ELECTRONIC SEISMOLOGIST

Software for Efficient Static Dislocation–Traction Calculations in Fault Simulators

by Andrew M. Bradley

Online Material: Software packages with documentation.

Quasistatic or quasidynamic rate–state friction (QRSF) simulators are used to study the mechanics of faults (e.g., Shibazaki and Shimamoto, 2007). The displacement discontinuity method (DDM; Crouch and Starfield, 1983) meshes the fault or faults into N elements and constructs a matrix of Green's functions (GFs) relating slip to stress. The simulator evolves strength and slip in time. Usually, the most expensive part of a simulation time step is the matrix–vector product (MVP) of the slip distribution with the DDM matrix; the straightforward implementation performs $O(N^2)$ operations. A simulator performs thousands to millions of MVPs with the same DDM matrix.

My free and open-source software (FOSS) package *hmmvp* speeds up simulations by an asymptotic factor of a little less than N faster than the straightforward implementation for both forming an approximation to and performing MVP with the GF matrix.

In QRSF simulations, rupture tip length scales as $f_r \propto \mu' \mathrm{d_c}/(\mathrm{b}\sigma)$ for the aging evolution law and f_r multiplied by a factor that depends on slip speed and background values for the slip law. In this case, $\mu' = \mu/(1-\nu)$, μ is the shear modulus, ν is Poisson's ratio, b is the constant multiplying the state term in rate–state friction, σ is the effective normal stress, and d_c is the characteristic slip distance for friction evolution (Rubin and Ampuero, 2009). Rupture tips must be well resolved in simulations. If f_r is nonuniform on the fault, discretization also can be nonuniform for efficiency. However, not all DDM operators are accurate on nonuniform meshes. My FOSS package *dc3dm* implements a method, IGA, that is as accurate on a nonuniform mesh as the standard method is on a uniform mesh. *dc3dm* uses *hmmvp* to efficiently approximate the IGA operator.

H-MATRIXES AND *hmmvp*

A variety of methods can speed up the MVP. Let a rectangular fault be discretized uniformly into $N = N_s N_d$ elements, in which N_s is the number of along-strike elements and N_d is the number along dip. The straightforward implementation is $O(N^2)$, in which O is big-O notation. In a homogeneous elastic (HE) full-space, the fast Fourier transform (FFT) can be used to calculate stress from slip in $O(N \log N)$ work. In an HE half-space, the FFT can be used in the along-strike direction to implement the convolution, resulting in $O(N_d^2 N_s \log N_s)$ time. In general, the FFT cannot be applied to a nonuniformly discretized or nonplanar fault. A class of methods, including fast multipole methods (Greengard and Rokhlin, 1987; Ying *et al.*, 2004), Barnes–Hut (Barnes and Hut, 1986), and hierarchical matrixes (H-matrixes; Hackbusch, 1999), are instead applicable. These methods take advantage of off-diagonal block (for an appropriate permutation) approximate low-rank structure. A block $\mathbf{B} \approx \mathbf{U}\mathbf{V}^T$, with $\mathrm{columns}(\mathbf{U}) \ll \min(\mathrm{rows}(\mathbf{B}), \mathrm{columns}(\mathbf{B}))$. For a mesh having N elements, work is close to $O(N)$. Among these methods, the H-matrix is best suited to QRSF simulators because it is fastest when the operator is formed infrequently relative to the number of times it is applied, it handles complicated GFs as easily as simple ones, and it can handle certain boundary conditions (BCs) more efficiently than the other methods.

Many others have used low-rank approximation (LRA). Ohtani *et al.* (2011) work in the same rate–state friction framework as I do and describe their use of the H-matrix package *Hlib* (Hackbusch, 1999; Börm *et al.*, 2003). They found that to get adequate performance, they had to empirically decrease the ranks (relative to those *Hlib* prescribed) of the far-off-diagonal blocks of their matrix; I believe method M I describe herein may solve this problem. Maerten (2010) focuses on using an H-matrix in static problems; the software is proprietary. Coulier *et al.* (2013) use H-matrix approximation for an elastodynamic boundary-element method (BEM); see also the references therein for other example applications in the framework of the BEM for elastostatics and elastodynamics. I believe that all these and similar applications of H-matrix approximation have used the less efficient method B (described below) to control error.

In this section, I focus on some details of the H-matrix method that I have implemented in *hmmvp*. Let $\mathbf{B}$ be the $M \times N$ matrix that implements the DDM operator and $\bar{\mathbf{B}}$ be the approximation to it. The procedure to construct an H-matrix has four parts. First, based on distance, a cluster tree over mesh elements is formed. If source and receiver element sets are not the same, a cluster tree is formed for each set. The cluster trees induce row and column permutations of $\mathbf{B}$. For notational brevity, hereafter I assume $\mathbf{B}$ is already permuted. Second, pairs of clusters are found that satisfy a criterion involving distance between the two clusters and their diameters; associated with pair i is a block of $\mathbf{B}$, $\mathbf{B}_i$. Third, the requested error tolerance ε (hereafter usually just "tolerance") is mapped to tolerances on each block $\mathbf{B}_i$. The tolerance specifies the maximum error allowed. Fourth, each block is approximated by an LRA that satisfies the block's tolerance.

An LRA to an $m \times n$ block $\mathbf{B}_i$ can be efficiently expressed as an outer product of two matrixes $\mathbf{U}$ and $\mathbf{V}$: $\mathbf{B}_i \approx \bar{\mathbf{B}}_i = \mathbf{U}\mathbf{V}^T$. Let r be the number of columns in $\mathbf{U}$; then r is the maximum rank of $\bar{\mathbf{B}}_i$ and the rank if $\mathbf{U}$ and $\mathbf{V}$ have independent columns, as is always the case in this work. $\mathbf{B}_i$ requires $O(mn)$ storage; $\bar{\mathbf{B}}_i$ requires $O[r(m+n)]$.

Let $\delta\mathbf{B} \equiv \mathbf{B} - \bar{\mathbf{B}}$. In *hmmvp*, the tolerance ε bounds the matrix error as $\|\delta\mathbf{B}\|_F \le \varepsilon\|\mathbf{B}\|_F$. This specification of the error bound must be mapped to one for each block. There are at least two methods. The standard method is what I call method B, for block-level relative error control (REC): $\|\delta\mathbf{B}_i\|_F \le \varepsilon\|\mathbf{B}_i\|_F$. Then $\|\delta\mathbf{B}\|_F^2 = \sum_i \|\delta\mathbf{B}_i\|_F^2 \le \varepsilon^2 \sum_i \|\mathbf{B}_i\|_F^2 = \varepsilon^2\|\mathbf{B}\|_F^2$, in which $\sum_i$ sums over the blocks $\mathbf{B}_i$ of $\mathbf{B}$. I have found that a second method, method M for matrix-level REC, yields greater compression: $\|\delta\mathbf{B}_i\|_F \le \varepsilon \frac{\sqrt{m_i n_i}}{\sqrt{MN}} \|\mathbf{B}\|_F$. As $MN = \sum_i m_i n_i$, $\|\delta\mathbf{B}\|_F^2 = \sum_i \|\delta\mathbf{B}_i\|_F^2 \le \varepsilon^2 (MN)^{-1} \|\mathbf{B}\|_F^2 \sum_i m_i n_i = \varepsilon^2 \|\mathbf{B}\|_F^2$. Method M tends to request less accuracy than does method B for the large far-off-diagonal blocks and more for the small near-diagonal blocks, which often must be exact in any case, and increases in relative efficiency with the order of singularity of the GF. At least one other researcher has described method M (Hackbusch, 2009). In Bradley (2011), I compared methods B and M on several model GFs.

The singular value decomposition (SVD) is the optimal method (in the 2 and Frobenius norms) to obtain an LRA, but it is far too expensive for large blocks for at least two reasons. First, the computation itself is expensive. Second, ideally only a small fraction of the elements in a block are computed, but the SVD requires them all. However, the SVD can be used efficiently in a certain instance. If $\mathbf{A} = \mathbf{U}\mathbf{V}^T$ and $\mathbf{U}$ has few columns, a combination of the thin QR factorizations of $\mathbf{U}$ and $\mathbf{V}$ (so that in $\mathbf{U} = \mathbf{Q}\mathbf{R}$, $\mathbf{Q}$ has only as many columns as $\mathbf{U}$) and the SVD of the product of their triangular factors gives the SVD of $\mathbf{A}$. This technique is often called the reduced SVD or SVD recompression in the context of LRA methods (Börm *et al.*, 2003).

A practical method to find the LRA of large blocks efficiently, though suboptimally, is adaptive cross approximation (ACA) (Bebendorf and Rjasanow, 2003). I followed the implementation in the software package AHMED (Bebendorf, 2008), with a modification for method M. In *hmmvp*, ACA gives an initial approximation $\bar{\mathbf{B}}_i^{(0)} \approx \mathbf{U}^{(0)}(\mathbf{V}^{(0)})^T$. SVD recompression using this initial approximation then efficiently gives the final approximation $\bar{\mathbf{B}}_i^{(f)} \approx \mathbf{U}^{(f)}(\mathbf{V}^{(f)})^T$, with $\mathrm{rank}(\bar{\mathbf{B}}_i^{(f)})$ often tens of percent smaller than $\mathrm{rank}(\bar{\mathbf{B}}_i^{(0)})$. SVD recompression can also be used to quickly construct a second H-matrix at looser tolerance from a first.

hmmvp contains a C++ program to compress a matrix and a library to compute MVP. It handles arbitrary 2D and 3D distributions of elements and smooth, decaying GFs. The user writes a C++ class specifying the element locations and implementing the GF. Parallelization uses OpenMP or MPI.

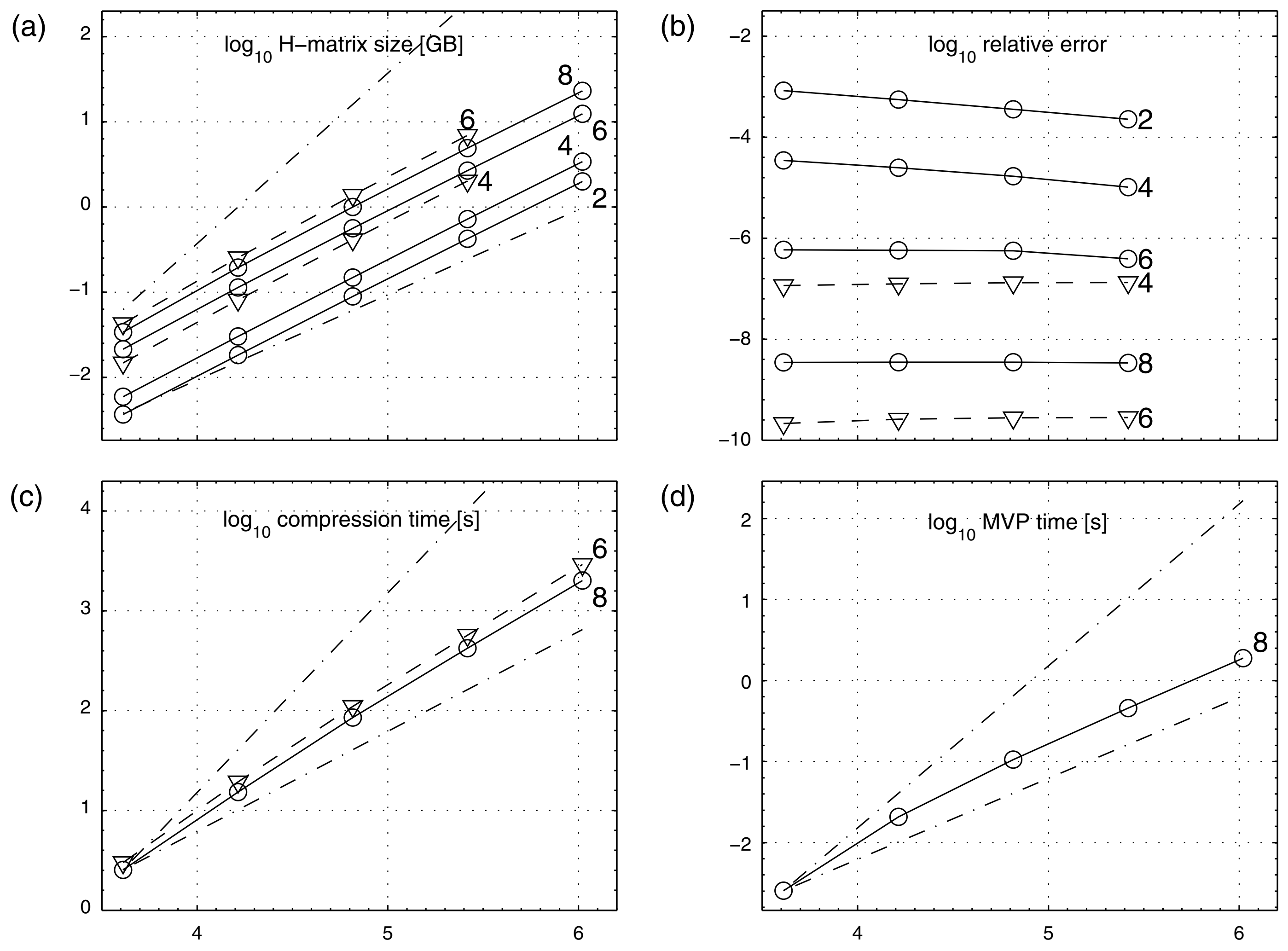
A numerical test is constructed as follows. A square planar fault dips at 12° in an HE half-space; the top of the fault is at the surface. The Poisson's ratio is set to 1/4. Shear modulus and length are nondimensionalized. The fault is uniformly discretized into N squares. Refinement divides each square into four. An entry in the $N \times N$ matrix $\mathbf{B}_N$ is computed using the GF for a constant-dislocation rectangular source (Okada, 1992). In this test, column i of $\mathbf{B}_N$ relates slip in a shear component of element i to that component of traction on each element; the shear component is a linear combination of strike and dip directions. H-matrixes are formed for a sequence of $N = 4^k$ for $k = 6$–10, with methods M and B, and at $\varepsilon = 10^{-k}$ for $k = 8, 6, 4$, and 2 for M and $k = 6, 4$ for B. The programs were run on a computer having these specifications: 16 cores, 2.6 GHz AMD Opteron 6212, and 32 GB memory.

Figure 1 shows the results. In all four plots, the x axis is $\log_{10} N$, and the y axis is indicated by the text title. Solid lines are for method M, dashed lines are for method B, and dash-dotted lines are reference lines.

In Figure 1a, H-matrix sizes, including metadata, are plotted. The top reference line is for storage of a single-precision full matrix. The bottom reference shows the slope for $O(N)$ scaling. The jump in compression for method M between the first two and second two sets of measurements results from an automatic switch from single to double precision. As a specific example, for $N = 1024^2$ and with $\varepsilon = 10^{-6}$, method M produces a 12.4 GB matrix, which is 330 times less storage than the 4 TB required for a single-precision full matrix of that size. The size of the files for method B on the largest mesh were not recorded. For $N = 512^2$ and with $\varepsilon = 10^{-6}$, method M yields a 2.7 GB matrix and method B yields a 6.9 GB matrix.

The measured relative errors $\|\delta\mathbf{B}_N\|_F / \|\mathbf{B}_N\|_F$ are plotted in Figure 1b. Every matrix has less error than requested. Any accuracy greater than that requested means work and storage are wasted. Every H-matrix of method M is within a factor of 10 of the requested tolerance except those for $\varepsilon = 10^{-2}$, which are within a factor of 100. In contrast, the H-matrixes of method B are a little less than 10^3 (for $\varepsilon = 10^{-4}$) and almost 10^4 (for $\varepsilon = 10^{-6}$) times more accurate than requested. The slight negative slope in the coarse-tolerance method-M curves results from approximations in the spatial decomposition that slightly alter the blocks at each resolution level. In general, I found that for DDM matrixes, method M produces a more efficient approximation than method B for a requested error tolerance by producing an approximation $\bar{\mathbf{B}}$ that is little more accurate than is requested; for identical achieved tolerances, methods M and B are about equally efficient.

Because ACA does not require every element of a block, H-matrixes can be formed in a time that scales better than $O(N^2)$. In Figure 1c, compression time in wall-clock seconds is shown for method M with $\varepsilon = 10^{-8}$ and method B with $\varepsilon = 10^{-6}$; 16 cores were used. Times are not shown for other values because those matrixes were derived from the most accurate matrixes using SVD recompression, which is fast enough that file input/output is the bottleneck. Reference lines are shown for $O(N)$ and $O(N^2)$ scaling. For $N = 1024^2$, com-



▲ **Figure 1.** Results for *hmmvp* numerical experiment. The *x* axis is shown as $\log_{10} N$, in which N is the number of elements in fault mesh; the *y* axis is indicated by plot titles. Solid curves are for method M; dashed lines are for method B; and dash-dotted lines are the reference lines of $O(N)$ and $O(N^2)$. Numbers k indicate tolerance 10^{-k}.

pression using method M took 34 minutes. Calculating every entry of this matrix using 16 cores and ignoring memory movement would take several tens of hours.

In Figure 1d, the time to compute an MVP using eight cores and OpenMP is plotted, along with $O(N)$ and $O(N^2)$ references. For $N = 1024^2$, $\varepsilon = 10^{-8}$, and method M, an MVP takes 1.9 s. Ignoring memory movement and assuming perfect efficiency, an MVP with a dense matrix of this size on 16 cores would take ten to a few tens of seconds. In practice, the time would be quite a bit longer because of memory movement.

IGA AND dc3dm

IGA is a DDM for a nonuniformly discretized rectangular planar fault in an HE half-space. The standard DDM for a uniform mesh in this physical setting, which I call DDMu and is essentially that of Crouch (1976), is as follows. Discretize the fault into uniformly sized rectangles. Use Okada's routine *DC3D* (Okada, 1992) (or an equivalent for a different physical setting) to calculate traction τ at the center of each element due to slip s, which is constant over each element. For each component n of slip and m of traction, these calculations generate the matrix $G_u^{(mn)}$, in which the element in row i and column j is the traction component m at element i's center due to a unit dislocation in component n in element j.

Components of interest are of several types: shear traction due to parallel shear dislocation (SS), shear traction due to orthogonal shear dislocation (SOS), normal traction due to normal dislocation, normal traction due to shear dislocation, and shear traction due to normal dislocation (SN). Only three need to be distinguished in what follows; I take these three to be SS, SOS, and SN.

The order of accuracy (OOA) of a 3D DDM is $-2\Delta \log \text{error}/\Delta \log N$, the change in the logarithm of the error for a given change in the logarithm of the number of elements, for a suitable mesh refinement scheme. DDMu has an OOA of 2 for SS and SN and 1 for SOS. All OOA I provide without explanation must be viewed as empirical; I do not have proofs for them.

A number of DDMs are available in addition to that of Crouch (1976). IGA is most similar to that described in Shou *et al.* (1997). They developed a DDM for a planar, rectangular, uniformly discretized crack based on the DDM in Shou and Crouch (1995) for a line crack. Biquadratic interpolation over a 3×3 patch of elements gives the slip in the middle source element. Extending their method to a tensor nonuniform mesh, or even to a logically rectangular mesh, might be possible. Vijayakumar *et al.* (2000) describe a method in which slip is piecewise linear on a triangular mesh. They report evaluating tractions at element vertices. As stress is singular at such locations for piecewise linear slip, I speculate they regularize the integral for the stress at the nodal points of the source element. See Shou *et al.* (1997) and Vijayakumar *et al.* (2000) for a list of other DDMs.

In all DDMs to date, elements are flat even if the slip representation is at least locally smooth. On a curved fault, certain components of stress are singular at flat element edges even if the slip field is smooth. On any fault, certain components of stress are singular at element edges if the slip field does not have a continuous derivative. Generally, if a singularity in the stress field is present due to either of these sources, only a uniform mesh or a mesh for which the nonuniformity is governed by a smooth function will give acceptable OOA. IGA uses a sufficiently smooth representation of slip on a nonuniform mesh and is intended to be used only on flat faults; it uses a discontinuous representation of slip on the underlying uniform mesh and, therefore, is able to use the efficient closed-form GF of Okada (1992).

The procedure of applying DDMu to a nonuniform mesh is called DDMu(n) in this work. There are at least two useful categories of nonuniform meshes having rectangular elements. The first is a tensor mesh having element sizes that are prescribed by a smooth function. The second is a mesh constructed by recursively splitting elements of an initially uniform mesh. On the first type of mesh, the OOA of DDMu(n) for component SS is 1; on the second, it is 1/2. IGA uses the second type of mesh. Usually the second type is more efficient than the first in terms of number of elements.

Let $\mathcal{M}_u$ be a uniform mesh. Each entry of DDMu's operator G_u corresponds to the output of one call to Okada's *DC3D*; such an entry is described here as a simple GF.

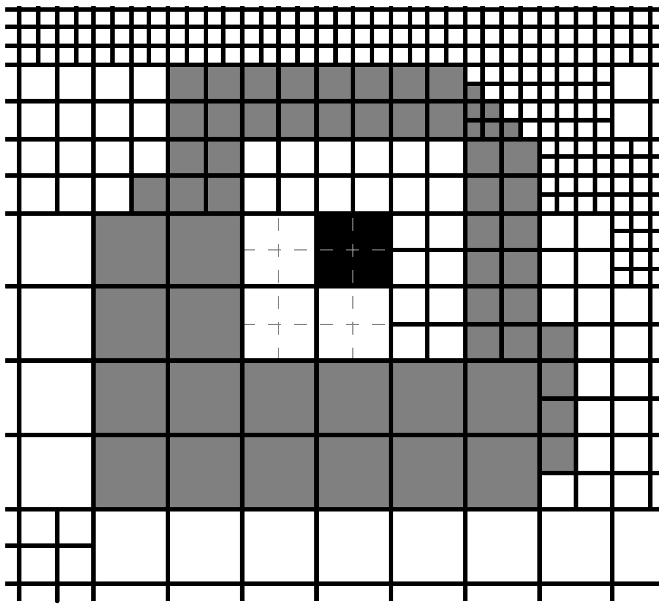
Let $\mathcal{M}_n$ be a nonuniform mesh having the following property: each element must tile each element larger than itself. Let the smallest element in $\mathcal{M}_n$ have the same size as an element in $\mathcal{M}_u$. By these two properties, every element $e \in \mathcal{M}_n$ has associated subelements in $\mathcal{M}_u$ that tile e. I also choose $\mathcal{M}_n$ so that each neighbor element of element e is no smaller than half or larger than twice e's length: the resulting mesh is referred to as "smooth." $\mathcal{M}_n$ is constructed according to the resolution function f_r, which maps a fault coordinate to a maximum permissible element size.

Let $\mathcal{T}_n$ be a triangulation induced by $\mathcal{M}_n$. Triangulations are nonunique. I choose the triangulation to have this property: every rectangular element is covered by the triangles radiating from its center. This property is one way to get fast triangle lookup, given a coordinate. On a smooth mesh, this triangulation is almost a Delaunay triangulation.

Let $\mathbf{I}_{n \to u}$ (sometimes written as just $\mathbf{I}$) be a linear operator mapping data on $\mathcal{M}_n$ to $\mathcal{M}_u$. It implements smooth (C^1) interpolation. I use the local cubic Clough–Tocher interpolant over $\mathcal{T}_n$, following the details in section 2 of Alfeld (1984). This interpolant needs a gradient estimate at each node. I fit a quadratic function to the data at the node and its neighbors and calculate the gradient of this quadratic function at the node. These calculations are implicit; because $\mathbf{I}$ is a linear operator, entries are calculated without reference to particular data. This operator has an OOA of at least 2.

Let $\mathbf{A}_{u \to n}$ (or just $\mathbf{A}$) be a linear operator mapping data on $\mathcal{M}_u$ to $\mathcal{M}_n$. Let $e \in \mathcal{M}_n$ be tiled by $E \subset \mathcal{M}_u$. $\mathbf{A}$ averages values at the centers of $f \in E$ to the center of e. Because the center of e is also the center of E, averaging is equivalent to a linear fit followed by interpolation. Hence $\mathbf{A}$ has an OOA of 2.

These three linear operators together implement exact IGA (EIGA): $\mathbf{G}_n \equiv \mathbf{A}_{u \to n} \mathbf{G}_u \mathbf{I}_{n \to u}$. To be clear, $\mathbf{G}_n$ is the output of the IGA method; $\mathbf{A}$, $\mathbf{G}_u$, and $\mathbf{I}$ are intermediate quantities. The interpretation of EIGA is as follows: it interpolates slip sufficiently smoothly to a fine uniform mesh, computes the DDMu solution on this mesh, and then coarsens traction to the nonuniform mesh. As $\mathbf{I}$ and $\mathbf{A}$ have OOA of at least 2 and $\mathbf{G}_u$ is the DDMu operator on a uniform mesh, the OOA of the method is that of DDMu on a uniform mesh.

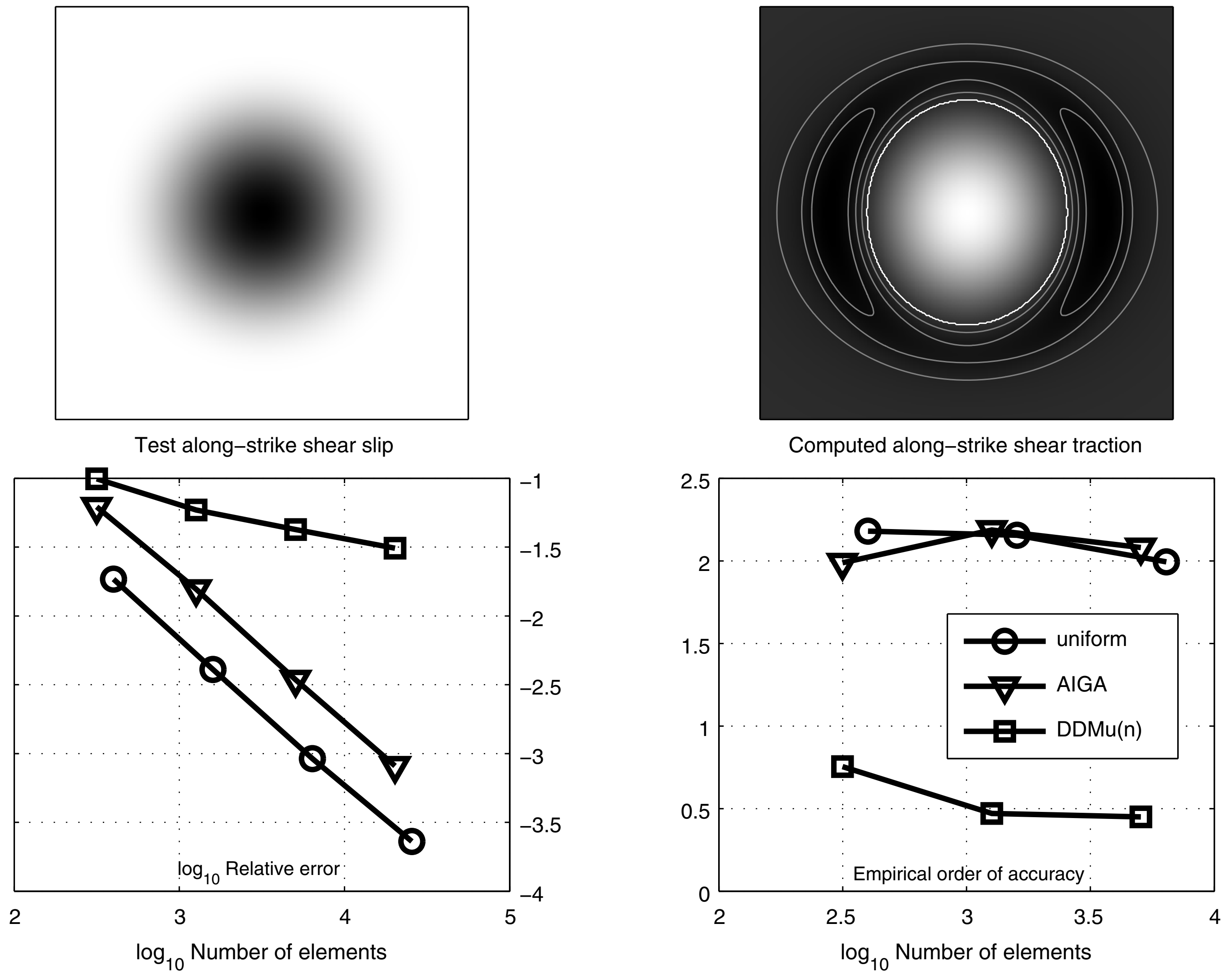


▲ **Figure 2.** Nonuniform mesh and element sets in the IGA algorithm. Solid lines indicate element edges. The black solid element is the receiver; all other elements are sources in this example. The inner white source elements and receiver element are subdivided into a uniform size, indicated by dashed lines, and contribute directly to the Green's function through full IGA calculations. The gray source elements contribute to interpolation calculations for the inner white elements. The Green's function for each gray and outer white source element is simple instead of an IGA calculation.

EIGA has the undesirable property that its computational complexity is determined by the smallest element in $\mathcal{M}_n$; this element induces the mesh $\mathcal{M}_u$ for which the three matrixes $\mathbf{A}$, G_u, and I must be computed. Approximate IGA (AIGA) uses an additional mechanism to solve this problem. Define a parameter δ_r to set the receiver neighborhood (nbhd) size. It must be a number between 0 (no neighborhood; in fact, identical to DDMu(n)) and a problem-dependent value at which EIGA is obtained.

Let $n_j \equiv \mathrm{nbhd}(e_j, \delta)$ be the set of elements such that each element $e \in n_j$ has distance (for two elements, the minimum 2-norm of the difference between a point in each element) from e_j no greater than $\delta\,\mathrm{length}(e_j)$. Let e_j be a receiver element. The initial neighborhood for e_j is $n_j^i \equiv \mathrm{nbhd}(e_j, \delta_r)$. The final neighborhood for element e_j is $n_j^f \equiv \mathrm{nbhd}(e_j, \delta_r^f)$, in which δ_r^f is the minimum value $\ge \delta_r$ such that if $e_j \in n_k^i$, then $e_k \in n_j^f$. In words, the final neighborhood is constructed such that if element 1 is in element 2's initial neighborhood, then each element is in each other's final neighborhood. Let e_j^S be the smallest element in n_j^f; $\mathrm{length}(e_j^S)$ sets the subelement size that tiles every element in n_j^f.

Figure 2 illustrates details of the IGA algorithm. The solid lines make up the mesh. The black solid element is the receiver e_j, and all element sets in this example are defined with respect to this receiver. Full IGA source–receiver calculations are carried out for sources $e \in n_j^f$ (white inside of the shaded layer, including the black element itself). The smallest element in this set governs the subelement (gray dashed lines) size for all elements in this set. A source e in a layer (shaded) around n_j^f may contribute to IGA calculations for sources inside n_j^f

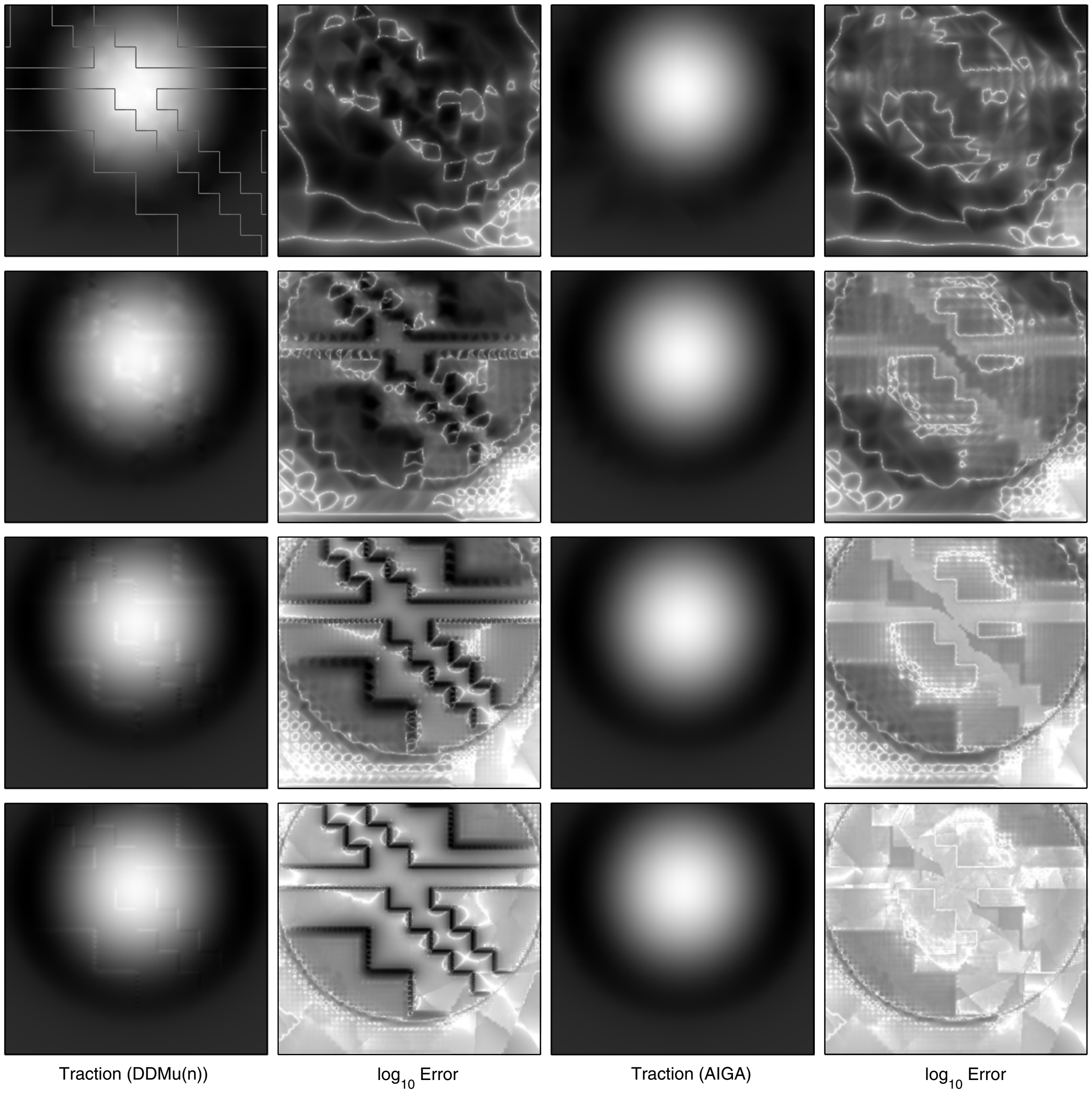


▲ **Figure 3.** Results for *dc3dm* numerical experiment. (Top) Plots for the test of along-strike shear slip function and the resulting computed plots of along-strike shear traction. In the slip image, white is zero, and dark is positive. In the traction image, the white contour separates the negative (inside) from the positive (outside), and gray contours are in the positive region only. On each edge is a zero-velocity boundary condition. (Bottom) The relative error in traction and empirical OOA.

through interpolation, but e itself is not broken into subelements. For all other sources (white outside of the shaded layer), the GFs are simple.

If δ_r^i (i.e., δ_r at mesh refinement level i) is chosen correctly, AIGA has the same OOA as EIGA. For example, if a nonuniform mesh is refined by splitting each element, set $\delta_r^i = [2^i(2\delta_r^0 + 1) - 1]/2$. This choice implements the following rule: let E^j be the set of elements that tile the element e^i at refinement level $i < j$. Then the areas of the neighborhoods around e^i and $f \in E^j$ must be the same in the limit of infinite refinement.

The FOSS package *dc3dm* implements AIGA. It uses *hmmvp* for H-matrix compression and to compute MVP. Recall that (for reasonable tolerance ε) *hmmvp* does not require every



▲ **Figure 4.** Traction (columns 1 and 3; same gray scale) and relative error in traction (columns 2 and 4; same gray scale; darker color indicates greater error) for DDMu(n) (columns 1 and 2) and AIGA (columns 3 and 4). Images correspond to the convergence test in Figure 3; images are slightly enlarged relative to the full fault to reveal details. From top to bottom, refinement increases successively by 2 in each dimension. (Top left) The lines indicate where the mesh changes element size.

entry of $\mathbf{G}_n$; for this and other reasons, $\mathbf{I}$, $\mathbf{G}_u$, and $\mathbf{A}$ and related quantities are not explicitly computed. DDMu is recovered when the mesh is uniform. Matrixes for all nine source–receiver dislocation–traction pairs and linear combinations of dislocations and tractions, respectively, can be calculated. BCs can be periodic in the surface-parallel direction in a half-space and in both directions in a whole space, velocity, and free surface. Periodicity is approximate: the domain is repeated periodically a finite number of times. For a given source–receiver pair, the periodically repeated source nearest the receiver is used as the primary source, and then a specified number of layers are constructed.

A suite of empirical convergence tests was developed. As in the experiment for Figure 1, the fault is a square dipping at 12° in an HE half-space, Poisson's ratio is set to 1/4, and shear modulus and length are nondimensionalized. The top of the fault is at the free surface in some tests. In the test in Figure 3, the top is beneath the free surface by half the depth range of the fault. The suite includes every corner combination of BCs and every source–receiver component. A base uniform mesh is created for DDMu and a base nonuniform mesh for AIGA and DDMu(n) at refinement level 0. In the nonuniform mesh, the smallest element is 16 times smaller in area than the largest element. At level i, each base element is divided uniformly into 4^i elements. In δ_r^i, $\delta_r^0 = 1$. Coarse solutions are mapped to a uniform fine mesh using IGA's interpolant, and the relative error is with respect to the DDMu solution on this fine mesh. Whether AIGA is more accurate than DDMu or the opposite is arbitrary, as the mesh and test slip function are chosen independently; only the OOA matters. The mesh is chosen to aggressively test AIGA. The lines in the top-left image in Figure 4 segment the image by element size. Small and medium elements cross through large elements in an X pattern, with one bar of the X aligned with the mesh and the other cutting it diagonally. The test slip function is chosen to permit converged results without refining the mesh too many levels and to respect the BCs. Figure 3 shows one example test in which 0 velocity is imposed on all boundaries; the source–receiver components are SS; and the test slip function is $s(x,y) = [1-(r/R)^2]^3$ for $r \le R$ and 0 otherwise, in which $s(x,y)$ is slip, $r = \sqrt{x^2+y^2}$, $R = 2L/5$, and L is the length of the square. The empirically obtained OOA support the hypothesis that DDMu(n) has an OOA of 1/2 and DDMu and AIGA have OOA of 2 for SS.

Figure 4 illustrates the reason for the different OOA. It shows images of traction and error in traction for the test in Figure 3, from coarsest mesh at top to finest mesh at bottom. DDMu(n) causes errors where adjacent elements differ in size (outlined by solid lines in the top-left image). The peak magnitude of the error stays approximately constant with refinement, but the area of large error decreases. Hence the OOA drops below that of DDMu but is above 0. In contrast, AIGA produces a smooth and accurate traction field.

The errors in DDMu(n) traction fields will of course affect QRSF simulations. Subtly, the QRSF model smoothes stress concentrations; therefore, time-dependent results are smooth (if elements are sufficiently small to meet the stability criterion) regardless of whether the DDM operator is accurate and, in particular, regardless of whether the traction for a smooth slip function is smooth. For this reason, we should not interpret smoothness of time-dependent results as indicating sufficient accuracy; rather, we must perform a convergence test using a method having acceptably high OOA, such as DDMu or AIGA.

DATA AND RESOURCES

No data were used in this paper. The current versions of *hmmvp* and *dc3dm* are available at pangea.stanford.edu/research/CDFM/software (last accessed July 2014). Ⓔ Additionally, the versions used in this paper are available in the electronic supplement to this paper.

ACKNOWLEDGMENTS

I would like to thank Paul Segall for helpful discussions and supporting this research, and Sylvain Barbot, one anonymous reviewer, and the Editor-in-Chief Zhigang Peng for their helpful comments. This research was supported by National Science Foundation Grant EAR-0838267, U.S. Geological Survey Grant G12AP20030, and Southern California Earthquake Center Grant 13096.

REFERENCES

Alfeld, P. (1984). A trivariate Clough–Tocher scheme for tetrahedral data, *Comput. Aided Geometr. Design* **1,** no. 2, 169–181.

Barnes, J., and P. Hut (1986). A hierarchical $O(N \log N)$ force-calculation algorithm, *Nature* **324,** 446–449.

Bebendorf, M. (2008). *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Springer, Berlin, Germany.

Bebendorf, M., and S. Rjasanow (2003). Adaptive low-rank approximation of collocation matrices, *Computing* **70,** no. 1, 1–24.

Börm, S., L. Grasedyck, and W. Hackbusch (2003). *Hierarchical Matrices*, Lecture Notes, Max-Planck-Institut fur Mathematik in den Naturwissenschaften, Leipzig, Germany.

Bradley, A. M. (2011). *H-Matrix and Block Error Tolerances*, available at http://arxiv.org/abs/1110.2807 (last accessed July 2014).

Coulier, P., S. François, G. Lombaert, and G. Degrande (2013). Application of hierarchical matrices to boundary element methods for elastodynamics based on Green's functions for a horizontally layered halfspace, *Eng. Anal. Bound. Elem.* **37,** no. 12, 1745–1758.

Crouch, S. L. (1976). Solution of plane elasticity problems by the displacement discontinuity method: I. Infinite body solution, *Int. J. Numer. Meth. Eng.* **10,** 301–343.

Crouch, S. L., and A. M. Starfield (1983). *Boundary Element Methods in Solid Mechanics*, George Allen and Unwin, London, United Kingdom.

Greengard, L., and V. Rokhlin (1987). A fast algorithm for particle simulations, *J. Comput. Phys.* **73,** no. 2, 325–348.

Hackbusch, W. (1999). A sparse matrix arithmetic based on H-matrices. Part I: Introduction to H-matrices, *Computing* **62,** 89–108.

Hackbusch, W. (2009). *Hierarchische Matrizen: Algorithmen und Analysis*, Springer, Berlin, Germany (in German).

Maerten, F. (2010). Adaptive cross approximation applied to the solution of system of equations and post-processing for 3D elastostatic problems using the boundary element method, *Eng. Anal. Bound. Elem.* **34,** no. 5, 483–491.

Ohtani, M., K. Hirahara, Y. Takahashi, T. Hori, M. Hyodo, H. Nakashima, and T. Iwashita (2011). Fast computation of quasi-dynamic earthquake cycle simulation with hierarchical matrices, *Procedia Comput. Sci.* **4,** 1456–1465.

Okada, Y. (1992). Internal deformation due to shear and tensile faults in a half-space, *Bull. Seismol. Soc. Am.* **82,** 1018–1040.

Rubin, A. M., and J.-P. Ampuero (2009). Self-similar slip pulses during rate-and-state earthquake nucleation, *J Geophys. Res.* **114,** no. B11, doi: 10.1029/2009JB006529.

Shibazaki, B., and T. Shimamoto (2007). Modelling of short-interval silent slip events in deeper subduction interfaces considering the frictional properties at the unstable–stable transition regime, *Geophys. J. Int.* **171,** no. 1, 191–205.

Shou, K. J., and S. L. Crouch (1995). A higher order displacement discontinuity method for analysis of crack problems, *Int. J. Rock Mech. Min. Sci. Geomech. Abstr.* **32,** 49–55.

Shou, K. J., E. Siebrits, and S. L. Crouch (1997). A higher order displacement discontinuity method for three-dimensional elastic problems, *Int. J. Rock Mech. Min. Sci.* **34,** no. 2, 317–322.

Vijayakumar, S., T. E. Yacoub, and J. H. Curran (2000). A node-centric indirect boundary element method: Three-dimensional displacement discontinuities, *Comput. Struct.* **74,** no. 6, 687–703.

Ying, L., G. Biros, and D. Zorin (2004). A kernel-independent adaptive fast multipole method in two and three dimensions, *J. Comp. Phys.* **196,** no. 2, 591–626.

Andrew M. Bradley
Department of Geophysics
Stanford University
397 Panama Mall, 3rd Floor
Stanford, California 94305 U.S.A.
ambrad@cs.stanford.edu

Published Online 1 October 2014