SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementation of Validation Benchmarks for the Elastic Equations in ExaHyPE 2

Tim Solfronk

SCHOOL OF COMPUTATION,
INFORMATION AND TECHNOLOGY

TECHNICAL UNIVERSITY OF MUNICH

Bachelor's Thesis in Informatics

Implementation of Validation Benchmarks for the Elastic Equations in ExaHyPE 2

Implementierung von Validierungsbenchmarks für die elastischen Gleichungen in ExaHyPE 2

Author:	Tim Solfronk
Examiner:	Univ.-Prof. Dr. Bader, Michael
Supervisor:	M.Sc. Marot-Lassauzaie, Marc
Submission Date:	16.02.2026

I confirm that this bachelor's thesis is my own work and I have documented all sources and material used.

Munich, 16.02.2026 Tim Solfronk

Acknowledgments

I would like to express my gratitude to my girlfriend Ina, who has helped me tremendously along the way by giving feedback, providing an outside view of my thesis and supporting me whenever possible. Further, I want to thank my friends and family for always lending me an ear when I needed one.

I am also especially grateful to my supervisor, Marc, for helping me choose the topic, always being patient when I had technical issues, explaining background knowledge whenever I needed it, and, in general, being available when I had questions and guiding me expertly through the project. Lastly, I want to thank Univ.-Prof. Dr. Bader, for giving me the opportunity to work on ExaHyPE 2 and learn more about simulations.

Abstract

ExaHyPE 2 is an exascale simulation engine that solves first-order hyperbolic partial differential equations. One application of ExaHyPE 2 is ExaSeis, which uses ExaHyPE's ADER-DG solver in combination with Riemann solvers to implement the elastic wave equations. In this thesis, multiple earthquake simulation benchmarks were implemented to test ExaSeis. These scenarios are widely known benchmarks used for validation, documented at the SISMOWINE web interface and the SCEC/USGS Spontaneous Rupture Code Verification Project. They include one kinematic point source and multiple dynamic rupture simulations. The focus is on rebuilding previously implemented ExaHyPE 1 simulations and adding more varied scenarios to precisely document the functionality of certain setups, such as non-planar fault geometry, multi-dimensional velocity structures, and locally dependent shear-stress input. Specifically, the successfully implemented benchmarks used for validation are: LOH1, TPV5, TPV6, TPV16, TPV26, TPV28 and TPV34. Furthermore, the thesis explores different ways to optimize simulation results by varying setup parameters, including domain size, cell size, boundary conditions, floating-point precision, and order of the ADER-DG solver.

Contents

1 Introduction

The US National Earthquake Information Center currently reports around 20,000 earthquakes worldwide per year [32]. This translates to about 55 earthquakes per day. While most of these earthquakes are rather small, the vast amount of data still helps us understand why earthquakes occur and how they propagate.

Recently, however, our understanding has likely increased even more through the means of earthquake simulations, especially because we can access data or create specific scenarios that we can't find in real life or are hard to measure. This helps our knowledge tremendously. For example, by testing how inelastic materials affect the energy release of an earthquake [33], experimenting with how long-term stress accumulation relates to instantaneous ruptures [18], or demonstrating how heat generated by frictional slips influences the transition between earthquakes and slow-slip events [35], just to name a few.

Most of the time, computing analytical solutions for earthquakes is highly impractical to impossible. That's why a big effort has been made to implement numerical simulation engines. This thesis focuses on the ExaHyPE 2 simulation framework. While ExaHyPE 2 can operate on many different kinds of wave problems [23], this work focuses specifically on seismic waves and verifying the implementation of the elastic wave equations in the ExaHyPE application ExaSeis. The results of the earthquake simulations are then validated by comparing them with solutions from other simulation codes uploaded to the SISMOWINE web interface [27] and the Statewide California Earthquake Center (SCEC)/United States Geological Survey (USGS) Spontaneous Rupture Code Verification Project [15]. Both are projects that provide different benchmark scenario descriptions and collect numerical solutions from multiple simulation engines to compare the results of the scenarios. In this thesis, a kinematic point-source simulation is used to verify the propagation of seismic waves in ExaSeis, and after that, multiple dynamic rupture simulations are implemented and optimized to test specific dynamic rupture features, such as slip-weakening friction and more complex fault geometries.

This work builds on the dissertation by Rannabauer, in which the ExaSeis framework for ExaHyPE 1 is introduced and functionalities, such as perfectly matched layers and ExaHyPE's Curvi tool, are implemented and explained [22]. As part of his paper, Rannabauer implements four dynamic rupture benchmarks to verify the implemen-

tation in ExaHyPE 1. Three of these four benchmarks have been reimplemented for ExaHyPE 2 and are discussed in this thesis. Furthermore, three additional dynamic rupture benchmarks were implemented in this thesis.

Another similar work is the research by Fülöp et al. [14] in which they test the dynamic rupture scenario TPV5 [7] for three different simulation codes, namely SeisSol, 3DEC and FLAC3D, to compare them by price, speed and accuracy among other things.

2 Background

2.1 ExaHyPE 2 Overview

ExaHyPE 2 is a rework of the original ExaHyPE simulation engine [23], which solves hyperbolic partial differential equations (PDEs). It is built on top of Peano, an open source framework for scientific simulations that handles dynamically adaptive Cartesian meshes. Furthermore, Peano implements grid traversal, memory management and load balancing [30]. ExaHyPE 2 uses these functionalities and defines the data inside and on the faces of the cells in Peano's Cartesian grid. It also supplies the solver, Arbitrary high-order DERivative + Discontinuous Galerkin (ADER-DG), for example, and adds the operations to the grid traversal. ExaHyPE 2 itself was implemented for wave problems in general and can therefore solve a variety of different PDEs, as long as they follow the form [20, 23]:

$$\frac{\partial Q}{\partial t} + \nabla \cdot F(Q, \nabla Q) + B(Q) \cdot \nabla Q = S(Q) + \sum_{i=1}^{n_{ps}} \delta_i. \tag{2.1}$$

Here $Q : \Omega \subset \mathbb{R}^d \mapsto \mathbb{R}^v$ is the state vector with v elements, with $\Omega \subset \mathbb{R}^d$ being the computational domain, $F(Q, \nabla Q))$ is the flux tensor and $B(Q)$ represents the Partial differential Equation (PDE)'s non-conservative part. Lastly, $S(Q)$ is the algebraic source term and δ_i are the given n_{ps} point sources [23].

To use ExaHyPE 2, one needs to implement all parts of the PDE (non-conservative product, flux, etc.) separately in C++ and can then use the ExaHyPE 2 Python API to link the PDE parts together and define the rest of the problem parameters. This includes the domain size, its variables and the simulation's end time. There are multiple applications that follow exactly these steps to implement PDE's like the shallow water equations or the elastic wave equations in ExaHyPE 2. This thesis focuses on the latter, which were implemented under the name ExaSeis and can be used to simulate numerous dynamic rupture scenarios. A highly summarized overview of the overall framework can be found in Figure 2.1. To learn more about ExaHyPE, one can either consult the publication from Reinarz et al. [23] or the official documentation [20].

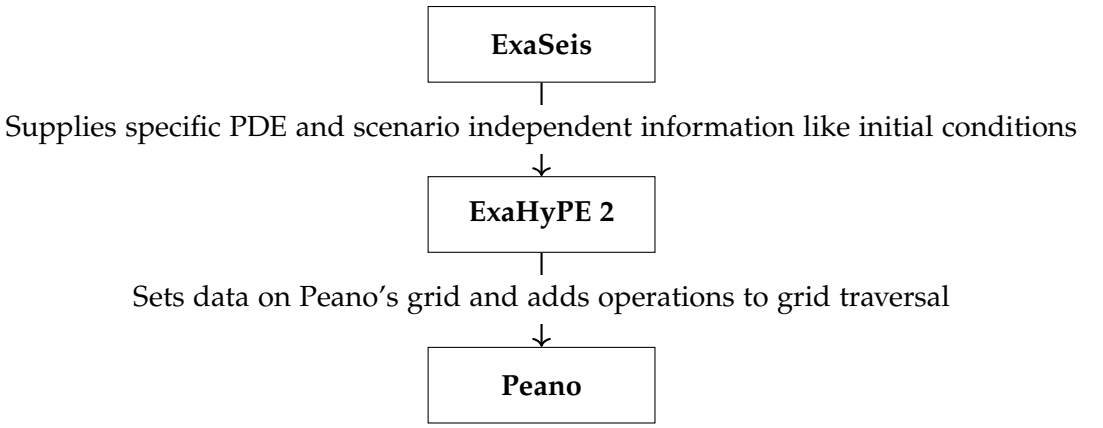


Figure 2.1: Overview of how ExaSeis, ExaHyPE 2 and Peano build on each other

2.2 Elastic Wave Equations

This section explains the one-dimensional linear elastic wave equations (one-dimensional waves in a three-dimensional space) and, afterward, briefly shows the equations for three-dimensional waves. It closely follows a work by LeVeque et al., where a more detailed introduction and explanation can be found [17]. After explaining the wave physics, the ADER-DG (Arbitrary high-order DERivative + Discontinuous Galerkin) and Riemann solver, which are used to solve the equations in ExaHyPE, are quickly introduced.

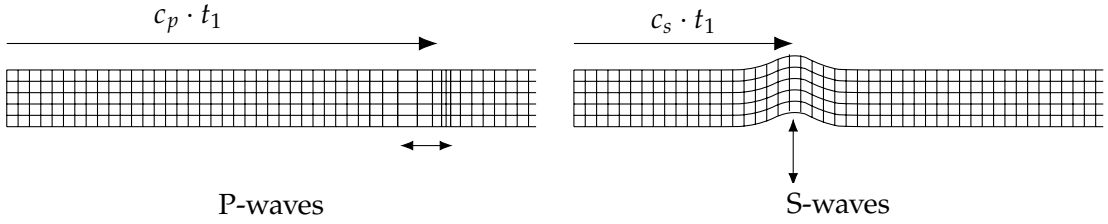


Figure 2.2: Illustration of P-waves and S-waves at timepoint $t_1 > 0$ after an earthquake, the hypocenter is at the left side of the respective grids

From the center of an earthquake, called the *hypocenter*, two types of waves originate in a solid. The first type of waves are primary waves (also called pressure waves or P-waves in short). These waves are essentially identical to sound waves, meaning they are waves that move the medium they are traveling through back in forth in the same direction as the wave is moving. Unlike in acoustics through gas or liquid, there is a second type of waves called S-waves in solids (also called shear waves or

secondary waves). Instead of moving the medium back and forth in the wave direction, these waves disturb the medium in an orthogonal direction. For three-dimensional waves, this leads to two S-waves, because there are two orthogonal directions to the wave direction. In our simplified case, we are dealing with one-dimensional waves in a three-dimensional domain. This means we have waves that only move in the x direction, leading all physical quantities to vary only with x. From this, it follows that both wave types extend infinitely far in the y (upward) and z (normal to the page) direction. Furthermore, we only have one S-wave that disturbs the medium in the y-direction. This approach was also copied from the work by LeVeque et al. [17]. As illustrated in Figure 2.2, S-waves tend to be much slower than P-waves.

First, an intuitive explanation of why the waves move: In Figure 2.2, the P-waves are shown to compress the medium in the x-direction over a small region in x. This compression increases the stress (force per unit area) in this region, which, in turn, results in an acceleration in the x-direction, moving the wave. This leads to the wave speed c_p. S-waves, on the other hand, disturb the medium in an orthogonal direction, leading to a shear displacement. Because of the strong chemical bonds in solids, however, this displacement is only allowed up to a point before the bonds' restoring force (stress) is strong enough to push the medium back to its original form. This way, the wave is pushed forward with speed c_s. It is important to note that if the displacement doesn't stay small, the relations in the elastic equations become non-linear, and at some point, when the deformation is so large that enough bonds in the medium are irreparably broken, the material may fail (snap or break), causing plastic deformation. In that case, the material does not return to its original resting position and one has to consult the theory of *plasticity* to describe its behavior.

Going back to the linear elastic case, we now show an explanation of the waves based on physical equations: We can define the displacement vector of a location (x, y) to be:

$$u(x,y,t) = \begin{pmatrix} u_x(x,y,t) \\ u_y(x,y,t) \end{pmatrix} = \begin{pmatrix} X(x,y,t) \\ Y(x,y,t) \end{pmatrix} - \begin{pmatrix} x \\ y \end{pmatrix}, \tag{2.2}$$

with $(X(x,y,t), Y(x,y,t))$ being the location of the material that was originally at (x, y) after time t . Based on that, we define the velocity vector v as the time derivative of the displacement,

$$v(x,y,t) = \begin{pmatrix} v_x(x,y,t) \\ v_y(x,y,t) \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial t} u_x(x,y,t) \\ \frac{\partial}{\partial t} u_y(x,y,t) \end{pmatrix}. \tag{2.3}$$

As previously mentioned when explaining the wave types, displacement can compress or stretch a material. This deformation is called *strain*. There are also displacements that do not generate strain, such as translation of the material as a whole.

Therefore, if the displacement was identical at all points throughout the medium, the strain would be zero. That's why the strain is only dependent on the spatial gradient of the displacement:

$$\nabla u(x,y,t) = \begin{pmatrix} \frac{\partial}{\partial x}u_x & \frac{\partial}{\partial y}u_x \\ \frac{\partial}{\partial x}u_y & \frac{\partial}{\partial y}u_y \end{pmatrix} = \begin{pmatrix} \frac{\partial}{\partial x}X - 1 & \frac{\partial}{\partial y}X \\ \frac{\partial}{\partial x}Y & \frac{\partial}{\partial y}Y - 1 \end{pmatrix}. \tag{2.4}$$

Furthermore, there is no deformation/strain if the displacement describes only the rigid rotation of the whole medium. As a consequence of that, the strain matrix ϵ is the displacement gradient ∇u minus the rotation matrix Ω,

$$\begin{aligned} \epsilon &= \begin{pmatrix} \epsilon_{xx} & \epsilon_{xy} \\ \epsilon_{xy} & \epsilon_{yy} \end{pmatrix} = \nabla u - \Omega \\ &= \begin{pmatrix} \frac{\partial}{\partial x}u_x & \frac{\partial}{\partial y}u_x \\ \frac{\partial}{\partial x}u_y & \frac{\partial}{\partial y}u_y \end{pmatrix} - \begin{pmatrix} 0 & \frac{1}{2}(\frac{\partial}{\partial y}u_x - \frac{\partial}{\partial x}u_y) \\ -\frac{1}{2}(\frac{\partial}{\partial y}u_x - \frac{\partial}{\partial x}u_y) & 0 \end{pmatrix} \\ &= \begin{pmatrix} \frac{\partial}{\partial x}u_x & \frac{1}{2}(\frac{\partial}{\partial y}u_x + \frac{\partial}{\partial x}u_y) \\ \frac{1}{2}(\frac{\partial}{\partial y}u_x + \frac{\partial}{\partial x}u_y) & \frac{\partial}{\partial y}u_y \end{pmatrix}. \end{aligned} \tag{2.5}$$

For our one-dimensional waves, $\epsilon_{yy} = 0$. The now described strain typically results in a restoring force called the *stress* σ. Because we are looking at solely small displacements, we can relate the stress in each direction linearly to the strain with the constitutive equations:

$$\begin{aligned} \sigma_{xx} &= (\lambda + 2\mu)\epsilon_{xx} \quad \text{with } \lambda + 2\mu > 0, \\ \sigma_{xy} &= 2\mu\ \epsilon_{xy} \qquad\qquad\qquad\quad \mu > 0. \end{aligned} \tag{2.6}$$

λ and μ are the Lamé parameters, which essentially describe the material properties. We can now use all of the equations above to write the linear elastic equations for one-dimensional waves. Assuming the density of the material $\rho > 0$, we can form the following equations for P-Waves:

$$\begin{aligned} \frac{\partial}{\partial t}\epsilon_{xx} - \frac{\partial}{\partial x}v_x &= 0, \\ \frac{\partial}{\partial t}\rho v_x - \frac{\partial}{\partial x}\sigma_{xx} &= 0. \end{aligned} \tag{2.7}$$

The second equation of Equation 2.7 uses Newton's second law since $\frac{\partial}{\partial t}v_x = a_x$. Using $\frac{\partial}{\partial y}v_x = 0$, similar equations can be constructed for the S-waves, by relating the shear-stress σ_{xy} and v_y:

$$\begin{aligned} \frac{\partial}{\partial t}\epsilon_{xy} - \frac{1}{2}\cdot\frac{\partial}{\partial x}v_y = 0, \\ \frac{\partial}{\partial t}\rho v_y - \frac{\partial}{\partial x}\sigma_{xy} = 0. \end{aligned} \tag{2.8}$$

Finally, we can use the constitutive equations from Equation 2.6 to replace ϵ_{xx} and ϵ_{xy} in Equation 2.7 and Equation 2.8 respectively, to create one system of equations for the linear elastic one-dimensional wave equations (one-dimensional waves in a three-dimensional domain):

$$\begin{aligned} \frac{\partial}{\partial t}\sigma_{xx} - (\lambda+2\mu)\frac{\partial}{\partial x}v_x = 0, \\ \frac{\partial}{\partial t}\rho v_x - \frac{\partial}{\partial x}\sigma_{xx} = 0, \\ \frac{\partial}{\partial t}\sigma_{xy} - \mu\frac{\partial}{\partial x}v_y = 0, \\ \frac{\partial}{\partial t}\rho v_y - \frac{\partial}{\partial x}\sigma_{xy} = 0. \end{aligned} \tag{2.9}$$

Similar to how the one-dimensional wave equations in Equation 2.9 were derived, one can also derive the linear elastic wave equations for three-dimensional waves, but now the equations for S- and P-waves are coupled:

$$\begin{aligned} \frac{\partial}{\partial t}\sigma_{xx} - (\lambda+2\mu)\frac{\partial}{\partial x}v_x - \lambda(\frac{\partial}{\partial y}v_y + \frac{\partial}{\partial z}v_z) = 0, \\ \frac{\partial}{\partial t}\sigma_{yy} - (\lambda+2\mu)\frac{\partial}{\partial y}v_y - \lambda(\frac{\partial}{\partial x}v_x + \frac{\partial}{\partial z}v_z) = 0, \\ \frac{\partial}{\partial t}\sigma_{zz} - (\lambda+2\mu)\frac{\partial}{\partial z}v_z - \lambda(\frac{\partial}{\partial x}v_x + \frac{\partial}{\partial y}v_y) = 0, \\ \frac{\partial}{\partial t}\sigma_{xy} - \mu(\frac{\partial}{\partial y}v_x + \frac{\partial}{\partial x}v_y) = 0, \\ \frac{\partial}{\partial t}\sigma_{xz} - \mu(\frac{\partial}{\partial z}v_x + \frac{\partial}{\partial x}v_z) = 0, \\ \frac{\partial}{\partial t}\sigma_{yz} - \mu(\frac{\partial}{\partial z}v_y + \frac{\partial}{\partial y}v_z) = 0, \\ \frac{\partial}{\partial t}\rho v_x - \frac{\partial}{\partial x}\sigma_{xx} - \frac{\partial}{\partial y}\sigma_{xy} - \frac{\partial}{\partial z}\sigma_{xz} = 0, \\ \frac{\partial}{\partial t}\rho v_y - \frac{\partial}{\partial x}\sigma_{xy} - \frac{\partial}{\partial y}\sigma_{yy} - \frac{\partial}{\partial z}\sigma_{yz} = 0, \\ \frac{\partial}{\partial t}\rho v_z - \frac{\partial}{\partial x}\sigma_{xz} - \frac{\partial}{\partial y}\sigma_{yz} - \frac{\partial}{\partial z}\sigma_{zz} = 0, \end{aligned} \tag{2.10}$$

where

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{pmatrix} \tag{2.11}$$

defines the symmetric stress tensor and $v = \begin{pmatrix} v_x & v_y & v_z \end{pmatrix}^T$ describes the particle velocity vector in the 3 dimensions x, y and z. For a detailed introduction to the linear elastic wave equations in three dimensions, the work of LeVeque et al. [17] is again recommended.

To simulate the elastic equations from Equation 2.10, one can rewrite them into the form of Equation 2.1, by defining the state vector

$$Q = \begin{pmatrix} \sigma_{xx} & \sigma_{yy} & \sigma_{zz} & \sigma_{xy} & \sigma_{xz} & \sigma_{yz} & \rho v_x & \rho v_y & \rho v_z \end{pmatrix}^T \tag{2.12}$$

as outlined by Reinarz et al. [23].

To solve the resulting PDE, ExaSeis uses an ADER-DG solver [11]. ADER-DG works by splitting the domain into finite elements (cells) and performing two main steps per timestep per cell: a predictor step and a corrector step [19]. In the predictor step, the next values in each cell are predicted by constructing a polynomial of the solution based on the weak formulation of the PDE and the current state variables. The now-predicted cell-local space-time solutions are then used at each element face in step 2 to incorporate the effects of neighboring cells into the final cell solution, ensuring better global results. This correction step is performed using a Riemann solver on the space-discretized PDE obtained from the Galerkin approach. For a more detailed explanation, the works by Zanotti et al. [34] and Duru et al. [11] are recommended. The Riemann solver is also where friction models for fault simulation are implemented, as we assume/always try to place the fault on cell interfaces.

2.3 Curvi

So far, the simulation is only considering a Cartesian grid. This means the grid is constructed from cubes (or squares in 2D) aligned with the Cartesian axes. In earthquake simulations, however, this does not always suffice, as non-planar geometry or geometry that is not aligned with the Cartesian axes, such as a leaning fault or non-planar topography, is often modeled. Because of that, ExaHyPE supplies the Curvi tool, which allows users to map a curvilinear mesh onto the Cartesian mesh for computation. A curvilinear mesh is also a hexahedral mesh (a mesh that is composed of elements with six faces), but the vertices do not have to be on a uniform rectilinear grid and the faces need not be planar. Therefore, fault and material interfaces can also

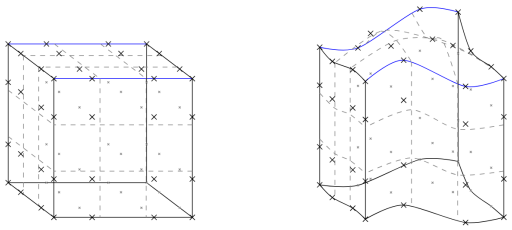


Figure 2.3: Comparison of a Cartesian cell (left) to a curvilinear cell (right) as shown in the work by Rannabauer [22]

be non-planar. This way, the mesh can follow the topography more closely and can ensure that even non-planar faults are always on the interface of two neighboring cells. An example of a curvilinear element compared to a Cartesian element is visualized in Figure 2.3.

To ensure this approach is valid, the transformation from Cartesian to curvilinear elements and vice versa must be bijective, and the neighbor relations of all elements must remain exactly the same. That means that if cell a has cell b as a neighbor in the curvilinear mesh, it also has cell b as a neighbor in the Cartesian mesh. It follows that if cell a does not have cell c as a neighbor in the curvilinear mesh, it will also not have it as a neighbor in the Cartesian mesh [22].

This transformation from curvilinear to Cartesian mesh and back obviously affects how the elastic equations from Equation 2.10 must be implemented, since we are still computing on a Cartesian grid while solving for a curvilinear one. This entails adding the metric derivative and its determinant of the element-wise transformation to the equation [22]. How exactly this is done is outside of the scope of this thesis, but can be reviewed in the dissertation by Rannabauer [22] or in the work by Duru et al. [11].

2.4 Perfectly Matched Layers (PMLs)

One problem that has not yet been explained is the boundary behavior of our domain. At the top domain boundary, we set a free-surface boundary condition, which means that normal stress and the respective shear stress (in this case, vertical) are kept at zero, as there is no material that pushes back. This simulates the Earth's surface (approximately, since we treat the air as a vacuum). On the other boundaries of the domain, however, we must simulate that the material extends much further to better approximate reality. This introduces a problem, as these types of boundaries lead to wave reflection with numerical methods.

One could just simply increase the domain up to a point where no waves reach the

boundaries of the domain. However, this would have to be done either by making cells coarser towards the boundary, which could cause wave reflections because the accuracy of the wave speeds would differ between elements [22], or by keeping the cell size the same everywhere, which would dramatically increase the number of cells and therefore the computing time. Neither outcome would be ideal. Historically, the problem was solved by keeping the domain size normal, but adding absorbing boundary conditions [12]. While these are effective in decreasing wave reflection on domain boundaries, they do not completely remove the reflections, also making them an imperfect solution.

Perfectly matched layers (PMLs) are another boundary approach. To simulate the boundaries, it uses *complex coordinate stretching* [2]. In the following, complex coordinate stretching is briefly explained based on the example given by Rannabauer [22].

We assume the solution of the PDE $w(x,t)$ is a simple 1D wave:

$$w(x,t) = u \cdot e^{i(kx-\omega t)}. \tag{2.13}$$

Here, k is the wave vector (in this case, a scalar), which defines the direction of the wave and its spatial frequency, ω is the angular frequency of the wave, which is responsible for how fast the wave travels in time t, and u is the amplitude of the wave. We can plot this wave by using Euler's formula $e^{i\theta} = \cos\theta + i\sin\theta$. An example of such a wave can be seen in Figure 2.4.

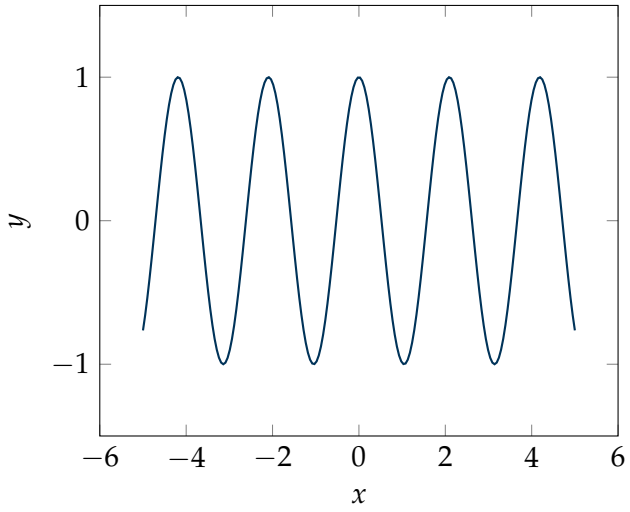


Figure 2.4: Real part of a 1D wave with $k = 3$, $\omega = 1$, $u = 1$ and $t = 0$

Next, let us assume, we want the waves to fade away for $x \geq 0$. This is achieved by extending the x coordinate into the complex domain:

$$\tilde{x} = x + ix \cdot H(x), \tag{2.14}$$

with H being the Heaviside step function

$$H(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{else} \end{cases} \tag{2.15}$$

Using these coordinates stretched into the complex domain, we can now rearrange the 1D wave equation:

$$\begin{aligned} w(\tilde{x}, t) &= u \cdot e^{i(k\tilde{x} - \omega t)} \\ &= u \cdot e^{i(kx + ik \cdot H(x) - \omega t)} \\ &= u \cdot e^{i(kx - \omega t)} \cdot e^{-kx \cdot H(x)}. \end{aligned} \tag{2.16}$$

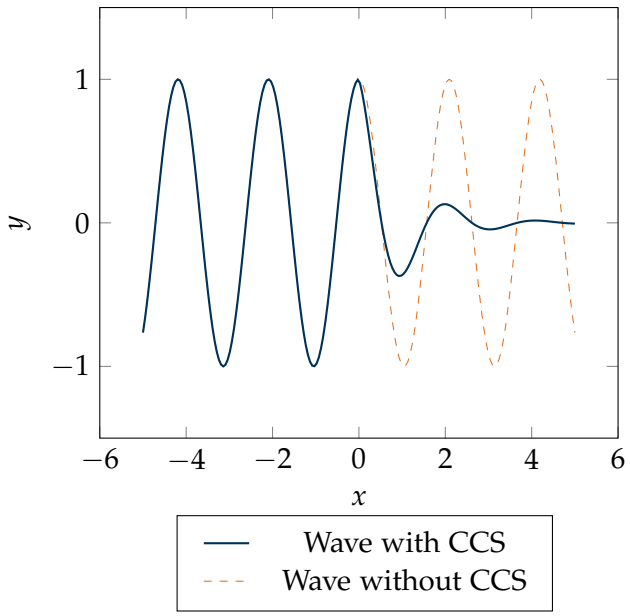


Figure 2.5: Comparison of real part of a 1D wave with $k = 3$, $\omega = 1$, $u = 1$ and $t = 0$ with and without complex coordinate stretching (CCS)

In Figure 2.5, the example wave from Figure 2.4 is plotted with the complex coordinate stretching included in Equation 2.16. We can see that for $x < 0$, the wave remains unchanged, but for $x > 0$, it decays exponentially. On the interface, the waves match exactly. This same mechanism can be used on more complex waves as well as solutions composed of multiple waves [22]. In ExaSeis, it is used by adding such a perfectly

matched layer around the domain to make the waves decay at the domain edges and avoid wave reflections. How this scheme is applied to the hyperbolic PDE of the elastic wave equations is beyond the scope of this thesis. For a brief explanation, the work by Rannabauer [22] is recommended, and for a more detailed explanation, the work by Duru et al. [10].

2.5 Seismological Terminology

When discussing earthquake simulations, we are most often dealing with *faults*. A fault is an often planar-like fracture in the Earth's crust, commonly at the boundary of two different tectonic plates. It is at these fracture zones that, if enough stress is built up, the rocks on each side slip past each other, either slowly, causing *creep*, or suddenly and rapidly, causing an *earthquake* [31]. Faults are often near vertical, but can essentially have any angle with respect to the surface. The downward direction of the fault is called *dip* and the horizontal direction of the fault is called *strike*. Following this, if the main direction of slip is horizontal, the fault is called a strike-slip fault, and otherwise a dip-slip fault. When standing near a fault, one distinguishes the sides of the fault as *near* and *far side*, with the far side being the side that is on the other side of the fault, and the near side being the closer one. This enables further classification of strike-slip faults into left-lateral and right-lateral strike-slip faults by describing the displacement direction on the far side of the fault (either to the right or to the left, when looking from the near side) [31].

The simulation of frictional slip on faults, coupled with wave propagation through elastic equations, is called dynamic rupture. A fault is prevented from slipping through two different means. The first is frictional cohesion c, which is continuously present to counteract slip and is independent of the stress acting on the fault. The second is the frictional resistance $\sigma_n \cdot f(s)$, which is composed of the frictional coefficient $f(s)$, where s (in the case of ExaSeis) is the magnitude of the slip so far, and the normal stress compressing the fault from both sides σ_n. These two terms are then added together, resulting in the fault strength

$$\tau_{str} = c + |\sigma_n \cdot f(s)|. \tag{2.17}$$

Only when the magnitude of the shear stress vector (also called *shear traction*) is bigger than τ_{str}, is the fault allowed to slip. If the shear traction is smaller than τ_{str}, the material stays still. Slipping is therefore innately dependent on the friction model. While there are different friction models $f(s)$ to simulate faults, all simulations covered in this thesis use a linear slip weakening friction model, like in the ExaSeis implementation of ExaHyPE 1 [22]. This simulates the idea that an increase in slip weakens fault friction,

which gives us linear slip weakening as introduced by Ida [16]:

$$f(s) = \mu_s - (\mu_s - \mu_d)\frac{min(s, d_c)}{d_c}. \tag{2.18}$$

Several new parameters are used here:

- μ_s, the static coefficient of friction,
- μ_d, the dynamic coefficient of friction (smaller than μ_s),
- d_c, the critical distance of slip-weakening.

The simulation starts with $f(s) = \mu_s$ (since slip s is initially zero), which then linearly decreases with slip to the dynamic coefficient of friction μ_d. This continues until the slip reaches the distance d_c, after which the coefficient of friction stays at μ_d.

To validate the dynamic rupture results, one places multiple *receivers* throughout the domain, which record the velocity, displacement, and stresses at the specified points during the entire simulation duration. This is similar to how seismographs record the ground motion during earthquakes in real life. These receivers are called *off-fault receivers*. Directly on a fault, something similar is done, but instead of recording velocity and displacement, one measures *slip-rate* and *slip*. Slip is usually defined as $u^+ - u^-$, where u^+ is the displacement on the side in the positive coordinate direction (in our case, this will always be the far side) and u^- the displacement on the other side of the fault [7]. Slip-rate is calculated the same way, but using velocities. We call these receivers *on-fault receivers*. Because ExaSeis does not currently support on-fault receivers natively, the measurements in this thesis were obtained by placing two off-fault receivers very close to the fault ($\sim$ 10 m away), one on the near side and one on the far side, and manually subtracting their displacement and velocity results.

3 Validation Benchmarks

3.1 Kinematic Point Sources

Before validating dynamic rupture scenarios, we first test the implementation of the elastic waves in a simplified way. Instead of including a fault with friction parameters, we initiate the waves with a kinematic point source. A kinematic point source represents a fault that is so small that it can be collapsed to a single point during computation [22] The implementation of kinematic point sources is done by adding body forces (given in N/m^3) to the right part of the equations in Equation 2.10, as already hinted at in Equation 2.1 [22].

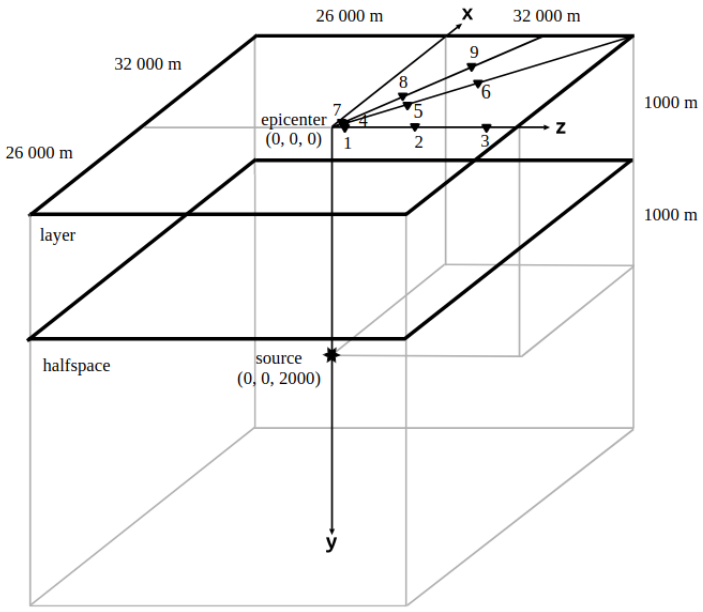


Figure 3.1: Diagram of LOH1 by the SISMOWINE web interface [27] with adapted coordinate system, receivers are marked with ▼

In ExaSeis, the kinematic point source scenario *Layer over half-space* (LOH1) [27] was implemented for validation prior to this thesis. The LOH1 benchmark lets us test

two things at once. First, whether wave propagation works correctly with ExaSeis's implementation of the elastic waves. Second, whether this wave propagation also works in an inhomogeneous domain, a domain composed of different materials. The domain in LOH1 consists of two distinct materials: a main material and a top-layer material. The top-layer material is less dense and has higher wave speeds than the main material. To model LOH1 in ExaSeis, a left-handed coordinate system was used in which the y-direction is pointing down. This leads to the following equations for the material parameters:

$$\rho = \begin{cases} 2.6\ \frac{t}{m} & \text{if } y > 1.0 \\ 2.7\ \frac{t}{m} & \text{else} \end{cases}, \quad c_p = \begin{cases} 4.0\ \frac{km}{s} & \text{if } y > 1.0 \\ 6.0\ \frac{km}{s} & \text{else} \end{cases}, \quad c_s = \begin{cases} 2.0\ \frac{km}{s} & \text{if } y > 1.0 \\ 3.464\ \frac{km}{s} & \text{else} \end{cases}.$$

LOH1 uses exactly one kinematic point source located at $(x_s, y_s, z_s) = (0, 2, 0)$ km. The moment-rate time history $f(t)$ originating from this point source is described as [27]:

$$f(t) = M_{xz}(\frac{t}{T^2}e^{-\frac{t}{T}}). \tag{3.1}$$

Here, M_{xz} is the shear component xz of the moment tensor, with $M_{xz} = 10^{18}$ Nm, and $T = 0.1$ s is the point of highest rate of the source. All other components of the moment tensor are set to zero.

The results of LOH1 discussed here were run with a domain of $[-2.025, 10.125] \times [0.0, 12.15] \times [-2.025, 10.125]$, with 81 cells in each direction, polynomial order of 5 and floating point precision of 32 bits, using PML at the domain edges. To verify the ExaSeis results, we compare them to a semi-analytic solution supplied by the SISMOWINE web interface [27]. As an example, the comparison of the receiver at $(x, y, z) = (0.577, 0.0, 0.384)$ km is shown in Figure 3.2. The noticeable oscillation before the spike at approximately $t_s = 0.8$ s is characteristic of the polynomial ADER-DG method and is also evident in other ADER-DG solutions at the SISMOWINE web interface (e.g., the solution by the simulation code SeisSol [26]). Otherwise, the results match SISMOWINE's solution fairly well.

3.2 TPV5: Planar Fault

After confirming wave propagation works sufficiently well with kinematic point sources, we can start validating the results of dynamic rupture scenarios. This is done by implementing several simulations proposed by the SCEC Spontaneous Rupture Code Verification Project [15]. These benchmarks start as very simple planar fault scenarios

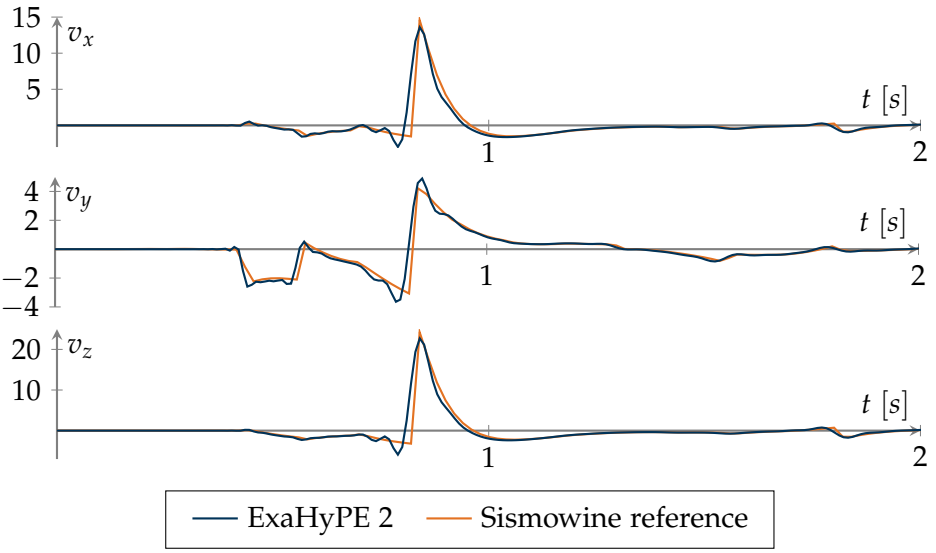


Figure 3.2: Loh1: Velocity vector $v = (v_x, v_y, v_z)$ at receiver $(x, y, z) = (0.577, 0.0, 0.384)$ km compared to semi-analytic Sismowine reference [27] (simulated with domain side length 12.15 km, 81^3 cells, order 5, 32-bit and Perfectly matched layer (PML) enabled)

in a homogeneous halfspace (halfspace = fault reaches the surface) and gradually increase in difficulty by adding more complex fault geometry and input parameters. In addition to the benchmark descriptions, the project also supplies reference solutions from other simulation codes. Henceforth, the results will be compared to two of these solutions, namely the low-order finite element code FaultMod [1] and the high-order finite difference code WaveQLab [9], where available. These are the same reference solutions that ExaHyPE 1 was validated against [22]. To streamline the comparison process, multiple Python tools were implemented to convert reference solutions into the same format as ExaHyPE and display them together using the *Matplotlib* Python library.

The first and simplest dynamic rupture benchmark in this thesis is TPV5. TPV5 models a vertical strike-slip fault in a homogeneous halfspace with mostly homogeneous initial conditions. Only the horizontal shear stress is varied in three patches on the fault [7]. The fault is a rectangular area that spans 30 km in length and 15 km in depth. The static and dynamic coefficients for the linear slip-weakening friction model are $\mu_s = 0.677$ and $\mu_d = 0.525$ (on the fault). The slip-weakening critical distance d_c is 0.4 m. As the material is homogeneous, it has the same material properties throughout the domain:

$$\rho = 2.670 \ \frac{t}{m}, \qquad c_p = 6.0 \ \frac{km}{s}, \qquad c_s = 3.464 \ \frac{km}{s}.$$

Regarding the initial stress, only the normal and horizontal shear stresses are non-zero. To describe them, a left-handed coordinate system centered on the fault's midpoint at the Earth's surface is introduced. Here, the positive y-axis is pointing down, representing the dip, and the positive x-axis is pointing towards the far side of the fault. The initial normal stress σ_{xx} is set to 120 MPa of compression, and the initial horizontal shear stress is defined as:

$$\sigma_{xz} = \begin{cases} 78.0 \text{ MPa} & \text{if } (y,z) \in [6.0, 9.0] \times [-9.0, -6.0] \\ 81.6 \text{ MPa} & \text{if } (y,z) \in [6.0, 9.0] \times [-1.5, 1.5] \\ 62.0 \text{ MPa} & \text{if } (y,z) \in [6.0, 9.0] \times [6.0, 9.0] \\ 70.0 \text{ MPa} & \text{else} \end{cases} \tag{3.2}$$

Around the fault is a *strength barrier*, which ensures that there is no movement outside of the fault. For TPV5, this is achieved by setting the static coefficient of friction outside of the fault very high, in this case $\mu_s = 10000$. ExaSeis, however, implements the strength barrier by increasing the frictional cohesion by 10^{10} in the Riemann solver. This has essentially the same effect and the results did not differ in tests with an adjusted Riemann solver that only uses μ_s for the strength barrier. A rough overview of the benchmark is shown in Figure 3.3. This benchmark is deliberately very simple, as it lets us validate the overall dynamic rupture implementation.

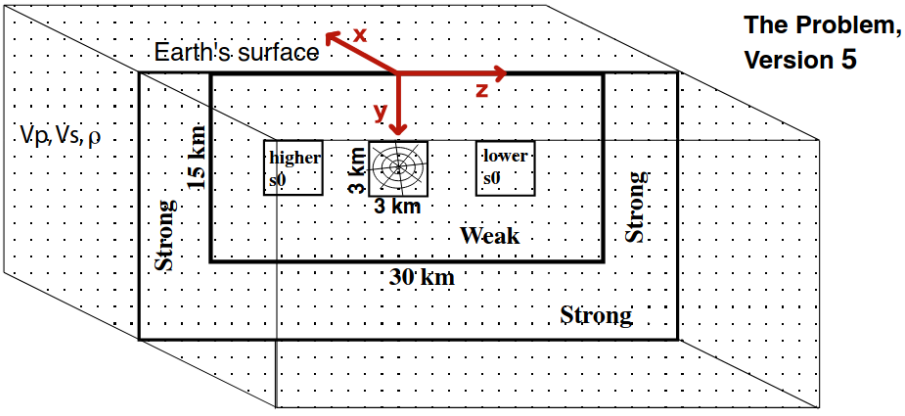


Figure 3.3: Diagram of TPV5 as shown on the official SCEC Website [7] with added coordinate system

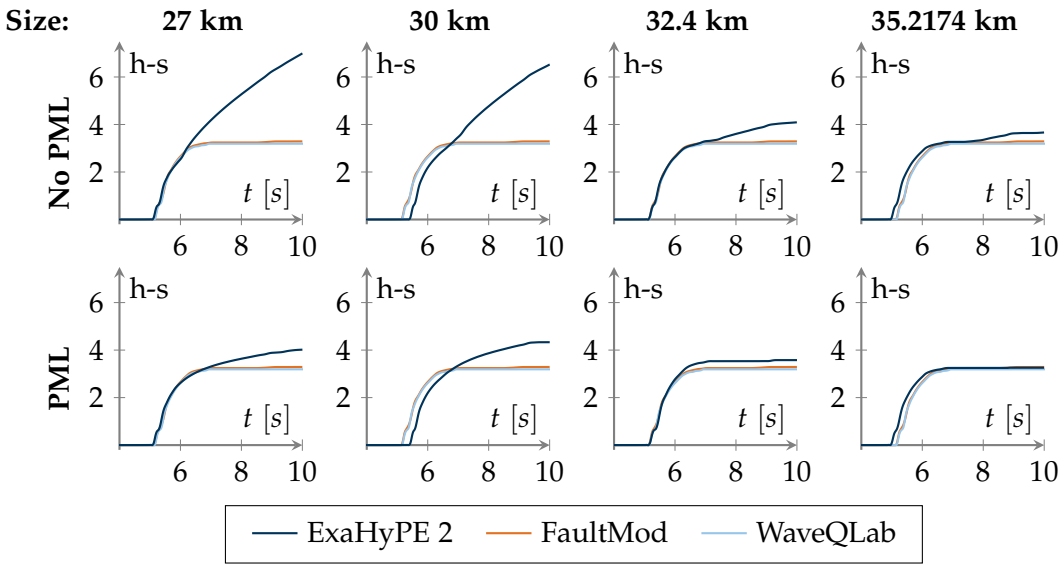


Figure 3.4: TPV5: Comparison of horizontal slip (h-s) measured at on-fault receiver (dip: 0 km, strike: -12 km) for different domain sizes, with and without PML (simulated with 27^3 cells, order 5 and 32-bit)

TPV5 was already implemented with ExaSeis prior to this thesis and the implementation showed somewhat similar results to the reference solutions. However, in later timesteps, the solutions diverged significantly, particularly without PML. This was the case because the simulated domain did not adequately account for the strength barrier. The domain was confined to $[-13.5, 13.5] \times [0.0, 27.0] \times [-13.5, 13.5]$ and was therefore too small to include it in the horizontal direction. After this realization, multiple domain-size tests were conducted to determine how much of the strength barrier should be included in the domain. The tests were done on a mesh with 27 cells in all dimensions. The first test size was the existing size, with the domain spanning 27 km in each direction. The second used 30 km in each direction, including the entire fault, but not the strength barrier. The third test included the entire fault in the domain with one cell of strength barrier on both sides of the fault (32.4 km), and the last test had two cells of strength barrier at the domain edges in the horizontal direction (35.2174 km). All tests were run with polynomial order of 5 and a 32-bit floating-point precision.

In Figure 3.4, all domain-size solutions are compared to the solutions by FaultMod and WaveQLab. We can see that including even just one cell of strength barrier on each side (32.4 km) already considerably improves the results. Furthermore, another positive effect of PML in fault simulations is clearly visible, as even in domains without the strength barrier, the solutions diverge much less. On closer inspection, one can notice a

slight delay or head start in some solution. This may be due to the growing cell size, which might have given the cells an unfavorable size for simulating the different wave speeds. As further inspected in section 3.5, this unwanted effect can be minimized by subdividing the mesh into more cells or using a higher polynomial order.

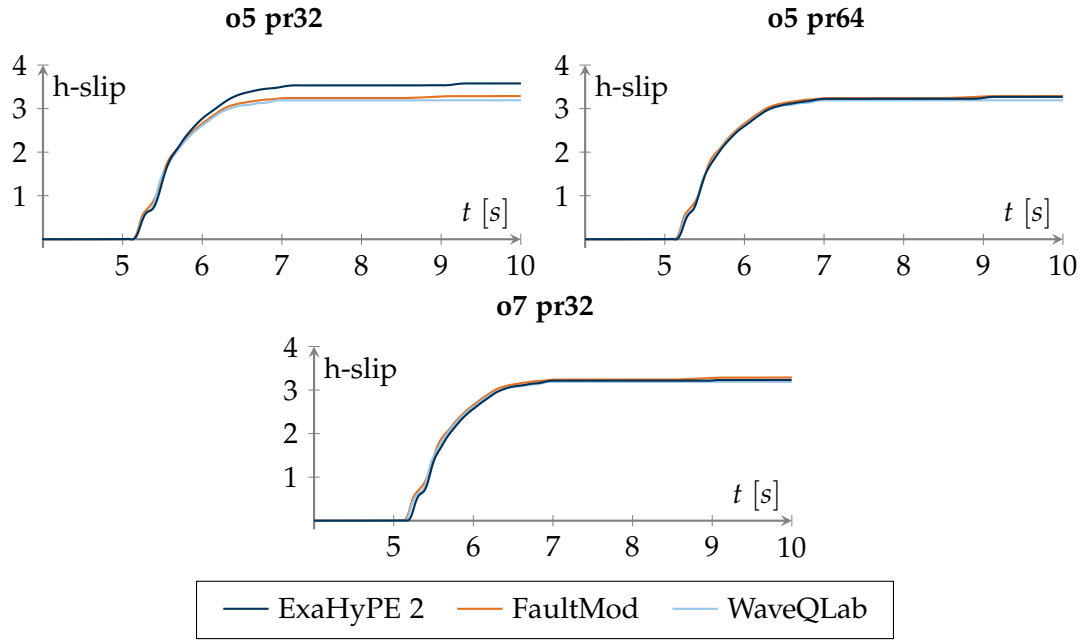


Figure 3.5: TPV5: Comparison of horizontal slip with different order (o) and floating point precision (pr) at (dip: 0 km, strike: -12 km) (simulated with domain side length 32.4 km, 27^3 cells and PML enabled)

Finally, there is still a discrepancy in the horizontal slip amplitude results for a domain size of 32.4 km with PML and the reference solutions. This discrepancy, however, can be minimized by running the simulation with either a polynomial order of 7 or with 64-bit. This is shown in Figure 3.5.

Figure 3.7 and Figure 3.6 show the final TPV5 results with a domain side length of 32.4 km, 27 cells per dimension, PML enabled, polynomial order of 7 and 64-bit floating point precision.

3.3 TPV6: Discontinuous Velocity Structure at Fault

The next benchmark, TPV6, is very similar to TPV5. The fault is also a vertical right-lateral strike-slip planar fault and has the same dimensions (30 km length, 15 km depth)

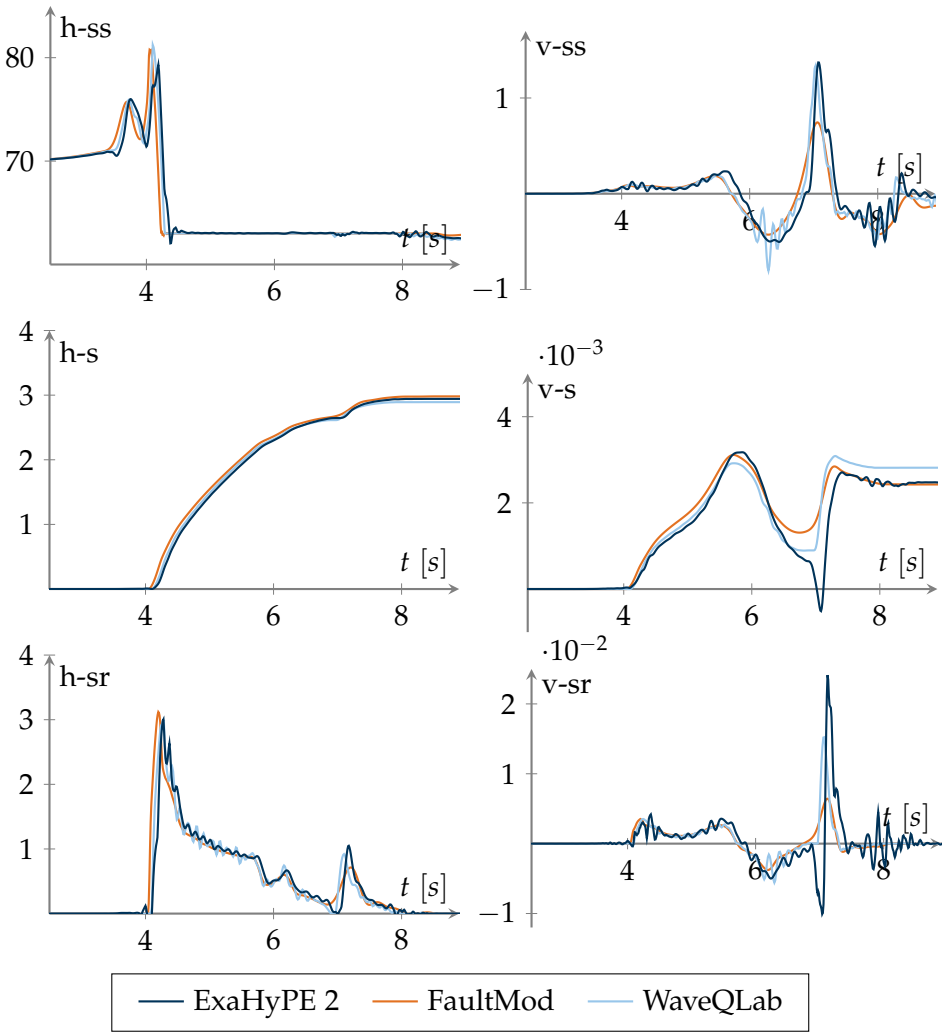


Figure 3.6: TPV5: Data at on-fault receiver (dip: 7.5 km, strike: -12 km), Shear-stress (ss), slip (s) and slip-rate(sr) in horizontal (h) and vertical (v) direction (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

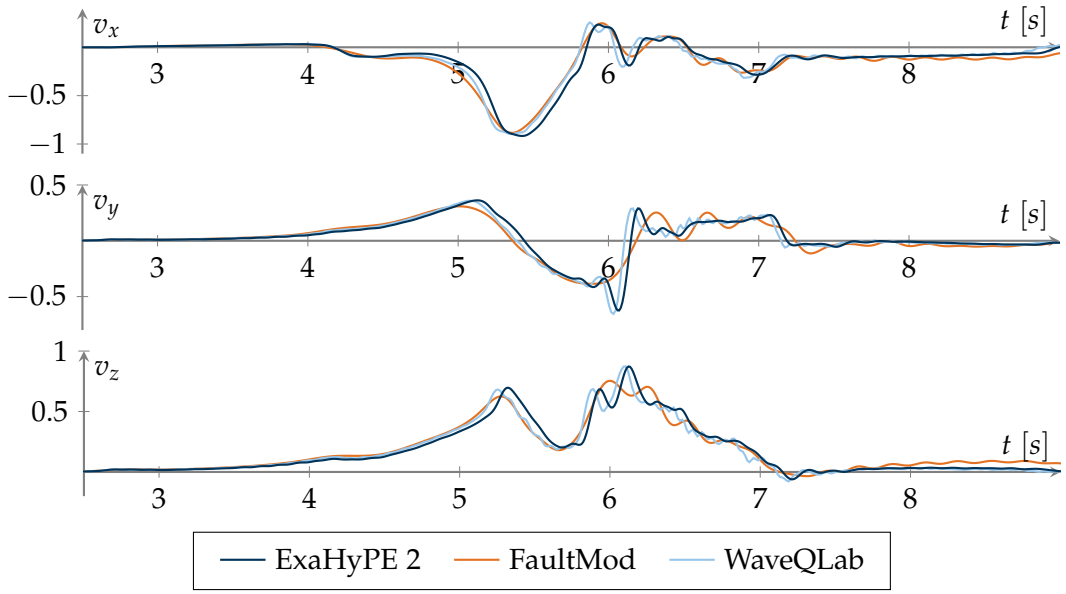


Figure 3.7: TPV5: Velocity vector $v = (v_x, v_y, v_z)$ at off-fault receiver (body: 3km, dip: 0 km, strike: -12 km) (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

as TPV5. The frictional coefficients μ_s and μ_d, and the slip-weakening critical distance d_c are also identical. What distinguishes it from TPV5 is that it uses a bimaterial domain. In this case, the material properties are split by the fault itself, meaning the far side has different wave speeds and densities than the near side. This leads to a large shear modulus (μ) contrast across the fault, making it a "well-posed" bimaterial problem [8]. When using the same coordinate system as for TPV5, the material properties are defined as:

$$\rho = \begin{cases} 2.225 \ \frac{t}{m} & \text{if } x > 0.0 \\ 2.67 \ \frac{t}{m} & \text{else} \end{cases}, \quad c_p = \begin{cases} 3.75 \ \frac{km}{s} & \text{if } x > 0.0 \\ 6.0 \ \frac{km}{s} & \text{else} \end{cases}, \quad c_s = \begin{cases} 2.165 \ \frac{km}{s} & \text{if } x > 0.0 \\ 3.464 \ \frac{km}{s} & \text{else} \end{cases}.$$

The initial stress is also very similar to TPV5, but the left and right patches of different horizontal shear stress are omitted. This makes $\sigma_{xx} = -120$ MPa (compression) and the horizontal shear stress:

$$\sigma_{xz} = \begin{cases} 81.6 \text{ MPa} & \text{if } (y, z) \in [6.0, 9.0] \times [-1.5, 1.5] \\ 70.0 \text{ MPa} & \text{else} \end{cases} \tag{3.3}$$

The remaining stresses are initialized as zero.

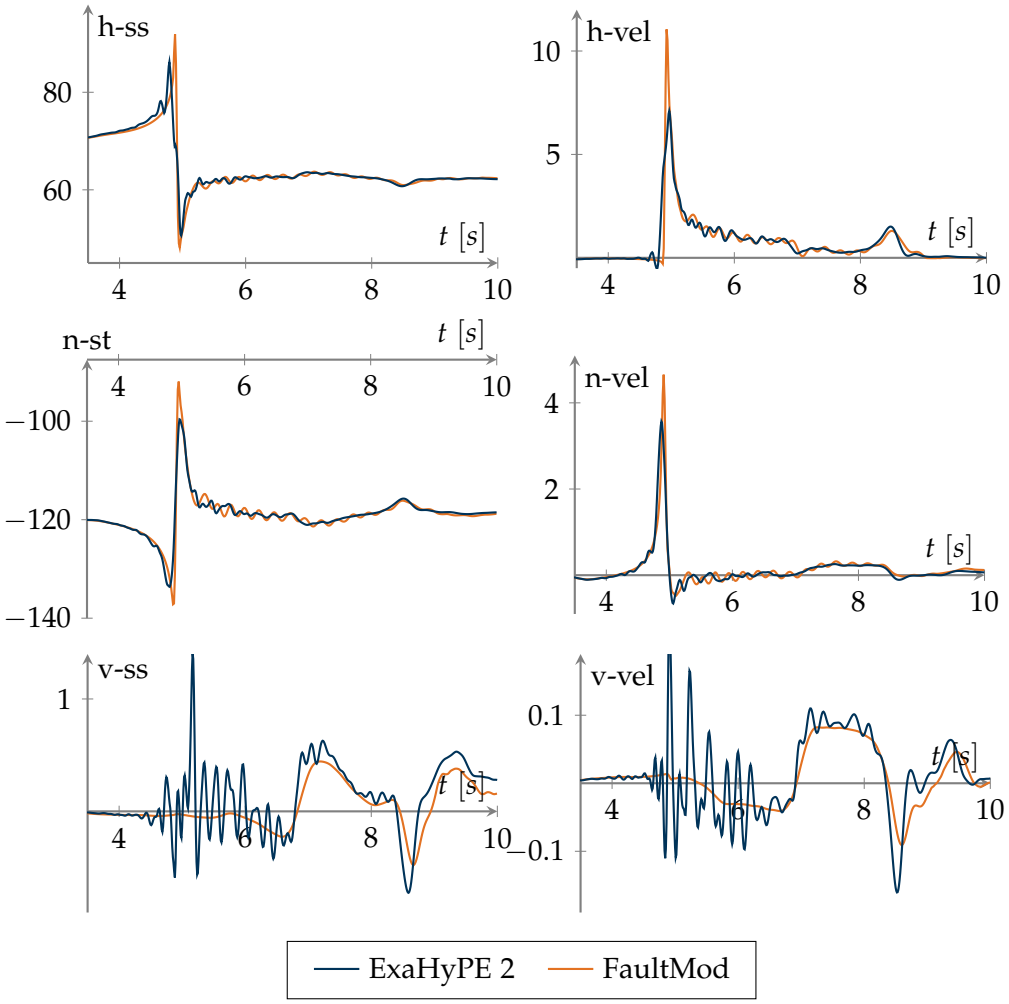


Figure 3.8: TPV6: Data at far side on-fault receiver (dip: 7.5 km, strike: 12 km); shear-stress (ss), stress (st) and velocity (vel) in horizontal (h), normal (n) and vertical (v) direction (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

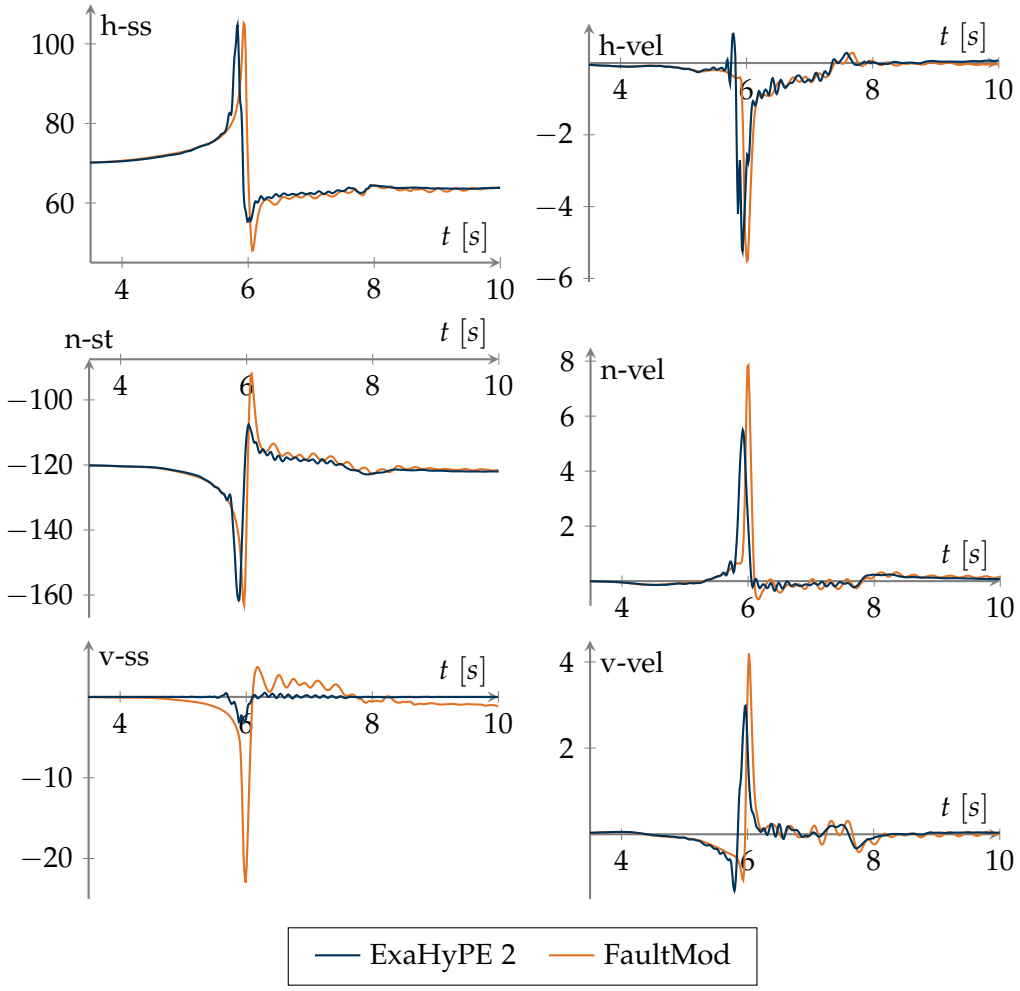


Figure 3.9: TPV6: Data at near side on-fault receiver (dip: 0.0 km, strike: 12 km); shear-stress (ss), stress (st) and velocity (vel) in horizontal (h), normal (n) and vertical (v) direction (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

Because of the similar parameters and identical structure, the same domain size as for the final results of TPV5 was chosen: $[-16.2, 16.2] \times [0.0, 32.4] \times [-16.2, 16.2]$. The results were computed with 27 cells in each dimension, leading to a cell size of 1.2 km. The receivers in TPV6 are defined differently from those in TPV5, as we have far- and near-side **on-fault** receivers, which are essentially off-fault receivers directly next to the fault. To reflect this, the ExaHyPE receivers were placed 10 m away from the fault ($x = 0.01$ km for the far-side and $x = -0.01$ km for the near-side). Given the benefits observed in TPV5 with higher polynomial order and precision, order 7 and 64-bit floating-point precision were used to simulate TPV6. The results (with PML enabled) of a below-surface far-side receiver are shown in Figure 3.8 and of a surface near-side receiver in Figure 3.9. Because WaveQLab does not provide a solution, the results are compared only to FaultMod's calculations.

The ExaHyPE 2 results seem to fit fairly well, but an increase in noise during the arrival of the earthquake is recorded in the vertical components of the below-surface receiver. This was the case for all below-surface receivers in TPV6. As later shown (see section 3.4 and section 3.5), this increase in noise shortly after the onset of the main slip is also evident in the results of some other benchmarks. When analyzing the results at the surface receiver (near side), one can notice that the vertical shear-stress is considerably smaller than the reference. This is due to the free-surface boundary condition in the Riemann solver, as it tries to keep the vertical-shear stress near zero at the surface. While this isn't necessarily incorrect, it does not give us any meaningful data to interpret. This effect of the free-surface boundary slowly fades away below the surface until a dip of approximately 0.1 km, where a similar vertical shear stress to the reference solution can be recorded. Because this is a general attribute of the Riemann solver, this near-zero vertical shear stress is observed at all surface receivers across all benchmarks.

3.4 TPV26: Forced Rupture

So far, the rupture has always been initiated by a square patch with increased shear stress at the fault's center. In TPV26, however, a new nucleation method is introduced: *forcing a rupture* [4]. The goal of this technique is to precisely control hypocenter placement and the time of nucleation by gradually reducing friction on specific parts of the fault. To implement this, two new parameters were introduced to the friction law:

- T, time of the forced rupture (in seconds)
- t_0, forced rupture decay time (in seconds)

Both parameters are constant in time but variable in space. With these parameters a new friction term $f_{forced}(T, t_0, t)$ is defined:

$$f_{forced}(T, t_0, t) = \begin{cases} 0.0 & \text{if } t < T \\ \frac{t-T}{t_0} & \text{if } T \leq t < T + t_0 \\ 1.0 & \text{else} \end{cases} \tag{3.4}$$

This term is then used to update the linear slip-weakening friction from Equation 2.18:

$$f(s) = \mu_s - (\mu_s - \mu_d) \max(f_{forced}, \frac{\min(s, d_c)}{d_c}). \tag{3.5}$$

As we can see, the linear slip-weakening part remains, but it is accompanied by f_{forced}. Now, the friction is decreased either by forced rupture over a time t_0 or by the normal slip criterion, whichever occurs first.

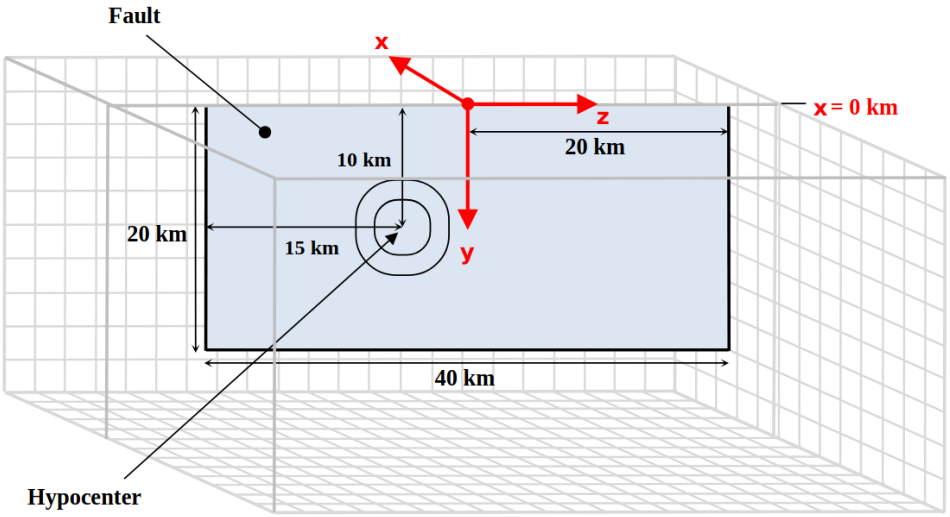


Figure 3.10: Diagram of TPV26 as shown on the official SCEC Website with adjusted coordinate system [4]

The fault in TPV26 is again a vertical, planar, strike-slip, right-lateral fault. It has a length of 40 km, a height of 20 km and it reaches the surface. We again define the same coordinate system centered on the fault's midpoint at the Earth's surface. The friction parameters are $\mu_s = 0.18$, $\mu_d = 0.12$, $d_c = 0.3$ m and $t_0 = 0.5$ s. The time of forced rupture T is dependent on the distance to the hypocenter at dip 10 km and strike -5 km:

$$T = \begin{cases} \frac{r}{0.7c_s} + \frac{0.081 r_{crit}}{0.7c_s}\left(\frac{1}{1-(r/r_{crit})^2} - 1\right) & \text{if } r < r_{crit} \\ 10^9 & \text{else} \end{cases} \tag{3.6}$$

Where $r(y,z) = \sqrt{(y+5\text{ km})^2 + (z-10\text{ km})^2}$ is the radius from the hypocenter and $r_{crit} = 4$ km. This leads to the forced rupture starting at $T = 0$ s at the hypocenter. A diagram of TPV26 can be found in Figure 3.10. The material properties ρ, c_p and c_s are the same as in TPV5 (see section 3.2). Furthermore, TPV26 also adds frictional cohesion c. While c is only 0.40 MPa at depths greater than 5 kilometers, it is linearly tapered in the uppermost 5 km to reach 4.0 MPa at the earth's surface. As an equation, c can be defined as:

$$c(y) = 0.40\text{ MPa} + 0.72\ \frac{\text{MPa}}{\text{km}} \cdot \max(5\text{ km} - y, 0). \tag{3.7}$$

The stress tensor in TPV26 includes fluid pressure P_f, simulating a hydrostatic (non-moving) water table at the surface, and gravity, which affects the vertical stress σ_{yy}. In summation, the initial stress tensor is set to:

$$\begin{aligned}
\sigma_{xx} &= \Omega(y)(b_{33}(\sigma_{yy} + P_f) - P_f) + (1.0 - \Omega(y))\sigma_{yy} + P_f, \\
\sigma_{yy} &= -2670\ \frac{\text{kg}}{\text{m}^3} \cdot 9.8\ \frac{\text{m}}{\text{s}^2} \cdot 1000\ \frac{\text{m}}{\text{km}} \cdot y + P_f, \\
\sigma_{zz} &= \Omega(y)(b_{11}(\sigma_{yy} + P_f) - P_f) + (1.0 - \Omega(y))\sigma_{yy} + P_f, \\
\sigma_{xy} &= 0.0\text{ MPa}, \\
\sigma_{xz} &= \Omega(y)(b_{13}(\sigma_{yy} + Pf)), \\
\sigma_{yz} &= 0.0\text{ MPa}.
\end{aligned} \tag{3.8}$$

The tapering coefficient $\Omega(y)$ used for the stress tensor is

$$\Omega(y) = \begin{cases} 1 & \text{if } y \leq 15\text{ km} \\ (20\text{ km} - y)/5\text{ km} & \text{if } 15\text{ km} < y \leq 20\text{ km} \\ 0 & \text{else} \end{cases} \tag{3.9}$$

and the coefficients b_{ij} are given in Table 3.1.

Coefficient	Value
b_{11}	0.926793
b_{33}	1.073206
b_{13}	-0.169029

Table 3.1: TPV26 stress coefficients as stated in the official documentation [4]

TPV26 was already implemented in ExaSeis prior to this thesis. Only a couple of tracers had to be shifted and the results had to be validated. The domain size was chosen similarly to the prior simulations to include one cell of strength barrier at the domain edges. For a fault with length 40 km this sets the domain to $[-21.6, 21.6] \times$

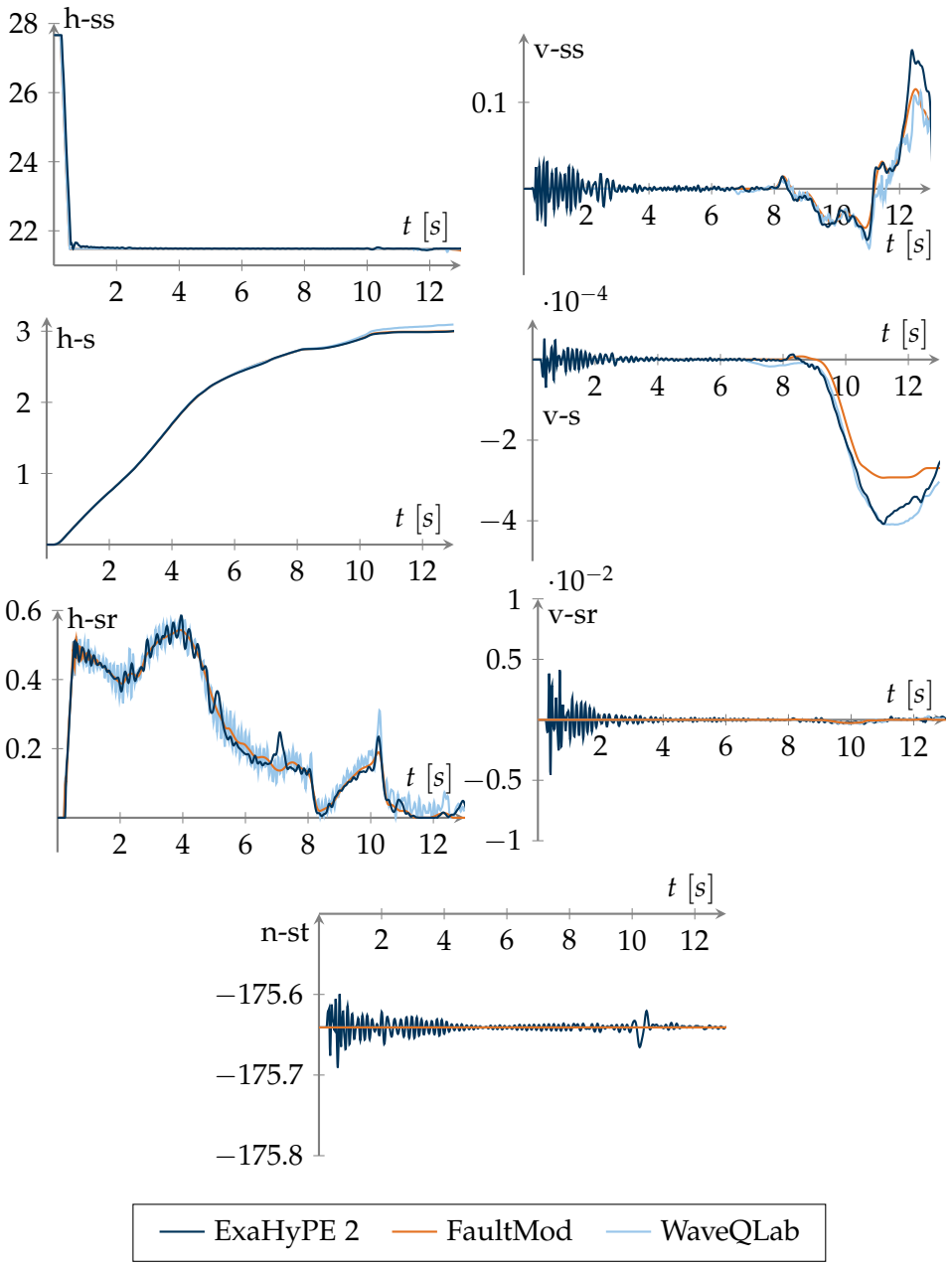


Figure 3.11: TPV26: Data at hypocenter (dip: 10.0 km, strike: -5.0 km); shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction in addition to normal stress (n-st) are shown (simulated with domain side length 43.2 km, 27^3 cells, order 7, 64-bit and PML enabled)

$[0.0, 43.2] \times [-21.6, 21.6]$. For validation, a resolution of 27 cells per dimension (cell size: 1.6 km), polynomial order of 7, and 64-bit floating-point precision were used. All results are with PML enabled.

In Figure 3.11, the results at the hypocenter are shown. As specified by T and t_0, the rupture indeed starts within the first 0.5 s. A similar noise to the one mentioned in section 3.3 can also be seen for this below-surface receiver right after nucleation. Furthermore, we can see a difference in vertical slip. As this discrepancy is also present when comparing the reference solutions, it is most likely not due to a false implementation in ExaSeis. Otherwise, the ExaHyPE solutions of this and other receivers closely follow the references.

3.5 TPV16: Random Initial Stress Conditions

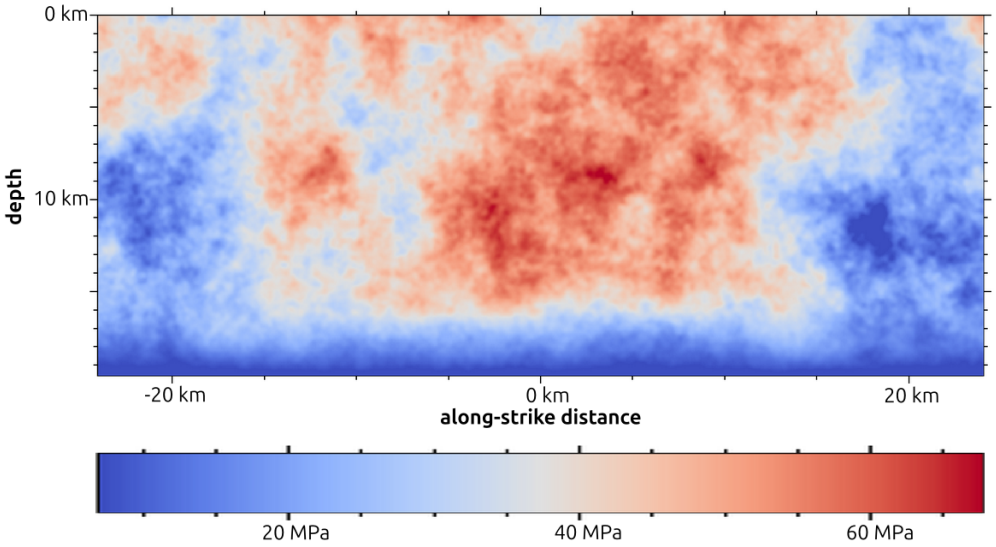
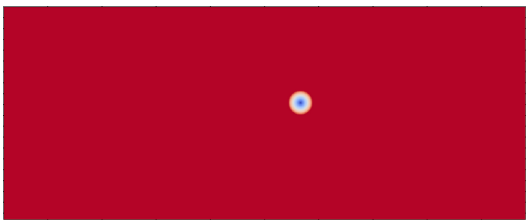
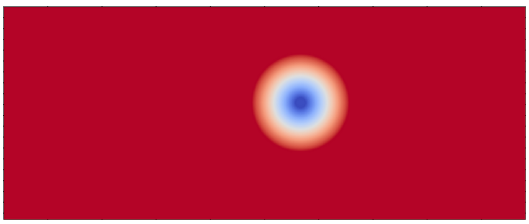


Figure 3.12: TPV16: Initial horizontal shear stress on the fault as shown on the official SCEC Website [3]

After validating the general implementation of forced ruptures, we can try simulating a similar fault, but now starting with randomly distributed initial shear stress, as shown in Figure 3.12. This shear-stress field is provided in the form of a text file by the SCEC Spontaneous Rupture Code Verification Project, with a space discretization of 75 m. To use it in ExaHyPE 2, the data was converted to the much more memory-efficient *Network Common Data Format 4 (Network Common Data Format (NetCDF) 4)*.

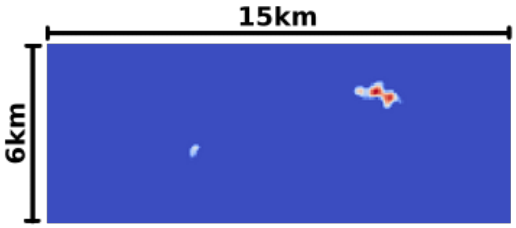


(a) Time of forced rupture T



(b) Critical distance d_c

(c) Frictional cohesion c



(d) Static coefficient of friction μ_s at central 15 km $\times$ 6 km of the fault

Figure 3.13: TPV16: Qualitative view of non-stress space-variable on-fault parameters from the official documentation [3] (red=high, blue=low)

Testing ExaHyPE with TPV16 is interesting in two ways. First, because it shows us how small the domain elements need to be to accurately reflect the input data's precision of 75 meters. Second, TPV16 implements a two-stage nucleation method [3]. The first stage, as already mentioned, is a forced rupture. This happens immediately after the simulation starts at an approximate strike of 3 km and dip of 9 km with a 1 km radius, as visualized in Figure 3.13a. Outside of this zone, the time of forced rupture T is set to 10^9 s. The second stage of the nucleation is done through regular slip-weakening friction, by reducing the critical distance d_c in a radius of approximately 4 km around the hypocenter, as shown in Figure 3.13b. Other than that, we have a similar cohesion c as in TPV26 (except that only the uppermost 3 km are used for tapering) and an increased static coefficient of friction μ_s at points, where the initial horizontal shear-stress is especially high. This is also visualized in Figure 3.13.

The rest of the parameters are set to the values in Table 3.2.

All not-mentioned stress components are initialized to zero. Outside of the fault, all space-variable parameters were set to the value they have at the edge of the fault.

The fault in total has a length of 48 km and a depth of 19.5 km. Because this fault is particularly long, the waves take considerably longer to reach the domain edges, raising the question of whether the strength barrier needs to be included in the domain. To answer this, another set of domain-size tests was conducted. The different domain sizes

Parameter	Value
ρ	2670.0 kg/m^3
c_p	6.0 km/s
c_s	3.464 km/s
σ_{xx}	60 MPa (compression)
μ_d	0.373
t_0	0.5 s

Table 3.2: TPV16: all non-space-variable parameters [3]

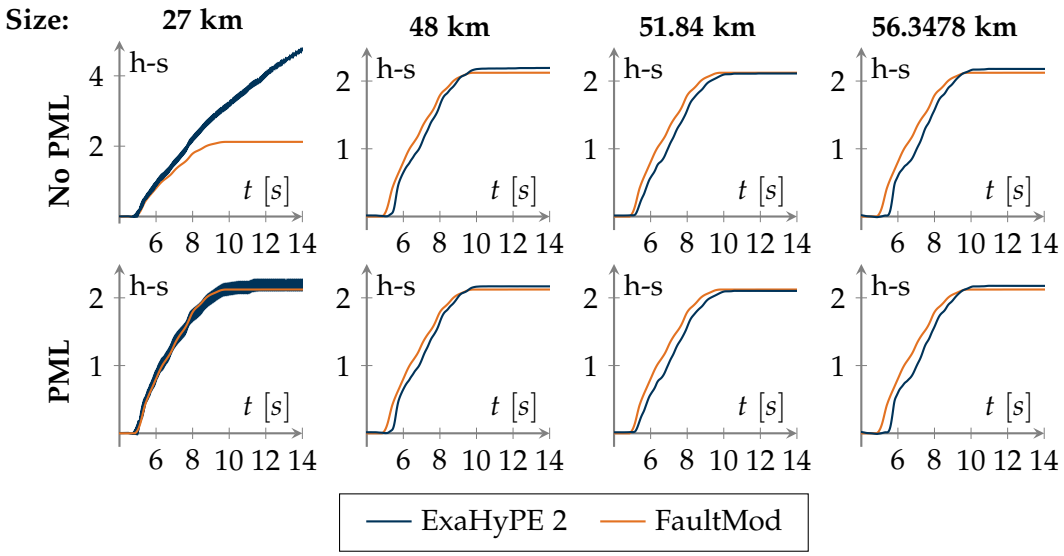


Figure 3.14: TPV16: Comparison of horizontal slip (h-s) measured at on-fault receiver (dip: 9 km, strike: -9 km) for different domain sizes, with and without PML (simulated with 27^3 cells, order 5 and 32-bit)

were picked using the same scheme as in section 3.2, giving us the sizes per dimension of: 27 km (base), 48 km (only fault), 51.84 km (one cell of strength barriers at domain edges) and 56.3478 km (two cells of strength barrier at domain edges). All tests were done with a floating point precision of 32 bits, polynomial order of 5 and 27 cells in each dimension. Again, because no reference solution by WaveQLab was publicly available, the results were only compared to FaultMod's solution. A comparison of the results can be seen in Figure 3.14. All solutions for domain side lengths of 48 km or higher look fairly similar to the reference solution and when comparing all tracers, no clear advantage of including the strength barrier can be seen anymore. The solutions for the smallest domain include a considerable amount of high-frequency noise, and similar to TPV5, the simulation with PML models the fault with the surrounding strength barrier much better than without PML. Lastly, as with TPV5, the results for larger domain sizes show a delay in their graphs. This might be because of worse accuracy of wave speeds with such big cell sizes, but could also be due to coarser mapping of the initial stress. Reducing the cell size improves the timing offsets significantly, as shown in Figure 3.15. Both results were run without PML, with 32-bit floating-point precision and polynomial order of 5. Because we are dealing with small offsets, an additional solution by the finite element code EQdyna [13] is introduced to ensure the offset is not limited to the FaultMod solution only. Indeed, while smaller, the ExaHyPE result still has a slight offset to the FaultMod solution with 81 cells per dimension, but it practically matches the EQdyna reference.

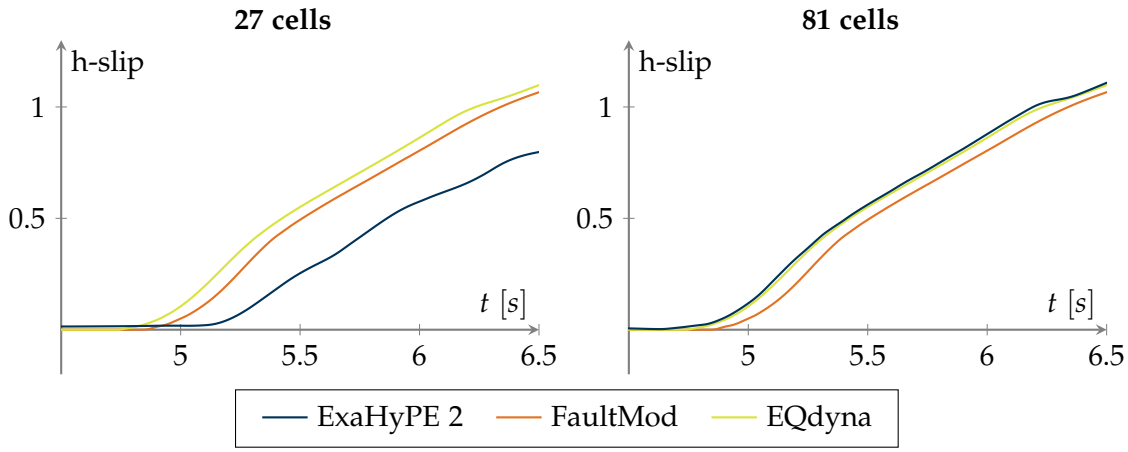


Figure 3.15: TPV16: Comparison of time difference in horizontal slip with different amounts of cells per dimension (simulated with domain side length 51.84 km, order 5, 32-bit and without PML)

The offset can also be decreased by keeping 27 cells per dimension, but running

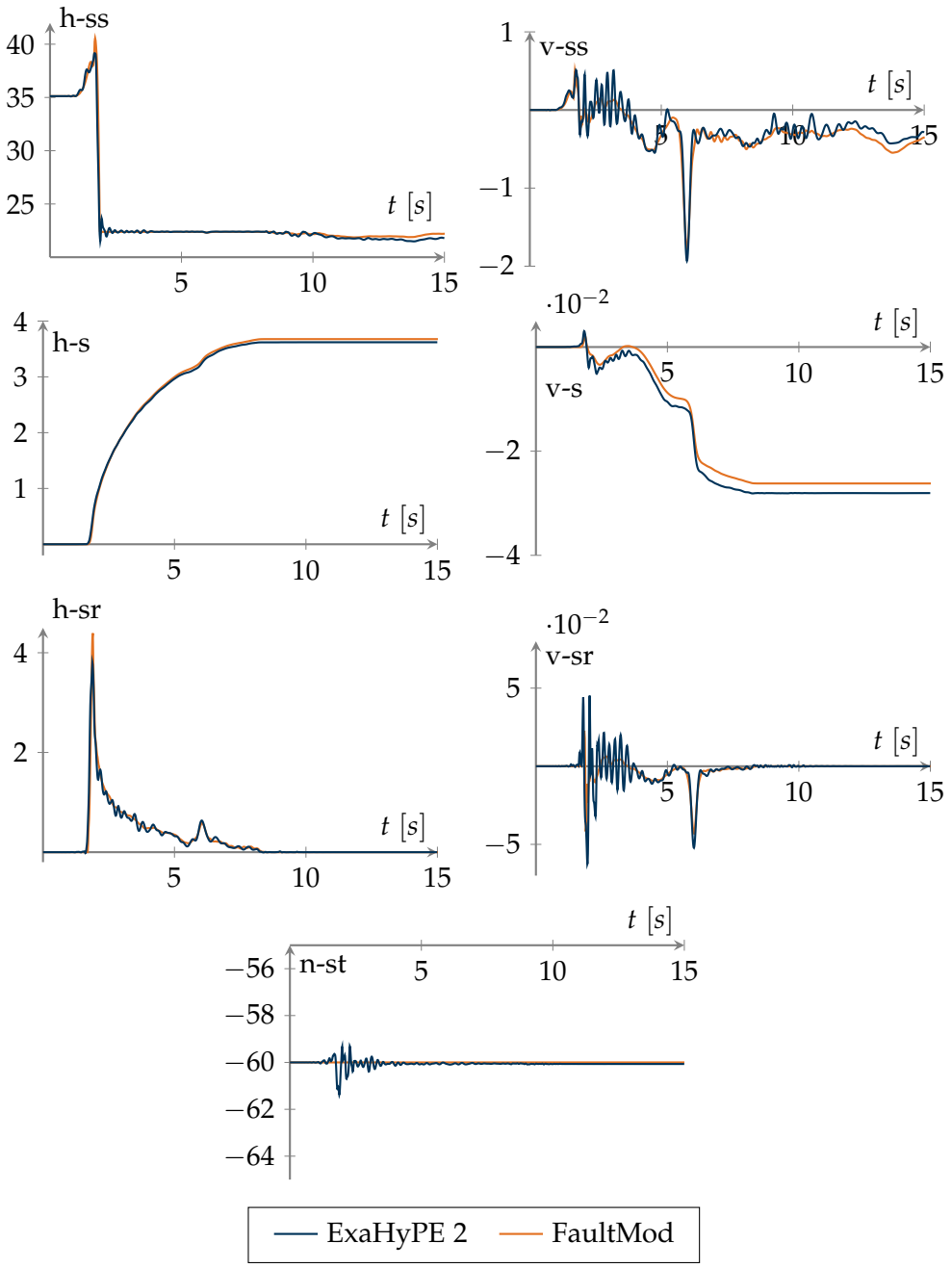


Figure 3.16: TPV16: Data at on-fault receiver (dip: 9.0 km, strike: -9.0 km); shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction in addition to normal stress (n-st) are shown (simulated with domain side length 51.84 km, 27^3 cells, order 7, 64-bit and PML enabled)

the simulation with polynomial order of 7 instead of 5, as done for the final results of TPV16. Changing the floating-point precision to 64 bits does not affect the offset.

The final tests were done with PML enabled and the domain $[-25.92, 25.92] \times [0.0, 51.84] \times [-25.92, 25.92]$ (domain side length 51.84 km), with 27 cells in each dimension (cell size of 1.92 km). Polynomial order of 7 was used and a floating-point precision of 64 bits. An example comparison of one on-fault receiver is shown in Figure 3.16. In general, the results for TPV16 fit the reference solution pretty well, but we do again see an increase in noise for about 2 seconds after the earthquake reaches the receiver. Unlike TPV6 and TPV26, this noise can also be observed in surface receivers.

3.6 TPV28: Non-planar Fault

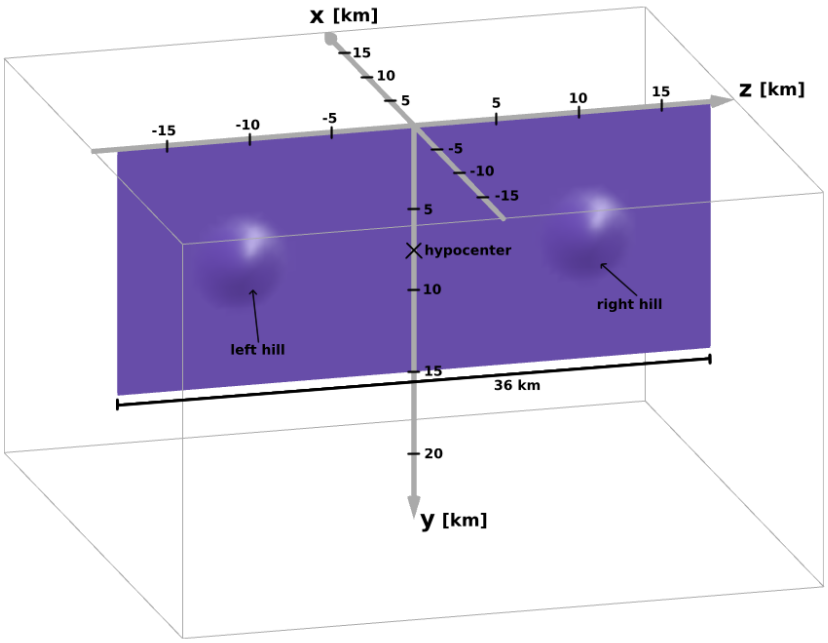


Figure 3.17: Diagram of TPV28 fault

The next validation benchmark, TPV28 [5], finally takes advantage of the curvilinear meshes. While all previous benchmarks used curvilinear meshes for the setup, the curvilinear mesh was technically unnecessary, as all faults were vertical and planar, and no other non-cartesian geometry was included. TPV28, however, defines a non-planar fault. The fault is 36 km long, 15 km deep, and partly planar, with two hills 6 km wide

and 0.6 km high. The hills in this case are indentations in the fault modeled with a cosine function that are placed symmetrically in the middle of the fault depth-wise and 10.5 km away from the hypocenter (middle of the fault). A simple visualization of the fault can be found in Figure 3.17. The geometry of the fault in x-direction is defined mathematically as:

$$x_{fault}(y,z) = \begin{cases} -0.3\text{ km}(1+\cos(\pi r_1/(3\text{ km}))) & \text{if } r_1 < 3.0\text{ km} \\ -0.3\text{ km}(1+\cos(\pi r_2/(3\text{ km}))) & \text{if } r_2 < 3.0\text{ km} \\ 0\text{ km} & \text{if } y \in [0,15] \wedge z \in [-18,18] \\ \text{no fault/strength barrier} & \text{else} \end{cases} \tag{3.10}$$

Where r_1 and r_2 are the radii from the respective center points of the hills located at dip 7.5 km and strike -10.5 km and 10.5 km:

$$\begin{aligned} r_1 &= \sqrt{(y-7.5\text{ km})^2 + (z+10.5\text{ km})^2}, \\ r_2 &= \sqrt{(y-7.5\text{ km})^2 + (z-10.5\text{ km})^2}. \end{aligned} \tag{3.11}$$

It is important to note that, even though the fault is non-planar, it has a continuous first derivative, classifying it as a smooth fault rather than a rough one. The fault sits in a homogeneous halfspace with material properties identical to TPV5:

$$\rho = 2.67\ \frac{\text{t}}{\text{m}^3}, \qquad c_p = 6.0\ \frac{\text{km}}{\text{s}}, \qquad c_s = 3.464\ \frac{\text{km}}{\text{s}}.$$

The stress tensor on the fault is given in the x-,y- and z-direction. Because the fault is non-planar, this stress tensor does not reflect the actual normal and shear stresses on the hills of the fault, only on the planar parts. On the hills, the stress tensor must be rotated into the correct coordinate system at each point. The non-translated stress tensor is defined as:

$$\begin{aligned} \sigma_{xx} &= -60\text{ MPa}, \\ \sigma_{yy} &= 0\text{ MPa}, \\ \sigma_{zz} &= -60\text{ MPa}, \\ \sigma_{xy} &= 0\text{ MPa}, \\ \sigma_{xz} &= 29.38\text{ MPa}, \\ \sigma_{yz} &= 0\text{ MPa}. \end{aligned} \tag{3.12}$$

The rupture is not forced. To nucleate the rupture, an additional horizontal shear stress $\tau_{nucleate}(r)$ is applied in a circular zone surrounding the hypocenter:

$$\tau_{nucleate}(r) = \begin{cases} 11.6 \text{ MPa} & \text{if } r \leq 1.4 \text{ km} \\ 5.8 \text{ MPa}(1 + \cos(\pi \frac{r - 1.4 \text{ km}}{0.6 \text{ km}}) & \text{if } 1.4 \text{ km} < r \leq 2.0 \text{ km} \\ 0 & \text{else} \end{cases} \tag{3.13}$$

With $r(y,z) = \sqrt{(y - 7.5 \text{ km})^2 + z^2}$ being the radius from the hypocenter. Usually, the nucleation shear stress has to be added after rotating the stress tensor, but, as $\tau_{nucleate}$ is zero on all non-planar parts of the fault, we can add it immediately to σ_{xz} of Equation 3.12. This results in a pure right-lateral strike-slip scenario. The friction parameters for TPV28 are displayed in Table 3.3.

Parameter	Value
μ_s	0.677
μ_d	0.373
d_c	0.4 m
c	0

Table 3.3: TPV28: all friction parameters [5]

TPV28 was already partially implemented prior to this thesis, but produced incorrect results. This was due to a wrong domain offset, which has now been fixed. Additionally, the positions of on-fault tracers were corrected and the domain size was adjusted to include one cell of strength barrier at the domain edges.

All ExaSeis solutions shown here were run with a domain of $[-19.44, 19.44] \times [0.0, 38.88] \times [-19.44, 19.44]$, 27 cells per dimension (and therefore cell size of 1.44 km), PML enabled, polynomial order of 7 and 64-bit floating-point precision. The results are again compared to the solutions by FaultMod and WaveQLab. Because we specifically want to test if the influence of the non-planar fault is simulated correctly, the focus is on receivers on and near the hills. Figure 3.18 shows the result of a receiver placed directly at the center of the left hill. It closely resembles the references, meaning wave propagation and slip work as expected up to the center of the hill. Next, we take a look at a receiver placed on the slope of the same hill, facing away from the hypocenter (dip: 7.5 km, strike: -12 km). Here, the output of ExaHyPE currently only lets us compare the slip and slip-rate, as ExaHyPE returns the stress tensor translated in x-, y- and z-direction, which are not the horizontal, vertical and normal directions of the hill geometry at this receiver. As visualized in Figure 3.19, the results also match the reference solutions quite well. This is representative of all tracers near and on the hills. Finally, the stress tensor was validated again on the planar segments of the fault on the opposite side of the hills with regard to the hypocenter. There, it matches the reference solutions with the same accuracy as the results shown in Figure 3.18.

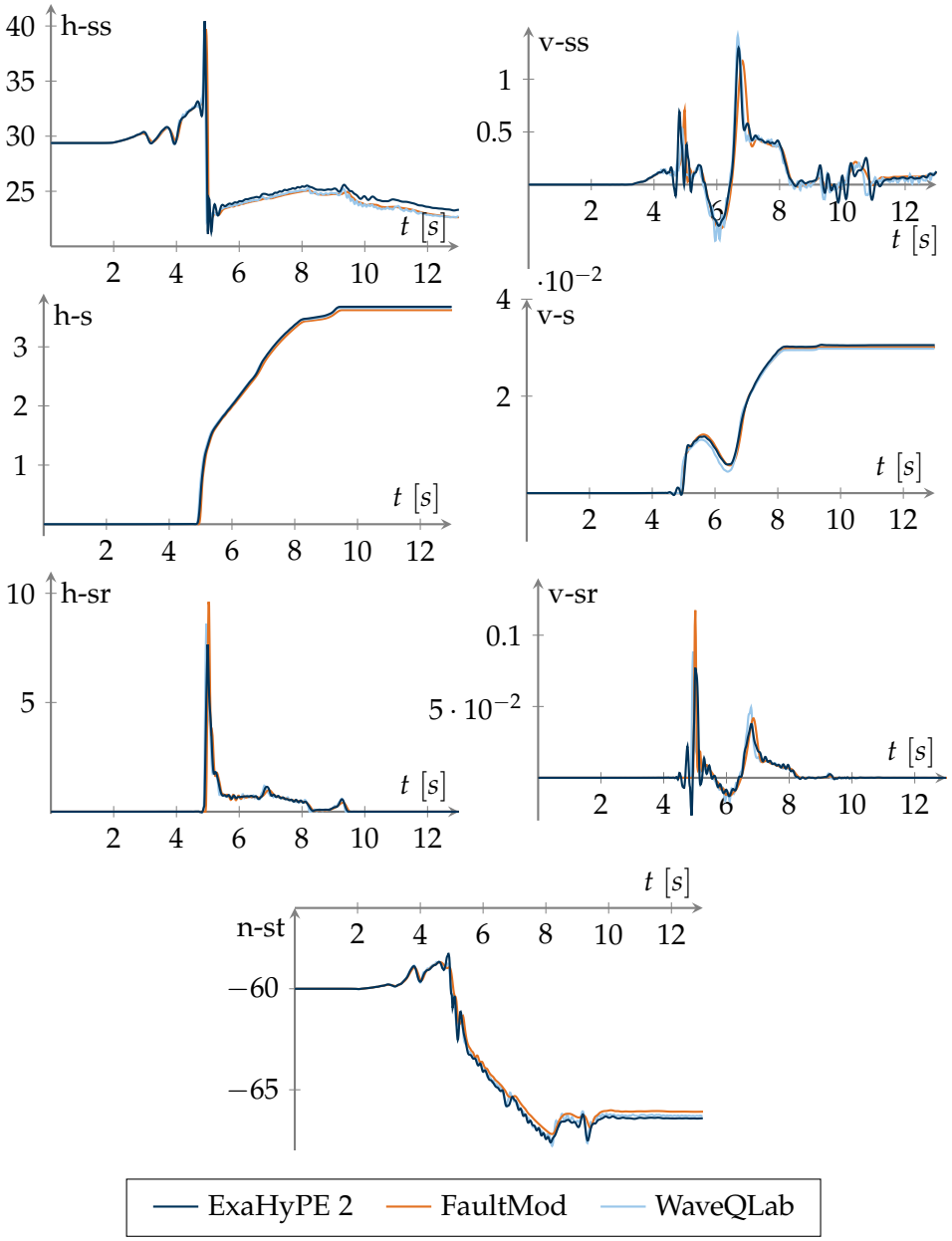


Figure 3.18: TPV28: Data at center of left hill (dip: 7.5 km, strike: -10.5 km); shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction in addition to normal stress (n-st) are shown (simulated with domain side length 38.88 km, 27^3 cells, order 7, 64-bit and PML enabled)

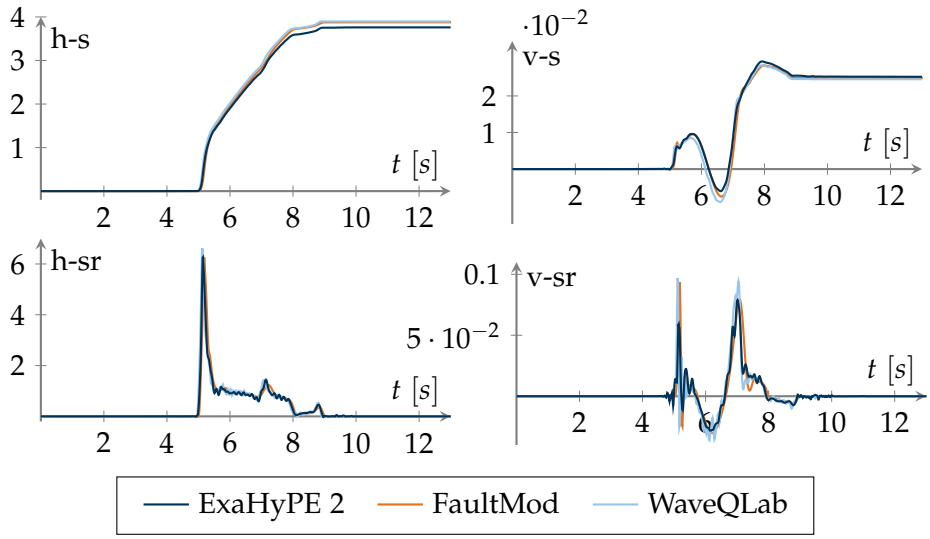


Figure 3.19: TPV28: Data at slope of left hill (dip: 7.5 km, strike: -12 km); slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction are shown (simulated with domain side length 38.88 km, 27^3 cells, order 7, 64-bit and PML enabled)

Separately from this thesis, changes were made to ExaSeis to correctly handle faults with even greater offsets in both x-directions from $x = 0$. These changes were verified independently of this thesis with an implementation of the rough non-planar fault scenario TPV29.

3.7 TPV34: Continuous 3D Velocity Structure

In this final validation benchmark, we aim to test ExaSeis using a more realistic fault example that combines previously tested features. This is done by modeling the Imperial Fault. The Imperial Fault is an approximately 45 km long and 15 km deep fault near the surface of the California-Mexico border [6], as visualized in Figure 3.20. Throughout the fault, the angle in respect to the surface ranges from 81 to 90 degrees (according to the SCEC Community Fault Model CFM-4 [21]). In the past, the Imperial Fault has caused multiple minor and major earthquakes, the most notable being a 6.0-6.7 local magnitude rupture in 1979 and a 6.5-7.0 moment magnitude rupture in 1940, according to the Southern California Earthquake Data Center [29]. TPV34 only includes the central vertical planar part of the fault, which is 30 km long (the middle 30 km of the red line in Figure 3.20).

Figure 3.20: Excerpt of the top view of the Imperial Fault (marked in red) shown in the official TPV34 documentation [6]

To simulate an approximation of the actual fault, the scenario uses the SCEC Community Velocity Model SCEC Community Velocity Model (CVM-H) [24] to set material properties (ρ, c_s and c_p) appropriately. This involves prompting CVM-H before the simulation to get the velocity structure throughout the entire domain. CVM-H lets us retrieve material properties at any point we specify, as long as the point is expressed in the Universal Transverse Mercator coordinate system. A conversion formula is given in the TPV34 documentation [6]. For the coordinate system in ExaSeis (centered on the fault's surface midpoint), it can be rewritten to:

$$\begin{aligned} x_{utm} &= 648446\ \text{m} - 0.5802386\ z - 0.8144465\ x, \\ y_{utm} &= 3625237\ \text{m} + 0.8144465\ z - 0.5802386\ x, \\ z_{utm} &= \max(y, 100\ \text{m}). \end{aligned} \tag{3.14}$$

Where x, y and z are the coordinates of a point given in ExaSeis's coordinate system in meters. To streamline the process of prompting for material data, a bash script that interfaces with two Python files was implemented to request the necessary domain data and directly convert it to NetCDF. Furthermore, the data is modified to restrict the minimum of every material property ($\rho_{min} = 2220.34\ \text{kg/m}^3$, $c_{p,min} = 2984\ \text{m/s}$

and $c_{s,min} = 1400$ m/s), as specified in the TPV34 documentation [6]. This pipeline was adapted from SeisSol's implementation [25]. The prompting is split into two parts: data on the fault D_1 and data throughout the domain in general D_2. Because D_1 is later used to calculate the initial stress conditions, the data is prompted with a relatively high resolution of 0.1 km (1/12 of a cell). D_2 is a much larger dataset, as it spans the entire three-dimensional domain. Therefore, it is sampled at a resolution of 0.6 km (1/2 of a cell). A spatial resolution of 0.3 km was also tested but showed no significant advantages. To dynamically read the material data D_2, the component used to read the initial conditions on the fault was repurposed. To verify that the material properties are set correctly, we can visualize ExaHyPE's domain in Paraview. An example of the visualization of c_s is shown in Figure 3.21. The S-wave speed is fairly slow at the top of the domain and gets faster further down, which is also shown in TPV34's documentation [6].

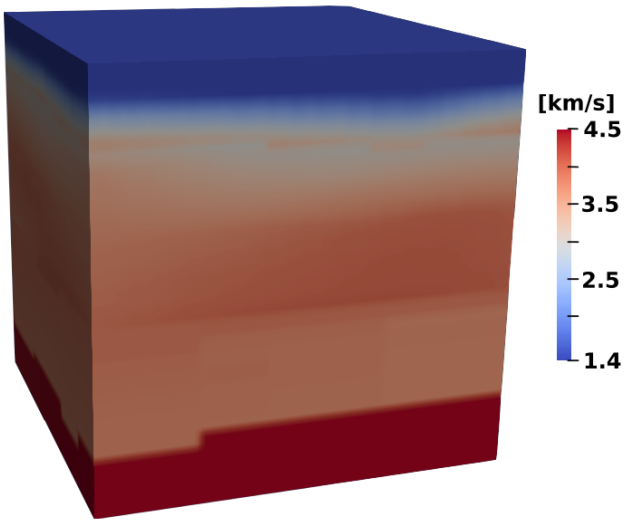


Figure 3.21: TPV34 c_s values across the domain visualized with Paraview

The important initial stress components in TPV34 are:

$$\begin{aligned} \sigma_{xx} &= -60 \text{ MPa} \cdot \frac{\mu}{\mu_0}, \\ \sigma_{xz} &= 30 \text{ MPa} \cdot \frac{\mu}{\mu_0}. \end{aligned} \tag{3.15}$$

Where μ is one of the Lamé parameters mentioned in section 2.2, also called the shear modulus. We can compute the shear modulus with the given material properties

in D_1 as follows:

$$\mu = {c_s}^2 \rho. \tag{3.16}$$

μ is then set into proportion with the shear modulus μ_0, which is set to

$$\mu_0 = 32.03812032 \text{ GPa}. \tag{3.17}$$

All other stress components are initialized to zero. To start the rupture, we again, almost identically to TPV28, add an additional nucleation shear stress $\tau_{nucleate}(r)$ to the horizontal shear stress σ_{xz}:

$$\tau_{nucleate}(r) = \begin{cases} 4.95 \text{ MPa} \cdot \mu/\mu_0 & \text{if } r \leq 1.4 \text{ km} \\ 2.475 \text{ MPa}(1 + \cos(\pi \frac{r - 1.4 \text{ km}}{0.6 \text{ km}}) \cdot \mu/\mu_0 & \text{if } 1.4 \text{ km} < r \leq 2.0 \text{ km} \\ 0 & \text{else} \end{cases} \tag{3.18}$$

Again, with $r(y,z) = \sqrt{(y - 7.5 \text{ km})^2 + z^2}$. In the friction model, cohesion c is linearly tapered at the uppermost domain section, similarly to TPV16 and TPV26:

$$c(y) = 0.425 \frac{\text{MPa}}{\text{km}} \cdot \max(2.4 \text{ km} - y, 0). \tag{3.19}$$

The rest of the friction parameters are documented in Table 3.4.

Parameter	Value
μ_s	0.58
μ_d	0.45
d_c	0.18 m

Table 3.4: Values for μ_s, μ_d and d_c from the TPV34 documentation [6]

Because TPV34's fault is 30 km long, the same domain size as for TPV5 and TPV6 was chosen: $[-16.2, 16.2] \times [0.0, 32.4] \times [-16.2, 16.2]$. The simulation was run with 27 cells per dimension, resulting in a cell size of 1.2 km. Same as for previous benchmarks, polynomial order of 7 and 64-bit floating-point precision was used. PML was enabled. WaveQLab does not provide a reference solution for TPV34, so the results are compared only to FaultMod's solution. To verify if the wave propagation in a heterogeneous domain is handled correctly, the results of an on-fault receiver semi-far away from the hypocenter (dip: 1 km, strike: -12 km) are shown in Figure 3.22 and the results of one of the furthest off-fault receiver's from the hypocenter (body: 15 km, dip: 0 km, strike: 15 km) are shown in Figure 3.23. The results of the on-fault receiver match well, except that the normal stress shows some deviation after the noise introduced at the onset of slip. The solution of the velocity vector at the off-fault receiver also matches well, with only slight discrepancies shown towards the later part of the simulation.

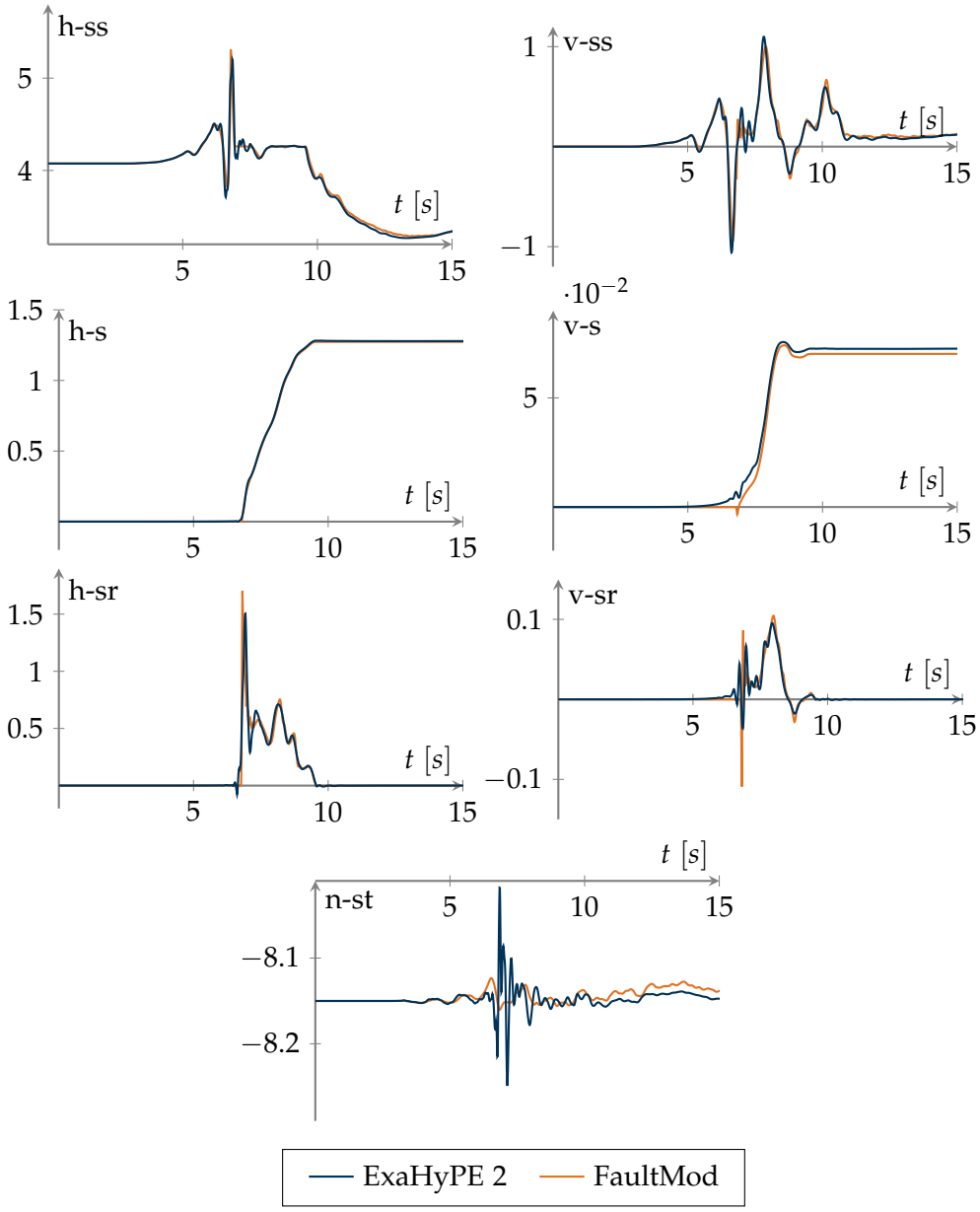


Figure 3.22: TPV34: Data at on-fault receiver (dip: 1.0 km, strike: -12.0 km); shear-stress (ss), slip (s) and slip-rate (sr) in horizontal (h) and vertical (v) direction in addition to normal stress (n-st) are shown (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

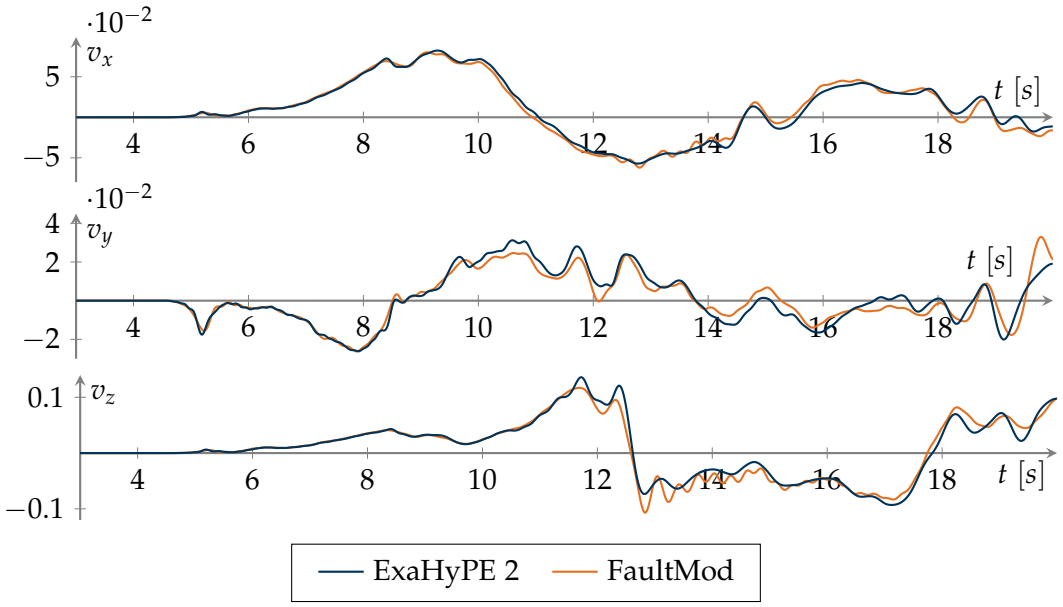


Figure 3.23: TPV34: Velocity vector $v = (v_x, v_y, v_z)$ at off-fault receiver (body: 15km, dip: 0 km, strike: 15 km) (simulated with domain side length 32.4 km, 27^3 cells, order 7, 64-bit and PML enabled)

4 Conclusion and Future Work

This work aimed to verify ExaSeis's implementation of the elastic wave equations in ExaHyPE 2. This was done by implementing multiple earthquake scenarios from the SISMOWINE web interface and the SCEC/USGS Spontaneous Rupture Code Verification Project. In total, four benchmarks, which were implemented prior to this work (LOH1, TPV5, TPV26 and TPV28), were validated and modeling of the scenarios was improved/fixed, when needed, and three benchmarks (TPV6, TPV16 and TPV34) were implemented from scratch and also validated. The results of these benchmarks were compared with solutions from other simulation codes, such as FaultMod and WaveQLab. In all cases, ExaSeis's results were similar to the reference solutions, though sometimes they included a small amount of noise at the onset of the fault slipping. This way, many setup conditions were shown to work with the current ExaSeis implementation, including bimaterial problems, forced ruptures, different ways of setting initial stress conditions, non-planar smooth faults and 3D velocity structures. Furthermore, the impact of simulation parameters like polynomial order of the ADER-DG solver, floating-point precision and cell size was explored. All results and implemented benchmarks are available in a public repository [28].

Future work could entail implementing even more benchmarks from the SCEC/USGS Spontaneous Rupture Code Verification Project to verify additional setup conditions, such as dip-slip faults, using TPV9 as an example. It would also be interesting to implement more real-life faults, like the Húsavík-Flatey fault (which is already being worked on), as these often combine multiple modeling challenges and sometimes include actual seismograph records to compare results with. There are also some technical improvements that could be worked on, mainly natively supporting on-fault receivers in ExaSeis. Currently, on-fault receivers are modeled by placing a receiver on each side of the fault and subtracting the results. Finding the right coordinates for these receivers takes a lot of time and it often leads to inaccuracy because their distance to the fault varies. A native implementation of on-fault receivers could therefore streamline simulation setup and improve results. Other friction models, including rate-state friction, could also be a focus of future work.

List of Figures

List of Tables

List of Abbreviations

SCEC Statewide California Earthquake Center 1

USGS United States Geological Survey 1

PDE Partial differential Equation 3

ADER-DG Arbitrary high-order DERivative + Discontinuous Galerkin 3

PML Perfectly matched layer 16

NetCDF Network Common Data Format 28

CVM-H SCEC Community Velocity Model 38

Bibliography

[1] M. 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. DOI: 10.1111/j.1365-246X.2009.04190.x. URL: https://pubs.usgs.gov/publication/70036782 (visited on 01/23/2026).

[2] W. C. Chew and W. 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. ISSN: 1098-2760. DOI: 10.1002/mop.4650071304. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.4650071304 (visited on 02/02/2026).

[3] *Documentation for TPV16 and TPV17*. URL: https://strike.scec.org/cvws/tpv16_17docs.html (visited on 02/04/2026).

[4] *Documentation for TPV26 and TPV27*. URL: https://strike.scec.org/cvws/tpv26_27docs.html (visited on 02/04/2026).

[5] *Documentation for TPV28*. URL: https://strike.scec.org/cvws/tpv28docs.html (visited on 02/04/2026).

[6] *Documentation for TPV34*. URL: https://strike.scec.org/cvws/tpv34docs.html (visited on 02/04/2026).

[7] *Documentation for TPV5*. URL: https://strike.scec.org/cvws/tpv5docs.html (visited on 01/26/2026).

[8] *Documentation for TPV6 and TPV7*. URL: https://strike.scec.org/cvws/tpv67docs.html (visited on 01/27/2026).

[9] K. Duru and E. M. Dunham. "Dynamic Earthquake Rupture Simulations on Nonplanar Faults Embedded in 3D Geometrically Complex, Heterogeneous Elastic Solids." In: *Journal of Computational Physics* 305 (Jan. 15, 2016), pp. 185–207. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2015.10.021. URL: https://www.sciencedirect.com/science/article/pii/S0021999115006853 (visited on 01/23/2026).

[10] K. Duru, L. Rannabauer, A.-A. Gabriel, G. Kreiss, and M. Bader. "A Stable Discontinuous Galerkin Method for the Perfectly Matched Layer for Elastodynamics in First Order Form." In: *Numerische Mathematik* 146.4 (Dec. 1, 2020), pp. 729–782. ISSN: 0945-3245. DOI: 10.1007/s00211-020-01160-w. URL: https://doi.org/10.1007/s00211-020-01160-w (visited on 02/02/2026).

[11] K. Duru, L. Rannabauer, A.-A. Gabriel, O. K. A. Ling, H. Igel, and M. 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 (Feb. 1, 2022), p. 114386. ISSN: 0045-7825. DOI: 10.1016/j.cma.2021.114386. URL: https://www.sciencedirect.com/science/article/pii/S0045782521006459 (visited on 01/19/2026).

[12] B. Engquist and A. Majda. "Absorbing Boundary Conditions for Numerical Simulation of Waves." In: *Proceedings of the National Academy of Sciences* 74.5 (May 1977), pp. 1765–1766. DOI: 10.1073/pnas.74.5.1765. URL: https://www.pnas.org/doi/10.1073/pnas.74.5.1765 (visited on 02/01/2026).

[13] *EQDYNA/EQdyna*. EQDYNA, Oct. 25, 2025. URL: https://github.com/EQDYNA/EQdyna (visited on 02/10/2026).

[14] L. Fülöp, V. Jussila, and O. Kaisko. *Benchmarking and Evolving Earthquake Fault-Rupture Simulations*. VTT Research Report. VTT Technical Research Centre of Finland, 2023.

[15] R. A. Harris, M. Barall, R. Archuleta, E. Dunham, B. Aagaard, J. P. Ampuero, H. Bhat, V. Cruz-Atienza, L. Dalguer, P. Dawson, S. Day, B. Duan, G. Ely, Y. Kaneko, Y. Kase, N. Lapusta, Y. Liu, S. Ma, D. Oglesby, K. Olsen, A. Pitarka, S. Song, and E. Templeton. "The SCEC/USGS Dynamic Earthquake Rupture Code Verification Exercise." In: *Seismological Research Letters* 80.1 (Jan. 1, 2009), pp. 119–126. ISSN: 0895-0695. DOI: 10.1785/gssrl.80.1.119. URL: https://doi.org/10.1785/gssrl.80.1.119 (visited on 01/23/2026).

[16] Y. Ida. "Cohesive Force across the Tip of a Longitudinal-Shear Crack and Griffith's Specific Surface Energy." In: *Journal of Geophysical Research (1896-1977)* 77.20 (1972), pp. 3796–3805. ISSN: 2156-2202. DOI: 10.1029/JB077i020p03796. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/JB077i020p03796 (visited on 02/02/2026).

[17] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge: Cambridge University Press, 2002. ISBN: 978-0-521-00924-9. DOI: 10.1017/CBO9780511791253.

[18] D. Li and A.-A. Gabriel. "Linking 3D Long-Term Slow-Slip Cycle Models With Rupture Dynamics: The Nucleation of the 2014 Mw 7.3 Guerrero, Mexico Earthquake." In: *AGU Advances* 5.2 (2024), e2023AV000979. ISSN: 2576-604X. DOI: 10.1029/2023AV000979. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/2023AV000979 (visited on 01/26/2026).

[19] M. Marot-Lassauzaie and M. Bader. *Mixed-Precision in High-Order Methods: The Impact of Floating-Point Precision on the ADER-DG Algorithm*. arXiv.org. Apr. 9, 2025. DOI: 10.48550/arXiv.2504.06889. URL: https://arxiv.org/abs/2504.06889v1 (visited on 02/13/2026).

[20] *Peano*. URL: https://tum-i5.github.io/ExaHyPE-Documentation/d3/d82/page_exahype2_home.html (visited on 01/23/2026).

[21] A. Plesch, S. Marshall, and J. Shaw. *SCEC Community Fault Model (CFM)*. Version 7.0. Statewide California Earthquake Center, Sept. 4, 2024. DOI: 10.5281/ZENODO.4651667. URL: https://zenodo.org/doi/10.5281/zenodo.4651667 (visited on 02/07/2026).

[22] L. A. Rannabauer. "Earthquake and Tsunami Simulation with High-Order ADER-DG Methods." Technische Universität München, 2022. URL: https://mediatum.ub.tum.de/1639212 (visited on 01/19/2026).

[23] A. Reinarz, D. E. Charrier, M. Bader, L. Bovard, M. Dumbser, K. Duru, F. Fambri, A.-A. Gabriel, J.-M. Gallard, S. Köppel, L. Krenz, L. Rannabauer, L. Rezzolla, P. Samfass, M. Tavelli, and T. Weinzierl. "ExaHyPE: An Engine for Parallel Dynamically Adaptive Simulations of Wave Problems." In: *Computer Physics Communications* 254 (Sept. 1, 2020), p. 107251. ISSN: 0010-4655. DOI: 10.1016/j.cpc.2020.107251. URL: https://www.sciencedirect.com/science/article/pii/S001046552030076X (visited on 01/23/2026).

[24] *SCECcode/Cvmh*. Statewide California Earthquake Center, Sept. 29, 2025. URL: https://github.com/SCECcode/cvmh (visited on 02/08/2026).

[25] SeisSol. *Examples/Tpv34 at Master · SeisSol/Examples*. GitHub. URL: https://github.com/SeisSol/Examples/tree/master/tpv34 (visited on 02/08/2026).

[26] *SeisSol — SeisSol Documentation*. URL: https://seissol.readthedocs.io/en/latest/ (visited on 02/06/2026).

[27] *SISMOWINE*. URL: https://sismowine.org/ (visited on 01/26/2026).

[28] T. Solfronk. *TimSolfronk/Bachelor_Data*. Feb. 10, 2026. URL: https://github.com/TimSolfronk/Bachelor_Data (visited on 02/11/2026).

[29] *Southern California Earthquake Data Center at Caltech*. URL: https://scedc.caltech.edu/earthquake/imperial.html (visited on 02/08/2026).

[30] T. Weinzierl. "The Peano Software—Parallel, Automaton-based, Dynamically Adaptive Grid Traversals." In: *ACM Trans. Math. Softw.* 45.2 (Apr. 18, 2019), 14:1–14:41. ISSN: 0098-3500. DOI: 10.1145/3319797. URL: https://dl.acm.org/doi/10.1145/3319797 (visited on 01/23/2026).

[31] *What Is a Fault and What Are the Different Types? | U.S. Geological Survey*. Dec. 31, 2023. URL: https://www.usgs.gov/faqs/what-a-fault-and-what-are-different-types (visited on 01/26/2026).

[32] *Why Are We Having so Many (or so Few) Earthquakes? Has Naturally Occurring Earthquake Activity Been Increasing? | U.S. Geological Survey*. Aug. 19, 2012. URL: https://www.usgs.gov/faqs/why-are-we-having-so-many-or-so-few-earthquakes-has-naturally-occurring-earthquake-activity (visited on 01/23/2026).

[33] S. Wollherr. "Inelastic material response in multi-physics earthquake rupture simulations." Text.PhDThesis. Ludwig-Maximilians-Universität München, Dec. 18, 2018. DOI: 10.5282/edoc.23609. URL: https://edoc.ub.uni-muenchen.de/23609/ (visited on 01/26/2026).

[34] O. Zanotti, F. Fambri, M. Dumbser, and A. Hidalgo. "Space–Time Adaptive ADER Discontinuous Galerkin Finite Element Schemes with *a Posteriori* Sub-Cell Finite Volume Limiting." In: *Computers & Fluids* 118 (Sept. 2, 2015), pp. 204–224. ISSN: 0045-7930. DOI: 10.1016/j.compfluid.2015.06.020. URL: https://www.sciencedirect.com/science/article/pii/S0045793015002030 (visited on 01/31/2026).

[35] X. Zhou and Y. Ben-Zion. "A Simulator of Earthquakes and Aseismic Slip on a Heterogeneous Strike-Slip Fault (HFQsim) With Static/Kinetic Friction and Temperature-Dependent Creep." In: *Journal of Geophysical Research: Solid Earth* 130.6 (June 2025), e2024JB030680. ISSN: 2169-9313. DOI: 10.1029/2024JB030680. URL: https://agupubs.onlinelibrary.wiley.com/doi/full/10.1029/2024JB030680 (visited on 01/26/2026).