Electronic Seismologist

FD3D_TSN: A Fast and Simple Code for Dynamic Rupture Simulations with GPU Acceleration

Jan Premus*[1], František Gallovič[1], Ladislav Hanyk[1], and Alice-Agnes Gabriel[2]

Abstract

We introduce FD3D_TSN—an open-source Fortran code for 3D dynamic earthquake rupture modeling based on the staggered grid fourth-order finite-difference method employing a regular cubical spatial discretization. Slip-weakening and fast-velocity-weakening rate-and-state fault friction laws are combined with vertical planar fault geometry, orthogonal to a planar free surface. FD3D_TSN demonstrates good agreement with other methods in a range of benchmark exercises of the Southern California Earthquake Center and U.S. Geological Survey dynamic rupture code verification project. Efficient graphic processing units (GPU) acceleration using the OpenACC framework yields a factor of 10 speed-up in terms of time to solution compared to a single-core solution for current hardware (Intel i9-9900K and Nvidia RTX 2070). The software is fast and easy-to-use and suitable explicitly for data-driven applications requiring a large number of forward simulations such as dynamic source inversion or probabilistic ground-motion modeling. The code is freely available for the scientific community and may be incorporated in physics-based earthquake source imaging and seismic hazard assessment, or for teaching purposes.

Cite this article as Premus, J., F. Gallovič, L. Hanyk, and A.-A. Gabriel (2020). FD3D_TSN: A Fast and Simple Code for Dynamic Rupture Simulations with GPU Acceleration, *Seismol. Res. Lett.* **91**, 2881–2889, doi: 10.1785/0220190374.

Supplemental Material

Introduction

Earthquakes are caused by the sudden release of accumulated elastic strain energy. Rupture propagates as a discontinuity on a pre-existing fault surface, governed by prestress and friction. The mathematical description of a so-called dynamic rupture model leads to a nonlinear mixed-boundary problem. Although some simple models of rupture propagating at prescribed speeds in 2D have analytical solutions available in closed forms (Kostrov, 1964; Aki and Richards, 2002), more complex models accounting for spontaneous propagation of the rupture require a numerical solver.

Dynamic rupture simulations have been undergoing rapid development in the past decades, starting with finite difference (FD; Andrews, 1976; Madariaga, 1976; Mikumo and Miyatake, 1978; Day, 1982) and boundary integral (e.g., Burridge, 1969; Das, 1980; Koller *et al.*, 1992) methods. Later, finite element (e.g., Oglesby *et al.*, 1998; Barall, 2006), spectral element (e.g., Festa and Vilotte, 2005; Kaneko *et al.*, 2008), and discontinuous Galerkin (de la Puente *et al.*, 2009; Pelties *et al.*, 2012; Tago *et al.*, 2012) methods were developed.

Empowered by supercomputing (e.g., Heinecke *et al.*, 2014; Ichimura *et al.*, 2014; Uphoff *et al.*, 2017), 3D dynamic rupture earthquake simulation software has reached the capability of accounting for increasingly complex geometrical (e.g., Ulrich *et al.*, 2019) and physical (e.g., Roten *et al.*, 2014, 2016; Wollherr *et al.*, 2019) modeling components in high-resolution single-event scenarios. Yet, the computational cost of such dynamic rupture models hinders applications requiring a large number of simulations.

Inverting strong ground motions using dynamic source models is challenging due to the nonlinear relationship between source parameters (prestress and parameters of the friction law) and seismograms. Typical numbers of required trial models are then hundreds of thousands to millions (Gallovič *et al.*, 2019a; Mirwald *et al.*, 2019). Similarly, high numbers of simulations are needed in physics-based scenario ground-motion modeling or parametric studies in general (e.g., Peyrat *et al.*, 2001; Cui *et al.*, 2013). Particular examples are seismic cycle simulations that span periods of thousands of years, generating synthetic catalogs of earthquakes (Erickson *et al.*, 2020). To enable such applications, the computational requirements of the forward simulation have to be reduced not only by simplifying the physical model of the fault and the surrounding medium but also using efficient numerical methods such as

1. Department of Geophysics, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic; 2. Department of Earth and Environmental Sciences, Geophysics, Ludwig-Maximilians-Universität München, München, Germany

*Corresponding author: premus@karel.troja.mff.cuni.cz

presented here. Having simplified and fast rupture dynamics software (Daub, 2016; Krischer *et al.*, 2018) is also useful for teaching and training purposes and allows the first acquaintance with dynamic rupture simulations. Such codes can be utilized by all researchers interested in earthquake physics even without immediate access to supercomputing power required by the most advanced codes.

The FD operators in FD3D_TSN are adapted and extended from the 3D FD code FD3D by R. Madariaga (Madariaga *et al.*, 1998). They utilize regular cubic grids, and are relatively simple and computationally efficient (Levander, 1988). Their use leads to a high speed of the calculation and is allowed by considering a simple fault geometry. Discontinuous velocity components at the fault are antisymmetric, allowing for the simplification of the fault boundary condition. Consequently, only one side of the fault needs to be calculated, cutting the required computational capacities (central processing unit [CPU] time and memory storage) in half. The major factor influencing the accuracy of FD3D_TSN is the here used implementation of the fault boundary condition (available in the supplemental material to this article). We use the traction-at-split-node approach (Dalguer and Day, 2007), which leads to an accuracy comparable with other codes.

Although many rupture propagation codes provide parallelization using OpenMP and Message Passing Interface (MPI), graphic processing units (GPU) acceleration is gaining attention only recently. GPU acceleration of wave-propagation codes using FD and finite-element methods was implemented by Michéa and Komatitsch (2010) and Komatitsch *et al.* (2009) using Compute Unified Device Architecture (CUDA). CUDA is also featured by the publicly available spectral element code SPECFEM3D (Komatitsch *et al.*, 2010) and the FD code AWP-ODC (Zhou *et al.*, 2013). The RAJA library is used to accelerate the SW4 code (Rodgers *et al.*, 2019).

To foster the GPU acceleration in the field of rupture dynamics, we utilize the easy-to-use OpenACC approach to optionally port the code to GPUs. We demonstrate that the speed-up can be significant, by one order of magnitude when using commonly available GPUs. We note that OpenACC, in comparison to the CUDA framework, allows also for non-Nvidia accelerator support, which will become increasingly important as demonstrated by future U.S. exascale machines relying on AMD GPU accelerators.

The present article introduces the main ingredients of FD3D_TSN, namely the FD method and implementation of boundary conditions. We describe the GPU acceleration of the code using OpenACC directives. Then, we perform verification exercises using benchmarks from the Southern California Earthquake Center and U.S. Geological Survey (SCEC-USGS) Spontaneous Rupture Code Verification Project (Harris *et al.*, 2018). We show that our results are on par with those of other codes and compare the wall clock times of the serial (one core) baseline and GPU-accelerated versions of our code. The code

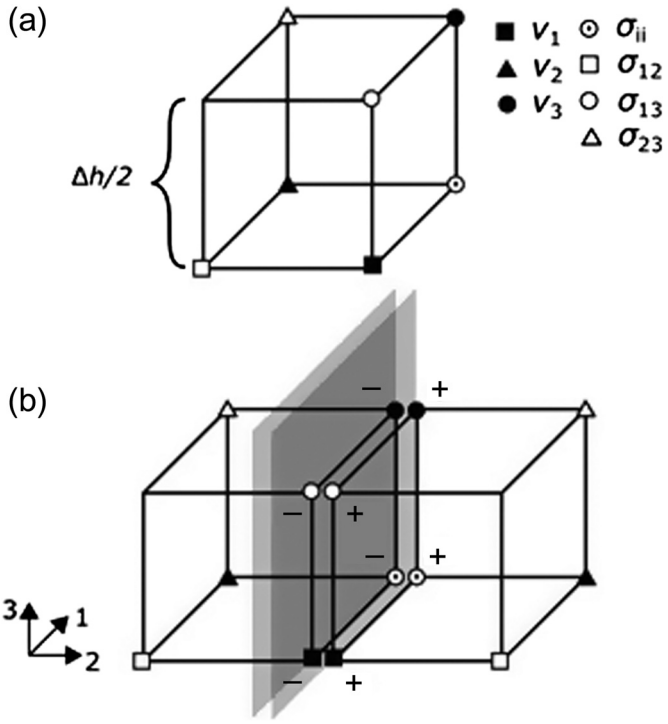


Figure 1. (a) Finite-difference (FD) grid cube with staggered positions of components of velocity v and stress σ (symbols). (b) Fault in the staggered grid. Fault (gray plane) is positioned to overlay with one of the grid planes. Discontinuous components of velocity and stress are treated separately for its "–" and "+" side.

FD3D_TSN is freely available on GitHub (see Data and Resources), together with a detailed method description and examples.

Method

Essential ingredients of a dynamic earthquake simulation code are: (1) discrete solver of the elastodynamic equation, (2) implementation of physical boundary conditions (fault and free surface), and (3) nonreflecting (artificial) boundaries simulating the free flow of energy outside of the computational domain. Here, we briefly overview these features; for details including the concrete operators used in the code, see the supplemental material and online documentation on GitHub (see Data and Resources).

FD3D_TSN solves the elastodynamic equation in the classical velocity–stress formulation. Central FD operators of the fourth-order in space (Levander, 1988; Madariaga *et al.*, 1998) are used to discretize it on a 3D-staggered regular grid. Figure 1a shows the spatial position of field variables in a grid cell.

The free surface is implemented using the stress imaging technique (Graves, 1996) at the upper boundary of the computational domain. Perfectly matched layers (PMLs; Berenger, 1994) in a classical split form (Kristek *et al.*, 2009) are placed at

the remaining borders as artificial absorbing conditions. The free surface can be optionally turned off and replaced by an absorbing boundary when, for example, a deep event is to be simulated. Although the PMLs are more demanding in terms of both computational power and memory storage, the absence of reflections for a wide range of incidence angles permits a significant reduction of the size of the computational domain in practical applications.

Frictional boundary conditions at the fault surface are implemented using the traction-at-split-node method (Dalguer and Day, 2007), assuming a general 2D slip vector. We consider the same material parameters on both sides of the fault. This simplifies the governing traction-at-split-node equations because discontinuous velocity and stress components are antisymmetric, whereas the continuous ones are symmetric across the fault. Moreover, the wave-propagation FD operators can be calculated on one side of the fault only, which cuts the computational and memory demands of solving the problem in half. The staggered grid complicates the enforcement of the fault boundary condition slightly because the horizontal and vertical components of the slip rate and traction (equal to shear stress for the vertical fault) are calculated at different nodes. To correct for this, we use bilinear interpolation of the missing components when evaluating the absolute values of slip rate and traction in the friction law. Details of the implementation are presented in the supplemental material.

Friction law

FD3D_TSN features two friction laws: (1) classical slip-weakening law (Ida, 1972; Andrews, 1976) and (2) rate-and-state law with rapid velocity weakening at high-slip rates (Ampuero and Ben-Zion, 2008; Beeler *et al.*, 2008), as proposed by Dunham *et al.* (2011). Although the numerical implementation of the former is straightforward, the latter needs a specific approach described in the following.

Following Rojas *et al.* (2009), we consider the state variable and slip rate staggered in time and extend their method to 3D. The evolution equation for the state variable can be analytically integrated over the timestep in which the slip rate is considered constant and equal to its value in the middle of the discrete time interval. A similar approach to the differential equation for the slip rate leads to a nonlinear equation requiring the application of a nonlinear solver. Because we can calculate the analytical derivative of the equation and provide the last timestep solution as the first guess, we employ the Newton method that is suitable under such conditions and converges rather quickly. The additional computational cost for solving the nonlinear equation at every node on the fault surface is therefore negligible, being just a few percent of the computational time for the cases presented in the Verification Exercise and Performance section. More details on the implementation of the fast-velocity-weakening friction can be found in the supplemental material.

GPU acceleration

GPU acceleration of the FD3D_TSN code employs the OpenACC programming model (see Data and Resources). OpenACC is a standardized directive-based system for parallel processing on heterogeneous platforms that consist of a host (CPU) and compute accelerators, in particular, GPUs. OpenACC directives make it possible to parallelize the code and offload computationally intensive parts to the accelerators, whereas the possibility to compile the same code for a serial run is retained, similarly to the OpenMP programming model (see Data and Resources). OpenACC has been implemented in a few C, C++, Fortran compilers. For x86-64 hardware architecture and Linux and Windows systems, the Portland Group, Inc. (PGI, see Data and Resources) offers no-cost community edition of compilers and tools, besides commercial professional edition, and also the free GNU compiler collection (GCC) is reaching with OpenACC mature state (see Data and Resources). PGI, as a brand of Nvidia Corporation, not only targets Nvidia accelerators solely but also allows running OpenACC-parallelized programs on multicore CPUs. GCC can offload to Nvidia GPUs already and plans to target AMD GPUs (see Data and Resources).

The FD3D_TSN code optimizes data locality to minimize relatively slow data transfers between the host and GPU memory through the PCIe interface. Specifically, input data are moved into the GPU memory by one single transfer using the OpenACC directive !$ACC DATA COPYIN; substantially, smaller output data are copied back to the host memory every timestep using the directive !$ACC DATA COPY, and no more data between the host and GPU are transferred in the course of computation. Parallelizable regions of nested loops are surrounded by the directives !$ACC PARALLEL and !$ACC LOOP COLLAPSE (3). Being more specific, for example, specifying more clauses of the LOOP directive or proposing nondefault values of gang and worker sizes and vector length (i.e., CUDA grid and block sizes for controlling the distribution of threads among GPU multiprocessors) did not improve the performance. On the other hand, the clause COLLAPSE (3) is essential for performance, as it allows accumulating a large number of iterations into one collapsed loop, which is a desirable feature for the GPU acceleration.

Using the Code

The Fortran source code with documentation, several examples, and basic MATLAB plotting tools are freely available in a public repository on GitHub (see Data and Resources). The examples on GitHub comprise four SCEC-USGS benchmark exercises, specifically TPV5, TPV8, TPV9, and TPV104, offering the possibility of verification of the code. In addition, a dynamic rupture model of the 2016 Amatrice earthquake (Gallovič *et al.*, 2019b) is shown, to demonstrate the use of a general input file forwardmodel.dat containing spatially varying dynamic parameters.

Compilation and input files

Slip-weakening and fast-velocity-weakening friction use different sets of dynamic parameters. Moreover, the slip rates are calculated by two different algorithmic implementations. The user can thus select between the friction laws by preprocessor flag -DFVW, switching from the default slip-weakening to the fast-velocity-weakening friction.

There are two ways to generate input parameters for a dynamic model in FD3D_TSN—using a hardcoded function or input dynamic parameters through the file forwardmodel.dat. Hardcoded function needs to be specified during compilation, using preprocessor flags. SCEC-USGS benchmarks require a combination of two flags: -DSCEC and a chosen problem (-DTPV5, -DTPV8, -DTPV9, or -DTPV104). When none of these flags are present, forwardmodel.dat file is used as the input instead. This file contains a spatial distribution of dynamic parameters in the form of one line of values of prestress, static friction coefficient, and characteristic slip-weakening distance on an equidistant grid at the fault, starting in the left corner. The grid can be coarser than the FD grid; the values are bilinearly interpolated to the FD grid (Gallovič *et al.*, 2019a,b). The size of the coarser grid (amount of points in the horizontal and vertical directions) only needs to be set in the file inputinv.dat.

FD3D_TSN works with any horizontally layered 1D velocity model; its parameters are saved in the file crustal.dat. File inputfd3d.dat contains the information necessary for the initialization of the FD discretization and the PMLs, namely the size of the grid, discretization intervals in space and time, and PML thickness and damping.

Because the source geometry is simple (2D planar and vertical), hard-coding another model, including its potential parameterization, is straightforward. This way, the code can be easily adapted for various user's needs such as a specific parametric study or a particular dynamic inversion. Code FD3D_TSN_PT represents an example of the latter following the inversion strategy of Gallovič *et al.* (2019a,b), which uses FD3D_TSN as the forward solver, see Data and Resources.

Output files

On-fault slip rate and shear stress time series are stored in files sliprateX.res, sliprateZ.res, shearstressX.res, shearstressZ.dat, that is, for the horizontal (X) and vertical (Z) components separately. When the fast-velocity-weakening friction is enabled, the time series of the state variable is stored in file psi.res. Given the amount of data stored, these files are in binary format. Two MATLAB files are provided to demonstrate the retrieval of data from the binary files. User can acquire and plot time series of components of slip rate and shear stress at a given position (PrintSeries.m), or their spatial distribution at a given time (PrintSnapshot.m).

Spatial distributions of slip, rupture time, rise time, rupture velocity, and stress drop are generated and stored in separate text files. Besides, a local estimate of the cohesive zone (the area behind the rupture front in which shear stress decreases from its static to its dynamic value, Day *et al.*, 2005; Wollherr *et al.*, 2018) is provided in the file czone.dat for inspection of the numerical accuracy.

Synthetic seismograms are generated at chosen nodes in the form of time series of separate velocity components stored in files stan%i.dat, in which %i numbers the node. The user needs to set the number of nodes and their position on the grid in the file inputfd3d.dat before simulation. A more detailed and up-to-date description of the input and output files can be found in the online documentation at the GitHub repository.

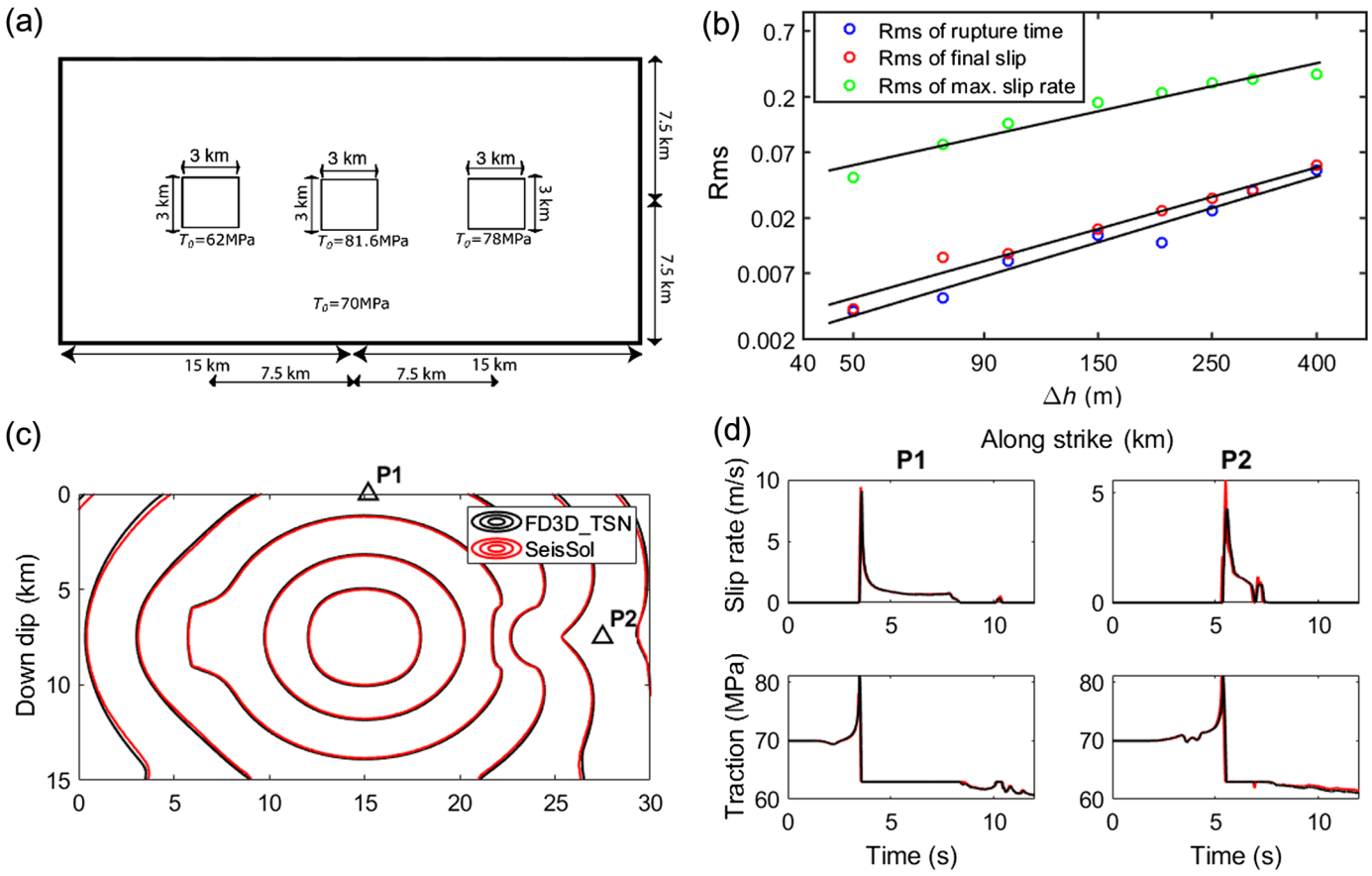
Verification Exercise and Performance

We have performed several exercises from the community benchmark suite of the SCEC-USGS Spontaneous Rupture Code Verification Project (Harris *et al.*, 2018). Their main webpage (see Data and Resources) contains descriptions of the exercises and results generated by 5–20 different solvers, depending on the exercise. There are three types of results for comparison: time evolution of physical quantities at prescribed points on the fault (slip, slip rate, shear stress, and state variable if applicable), seismograms at off-fault positions, and rupture time contour plots, that is, lines showing the rupture tip position at every half second. Here, we present our results for benchmarks TPV5 (slip-weakening friction) and TPV104 (fast-velocity-weakening friction). In addition, the results for the TPV8 and TPV9 benchmarks can be found on the SCEC-USGS webpage (see Data and Resources).

For the TPV5 benchmark, the size of the computational domain is $31 \times 10 \times 15.5$ km, whereas, for the TPV104 benchmark, it is larger ($40 \times 10 \times 20$ km) because the model contains additional velocity-strengthening layers around the fault following the benchmark definition. Figures 2 and 3 display results for TPV5 (spatial grid step $\Delta h = 100$ m) and TVP104 (spatial grid step $\Delta h = 50$ m), respectively. They are presented in terms of the rupture-time contours along the fault and the on-fault slip rate and shear–stress time series at two locations. To better illustrate the TPV5 simulation, Figure 4 shows snapshots of both slip rate and wavefield around the fault at equidistant times.

In both TPV5 and TVP104, the difference between the results of FD3D_TSN and code SeisSol (Heinecke *et al.*, 2014; Pelties *et al.*, 2014; Uphoff *et al.*, 2017) are well within the typical variation among other solvers (Harris *et al.*, 2009, 2011, 2018; Barall and Harris, 2015). We note that the full set of the FD3D_TSN simulation results are uploaded in the benchmark database of solutions. At the benchmark website, our solutions can be compared with other code outputs and physical quantities on and off the fault.

We note small differences in rupture times for different solvers in the fast-velocity-weakening friction benchmark TPV104. The solution of FD3D_TSN agrees very well with that of SPECFEM3D (Kaneko *et al.*, 2008), whereas the solutions of



FaultMod (Barall, 2006) and SeisSol show slightly slower rupture (Fig. 3). These differences remain visible with mesh refinement (compare to the $\Delta h = 50$ m solution). We understand these differences as an expression of differences in the implementation of rate-and-state friction evolution, but not restricted to staggered grid-specific choices.

Following Day *et al.* (2005), we calculate the root mean square (rms) difference of the spatially averaged slip, rupture time, and peak slip rate for various grid sizes Δh relative to a high-resolution reference solution with$\Delta h = 25$ m $\Delta h = 25$ m, see Figures 2b and 3b. Spatial discretizations $\Delta h = 100$ m and $\Delta h = 50$ m for benchmarks TPV5 and TPV104, respectively, are chosen for providing results sufficiently close to the reference solution (both rupture time and slip rms measures are less than 1%). The slopes of the rms misfits from both benchmarks are ~1.1, ~1.2, and ~1.3 for maximum slip rate, slip, and rupture time, respectively.

Wall clock times of the code for both grid spacings Δh and problems are shown in Tables 1 and 2, respectively, both for the serial (running on one CPU core) and GPU-accelerated versions. We used CPU Intel i9-9900K and GPU Nvidia RTX 2070 hardware to run these tests. When using the GPU, it is essential to keep all variables in the GPU memory, because any data transfer to and from GPU memory is the major bottleneck affecting the overall speed of simulation. The amount of GPU memory is generally smaller (several GB) than the computer RAM, which limits the maximum size of the model. Memory requirements of these simulations were comfortably below this limit of 8 GB for Nvidia RTX 2070, see Tables 1 and 2 for details.

Figure 2. Results for the TPV5 benchmark (heterogenous prestress). (a) Schematic showing dimensions of the fault and positions of prestress heterogeneities T_0 (adopted from the Southern California Earthquake Center and U.S. Geological Survey [SCEC-USGS] benchmark webpage). (b) A plot of the root mean square (rms) difference of the rupture time, slip, and maximum slip rate relative to a solution with $\Delta h = 25$ m as a function of Δh for FD3D_TSN. (c) Rupture front contours on the fault plane every 1 s. (d) Slip rates and shear stress (equal to the horizontal component of traction) at points P1 and P2, corresponding to fault receivers "st000dp000" and "st120dp075" in Harris *et al.* (2018). For their position, see panel (a). Simulation results for FD3D_TSN with $\Delta h = 100$ m and SeisSol (Pelties *et al.*, 2012) are denoted by black and red lines, respectively. The color version of this figure is available only in the electronic edition.

Discussion and Conclusions

This article presents FD3D_TSN, a simple-to-use FD code for dynamic earthquake source modeling assuming slip-weakening and fast-velocity-weakening rate-and-state friction laws. The code is written in the Fortran programming language and is freely available under the GNU General Public License license. The optional GPU acceleration using OpenACC

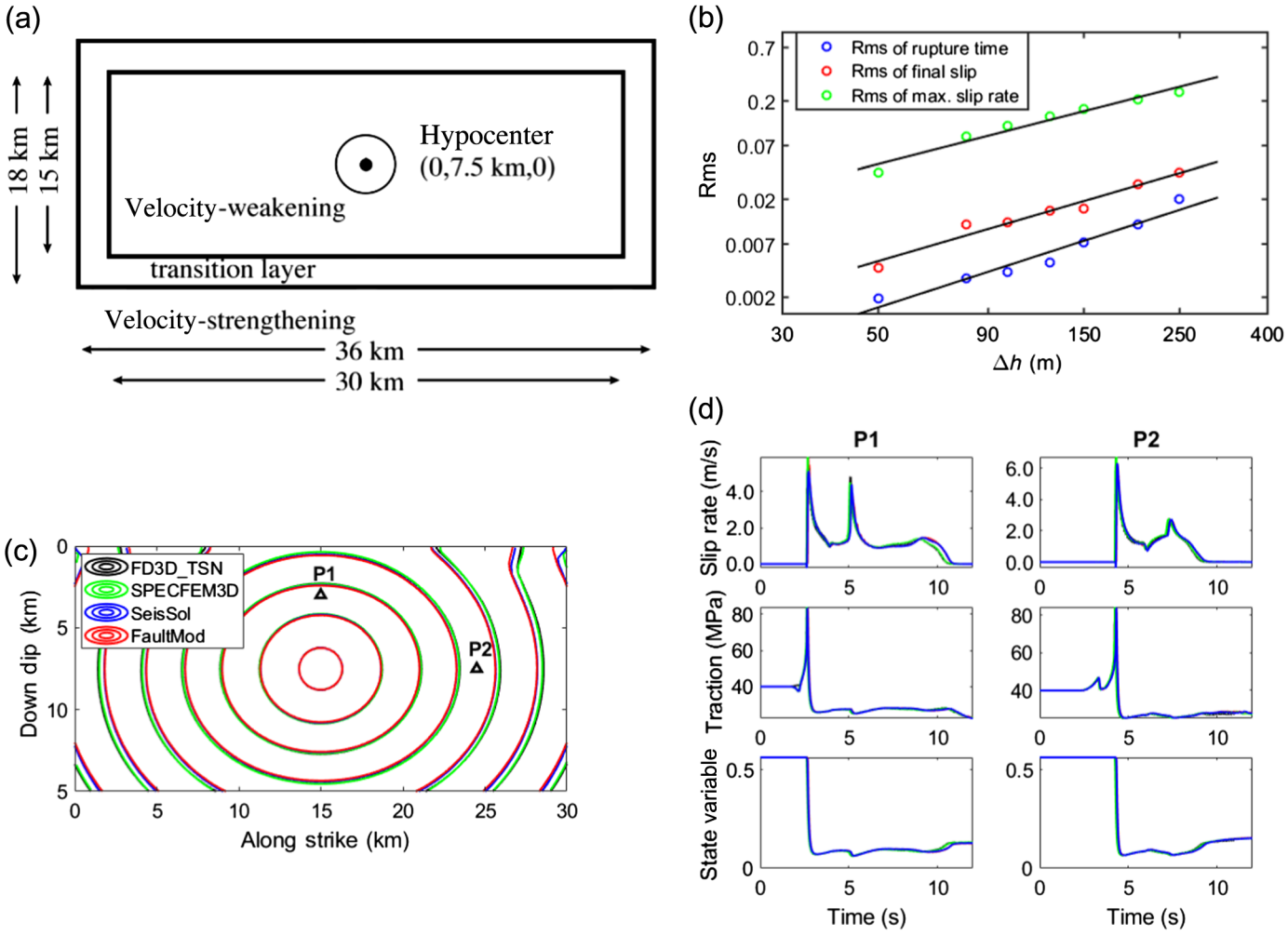


Figure 3. Same as Figure 2 but for the rate-and-state friction TPV104 benchmark. (a) Schematic diagram of the dimensions of the fault and positions of the velocity-weakening and strengthening zones (rectangles) and the nucleation zone (circle), as adopted from the SCEC-USGS webpage. (b) A plot of the root mean square (rms) difference of the rupture time, slip, and maximum slip rate relative to a solution with Δh 25 mas a function of Δh for FD3D_TSN. (c) Rupture front contours on the fault plane every 1 s. (d) The time evolution of the state variable. Solutions of FD3D_TSN with $\Delta h = 50$ m (black line), SPECFEM3d (green line), SeisSol (red line), and FaultMod (blue line) are shown. Nodes P1 and P2 correspond to fault receivers "faultst000dp030" and "faultst090dp075," respectively. The color version of this figure is available only in the electronic edition.

compiler directives (currently provided in PGI and GNU Fortran) can shorten the wall clock time by a factor of 10 compared to the same version of the code running on a single CPU core (for the hardware configuration with CPU Intel i9-9900K and GPU Nvidia RTX 2070).

FD3D_TSN is original in its high precision and fast speed. The latter is achieved by simplifying the geometry of the problem. At present, FD3D_TSN is limited to a vertical planar fault geometry, orthogonal to the planar free surface, embedded in an elastic medium with homogeneous or depth-dependent velocities and density. The planar fault orthogonal to the free surface is optimal for the regular grid discretization. Moreover, the consequent antisymmetry of the velocity and stress permits the calculations on one side of the fault only.

We admit that the model simplifications may restrict the range of possible applications of the code. However, in some cases, the simplified model can be considered as an approximation of the true model. For example, if the rupture does not reach the Earth's surface, a dipping fault geometry can be approximated by stretching the velocity model vertically to respect the original along-dip position of the fault intersections with the velocity model layers (Gallovič *et al.*, 2019b). In other cases, it may be straightforward to extend the code to incorporate specific features. For example, the effect of a low-velocity fault zone could be modeled by introducing velocity reduction around the fault, when obeying the model symmetry across the fault. Nevertheless, there are many more possible further developments such as the implementation of dipping free-surface topography (e.g., Robertsson, 1996) to correctly model the interaction between the free surface and the rupture propagating along a dipping fault.

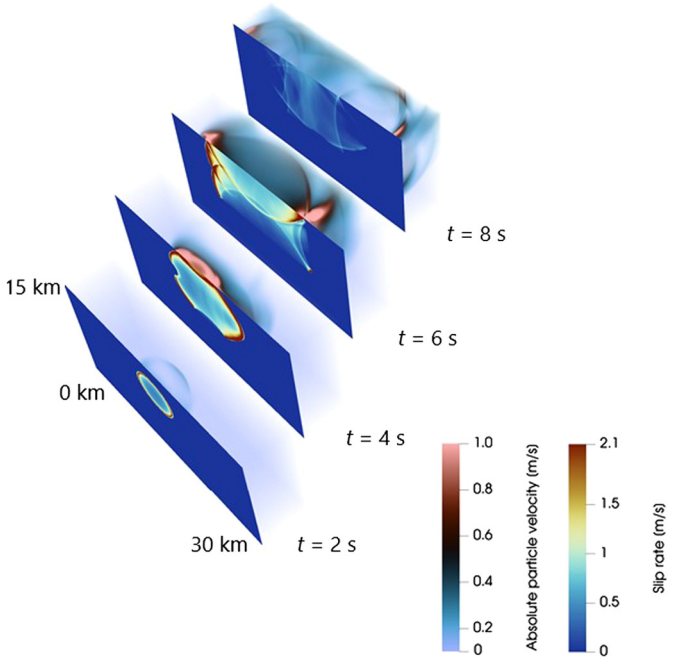


Figure 4. 3D plot animating the slip rate along the fault and the radiated wavefield for the TPV5 benchmark with heterogeneous initial stress at equidistant times, after the rupture nucleation in the middle of the fault. See also Figure 2 for more information on the problem setup. The color version of this figure is available only in the electronic edition.

Despite the simplification, FD3D_TSN has already proven useful in several applications. It is implemented as a new forward solver in Bayesian dynamic source inversion code FD3D_TSN_PT, see Gallovič *et al.* (2019a,b). Gallovič and Valentová (2020) utilized FD3D_TSN to simulate a large number of rupture scenarios complying with ground-motion prediction equations for stress-drop analysis. We foresee its usability also in parametric studies and in teaching the fundamentals of rupture dynamics to students, as it can be used without access to supercomputing facilities. Generally speaking, we believe that having such an efficient code and making it available to the community will boost research toward data-driven dynamic earthquake source studies and physics-based scenario modeling.

Data and Resources

The code FD3D_TSN and the online documentation of GitHub are available at https://github.com/JanPremus/fd3d_TSN, together with documentation and examples—benchmarks TPV5, TPV104, TPV8, TPV9, and the 2016 Amatrice earthquake model. The supplemental material for this article includes detailed description of the implementation of the fault boundary condition and the fast-velocity-weakening friction. Strong-motion inversion by parallel tempering code FD3D_TSN_PT that uses FD3D_TSN as a forward solver can be found at https://github.com/fgallovic/fd3d_tsn_pt. The original code FD3D by R. Madariaga (Madariaga *et al.*, 1998) can be downloaded at http://www.geologie.ens.fr/~madariag/Programs/programs.html. Southern California Earthquake Center and U.S. Geological Survey (SCEC-USGS) benchmark descriptions and results are available at http://scecdata.usc.edu/cvws/. Graphic processing units' (GPU) acceleration using OpenACC directives is possible using Portland Group, Inc. (PGI) community edition compilers and tools, available at https://www.pgroup.com/products/community.htm, or GNU compiler collection (GCC) version 8 (v.8 and higher with offloading support), at https://gcc.gnu.org/. The information on OpenACC programming model is available at https://www.openacc.org and the data about OpenMP programming model are available at https://www.openmp.org. Data about the PGI are available at https://www.pgroup.com. The information about free GCC is available at https://gcc.gnu.org/wiki/OpenACC and data about GCC that can offload to Nvidia GPUs already and plans to target AMD GPUs are available at https://gcc.gnu.org/wiki/Offloading. The MATLAB is available at www.mathworks.com/products/matlab. All websites were last accessed in March 2020.

TABLE 1
Wall Clock Times of the FD3D_TSN Code for the TPV5 Benchmark Model, Shown for Two Finite-Difference (FD) Spatial Discretizations and for Single Core (Intel i9-9900K) and GPU-Accelerated Versions (Nvidia RTX 2700)

Discrete Step (m)	Degrees of Freedom × Timesteps (RAM Requirement)	Single-Core Wall Clock Time (s)	GPU-Accelerated Wall Clock Time (s)
100	54,558,900 × 3000 (210 MB)	0:3:12	0:0:22
50	389,491,200 × 6000 (1490 MB)	0:39:13	0:3:47

Note that the times do not include saving of the results to the disk.

TABLE 2
Wall Clock Times of the FD3D_TSN Code for the TPV104 Benchmark Model, Shown for Two Finite-Difference (FD) Spatial Discretizations and for Single Core (Intel i9-9900K) and GPU-Accelerated Versions (Nvidia RTX 2700)

Discrete Step (m)	Degrees of Freedom × Timesteps (RAM Requirement)	Single-Core Wall Clock Time (s)	GPU-Accelerated Wall Clock Time (s)
100	88,149,600 × 3000 (340 MB)	0:4:28	0:0:22
50	638,517,600 × 6000 (2440 MB)	0:57:46	0:4:56

GPU, graphic processing units.

Acknowledgments

The authors thank R. Harris and an anonymous reviewer for their valuable comments that improved the presentation of their results. The authors acknowledge financial support through the bilateral project of the Czech Science Foundation and Deutsche Forschungsgemeinschaft (DFG), 18-06716J and GA 2465/2-1, respectively. Jan Premus acknowledges the support from Charles University Grant SVV 115-09/260447. Alice-Agnes Gabriel acknowledges the additional support by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (TEAR, Grant Agreement Number 852992 and ChEESE, Grant Agreement Number 823844).

References

Aki, K., and P. G. Richards (2002). *Quantitative Seismology*, University Science Books, Sausalito, California.

Ampuero, J. P., and Y. Ben-Zion (2008). Cracks, pulses and macroscopic asymmetry of dynamic rupture on a bimaterial interface with velocity weakening friction, *Geophys. J. Int.* **173,** no. 2, 674–692.

Andrews, D. J. (1976). Rupture velocity of plane strain shear cracks, *J. Geophys. Res.* **81,** no. 32, 5679–5687.

Barall, M. (2006). A grid-doubling finite-element technique for calculating dynamic three-dimensional spontaneous rupture on an earthquake fault, *Geophys. J. Int.* **178,** no. 2, 845–859.

Barall, M., and R. A. Harris (2015). Metrics for comparing dynamic earthquake rupture simulations, *Seismol. Res. Lett.* **86,** no. 1, 223–235.

Beeler, N., T. Tullis, and D. Goldsby (2008). Constitutive relationships and physical basis of fault strength due to flash heating, *J. Geophys. Res.* **113,** no. B01401, doi: 10.1029/2007JB004988.

Berenger, J.-P. (1994). A perfectly matched layer for the absorption of electromagnetic waves, *J. Comput. Phys.* **114,** no. 2, 185–200.

Burridge, R. (1969). The numerical solution of certain integral equations with non-integrable kernels arising in the theory of crack propagation and elastic wave diffraction, *Phil. Trans. Roy. Soc. Lond. A* **265,** 353–381.

Cui, Y., E. Poyraz, K. B. Olsen, J. Zhou, K. Withers, S. Callaghan, J. Larkin, C. D. Choi, A. Chourasia, Z. Shi, *et al.* (2013). Physics-based seismic hazard analysis on petascale heterogeneous supercomputers, *SC '13: Proc. of the International Conf. on High Performance Computing, Networking, Storage and Analysis*, Denver, Colorado, 1–12, doi: 10.1145/2503210.2503300.

Dalguer, L. A., and S. M. Day (2007). Staggered-grid split-node method for spontaneous rupture simulation, *J. Geophys. Res.* **112,** no. B02302, doi: 10.1029/2006JB004467.

Das, S. (1980). A numerical method for determination of source time functions for general three-dimensional rupture propagation, *Geophys. J. Int.* **62,** no. 3, 591–604.

Daub, E. G. (2016). Meet a new code: Daub finite difference, *Presentation at the March 2016 SCEC Rupture Dynamics Code Validation Workshop*, Pomona, California, 11 March 2016, available at http://scecdata.usc.edu/cvws/download/mar11_2016/Daub_cvws_0311.pdf (last accessed March 2020).

Day, S. M. (1982). Three-dimensional finite difference simulation of fault dynamics: Rectangular faults with fixed rupture velocity, *Bull. Seismol. Soc. Am.* **72,** 705–727.

Day, S. M., L. A. Dalguer, N. Lapusta, and Y. Liu (2005). Comparison of finite difference and boundary integral solutions to three-dimensional spontaneous rupture, *J. Geophys. Res.* **110,** no. B12307, doi: 10.1029/2005JB003813.

de la Puente, J., J.-P. Ampuero, and M. Käser (2009). Dynamic rupture modeling on unstructured meshes using a discontinuous Galerkin method, *J. Geophys. Res.* **114,** no. B10302, doi: 10.1029/2008JB006271.

Dunham, E. M., D. Belanger, L. Cong, and J. E. Kozdon (2011). Earthquake ruptures with strongly rate-weakening friction and off-fault plasticity, Part 1: Planar faults, *Bull. Seismol. Soc. Am.* **101,** 2296–2307.

Erickson, B., J. Jiang, M. Barall, N. Lapusta, E. M. Dunham, R. Harris, L. S. Abrahams, K. L. Allison, J.-P. Ampuero, S. Barbot, *et al.* (2020). The community code verification exercise for simulating sequences of earthquakes and aseismic slip (SEAS), *Seismol. Res. Lett.* doi: 10.1785/0220190248.

Festa, G., and J. Vilotte (2005). The Newmark scheme as velocity-stress time staggered: An efficient PML implementation for spectral element simulations of electrodynamics, *Geophys. J. Int.* **161,** 789–812.

Gallovič, F., and Ľ. Valentová (2020). Earthquake stress drops from dynamic rupture simulations constrained by observed ground motions, *Geophys. Res. Lett.* **47,** e2019GL085880, doi: 10.1029/2019GL085880.

Gallovič, F., Ľ. Valentová, J.-P. Ampuero, and A.-A. Gabriel (2019a). Bayesian dynamic finite-fault inversion: 1. Method and synthetic test, *J. Geophys. Res.* **124,** 6949–6969.

Gallovič, F., Ľ. Valentová, J.-P. Ampuero, and A.-A. Gabriel (2019b). Bayesian dynamic finite-fault inversion: 2. Application to the 2016 M_w 6.2 Amatrice, Italy, earthquake, *J. Geophys. Res.* **124,** 6970–6988.

Graves, R. W. (1996). Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences, *Bull. Seismol. Soc. Am.* **86,** no. 4, 1091–1106.

Harris, R. A., M. Barall, B. Aagaard, S. Ma, D. Roten, K. Olsen, B. Duan, D. Liu, B. Luo, K. Bai, *et al.* (2018). A suite of exercises for verifying dynamic earthquake rupture codes, *Seismol. Res. Lett.* **89,** no. 3, 1146–1162.

Harris, R. A., M. Barall, D. J. Andrews, B. Duan, S. Ma, E. M. Dunham, A.-A. Gabriel, Y. Kaneko, Y. Kase, B. T. Aagaard, *et al.* (2011). Verifying a computational method for predicting extreme ground motion, *Seismol. Res. Lett.* **82,** no. 5, 638–644.

Harris, R. A., M. Barall, R. Archuleta, E. Dunham, B. Aagaard, J. P. Ampuero, H. Bhat, V. Cruz-Atienza, L. Dalguer, P. Dawson, *et al.* (2009). The SCEC/USGS dynamic earthquake rupture code verification exercise, *Seismol. Res. Lett.* **80,** no. 1, 119–126.

Heinecke, A., A. Breuer, S. Rettenberger, M. Bader, A.-A. Gabriel, C. Pelties, A. Bode, W. Barth, X.-K. Liao, K. Vaidyanathan, *et al.* (2014). Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers, *Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis SC14*, New Orleans, Louisiana, 16–21 November.

Ichimura, T., K. Fujita, S. Tanaka, M. Hori, L. Maddegedara, Y. Shizawa, and H. Kobayashi (2014). Physics-based urban earthquake simulation enhanced by 10.7 BlnDOF × 30 K time-step unstructured FE non-linear seismic wave simulation, *SC14: International Conf. for High Performance Computing, Networking, Storage and Analysis*, New Orleans, Louisiana, 16–21 November, 15–26.

Ida, Y. (1972). Cohesive force across the tip of a longitudinal-shear crack and Griffith's specific surface energy, *J. Geophys. Res.* **77,** no. 20, 3796–3805.

Kaneko, Y., N. Lapusta, and J.-P. Ampuero (2008). Spectral-element modeling of spontaneous earthquake rupture on rate and state faults: Effect of velocity-strengthening friction at shallow depths, *J. Geophys. Res.* **113,** no. B0931, doi: 10.1029/2007JB005553.

Koller, M., M. Bonnet, and R. Madariaga (1992). Modelling of dynamical crack propagation using time-domain boundary integral equations, *Wave Motion* **16,** 339–366.

Komatitsch, D., P. Le Loher, D. Michéa, G. Erlebacher, and D. Göddeke (2010). Accelerating spectral-element and finite-difference wave propagation algorithms using a cluster of GPU graphics cards, *Workshop Défis Actuels De La Modélisation Électromagnétique: Gestion De La Complexité, Multi-Échelle, Multi-Physique, Gestion Des Incertitudes, Statistiques*, Saint Malo, France, 30 April 2010 (in French).

Komatitsch, D., D. Michea, and G. Erlebacher (2009). Porting a high-order finite-element earthquake modeling application to NVIDIA graphic cards using CUDA, *J. Parallel Distr. Comput.* **69,** no. 5, 451–460.

Kostrov, B. V. (1964). Selfsimilar problems of propagation of shear cracks, *J. Appl. Math. Mech.* **28,** no. 5, 1077–1087.

Krischer, L., Y. A. Aiman, T. Bartholomaus, S. Donner, M. van Driel, K. Duru, K. Garina, K. Gessele, T. Gunawan, S. Hable, *et al.* (2018). Seismo-Live: An educational online library of Jupyter notebooks for seismology, *Seismol.l Res. Lett.* **89,** no. 6, 2413–2419.

Kristek, J., P. Moczo, and M. Galis (2009). A brief summary of some PML formulations and discretizations for the velocity-stress equation of seismic motion, *Studia Geophysica et Geodaetica* **53,** 459–474.

Levander, A. (1988). Fourth-order finite-difference P-S, *Geophysics* **53,** 1425–1436.

Madariaga, R. (1976). Dynamics of an expanding circular fault, *Bull. Seismol. Soc. Am.* **66,** 639–666.

Madariaga, R., K. Olsen, and R. Archuleta (1998). Modeling dynamic rupture in a 3D earthquake fault model, *Bull. Seismol. Soc. Am.* **88,** no. 5, 1182–1197.

Michéa, D., and D. Komatitsch (2010). Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards, *Geophys. J. Int.* **182,** 389–402.

Mikumo, T., and T. Miyatake (1978). Dynamical rupture process on a three-dimensional fault with non-uniform frictions and near-field seismic waves, *Geophys. J. Int.* **54,** no. 2, 417–443.

Mirwald, A., V. M. Cruz-Atienza, J. Díaz-Mojica, A. Iglesias, S. K. Singh, C. Villafuerte, and J. Tago (2019). The 19 September 2017 (M_w 7.1) intermediate-depth Mexican earthquake: A slow and energetically inefficient deadly shock, *Geophys. Res. Lett.* **46,** 2054–2064.

Oglesby, D., R. Archuleta, and S. Nielsen (1998). Earthquakes on dipping faults: The effects of broken symmetry, *Science* **280,** 1055–1059.

Pelties, C., J. De la Puente, J.-P. Ampuero, G. B. Brietzke, and M. Käser (2012). Three-dimensional dynamic rupture simulation with a high-order discontinuous Galerkin method on unstructured tetrahedral meshes, *J. Geophys. Res.* **117,** no. B02309, doi: 10.1029/2011JB008857.

Pelties, C., A.-A. Gabriel, and J.-P. Ampuero (2014). Verification of an ADER-DG method for complex dynamic rupture problems, *Geosci. Model Dev.* **7,** no. 3), 847–866.

Peyrat, S., K. Olsen, and R. Madariaga (2001). Dynamic modeling of the 1992 Landers earthquake, *J. Geophys. Res.* **106,** no. B11, 26,467–26,482, doi: 10.1029/2001JB000205.

Robertsson, J. O. A. (1996). A numerical free-surface condition for elastic/viscoelastic finite-difference modeling in the presence of topography, *Geophysics* **61,** no. 6, 1921–1934.

Rodgers, A. J., R. Pankajakshan, B. Sjogreen, A. Petersson, and A. Pitarka (2019). Hayward fault earthquake ground motion simulations on GPU-accelerated platforms with SW4-RAJA, *Poster Presentation at 2019 SCEC Annual Meeting*, Palm Springs, California, 7–11 September 2019.

Rojas, O., E. Dunham, S. Day, L. Dalguer, and J. Castillo (2009). Finite difference modeling of rupture propagation with strong velocity-weakening friction, *Geophys. J. Int.* **179,** 1831–1858.

Roten, D., Y. Cui, K. B. Olsen, S. M. Day, K. Withers, W. H. Savran, P. Wang, and D. Mu (2016). High-frequency nonlinear earthquake simulations on petascale heterogeneous supercomputers, *SC '16: Proc. of the International Conf. for High Performance Computing, Networking, Storage and Analysis*, Salt Lake, Utah, 957–968, doi: 10.1109/SC.2016.81.

Roten, D., K. B. Olsen, S. M. Day, Y. Cui, and D. Fäh (2014). Expected seismic shaking in Los Angeles reduced by San Andreas fault zone plasticity, *Geophys. Res. Lett.* **41,** 2769–2777.

Tago, J., V. M. Cruz-Atienza, J. Virieux, V. Etienne, and F. J. Sánchez-Sesma (2012). A 3D hp-adaptive discontinuous Galerkin method for modeling earthquake dynamics, *J. Geophys. Res.* **117,** no. B09312, doi: 10.1029/2012JB009313.

Ulrich, T., A.-A. Gabriel, J.-P. Ampuero, and W. Xu (2019). Dynamic viability of the 2016 M_w 7.8 Kaikuoura earthquake cascade on weak crustal faults, *Nat. Comm.* **10,** 1213.

Uphoff, C., S. Rettenberger, M. Bader, E. Madden, T. Ulrich, S. Wollherr, and A. Gabriel (2017). Extreme scale multi-physics simulations of the tsunamigenic 2004 Sumatra megathrust earthquake, *SC '17: Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis*, Denver, Colorado, November 2017, available at https://dl.acm.org/doi/pdf/10.1145/3126908.3126948 (last accessed March 2020).

Wollherr, S., A.-A. Gabriel, and P. M. Mai (2019). Landers 1992 "reloaded": Integrative dynamic earthquake rupture modeling, *J. Geophys. Res.* **124,** 6666–6702.

Wollherr, S., A.-A. Gabriel, and C. Uphoff (2018). Off-fault plasticity in three-dimensional dynamic rupture simulations using a modal discontinuous Galerkin method on unstructured meshes: Implementation, verification and application, *Geophys. J. Int.* **214,** no. 3, 1556–1584.

Zhou, J., Y. Cui, E. Poyraz, D. J. Choi, and C. C. Guest (2013). Multi-GPU implementation of a 3D finite difference time domain earthquake code on heterogeneous supercomputers, *Procedia Comput. Sci.* **18,** 1255–1264.

Manuscript received 13 December 2019
Published online 17 June 2020