A grid-doubling finite-element technique for calculating dynamic three-dimensional spontaneous rupture on an earthquake fault

Michael Barall
United States Geological Survey, 345 Middlefield Road MS 977, MenloPark, CA 94025, USA; and Invisible Software Inc., San Jose, CA, USA.
E-mail: mbinv@invisiblesoft.com

Accepted 2009 March 19. Received 2009 March 19; in original form 2008 September 7

SUMMARY
We present a new finite-element technique for calculating dynamic 3-D spontaneous rupture on an earthquake fault, which can reduce the required computational resources by a factor of six or more, without loss of accuracy. The grid-doubling technique employs small cells in a thin layer surrounding the fault. The remainder of the modelling volume is filled with larger cells, typically two or four times as large as the small cells. In the resulting non-conforming mesh, an interpolation method is used to join the thin layer of smaller cells to the volume of larger cells. Grid-doubling is effective because spontaneous rupture calculations typically require higher spatial resolution on and near the fault than elsewhere in the model volume. The technique can be applied to non-planar faults by morphing, or smoothly distorting, the entire mesh to produce the desired 3-D fault geometry. Using our FaultMod finite-element software, we have tested grid-doubling with both slip-weakening and rate-and-state friction laws, by running the SCEC/USGS 3-D dynamic rupture benchmark problems. We have also applied it to a model of the Hayward fault, Northern California, which uses realistic fault geometry and rock properties. FaultMod implements fault slip using common nodes, which represent motion common to both sides of the fault, and differential nodes, which represent motion of one side of the fault relative to the other side. We describe how to modify the traction-at-split-nodes method to work with common and differential nodes, using an implicit time stepping algorithm.

Key words: Earthquake dynamics; Computational seismology.

1 INTRODUCTION
The paper presents a new grid-doubling technique for dynamic 3-D spontaneous rupture calculations. This technique substantially reduces the required computational resources compared to standard finite-element techniques, without loss of accuracy.

A dynamic 3-D spontaneous rupture calculation is a three-dimensional numerical simulation of the physical processes that occur during an earthquake (see, e.g. Harris 2004). It is constructed by specifying the relevant physical laws, which include the elastodynamic wave equation, the constitutive equations which describe the rock properties, and the frictional equations which describe the properties of the fault. It is also necessary to specify a set of initial conditions, including the initial stresses acting on the fault. Then, the computer calculates the resulting behaviour of the system, which includes the nucleation and propagation of the earthquake rupture, and the resulting seismic waves.

Such calculations are extremely computationally intensive. In finite-element approaches, space is discretized into a set of elements or cells. The standard technique is to use the same sized elements throughout all or most of the modelling volume. However, accurate modelling of fault frictional behaviour requires high spatial resolution, typically requiring an element size in the neighbourhood of 100 m or less. So, even relatively modest models may need many millions of elements. Models of that size consume so much computer memory and execution time that supercomputer-grade facilities must be used.

The goal of grid-doubling is to reduce the computational resources required for a spontaneous rupture model, by reducing the number of elements. The key idea is that the spatial resolution required to accurately model the frictional behaviour of the fault is typically much higher than the spatial resolution required to model the seismic waves propagating away from the fault. Grid-doubling takes advantage of this fact by using small-sized elements only in a thin layer surrounding the fault. Outside the thin layer, the rest of the modelling volume is filled with larger-sized elements. The difference in size between large and small elements is a power of 2, typically either 2 or 4 (which explains the name grid-doubling). Most often, the large cells are twice the size of the small cells, which reduces the total number of elements, and hence also the required computer memory and execution time, by approximately a factor of 6. (Naively one might expect reduction by a factor of almost 8,
but the layer of small cells, although thin, contains a large number of cells, which in practice limits the reduction factor.) Even greater savings can be achieved by using large cells that are four times the size of the small cells.

Our tests have shown that grid-doubling successfully models fault frictional behaviour at the spatial scale determined by the size of the small cells, even though most of the modelling volume is filled with large cells. However, as the size ratio between the large and small cells increases, there is a tendency for the model to show increasing amounts of spurious oscillation. Therefore, we have not attempted to push the technique beyond a size ratio of 4:1.

2 GRID-DOUBLING MESHES

In a finite-element model, a mesh consisting of elements or cells is used to discretize space. An example of a standard finite-element mesh is shown in Fig. 1. In this example, all the cells are cubes, 100 m on a side. The figure shows the top surface of the mesh, that is, the Earth’s surface in map view. All our meshes are 3-D, but for purposes of exposition the figures show only the 2-D top surface of the mesh. The number of cells shown in the figures is several orders of magnitude less than the number of cells that would be used in an actual spontaneous rupture calculation.

The fault trace is shown in Fig. 1 as a red line. A fault is a surface at which the displacement field can be discontinuous. The fault surface must conform to the mesh (or perhaps one should say that the mesh must conform to the fault surface) so that the fault surface is composed of element faces.

Fig. 2 shows the simplest grid-doubling mesh. In this mesh, 100 meter cells occur only in a thin layer surrounding the fault, with two layers of small cells on either side of the fault. The remainder of the modelling volume is filled with cells that are 200 m on a side. All the small cells are cubes 100 m on a side, and all the large cells are cubes 200 m on a side.

(It is possible to make a grid-doubling mesh with only one layer of small cells on each side of the fault. But we prefer to place the fault surface at a location which would be a boundary between large cells, if the entire volume were filled with large cells, because this makes it a bit easier to generate the mesh by starting with a coarse mesh and then refining the region adjacent to the fault.)

![Conventional finite-element mesh.](https://example.com/image1.png)

**Figure 1.** Conventional finite-element mesh. The mesh consists entirely of cubes, 100 m on a side. The fault trace is shown as the red line. The figure shows the top surface of the mesh (i.e., the surface of the Earth) in map view; in reality the mesh is 3-D.
The grid-doubling mesh shown in Fig. 2 has the same 100 m resolution on the fault surface as the standard mesh shown in Fig. 1. Therefore, we can expect to achieve the same spatial resolution when modelling the frictional processes that occur on the fault surface during rupture propagation. But, the total number of cells in the grid-doubling mesh is much less than the total number of cells in the standard mesh. This reduction in the total number of cells translates into a proportionate reduction in the required computer memory and execution time.

In terms of accuracy, the grid-doubling mesh has two disadvantages compared to the standard mesh. First, outside the layers of small cells, the maximum frequency of seismic waves that can be modelled is reduced by a factor of two. Second, there is the possibility of seismic wave reflection from the interface between small and large cells (Vichnevetsky 1987; Cathers et al. 1989). Our testing indicates that grid-doubling yields satisfactory results despite these effects, although they may contribute to increased spurious oscillations of the model.

It should be noted that the grid-doubling mesh is non-conforming. A mesh is conforming if the faces, edges, and vertices of each element match up with the faces, edges, and vertices of adjacent elements. The standard finite-element mesh shown in Fig. 1 is conforming. The grid-doubling mesh is non-conforming at the interface between small and large cells because, for example, a vertex of a small cell may lie in the middle of an edge or face of a large cell. Most finite-element codes and most mesh generators only work with conforming meshes. Our FaultMod finite-element software is able to generate and use non-conforming meshes. Technical details of how FaultMod handles non-conforming meshes are presented below.

Our grid-doubling mesh resembles the computer data structure known as an octree. Octree meshes have been used before in computational seismology (e.g. Kim et al. 2003; Bielak et al. 2005). Other types of non-conforming meshes have also been used for elastodynamic wave calculations (e.g. Chaljub et al. 2003). Our work differs from previous uses of octree meshes in that we apply octree...
meshes to dynamic earthquake source modelling, and place the transition from small to large cells close to the fault surface to achieve higher-resolution modelling of frictional processes occurring on the fault.

A non-conforming mesh is not the only possible way to make a rapid transition from small to large cells. Unstructured mesh generators can create conforming meshes with variable cell size, and are able to accommodate very complicated geometries (e.g. de la Puente et al. 2008). Some mesh generators can create a cell-size transition in a conforming mesh by introducing an intermediate layer of distorted cells (e.g. Shepherd et al. 2000; Komatitsch & Tromp 2002). Hughes (2000) describes a technique for creating a cell-size transition by using an intermediate layer of special transition elements that have non-smooth shape functions. However, all our work on grid-doubling has used the non-conforming meshes described here.

In Fig. 2, each large square represents a cube 200 m on a side, and each small square represents a cube 100 m on a side. This means that grid-doubling occurs in both the horizontal and vertical directions, that is, the small cells are half the size of the large cells both horizontally and vertically. In some of our early experiments we used an alternative form of grid-doubling in which the large cells are tiles measuring 200 m × 200 m × 100 m, with the 100 m side being vertical. In this alternative version, grid-doubling occurs in the horizontal direction only, and the mesh has a layered structure that makes it somewhat easier to generate. Our experiments showed that the alternative version gives similar results to a mesh with both horizontal and vertical grid-doubling. Since the alternative version has almost twice as many cells, we have abandoned it.

Our original idea was that only two layers of small cells would appear on each side of the fault, on the belief that the highest spatial resolution is needed only at the fault surface. However, Andrews (private communication, 2007) pointed out that the fault rupture tip is accompanied by an inhomogeneous mode wave (i.e., a wave with complex wavenumber), which should be modelled at the same spatial resolution as the fault surface. Since the inhomogeneous mode wave attenuates exponentially with increasing distance from the fault surface, Andrews suggested that the layer of small cells be thickened to allow better modelling of the inhomogeneous mode wave. Fig. 3 shows a grid-doubling mesh that has six layers of small cells on each side of the fault surface, as opposed to the two layers shown in Fig. 2. The thickness of the small cell layer can be adjusted as desired to obtain good performance.

Figure 3. Grid-doubling mesh with thickened layer of small cells. This figure illustrates a grid-doubling mesh with six layers of small cells on each side of the fault surface, as opposed to the two layers shown in Fig. 2. The thickness of the small cell layer can be adjusted as desired to obtain good performance.
cells on each side of the fault. In this work, we typically use 2–10 layers of small cells on each side of the fault surface.

All the examples so far show a 2:1 size ratio between large and small cells. It is possible to go further by using multiple stages of grid-doubling. Fig. 4 shows a mesh with two stages of grid-doubling, so that the small cells adjacent to the fault are 50 m on a side, whereas the large cells are 200 m. This grid offers 50 m resolution on and near the fault surface, at far less cost than filling the entire volume with 50 m cells. Note that the 50 m cells are separated from the 200 m cells by an intermediate layer of 100 m cells, so that the cell size doubles at each of two interfaces, rather than increasing by a factor of four all at once.

The prior examples all show a vertical planar fault, but grid-doubling is not limited to this case. Later we will describe a morphing technique for constructing a grid-doubling mesh with a non-vertical or non-planar fault, and give an example of using grid-doubling with a natural fault.

3 FaultMod Implementation of Grid-Doubling

This section provides technical details about how FaultMod implements the finite-element method on a grid-doubling mesh. It is assumed that the reader is generally familiar with the finite-element method (e.g. Zienkiewicz & Taylor 2000; Hughes 2000). FaultMod is finite-element software, developed for the U.S. Geological Survey Earthquake Hazards Program, and designed specifically for modelling earthquake faults. FaultMod is described in Barall (2008).

In the finite-element formalism, one selects a set of nodes which are numbered 1, 2, …, M. For each node m, one has a nodal displacement vector \( \mathbf{u}_m = (u_{m0}, u_{m1}, u_{m2}) \). In a dynamic model, the nodal displacement vector is implicitly a function of time. One also has a shape function \( \varphi_m(x) \), which is a function of position \( x = (x_0, x_1, x_2) \) and is independent of time. Then, at any point in the model...
Note that in a dynamic model, the displacement \( u(x) \) is a function of both position and time; the time dependence is implicit.

Similar expressions can be written for nodal velocity vectors \( \dot{u}_m \) and nodal acceleration vectors \( \ddot{u}_m \).

The shape functions \( \phi_m(x) \) are required to be continuous and piecewise smooth, except across the fault surface where discontinuities are allowed. In the paper, piecewise means within each cell, thus a piecewise smooth function is a function that is smooth within the interior of each cell.

The cells are used to construct the shape functions. For simplicity, consider a cubical cell, of edge length \( L \), oriented parallel to the coordinates axes, and with centre located at coordinates \((c_0, c_1, c_2)\) (see references for details). We place a node at each of the eight vertices of the cube. Then, we can define the eight corresponding element shape functions, within the cube, as

\[
\phi(x_0, x_1, x_2) = L^{-3} \left( (x_0 - c_0) \pm \frac{L}{2} \right) \left( (x_1 - c_1) \pm \frac{L}{2} \right) \left( (x_2 - c_2) \pm \frac{L}{2} \right).
\]  

Each of the eight possible combinations of plus and minus signs produces the element shape function for one of the eight nodes. For example, taking all plus signs produces the element shape function for the node at coordinates \((c_0 + L/2, c_1 + L/2, c_2 + L/2)\). We call these element shape functions because they give the shape function within a single element. Since a given node generally lies in several elements, the complete nodal shape function is obtained by assembling the element shape functions from all the elements that contain the node.

When the shape functions of eq. (2) are used in a standard conforming mesh, such as the one shown in Fig. 1, the resulting shape functions \( \phi_m(x) \) have the required properties: they are continuous everywhere except across the fault surface, and they are piecewise smooth. These particular shape functions produce a linear mesh, so-called because by appropriate choice of the nodal displacements \( u_m \) it is possible to produce any desired displacement function \( u(x) \), which is continuous except at the fault surface, and piecewise linear in the coordinates \( x_0, x_1, x_2 \). FaultMod can also produce a quadratic mesh by placing nodes at both the vertices and edge midpoints of each cell. With a quadratic mesh, it is possible to find nodal displacements \( u_m \) to produce any desired displacement function \( u(x) \), which is continuous except at the fault surface, and piecewise quadratic in the coordinates \( x_0, x_1, x_2 \). Since all our spontaneous rupture modelling is done using linear meshes, we confine the discussion to linear meshes.

Applying the element shape functions of eq. (2) to a non-conforming grid-doubling mesh, such as the one shown in Fig. 2, does not produce acceptable shape functions. The problem is that the resulting shape functions are discontinuous at the interface between the small cells and the large cells. This occurs because at the interface, the nodes, edges, and faces of the small cells do not match up with the nodes, edges, and faces of the large cells.

An examination of the figures shows that on the interface between small and large cells, the nodes fall into three categories.

1. Nodes that lie at a vertex of both small cells and large cells.
2. Nodes that lie at a vertex of small cells, and at the midpoint of an edge of large cells.
3. Nodes that lie at a vertex of small cells, and at the centre of a face of a large cell.

Figure 5. Interface between small and large cells. The figure is an enlargement of a grid-doubling mesh such as the one shown in Fig. 2. At locations \( b \) and \( c \), which are called free nodes, vertices of small cells are adjacent to vertices of large cells. At location \( a \), which is called a slave node, vertices of small cells are adjacent to the midpoint of an edge of a large cell. There is a second type of slave node, not shown, where vertices of small cells are adjacent to the centre of a face of a large cell.

(1) Nodes that lie at a vertex of both small cells and large cells.
(2) Nodes that lie at a vertex of small cells, and at the midpoint of an edge of large cells.
process repeatedly, it is possible to remove all the slave nodes from the system. Mathematically, it can be proved that after all the slave nodes are removed from the system, the remaining shape functions are continuous at the interface between small and large cells.

The FaultMod implementation of grid-doubling has two main components. One is a mesh generator that produces the grid-doubling meshes. The other is a component that automatically constructs the correct shape functions, by eliminating slave nodes from the system, as shown in eq. (5). The FaultMod implementation has been tested by applying a series of “patch tests” (e.g. Irons & Razzaque 1972; Taylor et al. 1986) to grid-doubling meshes. The patch tests are sensitive to any mesh connectivity errors or discontinuity of the shape functions, so successful completion of the patch tests gives confidence that the numerical implementation is correct.

On the fault surface, FaultMod places two coincident nodes at each cell vertex. One is called the common node and represents motion common to both sides of the fault; it is the node that would be at that location in the absence of the fault surface. The other is called the differential node and represents motion of one side of the fault relative to the other side; in other words, it represents fault slip. The shape function of the common node is assembled from element shape functions (e.g. eq. (2)) belonging to elements on both sides of the fault, and therefore is non-zero on both sides of the fault and everywhere continuous. The shape function of the differential node is assembled from element shape functions belonging to elements on one side of the fault only, and therefore is non-zero on only one side of the fault with a discontinuity at the fault surface.

Many authors require that shape functions satisfy the condition \( \psi_n(x_n) = \delta_{nn} \), where \( x_n \) is the location of node \( n \) and \( \delta_{nn} \) is the Kronecker delta. The shape functions associated with common and differential nodes violate this condition, however, this does not create any problem.

There are other methods that have been used to introduce faults into finite-element calculations. In the split node method (Melosh & Raefsky 1981), there are two coincident nodes at each vertex on the fault surface. In each coincident pair, one node represents motion on one side of the fault, and the other node represents motion on the other side of the fault, so both nodes have shape functions that are discontinuous at the fault surface. In the extended finite-element method (Belytschko & Black 1999), the mesh does not conform to the shape of the fault surface. Instead, one begins with a mesh for an unfaulted medium, and then adds additional degrees of freedom to represent slip on the fault.

The use of common and differential nodes is mathematically equivalent to the split node method, in the sense that both techniques have the same set of accessible physical states. Unlike split nodes, common and differential nodes provide a clean separation between the degrees of freedom that describe fault behaviour, such as slip and slip rate, and the degrees of freedom that describe the motion of an unfaulted medium. In this respect, common and differential nodes are similar to the extended finite-element method. Despite these similarities, we are not aware of another code that uses common and differential nodes in the same way as FaultMod.

With satisfactory shape functions in hand, it is now possible to perform finite-element analysis in the normal way. To begin, define the strain-displacement matrix \( B \) as

\[
B_{\alpha\beta m} = \frac{1}{2} \left[ \delta_{\alpha\mu} \frac{\partial \varphi_m}{\partial x_\beta} + \delta_{\beta\mu} \frac{\partial \varphi_m}{\partial x_\alpha} \right].
\]

Roman subscripts like \( m \) represent node numbers in the set \( \{1, 2, \ldots, M\} \), and Greek subscripts like \( \alpha, \beta, \) and \( \mu \) represent the three coordinate directions \( \{0, 1, 2\} \). And, \( \varphi_m \) represents the final shape functions remaining after the slave nodes are removed. In light of eq. (1), the strain tensor \( e \) can be written as

\[
e_{\alpha\beta} = \frac{1}{2} \left[ \frac{\partial u_\alpha}{\partial x_\beta} + \frac{\partial u_\beta}{\partial x_\alpha} \right] = \sum_{\alpha, \mu} B_{\alpha\beta m} u_m. \tag{7}
\]

An analogous formula can be written for the strain rate tensor \( \dot{e} \). There is a constitutive law, which specifies how the stress tensor depends on the strain and strain rate tensors. Constitutive laws can be complicated, and may involve both current and past values of the strain and strain rate tensors, as is the case for materials that exhibit plasticity or viscoelasticity (e.g. Findley et al. 1989). For simplicity, we assume that the stress tensor \( \sigma \) depends linearly on the current values of the strain and strain rate tensors

\[
\sigma_{\alpha\beta} = \sum_{\gamma, \delta} D_{\alpha\beta \gamma\delta} e_{\gamma\delta} + \sum_{\gamma, \delta} T_{\alpha\beta \gamma\delta} \dot{e}_{\gamma\delta}. \tag{8}
\]

The elasticity tensor \( D \) and the viscosity tensor \( T \) are constitutive parameters, whose values may vary as a function of position. Although they each appear to have \( 3^4 = 81 \) components, in fact, for isotropic materials they each have only two independent components. We now calculate the forces acting on each finite-element node, and require that these forces be in equilibrium (see references for details). The resulting force-balance equation is

\[
f = \int \sum_{\alpha\beta} B_{\alpha\beta m} \sigma_{\alpha\beta} dV + \int \sum_{\alpha\beta} \varphi_m \dot{u}_\alpha p dV, \tag{9}
\]

where \( f \) is the force acting on node \( n \) due to traction forces, and may include frictional forces acting on the fault, and forces applied to the boundaries of the mesh to suppress unwanted reflections of seismic waves (Lysmer & Kuhlemeyer 1969; Clayton & Engquist 1977). The first term on the right-hand side represents forces due to internal stresses of the material, and the second term represents inertial forces (mass times acceleration). The density is \( \rho \), and the integrals run over the entire modelling volume.

Substituting eqs (1), (7), and (8) into (9), we obtain the discretized equations of motion of the system

\[
f = \sum_{\alpha\beta} K_{\alpha\beta m} u_m + \sum_{\alpha\beta} C_{\alpha\beta m} \dot{u}_m + \sum_{\alpha\beta} \sum_{\gamma, \delta} M_{\alpha\beta \gamma\delta} \ddot{u}_m. \tag{10}
\]

The stiffness matrix \( K \) is

\[
K_{\alpha\beta m} = \int \sum_{\alpha\beta\gamma\delta} B_{\alpha\beta m} D_{\alpha\beta \gamma\delta} B_{\gamma\delta m} dV. \tag{11}
\]

The damping matrix \( C \) is

\[
C_{\alpha\beta m} = \int \sum_{\alpha\beta\gamma\delta} B_{\alpha\beta m} T_{\alpha\beta \gamma\delta} B_{\gamma\delta m} dV. \tag{12}
\]

The mass matrix \( M \) is

\[
M_{\alpha\beta m} = \int \varphi_m \delta_{\alpha\beta} \rho \varphi_m dV. \tag{13}
\]

It is possible to create different forms of the mass matrix by altering the form of the density \( \rho \). If \( \rho \) describes a continuous mass distribution (mathematically, if \( \rho \) is a piecewise smooth function) then we obtain the consistent mass matrix. If \( \rho \) describes a collection of point masses (mathematically, if \( \rho \) is a linear combination of Dirac delta-functions), then we obtain a lumped mass matrix. It should be noted that there is not a unique way to perform lumping, and that not all lumping procedures can be written in the form of eq. (13). A numerical advantage of lumping is that by placing point masses at the nodes, the mass matrix can be made (almost) diagonal.
FaultMod can generate either consistent or lumped mass matrices, but for spontaneous rupture calculations we use a lumped mass matrix, as recommended by Andrews (private communication, 2007). Lumping is performed separately within each element, by treating the element’s mass as set of point masses located at the element’s vertices. For example, given a cubical element, there would be eight point masses at the eight corners of the cube, each with one-eighth of the element’s mass. (For a general hexahedral element, the eight point masses may be unequal, and are proportional to the integrals of the shape functions over the element.)

Consider the situation depicted in Fig. 5, which shows the interface between small and large cells in a grid-doubling mesh. When performing mass lumping, the small cells have point masses located at the vertex labeled $a$, even though there is no finite-element node located at vertex $a$ (recall that nodes from vertices, such as $a$, which we call slave nodes, are removed from the system when constructing the shape functions). The presence of mass at location $a$ creates off-diagonal terms in the lumped mass matrix. Specifically, the off-diagonal mass matrix terms $M_{mbc}$ are non-zero. One could eliminate these off-diagonal terms by changing the mass lumping, eliminating the point masses at vertex $a$ and making a corresponding increase in the point masses at vertices $b$ and $c$. We have not done so, because FaultMod’s lumped mass matrix always has off-diagonal terms that arise from the common and differential nodes at the fault surface, so there is little to gain by eliminating the off-diagonal terms associated with the interface between small and large cells.

Finally, the equations of motion are integrated forward in time, using a time stepping algorithm, to obtain the values of $\dot{u}_{mbc}$, $\ddot{u}_{mbc}$, and $\dddot{u}_{mbc}$ at a series of times. FaultMod uses a version of the Newmark algorithm (Newmark 1959). At the fault surface, the motion of the differential nodes is calculated using a generalized version of the traction-at-split-nodes method (Andrews 1999). Details are provided in Appendix A. Any time stepping algorithm becomes numerically unstable or inaccurate if the time increment is too big. Roughly speaking, the largest possible time increment is approximately the time required for a seismic wave to cross the smallest cell in the mesh. So, for a grid-doubling mesh, the maximum time increment is determined by the size of the small cells adjacent to the fault, even though most of the mesh consists of large cells.

Grid-doubling was applied to all the benchmark problems, which include nine benchmarks with slip-weakening friction and four benchmarks with rate-and-state friction. We used the visual tools on the web site to compare the grid-doubling results to the results produced by other codes. In every case, grid-doubling yielded satisfactory results, which agree well with the results produced by other codes using other techniques. Some representative comparisons are shown below.

(Although this article is concerned only with 3-D calculations, we note that two of the benchmarks are 2-D calculations; thus demonstrating that grid-doubling is also applicable to the 2-D case.)

Fig. 6 shows results for the benchmark named TPV5. This benchmark has a uniform linear elastic half-space. The fault is a vertical plane, with slip-weakening friction. Initial shear and normal stresses on the fault are uniform, except for a high-shear-stress square patch in the centre of the fault which nucleates the rupture, and two other square patches on either side of the nucleation patch (refer to the web site for details). We add a small amount of artificial viscosity to help suppress spurious oscillations (Day et al. 2005, Dalguer & Day 2007). Ordinarily we add viscosity only in a thin layer surrounding the fault, which for grid-doubling can be conveniently achieved by adding viscosity only to the small cells, but for this comparison we added viscosity throughout the entire mesh.

### Table 1. S Cec/UsG s spontaneous rupture benchmark problems.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Friction</th>
<th>Fault</th>
<th>Motion</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPV5</td>
<td>Slip-weakening</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Slightly inhomogeneous initial stress</td>
</tr>
<tr>
<td>TPV6</td>
<td>Slip-weakening</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Shear modulus contrast across fault</td>
</tr>
<tr>
<td>TPV7</td>
<td>Slip-weakening</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Shear modulus contrast across fault</td>
</tr>
<tr>
<td>TPV8</td>
<td>Slip-weakening</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Initial stress varies linearly with depth</td>
</tr>
<tr>
<td>TPV9</td>
<td>Slip-weakening</td>
<td>Vertical</td>
<td>Dip-slip</td>
<td>Initial stress varies linearly with depth</td>
</tr>
<tr>
<td>TPV10</td>
<td>Slip-weakening</td>
<td>60° Dip</td>
<td>Dip-slip</td>
<td>Initial stress varies linearly with depth. Subshear rupture conditions</td>
</tr>
<tr>
<td>TPV10-2-D</td>
<td>Slip-weakening</td>
<td>60° Dip</td>
<td>Dip-slip</td>
<td>2-D version of TPV10</td>
</tr>
<tr>
<td>TPV11</td>
<td>Slip-weakening</td>
<td>60° Dip</td>
<td>Dip-slip</td>
<td>Initial stress varies linearly with depth. Supershear rupture conditions</td>
</tr>
<tr>
<td>TPV11-2-D</td>
<td>Slip-weakening</td>
<td>60° Dip</td>
<td>Dip-slip</td>
<td>2-D version of TPV11</td>
</tr>
<tr>
<td>TPV101</td>
<td>Rate-and-state</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Ageing law, in a whole-space</td>
</tr>
<tr>
<td>TPV102</td>
<td>Rate-and-state</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Ageing law</td>
</tr>
<tr>
<td>TPV103</td>
<td>Rate-and-state</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Slip law, in a whole-space</td>
</tr>
<tr>
<td>TPV104</td>
<td>Rate-and-state</td>
<td>Vertical</td>
<td>Strike-slip</td>
<td>Slip law</td>
</tr>
</tbody>
</table>

*Note:* Table shows all 13 benchmark problems posted on the S Cec/UsG s web site (Harris et al. 2009). The grid-doubling technique has been successfully applied to all 13 benchmarks. All benchmarks except TPV101 and TPV103 are performed in a half-space. All benchmarks assume linear elastic rock properties, and all except TPV6 and TPV7 have uniform rock properties throughout the model.
Figure 6. Comparison of rupture contour plots for grid-doubling and conventional meshes, for benchmark TPV5. The figure shows the fault surface, which measures 30 km × 15 km. Contour curves show the points on the fault surface that are just beginning to slip at a given time, and are drawn at intervals of 0.5 s. Red, grid-doubling mesh with four layers of 100 m cells adjacent to each side of the fault, and 200 m cells elsewhere. Black, conventional mesh where all cells are 100 m. Green, conventional mesh where all cells are 200 m. It is difficult to see the black curves because they coincide almost exactly with the red curves. The green curves propagate distinctly more slowly than the other two sets of curves. The comparison demonstrates that the grid-doubling mesh behaves like a conventional mesh composed entirely of small cells, and unlike a conventional mesh composed entirely of large cells.

Figure 7. Comparison of slip histories 12 km from the epicentre, for benchmark TPV5. The figure shows calculated horizontal slip rate (metre per second) as a function of time (seconds), for a station at the Earth’s surface, on the fault trace 12 km from the epicentre. Colours are the same as in Fig. 6. Such a slip history might be expected to show effects of any glancing reflections from the transition between small and large cells in the grid-doubling mesh. The results for the grid-doubling mesh (red) agree very well with the conventional mesh with 100 m cells (black). In contrast, the conventional mesh with 200 m cells produces quite different results (green).

Experience shows that a reliable way to compare benchmark results is to compare plots of the rupture front contours at various times (Harris et al. 2008). Each contour curve connects the points on the fault that are just beginning to slip at a given instant of time. We draw contour curves at intervals of 0.5 s.

The figure shows rupture front contour plots for three meshes: (i) a grid-doubling mesh with four layers of 100 m cells adjacent to each side of the fault and 200 m cells elsewhere; (ii) a conventional mesh where all cells are 100 m; and (iii) a conventional mesh where all cells are 200 m. The contours for the grid-doubling mesh are almost identical to the contours for the 100 m conventional mesh, but the contours for the 200 m conventional mesh are distinctly different than the contours for the other two meshes. In the code validation project it is commonly seen that ruptures propagate faster when the cell size is reduced. Consistent with that experience, the rupture for the 200 m mesh is propagating more slowly than the ruptures for the other two meshes, which are propagating at the same speed. The rms difference in rupture arrival time between the grid-doubling mesh and the 100 m conventional mesh is only 0.009 s, which is less than one time step interval (the time step interval is 0.01 s).

Fig. 7 shows another comparison for benchmark TPV5. In this figure, we show the calculated horizontal slip rate as a function of time, for a station located at the Earth’s surface, on the fault trace, 12 km from the epicentre. We selected a station far from the epicentre because such a station might be expected to show any adverse effects from glancing reflections of seismic waves off the cell-size transition in the grid-doubling mesh. The figure shows that the slip rate for the grid-doubling mesh agrees very well with the slip rate for the 100 m conventional mesh. But, the slip...
Table 2. Performance data for grid-doubling and conventional meshes, for benchmark TPV5.

<table>
<thead>
<tr>
<th>Mesh Type</th>
<th>Number of cells</th>
<th>Execution time (hr)</th>
<th>Computer memory (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid-doubling mesh</td>
<td>2,338,208</td>
<td>10.0</td>
<td>7.2</td>
</tr>
<tr>
<td>100 m cells adjacent to fault</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 m cells elsewhere</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conventional mesh</td>
<td>16,000,000</td>
<td>72.0</td>
<td>47.0</td>
</tr>
<tr>
<td>All cells 100 m</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conventional mesh</td>
<td>2,000,000</td>
<td>7.4</td>
<td>6.1</td>
</tr>
<tr>
<td>All cells 200 m</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: Table shows number of cells, execution time, and memory requirements for each of the three meshes that was used to create Figs 6 and 7. All three meshes had a total size of 40 km × 20 km × 20 km. All computations were done with the FaultMod finite-element software, using a SunFire x4600 computer. Each calculation consisted of 1000 time steps, with a time increment of 0.01 s. The grid-doubling mesh has four layers of small cells on each side of the fault surface. Compared to the 100 m conventional mesh, the resources required by the grid-doubling mesh are reduced by more than a factor of 6.

These results demonstrate that the grid-doubling mesh produces the same results as a conventional mesh composed entirely of small cells, and does not produce the same results as a conventional mesh composed entirely of large cells, even though the grid-doubling mesh consists mostly of large cells.

Table 2 shows the execution time and computer memory required for each of the three calculations used to produce Figs 6 and 7. Compared to the 100 m conventional mesh, the computational resources required by the grid-doubling mesh are reduced by more than a factor of 6. In particular, the computer memory required is reduced from 47.0 to 7.2 GB. In our experience, computer memory is often the limiting factor in the size of a spontaneous rupture model. So, grid-doubling greatly increases the size of the models that can be computed with available resources, without sacrificing accuracy.

Fig. 8 shows results from grid-doubling with a 4:1 size ratio, applied to benchmark problem TPV6. This benchmark has different material properties on the two sides of the fault, which is a vertical plane with slip-weakening friction (see web site for details). FaultMod was used to calculate rupture contour plots for two meshes: (i) a grid-doubling mesh with 10 layers of 50 m cells adjacent to each side of the fault, and 200 m cells elsewhere; and (ii) a conventional mesh where all cells are 200 m. Also shown for comparison are rupture contour plots calculated by Day and Dalguer using a finite-difference code with 50 m node spacing (Day et al. 2005; Dalguer & Day 2007). We are not able to run FaultMod with a 50 m conventional mesh due to limitations in available computing resources, thus, this example illustrates how grid-doubling can perform calculations that would not otherwise be possible with available resources. The contours for the grid-doubling mesh agree well with the Day and Dalguer contours for 50 m node spacing. The contours for the 200 m conventional mesh are distinctly different than the other two sets of contours, and propagate more slowly. Computations with 100 m resolution on the fault (not shown in the figure) propagate at an intermediate speed. The rms difference in the red, grid-doubling mesh with 10 layers of 50 m cells adjacent to each side of the fault, and 200 m cells elsewhere. Green, conventional mesh where all cells are 200 m. Black, contours calculated by Day and Dalguer using a finite-difference code with 50 m node spacing (Day et al. 2005; Dalguer & Day 2007). The red and black curves agree quite well, but the green curves propagate much more slowly. The comparison demonstrates that even with a 4:1 size ratio between large and small cells, the grid-doubling mesh behaves like a conventional mesh composed entirely of small cells, and unlike a conventional mesh composed entirely of large cells. (Data from Day and Dalguer are used with permission.)

Figure 8. Comparison of rupture contour plots for grid-doubling with a 4:1 size ratio, and conventional meshes, for benchmark TPV6. The figure shows the fault surface, which measures 30 km × 15 km. Contour curves show the points on the fault surface that are just beginning to slip at a given time, and are drawn at intervals of 0.5 s. Red, grid-doubling mesh with 10 layers of 50 m cells adjacent to each side of the fault, and 200 m cells elsewhere. Green, conventional mesh where all cells are 200 m. Black, contours calculated by Day and Dalguer using a finite-difference code with 50 m node spacing (Day et al. 2005; Dalguer & Day 2007). The red and black curves agree quite well, but the green curves propagate much more slowly. The comparison demonstrates that even with a 4:1 size ratio between large and small cells, the grid-doubling mesh behaves like a conventional mesh composed entirely of small cells, and unlike a conventional mesh composed entirely of large cells. (Data from Day and Dalguer are used with permission.)
Figure 9. Effect of changing the thickness of the small-cell layer. These are synthetic seismograms, showing the calculated horizontal ground motion parallel to the fault, at the epicentre of benchmark problem TPV6 (time in seconds and horizontal velocity in metre per second). Top: mesh constructed using two layers of small cells on each side of the fault. Centre: mesh constructed using 10 layers of small cells on each side of the fault. In both cases, the grid-doubling mesh has 50 m cells adjacent to the fault, and 200 m cells away from the fault. The top plot, with two layers of small cells, shows considerable spurious oscillations. The oscillations are greatly reduced in the centre plot, with 10 layers of small cells. For comparison purposes, the bottom plot shows a synthetic seismogram calculated by Day and Dalguer using a finite-difference code with 50 m node spacing (Day et al. 2005; Dalguer & Day 2007).

FaultMod calculated the synthetic seismograms for two different meshes: (i) a grid-doubling mesh with two layers of small cells on each side of the fault, and (ii) a grid-doubling mesh with 10 layers of small cells on each side of the fault. In both cases, the small cells adjacent to the fault are cubes 50 m on a side, and the cells elsewhere are cubes 200 m on a side. For the mesh with two layers of small cells, the synthetic rupture arrival time between the grid-doubling mesh and the Day and Dalguer contours is 0.027 s. Again, we see that the grid-doubling mesh behaves like a conventional mesh composed entirely of small cells, and unlike a conventional mesh composed entirely of large cells.

Fig. 9 shows the effect of changing the thickness of the layer of small cells. It shows synthetic seismograms for the epicentre of benchmark problem TPV6.
seismogram shows a significant amount of spurious oscillations. With 10 layers of small cells, the spurious oscillations are greatly reduced. Andrews (private communication, 2007) predicted that if the layer of small cells is not thick enough to adequately model the inhomogeneous mode wave that accompanies the fault rupture, then the energy associated with the inhomogeneous mode wave would instead appear as spurious oscillations. The penetration depth of the inhomogeneous mode wave should be comparable to the size of the cohesive zone of the rupture, which for this problem is on the order of several 100 m (Day et al. 2005). So the results of Fig. 9 seem to be consistent with Andrews’ prediction.

5 MESH MORPHING FOR NON-PLANAR FAULTS

All the examples so far have used vertical planar faults. Grid-doubling can also be applied to non-planar and non-vertical faults by using a process we call mesh morphing.

The process is illustrated in Fig. 10. The idea is to construct a mesh where the layers of small cells curve to follow the fault surface. The first step is to construct a grid-doubling mesh for a vertical planar fault, similar to Fig. 2. Then, the entire mesh is smoothly distorted, or morphed, so that the fault surface assumes the correct shape. We call this process morphing because the computer algorithm is similar to the well-known morphing procedure used in computer graphics. After morphing is applied, the cells are no longer cubes; they become general hexahedra.

Morphing can be applied both horizontally and vertically. A horizontal distortion, perpendicular to the average strike of the fault, generates the desired non-planar fault geometry. A vertical distortion can generate non-planar topography at the Earth’s surface.

Morphing was used to create grid-doubling meshes for the SCEC/USGS benchmark problems that have a dipping fault (see Table 1).

Morphing was also applied to create a grid-doubling mesh for a preliminary spontaneous rupture model of the Hayward fault, Northern California (Barall et al. 2008). The 3-D fault geometry

Figure 10. Grid-doubling mesh for a non-planar fault. A grid-doubling mesh for a non-planar fault surface can be constructed by morphing the mesh, so that the layers of small cells curve to follow the fault surface. First a grid-doubling mesh is constructed for a planar vertical fault, then the entire mesh is smoothly distorted, or morphed, so that the fault surface assumes the desired shape.
and the 3-D distribution of rock units are derived from a USGS geologic model of the Hayward fault (Graymer et al. 2005). Physical properties are assigned to rock units according to the USGS Bay Area Velocity Model (Brocher 2005; Brocher 2008). The model also includes surface topography. The resulting mesh has two layers of 150 m cells on each side of the fault surface, and 300 m cells elsewhere. The model fault surface measures 99 km × 13 km, much larger than the faults used in the SCEC/USGS benchmarks.

Fig. 11 shows a rupture front contour plot for the Hayward fault model. For this preliminary model, we applied the same initial stress conditions as benchmark problem TPV5 (which explains the similarity between Fig. 6 and the central part of Fig. 11). These initial stress conditions are not realistic, but they are a good choice for exercising the computer code. The result shows that grid-doubling can be successfully applied to a model of a natural fault, with realistic 3-D fault geometry, rock properties, and surface topography.

6 CONCLUSION

We have demonstrated a grid-doubling technique, which reduces the computational resources required for finite-element spontaneous rupture calculations by a factor of 6 or more, without loss of accuracy. A non-conforming mesh permits the use of small cells near the fault and large cells elsewhere. Grid-doubling has been successfully tested on the SCEC/USGS benchmark problems, and successfully applied to a model of a natural fault. A mesh-morphing technique makes it possible to use grid-doubling with a non-planar fault. Spontaneous rupture models are so computationally intensive that their size is often limited by the available computational resources. Grid-doubling allows larger spontaneous rupture models to be calculated with available computing facilities.

ACKNOWLEDGMENTS

Development of FaultMod is supported by the U.S. Geological Survey, Earthquake Hazards Program. Some of the computation is supported by the Southern California Earthquake Center and by the University of Southern California Center for High-Performance Computing and Communications. The author thanks Ruth Harris and Joe Andrews for many helpful discussions and suggestions. And also thanks Joe Andrews, Robert Simpson, and an anonymous reviewer for reviewing the manuscript.

REFERENCES


References


**APPENDIX A: FRICTION CALCULATION FOR COMMON AND DIFFERENTIAL NODES**

Our implementation of fault friction is based on the traction-at-split-nodes method of Andrews (1999). Andrews represents faults using split nodes (Melosh & Raefsky 1981), which have a diagonal mass matrix. Andrews also uses an explicit time stepping algorithm, where velocity is calculated at the middle of each time step whereas displacement and acceleration are calculated at the end of each time step. In contrast, our FaultMod software represents faults using common and differential nodes, which have a non-diagonal mass matrix. Also, FaultMod uses an implicit time stepping algorithm, where displacement, velocity, and acceleration are all calculated at the end of each time step. Here, we describe how we modified the traction-at-split-nodes method to accommodate these differences.

Let \( \vec{u}_m \), \( \vec{u}'_m \), and \( \vec{u}''_m \) be the displacement, velocity, and acceleration of node \( m \) at the end of time step \( i \), and let \( t \) be the time increment in seconds. The Newmark equations (Newmark 1959) are

\[
\begin{align*}
\vec{u}_{m+1}^{i+1} &= \vec{u}_m^{i+1} + t^2 \left( \theta_1 \vec{u}_m^i + \frac{1}{2} \theta_2 \vec{u}'_m^i \right) \\
\vec{u}'_{m+1}^{i+1} &= \vec{u}'_m^i + t \left( \theta_1 \vec{u}'_m^i + \theta_2 \vec{u}''_m^i \right) \\
\vec{u}''_{m+1}^{i+1} &= \vec{u}''_m^i + \frac{\tau}{(1 - \theta_2)} \vec{u}'_m^i + \tau t \theta_1 \vec{u}'_{m+1}^{i+1}. \tag{A1}
\end{align*}
\]

The parameters \( \theta_1 \) and \( \theta_2 \) can be selected as explained in Hughes (2000). For dynamic rupture calculations, we typically use \( \theta_1 = 0.65 \) and \( \theta_2 = 0.66125 \), which provides a moderate amount of algorithmic damping to help suppress unwanted oscillations.

If there were no earthquake faults, one could simply substitute eqs (A1) into the discretized equations of motion (10) to obtain a system of equations in which \( \vec{u}''_{m+1}^{i+1} \) are the only unknowns. These equations could then be solved for \( \vec{u}''_{m+1}^{i+1} \), and then eqs (A1) would give \( \vec{u}_m^{i+1} \) and \( \vec{u}'_m^{i+1} \). Our aim here is to explain how to modify this time stepping scheme to accommodate faults.

At each node location on the fault surface, we place two coincident nodes, called the common node and the differential node, as explained above. Let \( c \) denote a common node and \( d \) the corresponding differential node. The differential node is constrained to slide along the fault surface, which means that \( \vec{u}_c, \vec{u}_d, \) and \( \vec{u}'_d \) are always tangent to the fault surface. The area associated with the differential node is defined to be

\[
A_d = \int \varphi_d dA \tag{A2}
\]

where the integral runs over the fault surface. There is a friction law that we assume takes the form

\[
0 = H(\tau, \sigma|\vec{u}_d, \psi) \tag{A3}
\]

where \( \tau \) is the magnitude of the shear stress on the fault, \( \sigma \) is the normal stress on the fault (not to be confused with the stress tensor \( \sigma_{ij} \)), \( |\vec{u}_d| \) is the scalar slip rate (often denoted \( V \) in the literature), and \( \psi \) is a set of state variables. The shear stress vector must be parallel to the slip rate vector \( \vec{u}_d \). For slip-weakening friction, the state variable \( \psi \) would be equal to the total slip (Iida 1972). For rate- and state-friction, the state variable \( \psi \) would, of course, be the state variable of the rate-and-state law (Dietrich 1979; Ruina 1983). Let \( \sigma^+ \) and \( \psi^+ \) be the normal stress and state variable at the end of time step \( i \). As part of the time stepping procedure, we must calculate \( \sigma^{i+1} \) and \( \psi^{i+1} \).

The friction law also specifies how the state variable evolves with time. Often this is specified as a differential equation for \( \psi \). We assume that the evolution equation has been discretized, yielding an equation of the form

\[
\psi^{i+1} = G(\psi^i, |\vec{u}_d^{i+1} - \vec{u}_d^i|, \sigma^+, \sigma^{i+1}, t) \tag{A4}
\]

where \( |\vec{u}_d^{i+1} - \vec{u}_d^i| \) is the amount of slip that occurs during the time step. (If the slip rate appears in the evolution equation for \( \psi \), it is replaced by \( t^{-1} |\vec{u}_d^{i+1} - \vec{u}_d^i| \) which is the average slip rate during the time step.) For example, in the case of slip-weakening friction, where \( \psi \) is total fault slip, the discretized evolution law would be \( \psi^{i+1} = \psi^i + |\vec{u}_d^{i+1} - \vec{u}_d^i| \). For some rate-and-state laws, the evolution of the state variable depends on normal stress (Linker & Dieterich 1992).

For any node \( m \) define the node-equivalent mass to be

\[
M_m = \int \rho \varphi_m dV. \tag{A5}
\]

It should be noted that due to the presence of the differential nodes, \( \sum_m M_m \) exceeds the total mass of the system. We assume that cells adjacent to the fault have lumped mass, that is, their mass is concentrated at the nodes. Then, we can write out the mass matrix components (see eq. (13)) for the common and differential nodes

\[
M_{cm} = M_c \delta_{cm},
\]

\[
M_{dmd} = M_{dmd} = M_{dmd} = M_d \delta_{dm}. \tag{A6}
\]

Note the presence of non-zero off-diagonal components \( M_{cimd} \) and \( M_{dmcd} \). We also define the reduced mass to be

\[
M_r = \left( 1/M_c + 1/(M_d - M_c) \right)^{-1}. \tag{A7}
\]
With these preliminaries out of the way, we can now describe our time stepping procedure.

**Step 1: Take a trial step.** Select a trial acceleration \( \ddot{u}_{\text{trial}} \) for time step \( i + 1 \), and then use eqs (A1), with \( \ddot{u}_{\text{trial}} \) inserted in place of \( \ddot{u} \), to obtain \( \ddot{u}_{\text{trial}} \) and \( \dot{u}_{\text{trial}} \). In our work, we usually select \( \ddot{u}_{\text{trial}} = \ddot{u}_{\text{trial}} \). We can then insert these trial values into eq. (10) to get the resulting node-equivalent force \( F_n \) acting on node \( n \) in the trial state

\[
F_n = F_{nv} - \sum_{m,i} K_{x,mu} \ddot{u}_{\text{trial}}^{m,i} - \sum_{m,i} C_{x,mu} \dot{u}_{\text{trial}}^{m,i} - \sum_{m} M_{x,mu} \ddot{\varepsilon}_{\text{trial}}^{m}.
\]  

(A8)

**Step 2: Calculate the traction force acting on the fault, in the trial state.** For steps 2, 3, and 4, we focus our attention on one particular differential node \( d \) and the corresponding common node \( c \). Suppose that there was no friction on the fault, and also that there was no constraint on the motion of the differential node (i.e., motion normal to the fault surface is allowed). Then the differential node would undergo an acceleration \( \ddot{u}_d = F_d/M_d \), and the common node would undergo an acceleration \( \ddot{u}_c = F_c/M_c \). The relative acceleration is \( \ddot{u} = \ddot{u}_d - \ddot{u}_c \). Then, the node-equivalent traction force in the trial state is

\[
F^{\text{trial}} = M_d \ddot{u}_d - M_c \ddot{u}_c.
\]  

(A9)

Note that applying an additional force \( -F^{\text{trial}} \) to the differential node would reduce the relative acceleration to zero. We now decompose \( F^{\text{trial}} \) into the sum of two forces that are normal and tangent to the fault surface

\[
F^{\text{trial}} = \sigma^{\text{trial}} A_d w + T^{\text{trial}},
\]  

(A10)

where \( \sigma^{\text{trial}} \) is the trial value of the normal stress, \( w \) is a unit vector normal to the fault surface, and \( T^{\text{trial}} \) is a vector tangent to the fault surface. We call \( T^{\text{trial}} \) the trial value of the node-equivalent shear traction force.

**Step 3: Calculate the relationship between acceleration and shear traction.** Suppose that we change the acceleration of the differential node from \( \ddot{u}_d^{\text{trial}} \) to \( \ddot{u}_d^{\text{trial}+1} \) (we do not yet know the value of \( \ddot{u}_d^{\text{trial}+1} \)). Substitute \( \ddot{u}_d^{\text{trial}+1} \) into eq. (A8) in place of \( \ddot{u}_d^{\text{trial}} \) and recompute the node-equivalent traction force shown in eq. (A9). Using eqs (A6) and (A7), the new value of the node-equivalent traction force is

\[
F = F^{\text{trial}} - M_t \left( \ddot{u}_d^{\text{trial}+1} - \ddot{u}_d^{\text{trial}} \right).
\]  

(A11)

Using the fact that \( \ddot{u}_d^{\text{trial}} \) and \( \ddot{u}_d^{\text{trial}+1} \) are both tangent to the fault surface, \( F \) can now be split into force components normal and tangent to the fault surface

\[
F = \sigma^{\text{trial}} A_d w + T,
\]  

(A12)

where the new node-equivalent shear traction force is

\[
T = T^{\text{trial}} - M_t \left( \ddot{u}_d^{\text{trial}+1} - \ddot{u}_d^{\text{trial}} \right).
\]  

(A13)

**Step 4: Solve the friction law, and calculate fault slip.** Let \( \tau \) be the magnitude of the shear stress (we do not yet know the value of \( \tau \)). We choose the node-equivalent shear traction force \( T \) as follows:

\[
T = \tau A_d \frac{u_d^{\text{trial}} + t_0 M^{-1} T^{\text{trial}}}{|u_d^{\text{trial}} + t_0 M^{-1} T^{\text{trial}}|}.
\]  

(A14)

Note that \( u_d^{\text{trial}} \) and \( T^{\text{trial}} \) are both tangent to the fault surface, and so \( T \) is also tangent to the fault surface. Also note that \( |T| = \tau A_d \) as required. We can then use eqs (A1), (A13), and (A14) to calculate the values of \( u_d^{\text{trial}+1} \), \( \dot{u}_d^{\text{trial}+1} \), and \( \ddot{u}_d^{\text{trial}+1} \) as functions of \( \tau \). Because of eq. (A14), the slip velocity vector \( \dot{u}_d^{\text{trial}+1} \) is automatically parallel to the shear traction force \( T \), as required by the friction law. We can then calculate a trial value of the friction state variable, also as a function of \( \tau \), as follows:

\[
\psi^{\text{trial}}(\tau, \sigma^{\text{trial}}, |u_d^{\text{trial}+1}|, \dot{u}_d^{\text{trial}+1}) = G(\psi, |u_d^{\text{trial}+1}|, \sigma, \sigma^{\text{trial}}, \dot{u}_d^{\text{trial}+1}).
\]  

(A15)

Next, we can write down the friction law at the end of the time step as

\[
0 = H(\tau, \sigma^{\text{trial}}, |u_d^{\text{trial}+1}|, \psi^{\text{trial}}).
\]  

(A16)

The value of \( \sigma^{\text{trial}} \) is known, whereas \( \dot{u}_d^{\text{trial}+1} \) and \( \psi^{\text{trial}} \) are known functions of \( \tau \). Therefore, eq. (A16) is just a single equation in the single variable \( \tau \). So we can solve it for \( \tau \) in FaultMod, this equation is solved using the bisection method. Although slower than other techniques, such as Newton’s method, the bisection method has the advantage that convergence is guaranteed, even if the functions \( G \) and \( H \) have poor numerical properties. Once the value of \( \tau \) is known, the slip acceleration \( \ddot{u}_d^{\text{trial}+1} \) can be computed from eqs (A13) and (A14).

**REMARK 1.** It may happen that eq. (A16) has no solution. In this case, the shear stress is too small for the fault to slide, and we just choose \( \ddot{u}_d^{\text{trial}+1} \) so that the slip rate goes to zero. We do the same thing if the denominator in eq. (A14) is close to zero, or if the calculated slip velocity vector \( \dot{u}_d^{\text{trial}+1} \) and shear traction force \( T \) point in opposite directions (i.e., if they are antiparallel instead of parallel).

**REMARK 2.** When rate-and-state friction is used, it is sometimes necessary to work with very small values of slip rate \( V = |\dot{u}_d^{\text{trial}+1}| \). In such cases, better numerical accuracy is obtained by treating \( V \) as the independent variable, and expressing \( \tau \) as a function of \( V \). Then eq. (A16) becomes an equation for \( V \), and we use the bisection method to solve for \( V \).

**Step 5: Solve the equations of motion.** As mentioned before, we can substitute eq (A1) into the discretized equations of motion (10) to obtain a system of equations in which \( \dot{u}_d^{\text{trial}+1} \) are the only unknowns. But we have already computed \( \ddot{u}_d^{\text{trial}+1} \) for all differential nodes \( d \). By treating all the \( \ddot{u}_d^{\text{trial}+1} \) values as known, we can derive a reduced system of equations in which the unknowns are \( \ddot{u}_d^{\text{trial}+1} \), where now \( m \) ranges over all nodes in the system except the differential nodes. We can solve the reduced system of equations, and then apply eq (A1), to obtain displacement, velocity, and acceleration for every node in the system.

**REMARK.** This step is very similar to performing one step of a *kinematic simulation*, where the fault slip is considered to be known, and the finite-element method is used to calculate the motions of the rest of the system. So, one way to describe our time stepping algorithm is to say that the finite-element code is performing a kinematic simulation, whereas the friction law is used to dynamically calculate the fault-slip boundary conditions.

**Step 6: Update the friction state variables.** We can now substitute the computed values of \( u_d^{\text{trial}+1}, \dot{u}_d^{\text{trial}+1}, \) and \( \ddot{u}_d^{\text{trial}+1} \) into eq. (A8) in place of the trial values. Then, we repeat the calculations leading up to eqs (A9) and (A10), and obtain the normal stress \( \sigma^{\text{trial}+1} \) at the end of the time step. Finally, we use eq. (A4) to calculate the friction state variable \( \psi^{\text{trial}+1} \) at the end of the time step.