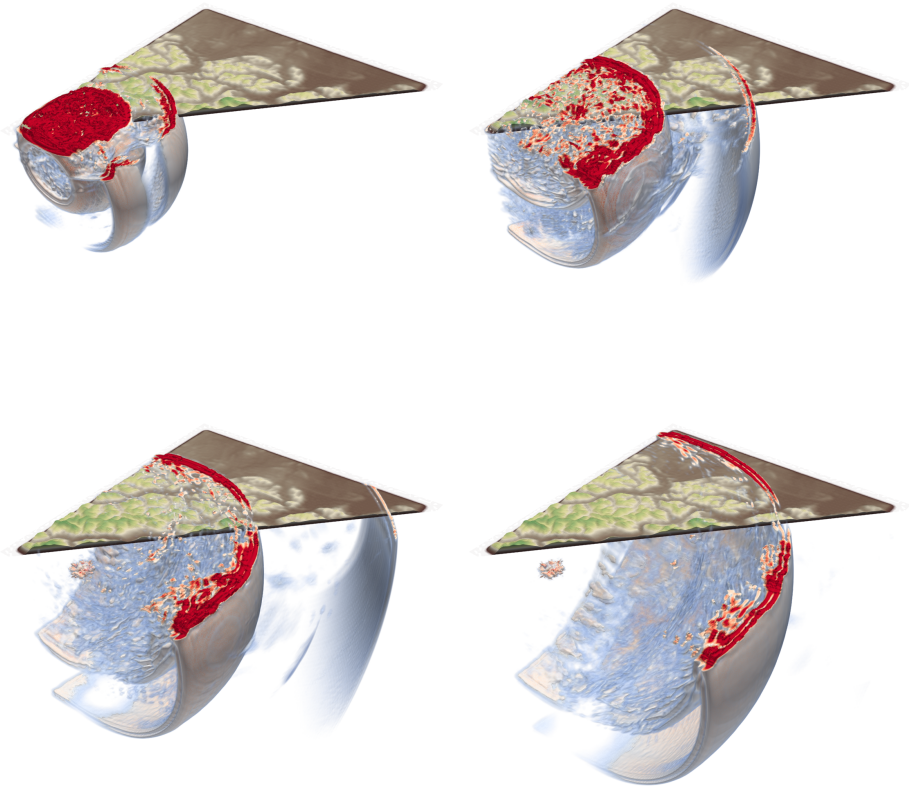


Figure 8.18.: 3D illustration of the absolute velocity of the propagating seismic wavefield for the Zugspitze model at t=5 s, t=10 s, t=15 s and t=20 s simulated with ExaSeis. Adapted from Figure 10 in [43].

8.4.1 Topography filtering

In order to quantify what effects the complexity of the topography has on several aspects of our model, we filtered the wave number content of the original Zugspitze topography. Our filtering approach uses the two dimensional discrete Fourier transformation of a representation of the topography on a uniform grid with a mesh size of $\Delta x = \Delta y = 25\,\mathrm{m}$. We get the coefficients $\mathcal{T}_{ij}$ to the wave numbers ${}^1\omega_i, {}^2\omega_i = i \cdot 80\,\mathrm{km}^{-1}, i \in [0, 1600]$. To reduce the complexity, we zero out those coefficients for which the Euclidean wave length $\omega^r_{ij} = \sqrt{({}^1\omega_i)^2 + ({}^2\omega_j)^2}$ is less than a predefined cut-off wave number ω^t

$$\overline{\mathcal{T}}_{ij} = \begin{cases} \mathcal{T}_{ij}, & \text{if } \omega^r_{ij} < \omega^t \\ 0, & \text{else.} \end{cases} \tag{8.20}$$

We look at six topographies, starting at a flat surface for $\omega^t = 0\ \mathrm{km}^{-1}$ and five configurations with exponentially increasing cut-off wave number $\omega^t \in \{4, 8, 16, 32, 64\}\ \mathrm{km}^{-1}$. The resulting topographies are presented in Figure 8.19.

Verification

In order to verify our implementation of the curvilinear mesh, we compare the receivers for all six topographies against high resolved results generated with the finite-difference code WaveQLab [40] and the discontinuous Galerkin code SeisSol [34]. WaveQLab runs on a uniform mesh of 100 m grid spacing, reflections are removed with a PML mechanism, equivalent to the one we use in ExaSeis. For the result in SeisSol the domain is extended to $[-65, 85] \times [-3, 45] \times [-65, 85]$ km, such that reflections from the boundaries are reduced. The results are computed with an order 6 method on a tetrahedral static adaptive mesh of 300 m cell size around the point-source that gradually increases to 3.5 km at the outer layers of the domain.

In Figure 8.20 we compare the components of the velocity vector in all four stations placed on the surface, for the topography with the highest cut-off frequency $\omega^t = 64\ \mathrm{km}^{-1}$. We get a perfect match between WaveQLab and ExaSeis in all cases. In the results from the SeisSol code, we see small reflections appearing at the end of each recorded station, which can be removed by further extending the domain.

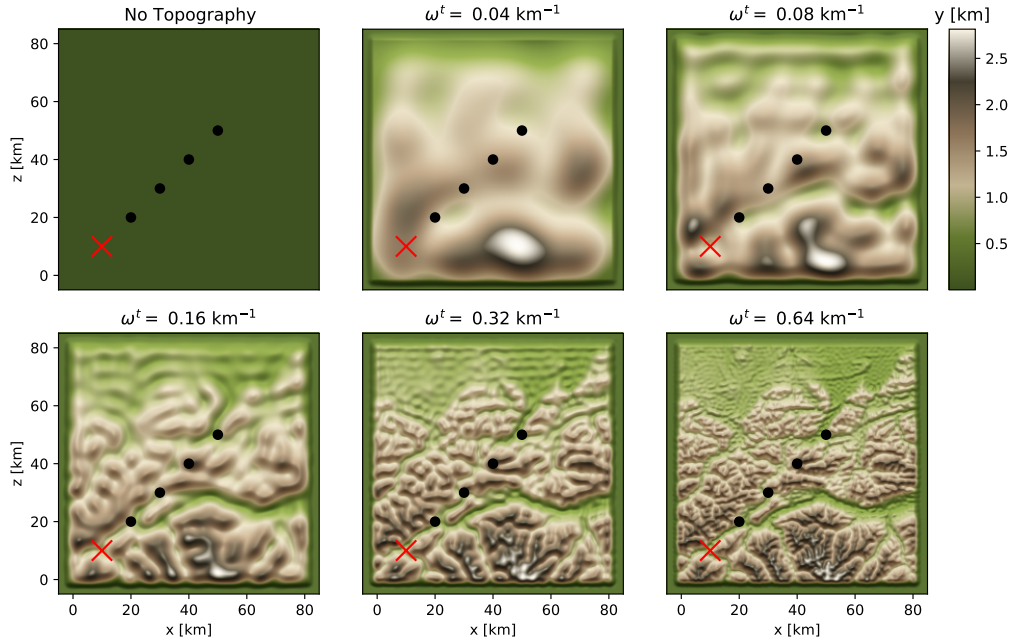


Figure 8.19.: Filtered topographies of the Zugspitze scenario for $\omega^t \in \{0, 4, 8, 16, 32, 64\}$ km^{-1}. Receivers are marked with a black circle, the point source location with a red cross. Adapted from Figure 9 in [43].

Influence on the time-step size

The complexity of the topography has a direct influence on the maximal admissible time-step size. As we showed in Section 7.4.4, the time-step size directly depends on the row of the Jacobian with the highest norm. In case of a flat surface the transformation from computational to physical space is the identity, the Jacobian is one everywhere. As we introduce complexity to the topography, the values of the element local Jacobians in the curvilinear mesh increases, as the transformation from computational to physical space leads to a "higher" deformation of the elements.

In Table 8.4, we show the highest norm for the rows of all Jacobians in the mesh for an ADER-DG method of order 3 and 241 elements, depending on the cut-off wave number ω^t. We see that the values increase with the cut-off wave number. For $\omega_t = 0.64$ the simulation requires 2.4× more time-steps, than a simulation with flat topography. Significant is the effect for an unfiltered topography, for which we would require 19.5× more time-steps, than for a flat surface. Compared to the highest cut-off frequency $\omega_t = 0.64$

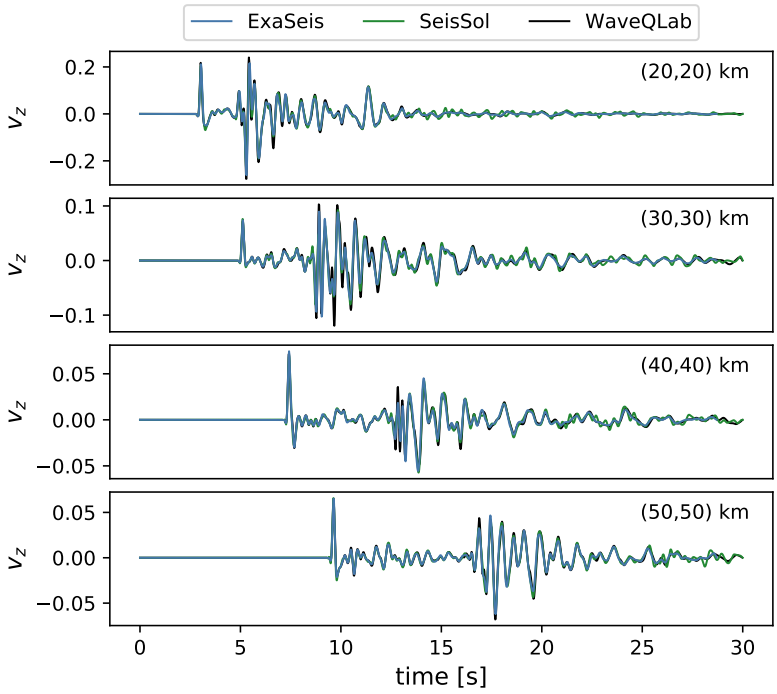


Figure 8.20.: Comparison of the velocity component in z for all four receivers, for the topography with the highest cut-off frequency $\omega^t = 64$ km^{-1}.

Table 8.4.: Maximal determinant of the element local metric derivatives ($\max \det\{J\}$) in the Zugspitze model for various cut-off frequencies ω^t. With increasing frequency content in the topography the maximal determinant significantly decreases.

ω^t	0.0	0.04	0.08	0.16	0.32	0.64	∞
$\max \|J_i^{\boldsymbol{k}}\|_2$	1.0	1.38	1.61	1.86	2.27	2.4	19.5

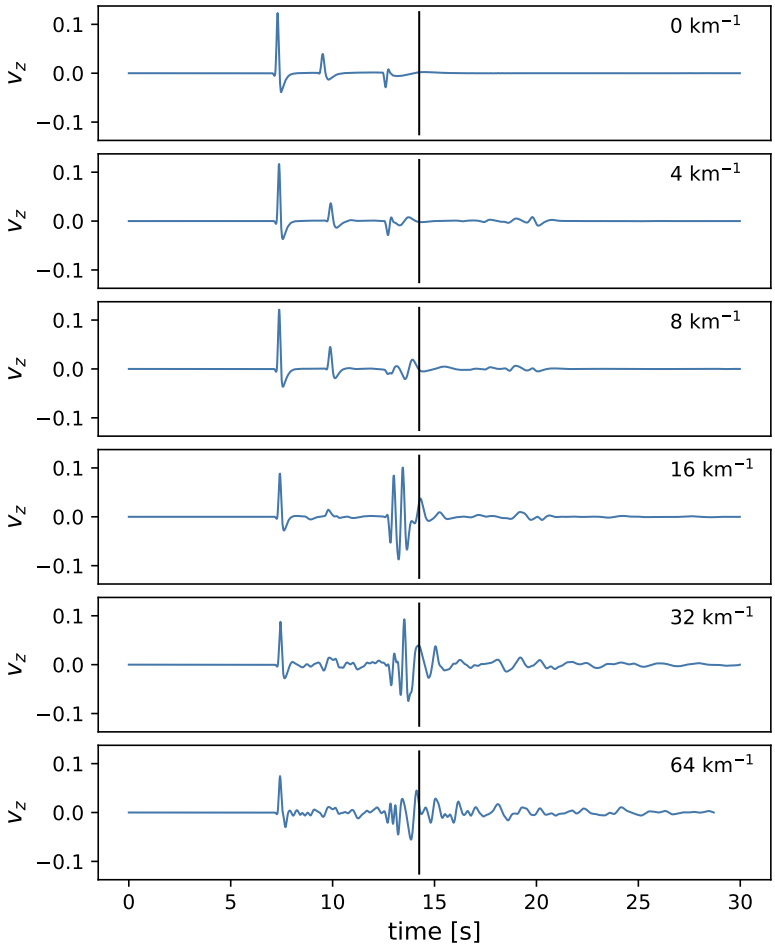


Figure 8.21.: Seismograms at $(x, y) = (40, 40)$ for all six versions of the complex topography.

for the simulation with the unfiltered topography, the number of time-steps increases by a factor of $\sim 8.1\times$.

Scattering effects

We want to quantify the effect of the complex topography on the scattering of the wavefield. In Figure 8.21 we see the recorded seismograms for the station at $(x, y) = (40, 40)$ km for the z component of the velocity vector. We can clearly observe that the frequency content increases with the frequency content of the topography.

In order to quantify the effect, we look at the seismogram after the S-

Table 8.5.: Spectral centroid of the four velocity components measured at the seismogram at $(40, 40)$ km for all six complex topographies.

ω^t	0.0	0.04	0.08	0.16	0.32	0.64
u	0.5	2.3	2.2	5.1	9.2	11.3
v	1.0	4.3	4.5	7.5	10.9	15.4
w	0.5	1.9	1.8	6.6	8.9	8.7

wave has been recorded on the flat surface at $t > 14.25\,\mathrm{s}$. As measure we compute the spectral centroid of the signal

$$C = 100 \cdot \frac{\sum_{t=0}^{N} f_i \cdot \omega_i}{\sum_{t=0}^{N} \omega_i}, \tag{8.21}$$

where $\omega_i = i/T$ are the sampled frequencies and f_i the corresponding coefficients of the signal. The measure tells us where the *center of mass* within a spectrum of a seismogram is located and allows us to compare the frequency contents of different seismograms. For the seismogram at $(40, 40)$ km the spectral centroids for all three velocity components and topographies are summarized in Table 8.5. We see that for all components the measure increases with the frequency content of the topography. This confirms our initial observation that the scattering effect increases with the complexity of the topography.

Peak ground velocity and amplification

In order to quantify the amplification and deamplification effects of the topography we compute the peak ground velocity (PGV) from the particle velocity measured at the surface for all six set-ups

$$PGV(x, y) = \max_t \sqrt{v_x(x, y, t)^2 + v_z(x, y, t)^2}. \tag{8.22}$$

In Figure 8.22 we show the PGV for all six topographies. We can see that with increasing complexity of the topography, the frequency of PGV increases. Comparing to the topography reveals, that we get the highest values of PGV at the mountain peaks, in valleys the values are reduced. To quantify these amplification (AMP) and deamplification (DMP) effects, we compute the maximal and minimal difference between PGVs for the complex topographies against the flat topography. The resulting values are

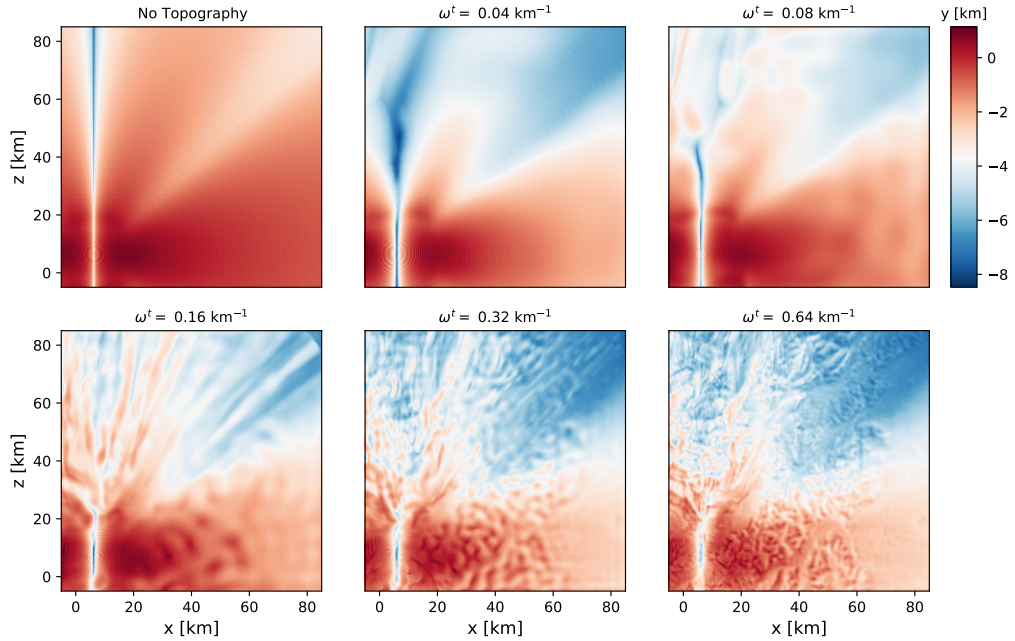


Figure 8.22.: PGV maps for all six topographies of the Zugspitze scenario.

Table 8.6.: Amplification (AMP) and Deamplification (DMP) in $\mathrm{m\,s^{-1}}$, depending on the cut-off frequency of the topography.

ω^t	0.04	0.08	0.16	0.32	0.64
AMP $[\mathrm{m\,s^{-1}}]$	0.99	1.3	1.1	1.2	1.7
DMP $[\mathrm{m\,s^{-1}}]$	-1.8	-1.9	-1.8	-2.0	-2.1

presented in Table 8.6. We get the largest amplification (AMP $1.7\,\mathrm{m\,s^{-1}}$) and deamplification (DMP $-2.1\,\mathrm{m\,s^{-1}}$) for the topography with the highest complexity. Vice versa both measures are reduced for the topography with the lowest complexity (AMP $0.99\,\mathrm{m\,s^{-1}}$ and DMP $-1.8\,\mathrm{m\,s^{-1}}$). For $\omega_t = 0.08, 0.16, 0.32$ all values for AMP lie close to each other.

8.5 The Húsavík-Flatey fault

Finally, we simulate an earthquake located in the north Iceland region. We look at the Húsavík-Flatey fault which is a $\sim 90\,\mathrm{km}$ long right lateral strike-slip fault. The set-up is based on the work by Li et al. [88]. The topography and fault are illustrated in Figure 8.23a. The structure of the fault is con-

structed from 55 vertical segments, that are summarized in one smoothed curve (red line). As our meshing approach becomes significantly more efficient, when faults are approximately parallel to one of the boundaries of the domain, we rotate the original set-up from [88] by 60° east. With PML we can restrict the domain to 96.8 km in each direction.

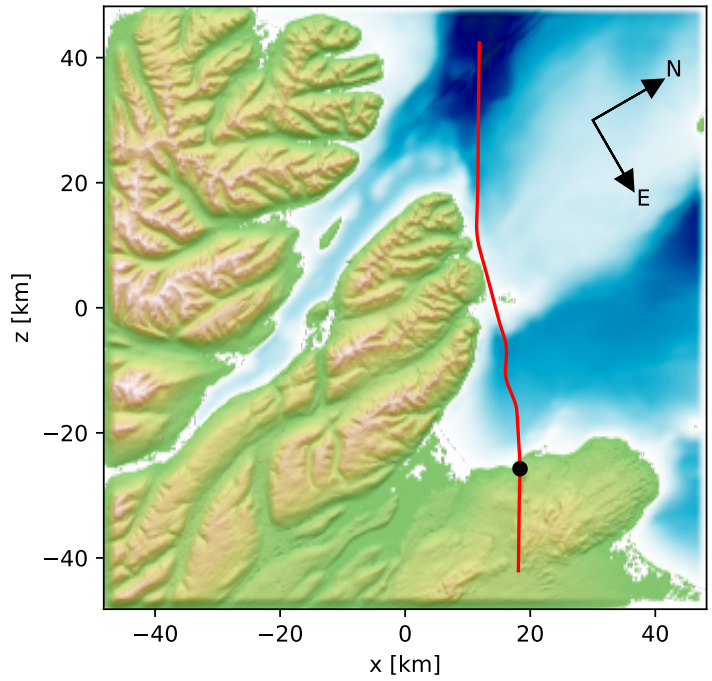
Initially we set Andersonian stresses with the method by Ulrich et al. [129]. Material parameters are determined by a depth dependent velocity model that we initialize using the easi framework [130]. The fault strength is modeled with a linear slip weakening law, for which the parameters are set to $\mu_s = 0.55$, $\mu_d = 0.1$ and $D_c = 0.5$.

The hypocenter is located at $(x_0, z_0) = (13.8, 2.5)$ km and at a depth of 5 km. Nucleation is initialized with the method we presented in Equation (8.9), for parameters $V_s = 3.8\,\mathrm{km\,s^{-1}}$, $r_{crit} = 3.0$ km and $t_0 = 0.0$ s. Cohesion linearly decreases in the top 10 km from 0.4 GPa to 2.4 GPa

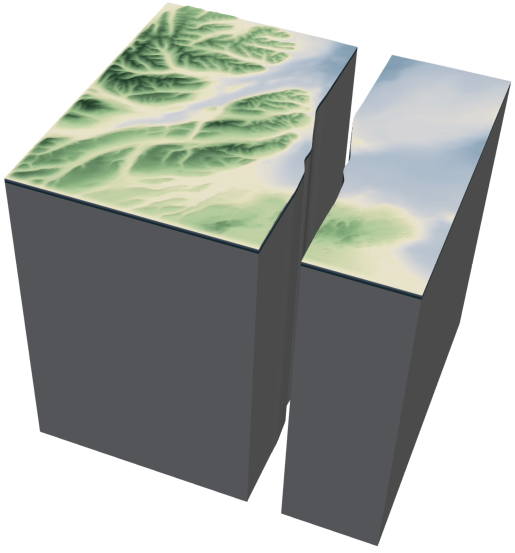
In order to generate the mesh, we split the domain into two sub-domains, such that the fault is modeled at their shared interface Figure 8.23b. The mesh consists of a total of 241 elements in each direction for an order three ADER-DG method, which results in a resolution of $\Delta x = 0.4$ km and a sub-cell resolution of $\Delta x_c = 0.1$ km. The resulting time-step size is $\Delta t = 0.7$ ms. In Figure 8.24, we compare the resulting rupture on the fault against a simulation performed with the SeisSol framework. The SeisSol reference has an on-fault resolution of 0.2 km and gradually coarsened to a maximum cell size of 5.0 km.

Both approaches show the exact same evolution of the rupture contour (top plot), which propagates homogeneously from the hypocenter towards the boundaries of the fault. After 16 s the rupture has propagated over the entire fault. For the absolute slip (ASL), we get well agreeing results for both codes. Close to the surface we spot small differences, which we explain with the different representations of the surface in both codes. While in SeisSol the surface is represented with linear functions on triangles, ExaSeis represent the surface in this set-up with cubic polynomials. The highest slip is right at the hypocenter with 4.8 m. South-west from the hypocenter, we locate a second patch with increased slip of 4.1 m. Using the absolutes slip on the fault, we measure a moment magnitude of $M_w = 7.2$.

In order to assess the impact of the earthquake on the surface, we plot the ground motion (i.e. the maximal acceleration in each location as multiples of $g = 9.81\,\mathrm{m\,s^{-2}}$) and the recorded particle velocity in Húsavík in Figure 8.25. We see that the ground motion is symmetric on both halves of the fault, with the highest measured ground motion of $7.0\,\mathrm{m\,s^{-2}}$. In Húsavík the highest ground motion is $3.0\,\mathrm{m\,s^{-2}}$.



(a) Topography and fault structure (red line) of the Húsavík-Flatey fault.



(b) Mesh for the Húsavík-Flatey fault.

Figure 8.23.: Rotated set-up for the Húsavík-Flatey fault (a). The fault is marked as red line, Húsavík as circle. Our meshing approach from Section 7.6 splits the mesh into two blocks (b).

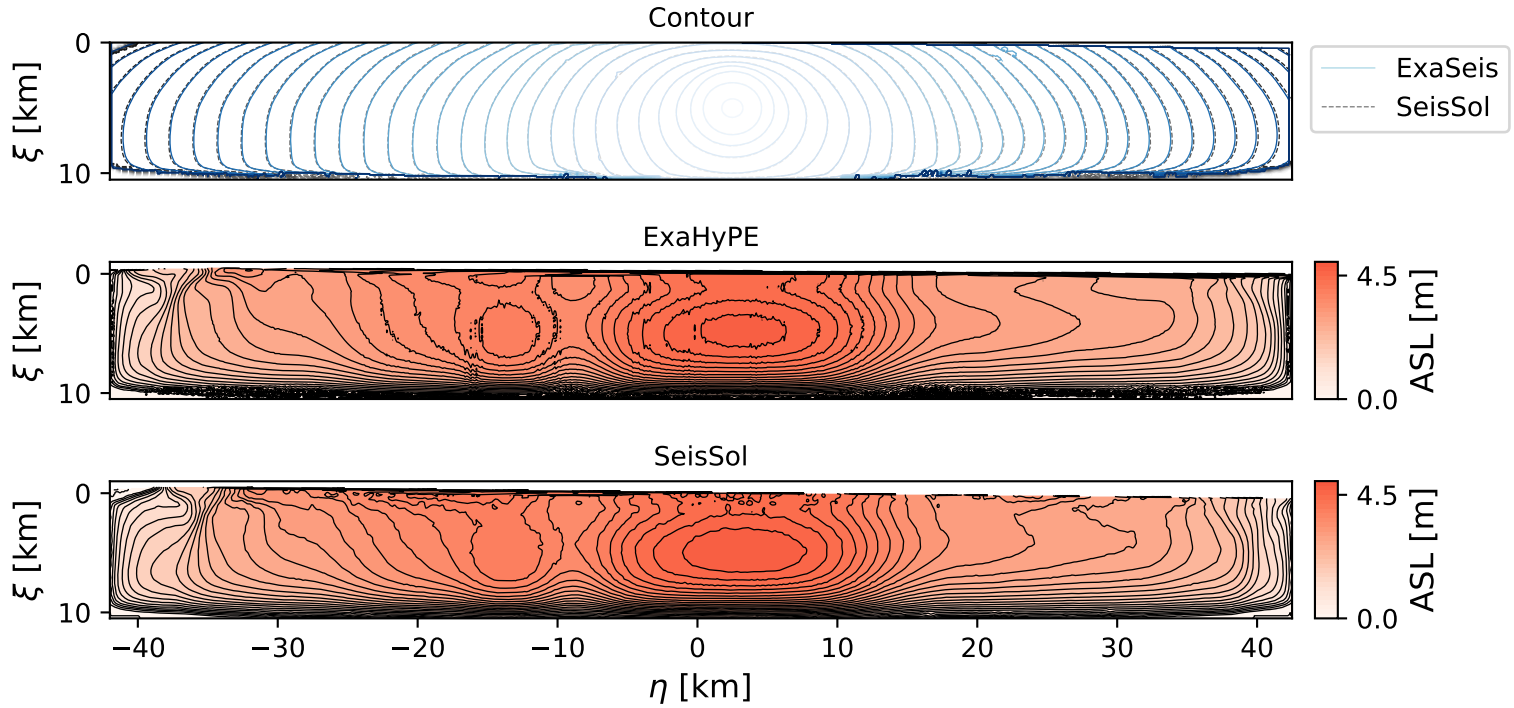
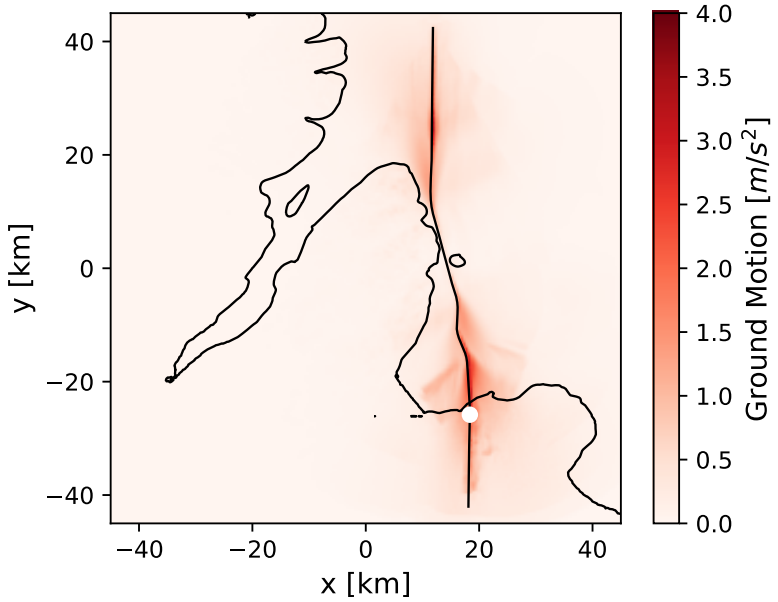
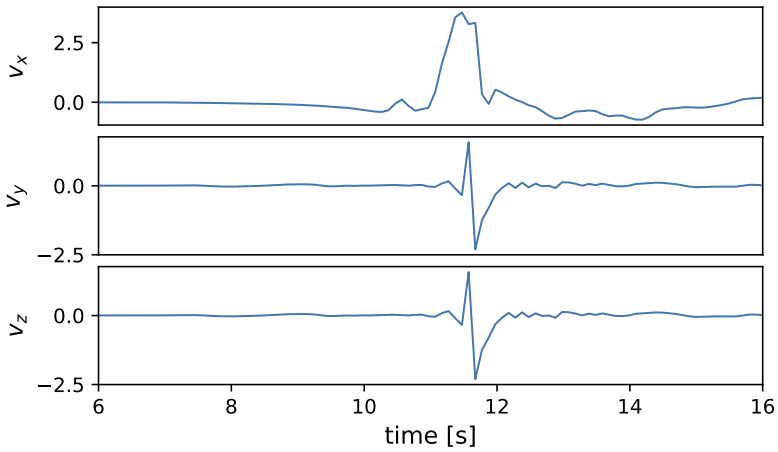


Figure 8.24.: Comparison of ExaSeis ans SeisSol for the rupture dynamics for the North-Iceland scenario. Top comparison of the rupture contour, both codes match perfectly. Middle and Bottom Absolute Slip Rate (ASL) for ExaSeis and SeisSol. Both codes show a very good agreement, on the surface some difference appear caused by the differing representations.



(a) Ground motion.



(b) Particle velocity at Húsavík.

Figure 8.25.: Ground Motion $[m/s^2]$ and particle velocity measured at Húsavík for the earthquake on the Húsavík-Flatey fault.

CHAPTER

9

Conclusion and discussion

Efficient earthquake and tsunami simulations are the basis of urgent computing pipelines, probabilistic seismic and tsunami hazard assessment and uncertainty quantification. In this context, we developed and implemented numerical schemes based on the ADER-DG method. To source tsunamis from the results of an earthquake simulation, we introduced a linking pipeline.

In order to conclude on the efficiency of the implemented methods, we performed a time-to-solution comparison of Runge-Kutta and ADER-DG methods for the shallow water equations.

In many ways, the non-linear shallow water equations we use throughout this work are limited. While several studies show that gravity waves, the essential wave type in tsunamis, are resolved accurately by the model, solitary waves or the dispersion of non-hydrostatic waves are not included in this model. Our realization of the SWE in sam(oa)^2 is an ideal basis to extend the code to more complex models, as the Boussinesq approximation [76] or new dispersive approaches [46]. In order to include vertical flow a multi-layer SWE approach can be integrated [48]. The bandwidth model, which we developed for sam(oa)^2 in the time-to-solution comparison (Chapter 4), provides a guideline for the efficiency of these extensions.

Our full time-to-solution comparison is performed exclusively for the sam(oa)^2 framework. We showed that the payload we put on each cell is crucial for the maximum attainable bandwidth of the implemented model and also influences the time-per-dof. However, our findings show the most important characteristics of both schemes: In case of smooth solutions, we

can benefit from the high order convergence of the ADER-DG method. For high orders, the main kernels can reach higher performance. For inundation at the coast, the RK-method profits from the Barth-Jespersen typed limiter which also reduces the number of required time-steps as the wave-speeds are kept small. Replacing the first order finite volume scheme used in the limiter of the ADER-DG method, with a second-order finite volume scheme, as the MUSCL-Hancock scheme that incorporates the Barth-Jespersen typed limiter, would be a promising idea [133].

Implementations of patches of low-order DG elements are one way to increase the payload on single elements [49, 50, 83]. With this approach we can increase the maximum attainable bandwidth for low orders.

The linking approach with novel Fourier filter, enables the linking of dynamic rupture codes and tsunami models. One essential weakness of this approach is that the parameters that define how strong a displacement field is filtered, have to be determined empirically. A set of rules for the parameters has to be found. With increasing computing power fully coupled models of earth and water layer become feasible and allow modeling the interaction of sea-surface and sea floor more accurately [1, 81]. However, no fully coupled model that resolves inundation has been developed to the best of our knowledge, such that the linking to designated tsunami codes remains without alternative.

Finally, we presented the ExaSeis framework, with which the ExaHyPE-Engine becomes ready for dynamic rupture simulations with dynamically adaptive refined curvilinear meshes. While in this work, we reproduced a synthetic community benchmark with dynamic AMR, the method next has to be applied to reproduce actual earthquake events. One candidate is the simulation of earthquakes in the north Iceland region which we performed on uniform meshes in this work. An essential component of these simulations is the automatic mesher that we developed. The mesher allows to automatically set-up different geometries for fault structures or surfaces.

We saw that the time-step size depends on the complexity of the topography, and single elements can have a significant effect on time required for a whole simulation. A known method to address this problem is local time stepping [19, 132], which is realizable for the Cauchy-Kovalevskaya method we introduced.

A comparison in time-to-solution to alternative approaches as the Diffuse Interface Method by Tavelli et al. [123] or the Godunov-Peshkov-Romensik model [37, 51], where the time-step size is independent from the topography would be desirable.

Overall this work enabled the simulation of earthquakes and tsunamis

with the ADER-DG method. We showed the whole picture of the method's potential for the simulation of tsunamis with the SWE and provided a performance model that can be used for the extension of sam(oa)^2. For computational seismology, we developed one of the first full functioning codes for dynamic rupture simulations on dynamically refined meshes. The method is set-up fully automatic, such that it is ready for applications in UQ, PSHA and urgent computing.
- I hope you enjoyed reading my dissertation.

APPENDIX

A

Fourier transformation of displacements

First we compute the Fourier transformation to the S-wave:

$$U_v(x,t) = \frac{1}{v^2}\left(1 - \frac{d^2}{r^2}\right)\exp\left(-\left(t - \frac{r}{v}\right)^2\right), \tag{A.1}$$

where v is the velocity of the wave in the material and r is the distance to the point source location $r = \sqrt{x^2 + 1^2}$.

In the proof we use the approximation

$$\frac{r}{v} = \frac{\sqrt{x^2 + d}}{v} \approx \frac{x}{v}, \tag{A.2}$$

for which we assume $\frac{d}{v^2} \ll 1$. We start with

$$\begin{aligned}
&\mathcal{F}_{t,x}(f,\omega)\left\{U_v(x,t)\right\} = \\
&\mathcal{F}_{t,x}(f,\omega)\left\{\frac{1}{v^2}\left(1 - \frac{d^2}{r^2}\right)\exp\left(-\left(t - \frac{r}{v}\right)^2\right)\right\} = \\
&\delta(f)\,\delta(\omega)\,\frac{\sqrt{\pi}}{\sqrt{2}v^2} - \frac{1}{v^2}\mathcal{F}_{t,x}(f,\omega)\left\{\frac{d^2}{r^2}\exp\left(-\left(t - \frac{r}{v}\right)^2\right)\right\}.
\end{aligned} \tag{A.3}$$

For the last term we get

$$\begin{aligned}
&\mathcal{F}_{t,x}(f,\omega)\left\{\frac{d^2}{r^2}\exp\left(-\left(t-\frac{r}{v}\right)^2\right)\right\} = \\
&\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\mathcal{F}_t(f)\left\{\exp\left(-\left(t-\frac{r}{v}\right)^2\right)\right\}\right\} = \\
&\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\exp\left(-if\frac{r}{v}\right)\mathcal{F}_t(f)\left\{\exp\left(-t^2\right)\right\}\right\} = \\
&\exp\left(-\frac{f^2}{4}\right)\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\exp\left(-if\frac{r}{v}\right)\right\}. \qquad \text{(A.4)}
\end{aligned}$$

(A.5)

From which we again take the last term

$$\begin{aligned}
&\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\exp\left(-if\frac{r}{v}\right)\right\} = \\
&\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\right\} * \mathcal{F}_x(\omega)\left\{\exp\left(-if\frac{r}{v}\right)\right\} \overset{A.2}{\approx} \\
&\mathcal{F}_x(\omega)\left\{\frac{d^2}{r^2}\right\} * \mathcal{F}_x(\omega)\left\{\exp\left(-if\frac{x}{v}\right)\right\} = \\
&\frac{\pi}{dv^2}\exp\left(-d|\omega|\right) * \delta(\omega-\frac{f}{v}) = \\
&\frac{\pi}{dv^2}\exp\left(-d|\omega-\frac{f}{v}|\right). \qquad \text{(A.6)}
\end{aligned}$$

We can accumulate all three results to

$$\begin{aligned}
&\mathcal{F}_{t,x}(f,\omega)\left\{U_v(x,t)\right\} = \\
&\qquad \delta\left(f\right)\delta\left(\omega\right)\frac{\sqrt{\pi}}{\sqrt{2}v^2} - \frac{\pi}{dv^2}\exp\left(-\frac{f^2}{4}\right)\exp\left(-d|\omega-\frac{f}{v}|\right). \qquad \text{(A.7)}
\end{aligned}$$

Second the Fourier transformation of the permanent displacement:

The time-rate function and the final displacement are defined on independent variables, we can thus evaluate their Fourier transformations separately.

Using the rules for shifting and the transformation for a Gaussian hill the

Fourier transformation of the final displacement is

$$\mathcal{F}_x(\omega)\left\{d(x)\right\} = \mathcal{F}_x(\omega)\left\{\exp\left(-\frac{x^2}{2\cdot\sigma_r^2}\right)\right\} = \sigma_r \cdot \exp\left(-\frac{(\sigma_r\cdot\omega)^2}{2}\right). \quad \text{(A.8)}$$

For the time-rate function we get

$$\begin{aligned}
&\mathcal{F}_t(f)\left\{\chi(t)\right\} = \\
&\mathcal{F}_t(f)\left\{\frac{1}{\sqrt{\pi}}\frac{4}{t_c}\exp\left(-\left(\frac{(t-t_0)\cdot 4}{t_c}\right)^2\right)\right\} = \\
&\frac{\sqrt{2}}{\sqrt{\pi}}\exp\left(it_0 f\right)\exp\left(-\frac{t_c}{8}\cdot f^2\right). \qquad \text{(A.9)}
\end{aligned}$$

The integration property of the Fourier transformation gives us the transformation of the time-rate integral. For $f \neq 0$

$$\mathcal{F}_T(f)\left\{\int_{-\infty}^{T}\chi(t)dt\right\} = \quad \text{(A.10)}$$

$$\frac{1}{if}\mathcal{F}_t(f)\left\{\chi(t)\right\} + \pi\cdot\chi(0)\delta(f) \approx \quad \text{(A.11)}$$

$$\frac{\sqrt{2}}{f\sqrt{\pi}}\exp\left(i(t_0 f - \frac{\pi}{2})\right)\exp\left(-\frac{t_c}{8}\cdot f^2\right), \quad \text{(A.12)}$$

where we used $\chi(0) \approx 0$ and $i^{-1} = \exp\left(-i\frac{\pi}{2}\right)$.

APPENDIX

B

The three dimensional transfinite interpolation

The three dimensional transfinite interpolation $\vec{\mathcal{C}}^3\left[\Lambda^n\right](u,v,w) : [0,1]^3 \rightarrow \mathbb{R}^n$, to a set of boundary curves

$$\Lambda^n = \Big(\vec{F}_{0vw}(v,w)\,,\ \vec{F}_{1vw}(v,w)\,, \vec{F}_{u0w}(u,w)\,, \\ \vec{F}_{u1w}(u,w)\,,\ \vec{F}_{uv0}(u,v)\ \,, \vec{F}_{uv1}(u,v)\ \Big),$$

is defined by

$$\vec{\mathcal{C}}^3\left[\lambda^n\right](u,v,w) = A(u,v,w) + B(u,v,w) + C(u,v,w) \\ - AB(u,v,w) - AC(u,v,w) - BC(u,v,w) + ABC(u,v,w). \quad \text{(B.1)}$$

With helper functions

$$A(u,v,w) = (1-u)\vec{F}_{0vw}(v,w) + u\vec{F}_{1vw}(v,w)\,, \quad \text{(B.2)}$$
$$B(u,v,w) = (1-v)\vec{F}_{u0w}(v,w) + v\vec{F}_{u0w}(v,w)\,, \quad \text{(B.3)}$$
$$C(u,v,w) = (1-w)\vec{F}_{uv0}(u,v) + w\vec{F}_{uv1}(u,v)\,. \quad \text{(B.4)}$$

$$AB(u,v,w) = (1-u)(1-v)\vec{F}_{0vw}(0,w) + u(1-v)\vec{F}_{1vw}(0,w) + v(1-u)\vec{F}_{0vw}(1,w) + uv\vec{F}_{1vw}(1,w), \tag{B.5}$$

$$AC(u,v,w) = (1-u)(1-w)\vec{F}_{uv0}(0,v) + u(1-w)\vec{F}_{uv0}(1,v) + w(1-u)\vec{F}_{uv1}(0,v) + uw\vec{F}_{uv1}(1,v), \tag{B.6}$$

$$BC(u,v,w) = (1-v)(1-w)\vec{F}_{u0w}(u,0) + v(1-w)\vec{F}_{u1w}(u,0) + w(1-v)\vec{F}_{u0w}(u,1) + vw\vec{F}_{u1w}(u,1). \tag{B.7}$$

and

$$\begin{aligned} ABC(u,v,w) = {} & (1-u)(1-v)(1-w)\vec{F}_{u0w}(0,0) \\ & + w(1-u)(1-v)\vec{F}_{u0w}(0,1) \\ & + u(1-v)(1-w)\vec{F}_{u0w}(1,0) \\ & + v(1-u)(1-w)\vec{F}_{u1w}(0,0) \\ & + uw(1-v)\vec{F}_{u0w}(1,1) \\ & + vw(1-u)\vec{F}_{u1w}(0,1) \\ & + uv(1-w)\vec{F}_{u1w}(1,0) \\ & + uwv\vec{F}_{u1w}(1,1). \end{aligned} \tag{B.8}$$

Bibliography

[1] Lauren S Abrahams, Lukas Krenz, Eric M Dunham, Alice-Agnes Gabriel, and Tatsuhiko Saito. *Comparison of methods for coupled earthquake and tsunami modeling*. 2022.

[2] Robert A Adams and John JF Fournier. *Sobolev spaces*. Elsevier, 2003.

[3] Keiiti Aki and Paul G. Richards. *Quantitative Seismology*. 2nd edition. University Science Books, 2002. ISBN: 0-935702-96-2.

[4] Sara Aniko Wirp et al. "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), p. 177.

[5] Daniel Appelö and Gunilla Kreiss. "A new absorbing layer for elastic waves". In: *Journal of Computational Physics* 215.2 (2006), pp. 642–660.

[6] Daniel Arndt et al. "ExaDG: High-order discontinuous Galerkin for the exa-scale". In: *Software for Exascale Computing-SPPEXA 2016-2019*. Springer, Cham, 2020, pp. 189–224.

[7] Marc de la Asunción, Manuel J Castro, Enrique Domingo Fernández-Nieto, José M Mantas, Sergio Ortega Acosta, and José Manuel González-Vida. "Efficient GPU implementation of a two waves TVD-WAF method for the two-dimensional one layer shallow water system on structured meshes". In: *Computers & Fluids* 80 (2013), pp. 441–452.

[8] A. Y. Babeyko, A. Hoechner, and S. V. Sobolev. "Source modeling and inversion with near real-time GPS: a GITEWS perspective for Indonesia". In: *Natural Hazards and Earth System Sciences* 10.7 (2010), pp. 1617–1627. DOI: 10.5194/nhess-10-1617-2010.

[9] Michael Barall. “A grid-doubling finite-element technique for calculating dynamic three-dimensional spontaneous rupture on an earthquake fault”. In: *Geophysical Journal International* 178.2 (2009), pp. 845–859.

[10] Pierre-Yves Bard et al. “Effects of surface geology on ground motion: recent results and remaining issues”. In: *Proc. 10 European Conf. Earth. Eng., ed. Duma, Balkema, Rotterdam*. 1995, pp. 305–323.

[11] Timothy Barth and Dennis Jespersen. “The design and application of upwind schemes on unstructured meshes”. In: *27th Aerospace sciences meeting*. 1989, p. 366.

[12] Peter Bastian, Markus Blatt, Andreas Dedner, Christian Engwer, Robert Klöfkorn, Mario Ohlberger, and Oliver Sander. “A generic grid interface for parallel and adaptive scientific computing. Part I: abstract framework”. In: *Computing* 82.2-3 (2008), pp. 103–119.

[13] Peter Bastian et al. “A generic grid interface for parallel and adaptive scientific computing. Part II: Implementation and tests in DUNE”. In: *Computing* 82.2-3 (2008), pp. 121–138.

[14] Peter Bauer, Alan Thorpe, and Gilbert Brunet. “The quiet revolution of numerical weather prediction”. In: *Nature* 525.7567 (2015), pp. 47–55.

[15] J Behrens and F Dias. “New computational methods in tsunami science”. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 373.2053 (2015), p. 20140382.

[16] Jörn Behrens et al. “Probabilistic Tsunami Hazard and Risk Analysis: A Review of Research Gaps”. In: *Frontiers in Earth Science* 9 (2021), p. 114.

[17] Jean-Pierre Berenger. “A perfectly matched layer for the absorption of electromagnetic waves”. In: *Journal of computational physics* 114.2 (1994), pp. 185–200.

[18] Marsha J Berger, David L George, Randall J LeVeque, and Kyle T Mandli. “The GeoClaw software for depth-averaged flows with adaptive refinement”. In: *Advances in Water Resources* 34.9 (2011), pp. 1195–1206.

[19] 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)*. IEEE. 2016, pp. 854–863.

[20] Alexander Breuer, Alexander Heinecke, Leonhard Rannabauer, and Michael Bader. "High-order ADER-DG minimizes energy-and time-to-solution of SeisSol". In: *International Conference on High Performance Computing*. Springer. 2015, pp. 340–357.

[21] Shintaro Bunya, Ethan J. Kubatko, Joannes J. Westerink, and Clint Dawson. "A wetting and drying treatment for the Runge–Kutta discontinuous Galerkin solution to the shallow water equations". In: *Computer Methods in Applied Mechanics and Engineering* 198.17 (2009), pp. 1548–1562. DOI: https://doi.org/10.1016/j.cma.2009.01.008.

[22] Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas. "p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees". In: *SIAM Journal on Scientific Computing* 33.3 (2011), pp. 1103–1133. DOI: 10.1137/100791634.

[23] John C Butcher. "Coefficients for the study of Runge-Kutta integration processes". In: *Journal of the Australian Mathematical Society* 3.2 (1963), pp. 185–201.

[24] John C Butcher. "Implicit runge-kutta processes". In: *Mathematics of Computation* 18.85 (1964), pp. 50–64.

[25] G. F. Carrier and H. P. Greenspan. "Water waves of finite amplitude on a sloping beach". In: *Journal of Fluid Mechanics* 4.1 (1958), pp. 97–109. DOI: 10.1017/S0022112058000331.

[26] Dominic Etienne Charrier. "Communication-avoiding algorithms for a high-performance hyperbolic PDE engine". PhD thesis. Durham University, 2020.

[27] Weng Cho Chew and William H Weedon. "A 3D perfectly matched medium from modified Maxwell's equations with stretched coordinates". In: *Microwave and optical technology letters* 7.13 (1994), pp. 599–604.

[28] Bernardo Cockburn and Chi-Wang Shu. "Runge–Kutta discontinuous Galerkin methods for convection-dominated problems". In: *Journal of scientific computing* 16.3 (2001), pp. 173–261.

[29] Bernardo Cockburn and Chi-Wang Shu. "The Runge–Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems". In: *Journal of Computational Physics* 141.2 (1998), pp. 199–224.

[30] Steven M. Day. *Tests of 3D elastodynamics codes*. https://steveday.sdsu.edu/BASINS/Final_Report_1A01.pdf. Accessed June 17th 2021. 2001.

[31] Willem Deconinck et al. "Atlas : A library for numerical weather prediction and climate modelling". In: *Computer Physics Communications* 220 (2017), pp. 188–204. DOI: https://doi.org/10.1016/j.cpc.2017.07.006.

[32] Michael Dumbser and Dinshaw S. Balsara. "A new efficient formulation of the HLLEM Riemann solver for general conservative and non-conservative hyperbolic systems". In: *Journal of Computational Physics* 304 (2016), pp. 275–319. DOI: https://doi.org/10.1016/j.jcp.2015.10.014.

[33] 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: https://doi.org/10.1016/j.jcp.2008.05.025.

[34] 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.

[35] Michael Dumbser and Martin Käser. "Arbitrary high order non-oscillatory finite volume schemes on unstructured meshes for linear hyperbolic systems". In: *Journal of Computational Physics* 221.2 (2007), pp. 693–723.

[36] Michael Dumbser and Raphaël Loubère. "A simple robust and accurate a posteriori sub-cell finite volume limiter for the discontinuous Galerkin method on unstructured meshes". In: *Journal of Computational Physics* 319 (2016), pp. 163–199. DOI: https://doi.org/10.1016/j.jcp.2016.05.002.

[37] Michael Dumbser, Ilya Peshkov, Evgeniy Romenski, and Olindo Zanotti. "High order ADER schemes for a unified first order hyperbolic formulation of continuum mechanics: Viscous heat-conducting fluids and elastic solids". In: *Journal of Computational Physics* 314 (2016), pp. 824–862. DOI: https://doi.org/10.1016/j.jcp.2016.02.015.

[38] Michael Dumbser, Olindo Zanotti, Raphaël Loubère, and Steven Diot. "A posteriori subcell limiting of the discontinuous Galerkin finite element method for hyperbolic conservation laws". In: *Journal of Computational Physics* 278 (2014), pp. 47–75. DOI: https://doi.org/10.1016/j.jcp.2014.08.009.

[39] 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.

[40] 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: https://doi.org/10.1016/j.jcp.2015.10.021.

[41] Kenneth Duru, Leonhard Rannabauer, Alice-Agnes Gabriel, and Heiner Igel. "A new discontinuous Galerkin method for elastic waves with physically motivated numerical fluxes". In: *Journal of Scientific Computing* 88.3 (2021), pp. 1–32.

[42] Kenneth Duru, Leonhard Rannabauer, Alice-Agnes Gabriel, Gunilla Kreiss, and Michael Bader. "A stable discontinuous Galerkin method for the perfectly matched layer for elastodynamics in first order form". In: *Numerische Mathematik* 146.4 (2020), pp. 729–782.

[43] Kenneth Duru, Leonhard Rannabauer, Alice-Agnes Gabriel, On Ki Angel Ling, Heiner Igel, and Michael Bader. "A stable discontinuous Galerkin method for linear elastodynamics in 3D geometrically complex elastic solids using physics based numerical fluxes". In: *Computer Methods in Applied Mechanics and Engineering* 389 (2022), p. 114386.

[44] Albert Einstein. "Die grundlage der allgemeinen relativitätstheorie". In: *Das Relativitätsprinzip*. Springer, 1923, pp. 81–124.

[45] Björn Engquist and Andrew Majda. "Absorbing boundary conditions for numerical simulation of waves". In: *Proceedings of the National Academy of Sciences* 74.5 (1977), pp. 1765–1766.

[46] C. Escalante, M. Dumbser, and M.J. Castro. "An efficient hyperbolic relaxation system for dispersive non-hydrostatic water waves and its solution with high order discontinuous Galerkin schemes".

In: *Journal of Computational Physics* 394 (2019), pp. 385–416. DOI: https://doi.org/10.1016/j.jcp.2019.05.035.

[47] Niklas Fehn, Wolfgang A. Wall, and Martin Kronbichler. "Efficiency of high-performance discontinuous Galerkin spectral element methods for under-resolved turbulent incompressible flows". In: *International Journal for Numerical Methods in Fluids* 88.1 (2018), pp. 32–54. DOI: https://doi.org/10.1002/fld.4511.

[48] Enrique Domingo Fernández-Nieto, MJ Castro Díaz, and Carlos Parés. "On an intermediate field capturing Riemann solver based on a parabolic viscosity matrix for the two-layer shallow water system". In: *Journal of Scientific Computing* 48.1 (2011), pp. 117–140.

[49] Chaulio R Ferreira and Michael Bader. "Load balancing and patch-based parallel adaptive mesh refinement for tsunami simulation on heterogeneous platforms using Xeon Phi coprocessors". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. 2017, pp. 1–12.

[50] Chaulio R Ferreira, Kyle T Mandli, and Michael Bader. "Vectorization of Riemann solvers for the single-and multi-layer Shallow Water Equations". In: *2018 International Conference on High Performance Computing & Simulation (HPCS)*. IEEE. 2018, pp. 415–422.

[51] A-A Gabriel, Duo Li, Simone Chiocchetti, Maurizio Tavelli, Ilya Peshkov, Evgeniy Romenski, and Michael Dumbser. "A unified first-order hyperbolic model for nonlinear dynamic rupture processes in diffuse fracture zones". In: *Philosophical Transactions of the Royal Society A* 379.2196 (2021), p. 20200130.

[52] Peter Galison. *Image and Logic*. Chicago: University of Chicago Press, 1997.

[53] Jean-Matthieu Gallard, Lukas Krenz, Leonhard Rannabauer, Anne Reinarz, and Michael Bader. "Role-Oriented Code Generation in an Engine for Solving Hyperbolic PDE Systems". In: *Tools and Techniques for High Performance Computing*. Ed. by Guido Juckeland and Sunita Chandrasekaran. Cham: Springer International Publishing, 2020, pp. 111–128. ISBN: 978-3-030-44728-1.

[54] Jean-Matthieu Gallard, Leonhard Rannabauer, Anne Reinarz, and Michael Bader. "Vectorization and Minimization of Memory Footprint for Linear High-Order Discontinuous Galerkin Schemes". In:

2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2020, pp. 711–720. DOI: 10.1109/IPDPSW50202.2020.00126.

[55] 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.

[56] Louis Geli, Pierre-Yves Bard, and Béatrice Jullien. "The effect of topography on earthquake ground motion: a review and new results". In: *Bulletin of the Seismological Society of America* 78.1 (1988), pp. 42–63.

[57] David L George. "Augmented Riemann solvers for the shallow water equations over variable topography with steady states and inundation". In: *Journal of Computational Physics* 227.6 (2008), pp. 3089–3113.

[58] Steven J Gibbons et al. "Probabilistic Tsunami Hazard Analysis: High Performance Computing for Massive Scale Monte Carlo type Inundation Simulations". In: *EGU General Assembly Conference Abstracts*. 2020, p. 8041.

[59] F.X. Giraldo, J.S. Hesthaven, and T. Warburton. "Nodal High-Order Discontinuous Galerkin Methods for the Spherical Shallow Water Equations". In: *Journal of Computational Physics* 181.2 (2002), pp. 499–525. DOI: https://doi.org/10.1006/jcph.2002.7139.

[60] Paul Glaister. "Shallow water flow with cylindrical symmetry". In: *Journal of Hydraulic Research* 29.2 (1991), pp. 219–227.

[61] Sylfest Glimsdal et al. "A new approximate method for quantifying tsunami maximum inundation height probability". In: *Pure and Applied Geophysics* 176.7 (2019), pp. 3227–3246.

[62] Herman H Goldstine and Adele Goldstine. "The electronic numerical integrator and computer (eniac)". In: *Mathematical Tables and Other Aids to Computation* 2.15 (1946), pp. 97–110.

[63] Kazushige Goto and Robert A van de Geijn. "Anatomy of high-performance matrix multiplication". In: *ACM Transactions on Mathematical Software (TOMS)* 34.3 (2008), pp. 1–25.

[64] George Green et al. "On the motion of waves in a variable canal of small depth and width". In: *Transactions of the Cambridge Philosophical Society* 6 (1838), p. 457.

[65] Anita Grezio et al. “Probabilistic tsunami hazard analysis: Multiple sources and global applications”. In: *Reviews of Geophysics* 55.4 (2017), pp. 1158–1198.

[66] Bertil Gustafsson, Heinz-Otto Kreiss, and Joseph Oliger. *Time dependent problems and difference methods*. Vol. 24. John Wiley & Sons, 1995.

[67] Ruth A Harris et al. “A suite of exercises for verifying dynamic earthquake rupture codes”. In: *Seismological Research Letters* 89.3 (2018), pp. 1146–1162.

[68] Ruth A Harris et al. “The SCEC/USGS dynamic earthquake rupture code verification exercise”. In: *Seismological Research Letters* 80.1 (2009), pp. 119–126.

[69] 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*. 2016, pp. 981–991. DOI: 10.1109/SC.2016.83.

[70] Jan S Hesthaven and Tim Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[71] Juan J Horrillo, Zygmunt Kowalik, and Edward Kornkven. “The third international workshop on long-wave runup models”. In: (2004).

[72] Yoshiaki Ida. “Cohesive force across the tip of a longitudinal-shear crack and Griffith’s specific surface energy”. In: *Journal of Geophysical Research* 77.20 (1972), pp. 3796–3805.

[73] Claes Johnson and Juhani Pitkäranta. “An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation”. In: *Mathematics of computation* 46.173 (1986), pp. 1–26.

[74] Steven G. Johnson. *Notes on Perfectly Matched Layers (PMLs)*. 2021. arXiv: 2108.05348 [cs.CE].

[75] Kinjiro Kajiura. “The leading wave of a tsunami”. In: *Bulletin of the Earthquake Research Institute, University of Tokyo* 41.3 (1963), pp. 535–571.

[76] Jihwan Kim, Geir K Pedersen, Finn Løvholt, and Randall J LeVeque. "A Boussinesq type extension of the GeoClaw model-a study of wave breaking phenomena applying dispersive long wave models". In: *Coastal engineering* 122 (2017), pp. 75–86.

[77] Reinhard Klette. *Concise computer vision*. Springer, 2014.

[78] 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.

[79] JE Kozdon and EM Dunham. "Adaptive Mesh Refinement for Dynamic Rupture Simulations". In: *AGU Fall Meeting Abstracts*. Vol. 2010. 2010, S54A–08.

[80] Jeremy E Kozdon, Eric M Dunham, and Jan Nordström. "Simulation of dynamic earthquake ruptures in complex geometries using high-order finite difference methods". In: *Journal of Scientific Computing* 55.1 (2013), pp. 92–124.

[81] 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: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2021, pp. 1–14.

[82] Miriam Kristeková, Jozef Kristek, Peter Moczo, and Steven M Day. "Misfit criteria for quantitative comparison of seismograms". In: *Bulletin of the seismological Society of America* 96.5 (2006), pp. 1836–1850.

[83] Martin Kronbichler and Katharina Kormann. "Fast Matrix-Free Evaluation of Discontinuous Galerkin Finite Element Operators". In: *ACM Trans. Math. Softw.* 45.3 (Aug. 2019). DOI: 10.1145/3325864.

[84] Mustafa Kuzuoglu and Raj Mittra. "Frequency dependence of the constitutive parameters of causal perfectly matched anisotropic absorbers". In: *IEEE Microwave and Guided wave letters* 6.12 (1996), pp. 447–449.

[85] Thorne Lay et al. "The great Sumatra-Andaman earthquake of 26 december 2004". In: *science* 308.5725 (2005), pp. 1127–1133.

[86] Randall J LeVeque et al. *Finite volume methods for hyperbolic problems*. Vol. 31. Cambridge university press, 2002.

[87] Randall J LeVeque, David L George, and Marsha J Berger. “Tsunami modelling with adaptively refined finite volume methods”. In: *Acta Numerica* 20 (2011), pp. 211–289.

[88] Bo Li, Alice-Agnes Gabriel, Thomas Ulrich, Claudia Abril Lopez, Benedikt Halldorsson, and Michael Bader. “Physics-based rupture models and ground shaking simulations in the Húsavík–Flatey fault zone, Northern Iceland”. In: *AGU Fall Meeting 2021*. AGU. 2021.

[89] On Ki Angel Ling. “Simulating Seismic Wave Propagation in the European Alps with WaveQLab3D and ExaHyPE”. MA thesis. Ludwig-Maximilians-Universität, 2018.

[90] 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.

[91] 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.

[92] EH Madden et al. “Linked 3D modeling of megathrust earthquake-tsunami events: from subduction to tsunami run up”. In: *Geophysical Journal International* (2020).

[93] Masafumi Matsuyama and Hiroyoshi Tanaka. “An experimental study of the highest run-up height in the 1993 Hokkaido Nansei-oki earthquake tsunami”. In: *National Tsunami Hazard Mitigation Program Review and International Tsunami Symposium (ITS)*. 2001, pp. 879–889.

[94] John D McCalpin et al. “Memory bandwidth and machine balance in current high performance computers”. In: *IEEE computer society technical committee on computer architecture (TCCA) newsletter* 2.19–25 (1995).

[95] Oliver Meister. “Sierpinski Curves for Parallel Adaptive Mesh Refinement in Finite Element and Finite Volume Methods”. PhD thesis. Technische Universität München, 2016.

[96] Oliver Meister, Kaveh Rahnema, and Michael Bader. “Parallel memory-efficient adaptive mesh refinement on structured triangular meshes with billions of grid cells”. In: *ACM Transactions on Mathematical Software (TOMS)* 43.3 (2016), pp. 1–27.

[97] Diego Melgar, Amy L Williamson, and E Fernando Salazar-Monroy. "Differences between heterogenous and homogenous slip in regional tsunami hazards modelling". In: *Geophysical Journal International* 219.1 (July 2019), pp. 553–562. DOI: 10.1093/gji/ggz299.

[98] Peter Moczo et al. "Comparison of numerical methods for seismic wave propagation and source dynamics—the SPICE code validation". In: *ESG 2006, Third Intl. Symposium on the Effects of Surface Geology on Seismic Motion*. Vol. 1. LCPC Editions. 2006, pp. 495–504.

[99] Beth Mortimer, William Lake Rees, Paula Koelemeijer, and Tarje Nissen-Meyer. "Classifying elephant behaviour through seismic vibrations". In: *Current Biology* 28.9 (2018), R547–R548. DOI: https://doi.org/10.1016/j.cub.2018.03.062.

[100] S Nielsen, Elena Spagnuolo, Marie Violay, S Smith, Giulio Di Toro, and A Bistacchi. "G: Fracture energy, friction and dissipation in earthquakes". In: *Journal of seismology* 20.4 (2016), pp. 1187–1205.

[101] 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.

[102] Josep de la Puente, Juan Esteban Rodriguez, Marisol Monterrubio-Velasco, Otilio Rojas, and Arnau Folch. "Urgent Supercomputing of Earthquakes: Use Case for Civil Protection". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '20. Geneva, Switzerland: Association for Computing Machinery, 2020. ISBN: 9781450379939. DOI: 10.1145/3394277.3401853.

[103] Leonhard Rannabauer, Michael Dumbser, and Michael Bader. "ADER-DG with a-posteriori finite-volume limiting to simulate tsunamis in a parallel adaptive mesh refinement framework". In: *Computers & Fluids* 173 (2018), pp. 299–306.

[104] Anne Reinarz et al. "ExaHyPE: an engine for parallel dynamically adaptive simulations of wave problems". In: *Computer Physics Communications* 254 (2020), p. 107251.

[105] Gerard R Richter. "An optimal-order error estimate for the discontinuous Galerkin method". In: *Mathematics of Computation* 50.181 (1988), pp. 75–88.

[106] Johann Rudi et al. “An extreme-scale implicit solver for complex PDEs: highly heterogeneous flow in earth’s mantle”. In: *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 2015, pp. 1–12.

[107] Carl Runge. “Über die numerische Auflösung von Differentialgleichungen”. In: *Mathematische Annalen* 46.2 (1895), pp. 167–178.

[108] Tatsuhiko Saito. “Dynamic tsunami generation due to sea-bottom deformation: Analytical representation based on linear potential theory”. In: *Earth, Planets and Space* 65.12 (2013), pp. 1411–1423.

[109] Tatsuhiko Saito. *Tsunami generation and propagation*. Springer, 2019.

[110] 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.

[111] 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.

[112] Philipp Samfass, Jannis Klinkenberg, and Michael Bader. “Hybrid MPI+ OpenMP Reactive Work Stealing in Distributed Memory in the PDE Framework sam (oa)ˆ 2”. In: *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE. 2018, pp. 337–347.

[113] Philipp Samfass, Tobias Weinzierl, Dominic E Charrier, and Michael Bader. “Lightweight task offloading exploiting MPI wait times for parallel adaptive mesh refinement”. In: *Concurrency and Computation: Practice and Experience* 32.24 (2020), e5916.

[114] Kenji Satake. “Advances in earthquake and tsunami sciences and disaster risk reduction since the 2004 Indian ocean tsunami”. In: *Geoscience Letters* 1.1 (2014), pp. 1–13.

[115] Linus Seelinger, Anne Reinarz, Leonhard Rannabauer, Michael Bader, Peter Bastian, and Robert Scheichl. “High performance uncertainty quantification with parallelized multilevel Markov chain Monte Carlo”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2021, pp. 1–15.

[116] Jacopo Selva et al. "Probabilistic tsunami forecasting for early warning". In: *Nature communications* 12.1 (2021), pp. 1–14.

[117] Bruno Seny, Jonathan Lambrechts, Thomas Toulorge, Vincent Legat, and Jean-François Remacle. "An efficient parallel implementation of explicit multirate Runge–Kutta schemes for discontinuous Galerkin computations". In: *Journal of Computational Physics* 256 (2014), pp. 135–160.

[118] Chi-Wang Shu and Stanley Osher. "Efficient implementation of essentially non-oscillatory shock-capturing schemes, II". In: *Upwind and High-Resolution Schemes*. Springer, 1989, pp. 328–374.

[119] Y Tony Song, L-L Fu, Victor Zlotnicki, Chen Ji, Vala Hjorleifsdottir, CK 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.

[120] Holger Stengel, Jan Treibig, Georg Hager, and Gerhard Wellein. "Quantifying performance bottlenecks of stencil computations using the execution-cache-memory model". In: *Proceedings of the 29th ACM on International Conference on Supercomputing*. 2015, pp. 207–216.

[121] Ryütarü Takahashi. "On the seismic sea waves caused by deformation of the sea bottom". In: *Bull. Earthq. Res. Inst., Univ. Tokyo* 20 (1942), pp. 357–400.

[122] Yuichiro Tanioka and Kenji Satake. "Tsunami generation by horizontal displacement of ocean bottom". In: *Geophysical research letters* 23.8 (1996), pp. 861–864.

[123] Maurizio Tavelli, Michael Dumbser, Dominic Etienne Charrier, Leonhard Rannabauer, Tobias Weinzierl, and Michael Bader. "A simple diffuse interface approach on adaptive Cartesian grids for the linear elastic wave equations with complex topography". In: *Journal of Computational Physics* 386 (2019), pp. 158–189.

[124] William Carlisle Thacker. "Some exact solutions to the nonlinear shallow-water wave equationsa". In: *Journal of Fluid Mechanics* 107 (1981), pp. 499–508. DOI: 10.1017/S0022112081001882.

[125] Jeremy Thomas. *https://www.llnl.gov/news/llnl-and-hpe-partner-amd-el-capitan-projected-worlds-fastest-supercomputer*. https://www.llnl.gov/news/llnl-and-hpe-partner-amd-el-capitan-projected-worlds-fastest-supercompute. Accessed: January 9th 2022.

[126] Eleuterio F Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer Science & Business Media, 2013.

[127] Eleuterio F Toro. *Shock-capturing methods for free-surface shallow flows*. Wiley-Blackwell, 2001.

[128] Thomas Ulrich, Alice-Agnes Gabriel, and Elizabeth Madden. "Stress, rigidity and sediment strength control megathrust earthquake and tsunami dynamics". In: (2020).

[129] Thomas Ulrich et al. "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.

[130] Carsten Uphoff. "Flexible model extension and optimisation for earthquake simulations at extreme scales". PhD thesis. Technische Universität München, 2020.

[131] Carsten Uphoff and Michael Bader. "Yet another tensor toolbox for discontinuous Galerkin methods and other applications". In: *ACM Transactions on Mathematical Software (TOMS)* 46.4 (2020), pp. 1–40.

[132] 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: *Proceedings of the international conference for high performance computing, networking, storage and analysis*. 2017, pp. 1–16.

[133] Bram van Leer. "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method". In: *Journal of Computational Physics* 32.1 (1979), pp. 101–136. DOI: https://doi.org/10.1016/0021-9991(79)90145-1.

[134] Iris Van Zelst, Stephanie Wollherr, A-A Gabriel, Elizabeth H Madden, and Ylona van Dinther. "Modeling megathrust earthquakes across scales: One-way coupling from geodynamics and seismic cycles to dynamic rupture". In: *Journal of Geophysical Research: Solid Earth* 124.11 (2019), pp. 11414–11446.

[135] Stefan Vater, Nicole Beisiegel, and Jörn Behrens. "A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: One-dimensional case". In: *Advances in water resources* 85 (2015), pp. 1–13.

[136] Stefan Vater, Nicole Beisiegel, and Jörn Behrens. "A limiter-based well-balanced discontinuous Galerkin method for shallow-water flows with wetting and drying: Triangular grids". In: *International Journal for Numerical Methods in Fluids* 91.8 (2019), pp. 395–418.

[137] Cornelis Boudewijn Vreugdenhil. *Numerical methods for shallow-water flow*. Vol. 13. Springer Science & Business Media, 1994.

[138] T. Weinzierl. "The Peano Software ;Parallel, Automaton-based, Dynamically Adaptive Grid Traversals". In: *ACM Trans. Math. Softw.* 45.2 (2019), 14:1–14:41.

[139] Samuel Williams, Andrew Waterman, and David Patterson. "Roofline: an insightful visual performance model for multicore architectures". In: *Communications of the ACM* 52.4 (2009), pp. 65–76.

[140] Sara Aniko Wirp et al. "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), p. 177.

[141] Yulong Xing and Xiangxiong Zhang. "Positivity-preserving well-balanced discontinuous Galerkin methods for the shallow water equations on unstructured triangular meshes". In: *Journal of Scientific Computing* 57.1 (2013), pp. 19–41.

[142] Yulong Xing, Xiangxiong Zhang, and Chi-Wang Shu. "Positivity-preserving high order well-balanced discontinuous Galerkin methods for the shallow water equations". In: *Advances in Water Resources* 33.12 (2010), pp. 1476–1493. DOI: `https://doi.org/10.1016/j.advwatres.2010.08.005`.

[143] Iris van Zelst, Leonhard Rannabauer, Alice-Agnes Gabriel, and Ylona van Dinther. "Earthquake rupture on multiple splay faults and its effect on tsunamis". In: (2021).

[144] Xiangxiong Zhang and Chi-Wang Shu. "Maximum-principle-satisfying and positivity-preserving high-order schemes for conservation laws: survey and new developments". In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 467.2134 (2011), pp. 2752–2776.

[145] Xi Zhao, Benlong Wang, and Hua Liu. "Characteristics of tsunami motion and energy budget during runup and rundown processes over a plane beach". In: *Physics of Fluids* 24.6 (2012), p. 062107. DOI: `10.1063/1.4729597`.

[146] G. J. van Zwieten, R. F. Hanssen, and M. A. Gutiérrez. “Overview of a range of solution methods for elastic dislocation problems in geophysics”. In: *Journal of Geophysical Research: Solid Earth* 118.4 (2013), pp. 1721–1732. DOI: https://doi.org/10.1029/2012JB009278.