Andrew M. Bradley (ambrad@cs.stanford.edu) and Paul Segall (PI; segall@stanford.edu)
SCEC 2013 Report
March 11, 2014

1 Abstract

`dc3dm` is a software package that efficiently forms and applies the linear operator relating quasistatic dislocation and traction components on a nonuniformly discretized rectangular fault with rectangular elements in a homogeneous elastic (HE) half space. This linear operator implements what is called the displacement discontinuity method (DDM).

The key properties of `dc3dm` and the algorithms it implements are: 1. The mesh can be nonuniform. 2. Work and memory scale roughly linearly in the number of elements (rather than quadratically). 3. The order of accuracy (OOA) on a nonuniform mesh is the same as that of the standard method on a uniform mesh.

Property 2 is achieved using our package `hmmvp`, which implements hierarchical-matrix (H-matrix) compression.

A nonuniform mesh (property 1) is natural for some problems. For example, in a rate-state friction simulation, nucleation length, and so required element size, scales reciprocally with effective normal stress, and the factor difference between smallest and largest required element sizes is frequently 16 to 100 by area.

On a uniform mesh, straightforward application of a constant-slip Green's function (GF) yields a DDM we refer to as DDMu. On a nonuniform mesh, this same procedure leads to artifacts that degrade the OOA of DDMu. We have developed a method we call IGA that implements the DDM using linear combinations of the same GF for a nonuniformly discretized mesh. `dc3dm` implements an approximate form of IGA.

Importantly, IGA's OOA on a nonuniform mesh is the same as DDMu's on a uniform one (property 3).

`dc3dm` and `hmmvp` are available at `pangea.stanford.edu/research/CDFM/software`.

2 Technical Report

2.1 Objectives

Quasistatic rate-state friction (QRSF) simulators are used to study the mechanics of faults. 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 (GF) 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.

In our proposal for this work, we had two objectives. First, we wanted to complete scalable implementations of all parts of `hmmvp`, a software package that speeds ups the DDM MVP to about $O(N \log N)$ asymptotically and in practice gives speedups on the order of tens to thousands, depending on problem size, with the same reduction in memory use.

Second, we intended to develop algorithms and software to calculate the Green's function for a layered elastic half space. For now, we have postponed working on this second problem because we became aware of an important problem that we felt we should solve first. The standard DDM method in a 3D half space uses constant slip elements. This method is convergent only on a subset of all fault geometries and meshes, and has an acceptably high order of accuracy (also called convergence rate) on only uniform meshes. However, nonuniform meshes are very important when performing simulations with highly variable rheology, as required resolution scales with $L_b \sim \mu' d_c/(b\sigma)$, where $\mu' \equiv \mu/(1-\nu)$, 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. Hence we developed the algorithm IGA and implemented it in and released the software `dc3dm`. The method achieves the same order of accuracy as the standard method but on a nonuniform mesh. `dc3dm` builds on `hmmvp`.

2.2 Methodology and Results

Constructing and applying a DDM has roughly three components: the Green's function, the representation of the linear operator, and the mesh and geometry. We describe our contributions in each area in what follows.

2.2.1 Greater accuracy in Okada's `DC3D`

The routine `DC3D` of [Okada, 1992], usually in the file `dc3.f`, is widely used to compute the Green's function for a rectangular dislocation in an elastic half space. This software is efficient and generally very accurate. Okada recognized several sources of numerical error and compensated for these. However, our high-resolution DDM applications, especially those on a nonuniform mesh, have revealed that more must be done.

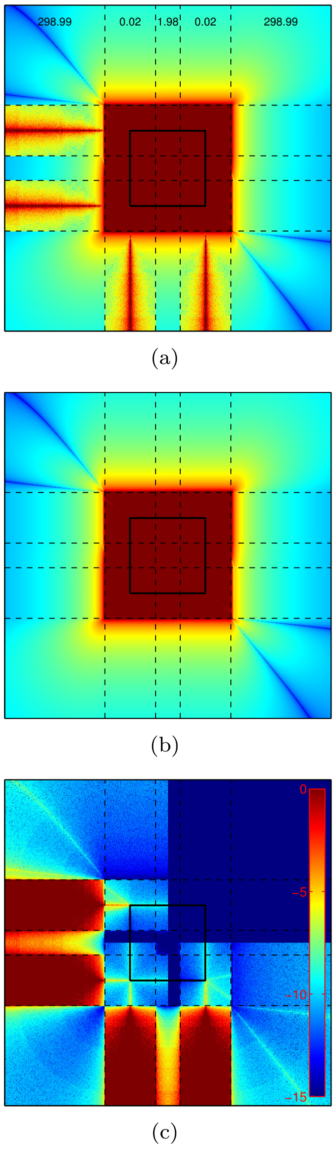


Figure 1: A demonstration of the symmetry-based increase in accuracy of Okada's routine `DC3D`.

Fig. 1 shows an example rectangular dislocation (outlined by the black square) and receivers placed (highly nonuniformly to emphasize regions of numerical error) in the plane around it. Colors in (a) and (b) correspond to $\log_{10}$ of the absolute value of the sum over displacement derivatives. (This quantity is not meant to be physical; rather, it provides a good summary of all components.) In (a), there are four cones of numerical error in the derivative calculation out of a possible eight. Each cone emerges from an edge of the rectangular dislocation. The cause of the error is numerical cancellation in expressions of the form $R+y$ for $y = \eta < 0$ or $y = \xi < 0$, where $R = (\xi^2+\eta^2+q^2)^{1/2}$ and ξ, η, q are element coordinate directions, with q normal to the element.

The solution is to use the symmetry of the problem to reflect the source-receiver geometry across the q-ξ and q-η planes so that the transformed receiver is in the $\xi, \eta \geq 0$ quadrant. Calculations for receiver points in this one quadrant are not subject to cancellation error. Then reflect the transformed solution back to the original space. The software operations are a small number of comparisons and sign changes and so are negligible relative to the other calculations. The result is shown in (b), and the relative difference between the two images in shown in (c). In (c), color corresponds to $\log_{10}$ of the absolute value of the difference between (a) and (b) divided by the maximum value over the receivers.

The modified version of Okada's software is distributed in the file `dc3omp.f` on our website's Software page in the packages `disloc3d` and `dc3dm`. `dc3omp.f` also includes OpenMP pragmas to make the code threadsafe (not parallel: it remains a serial code, as is probably best; rather, multiple threads can call `DC3D` safely without overwriting each other's `COMMON` data).

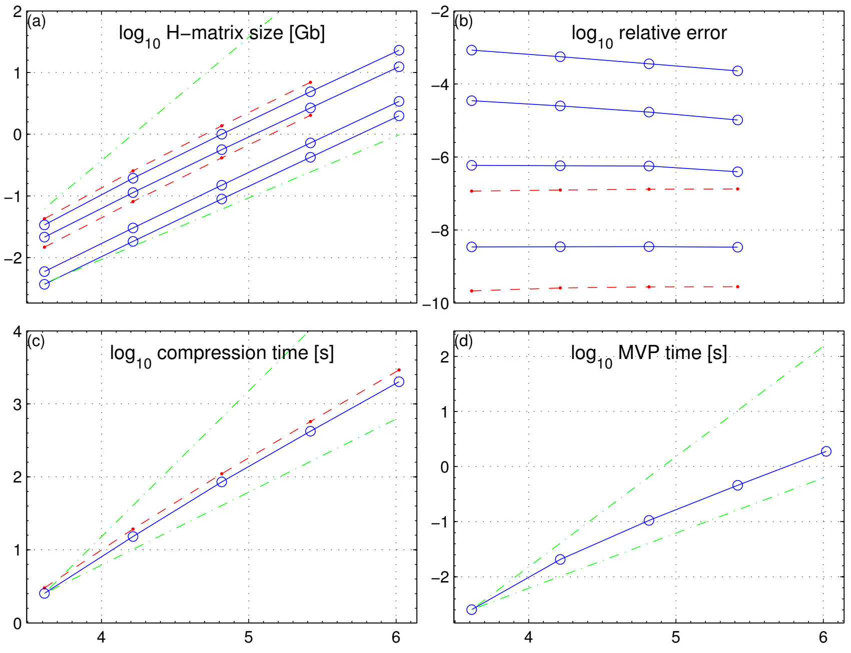


Figure 2: Results for `hmmvp` numerical experiment. On x axis is $\log_{10} N$, where N is number of elements in fault mesh. y axis is indicated by plot titles. Blue solid curves are for (M); red dashed, (B); green dash-dotted, reference slopes for linear and quadratic scaling.

2.2.2 H-matrices and `hmmvp`

A variety of methods can speed up the MVP. The FFT is the fastest when it is applicable. H-matrix approximation generally achieves the fastest MVP but slowest construction time among alternatives and so is best for problems in which the operator is formed very infrequently relative to its application; they are also useful when the GF is very complicated. H-matrices are well suited to QRSF simulators.

Let B be the $M \times N$ matrix that implements the DDM operator and $\bar{B}$ be the approximation to it, and similarly with all approximations. The procedure to construct an H-matrix has four parts. First, a cluster tree over the elements is formed. The cluster tree induces a permutation of B. For notational brevity, hereafter we assume 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 B, B_i. Third, the requested error tolerance ε is mapped to tolerances on each block B_i. The tolerance specifies the maximum error allowed. Fourth, each block is approximated by a low-rank approximation (LRA) that satisfies the block's tolerance.

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

Let $\delta B \equiv B - \bar{B}$. In `hmmvp`, the tolerance ε bounds the matrix error as $\|\delta B\|_2 \le \varepsilon \|B\|_2$. This specification of the error bound must be mapped to one for each block. There are at least two methods. The most common is what we call method (B) for *block*-level relative error control (REC): $\|\delta B_i\|_{\mathrm{F}} \le \varepsilon \|B_i\|_{\mathrm{F}}$. Proof: $\|\delta B\|_{\mathrm{F}}^2 = \sum_i \|\delta B_i\|_{\mathrm{F}}^2 \le \varepsilon^2 \sum_i \|B_i\|_{\mathrm{F}}^2 = \varepsilon^2 \|B\|_{\mathrm{F}}^2$, where $\sum_i$ sums over the blocks B_i of B. We have found that a second method yields greater compression, method (M) for *matrix*-level REC: $\|\delta B_i\|_{\mathrm{F}} \le \varepsilon \frac{\sqrt{m_i n_i}}{\sqrt{MN}} \|B\|_{\mathrm{F}}$. Proof: As $MN = \sum_i m_i n_i$, $\|\delta B\|_{\mathrm{F}}^2 = \sum_i \|\delta B_i\|_{\mathrm{F}}^2 \le \varepsilon^2 (MN)^{-1} \|B\|_{\mathrm{F}}^2 \sum_i m_i n_i = \varepsilon^2 \|B\|_{\mathrm{F}}^2$.

Numerical test. A square planar fault (but `hmmvp` can handle arbitrary 3D distributions of elements) dips at 12 degrees in an HE half space; the top of the fault is at the surface. The fault is uniformly discretized into N squares. Refinement divides each square into four. The Okada Green's function for constant-dislocation rectangular sources determines the values in the $N \times N$ matrix B_N. Column i of B_N relates slip in a shear component of element i to traction on that same component in all elements. H-matrices are formed for $N = 2^k$ for $k = 6$ to 10; with methods (M) and (B); and at different values of ε: 10^{-k} for $k = -8, -6, -4, -2$ for (M) and $k = -6, -4$ for (B). Compression and MVP were tested on a computer having these specifications: 16 cores, 2.6 GHz AMD Opteron 6212, 32 GB memory. Compression uses 16 cores with OpenMP; the MVP, 8.

Figure 2 shows results. In all four plots, the x axis is $\log_{10} N$ and the y axis is indicated by the text title. Solid blue lines are for (M) and red dashed for (B); green dash-dotted are reference lines.

In (a), H-matrix sizes, including metadata, are plotted. The top reference line is for single-precision storage of a full matrix. The bottom reference shows the slope for $O(N)$ scaling. The jump in compression for (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 $\varepsilon = 10^{-6}$, (M) produces a 12.4 GB matrix, which is 330 times smaller than the 4 TB required for a single-precision full matrix of that size. The size of the files for (B) on the largest mesh were not recorded. For $N = 512^2$ and with $\varepsilon = 10^{-6}$, (M) yields a 2.7 GB matrix and (B) a 6.9 GB matrix, which is 2.6 times larger.

In (b), measured relative errors $\|\delta B_N\|_F/\|B_N\|_F$ are plotted. Every matrix has less error than requested. Any accuracy greater than that requested means work and storage are wasted. Every (M) H-matrix 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 (B) H-matrices a little less than 10^3 (for $\varepsilon = 10^{-4}$) and almost 10^4 (for $\varepsilon = 10^{-6}$) times more accurate than requested. In general, we have found that for DDM matrices, (M) produces a more efficient approximation than (B) for a requested error tolerance by producing an approximation $\bar{B}$ that is little more accurate than is requested; for identical achieved tolerances, (M) and (B) are about equally efficient.

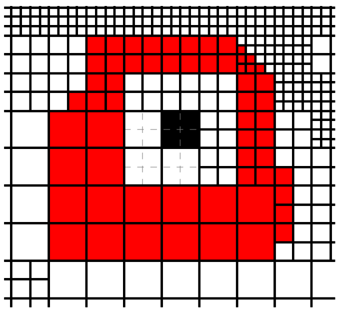


Figure 3: An IGA mesh.

Because the compression procedure accesses a subset of elements, H-matrices can be formed in a time that scales better than $O(N^2)$. In (c), compression time in wall-clock seconds is shown for (M) with $\varepsilon = 10^{-8}$ and (B) with $\varepsilon = 10^{-6}$. Times are not shown for other values because the matrices were derived from the most accurate using SVD recompression, which is fast enough that reading and writing the files is the bottleneck. Reference lines are shown for $O(N)$ and $O(N^2)$ scaling. For $N = 1024^2$, (M) took 34 minutes.

In (d), the time to compute a matrix-vector product using 8 cores 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 seconds.

This year, `hmmvp` version 1 was completed and released. All parts are implemented in `C++` and parallelized by (according to build options) MPI and OpenMP. Compression is now approximately 20 times faster than in version 0 on a 16-core machine. The MVP was also improved in parts, though speedup over the version 0 release is probably $< 2\times$.

2.2.3 IGA and `dc3dm`

Exact IGA. Let M be a mesh. A DDM constructs a matrix G to relate traction and slip by $\tau = Gs$. There is one such equation for each pair of slip and traction components.

The order of accuracy of a discretization of a PDE is the negative of the slope of log relative error as a function of log number of elements. This number is by convention multiplied by the dimension of the manifold that is discretized, which in this case is 2. (Otherwise, the OOA for a method would differ with the dimensionality of the problem.) If the OOA is > 0, the method is convergent: as the mesh is refined, accuracy of the solution increases without bound. If the OOA is ≤ 0, the method is nonconvergent. If it is < 0, then it is divergent, which means that accuracy drops with increasing refinement. If it is exactly 0, accuracy can increase with refinement until some refinement level, at which point no further gain in accuracy is possible. As a rule, we must always use convergent methods. One practical reason is that a basic and important test of the validity of a solution is to refine the mesh and check whether the solution changes. If it does not and the method is convergent, then the solution is likely valid. But if the method is nonconvergent, then getting the same solution may only mean that the maximum possible accuracy has been attained; that accuracy may not be sufficient, and one cannot know. We also prefer high-order methods if they are available, as a higher-order method does less work during the simulation for a requested accuracy.

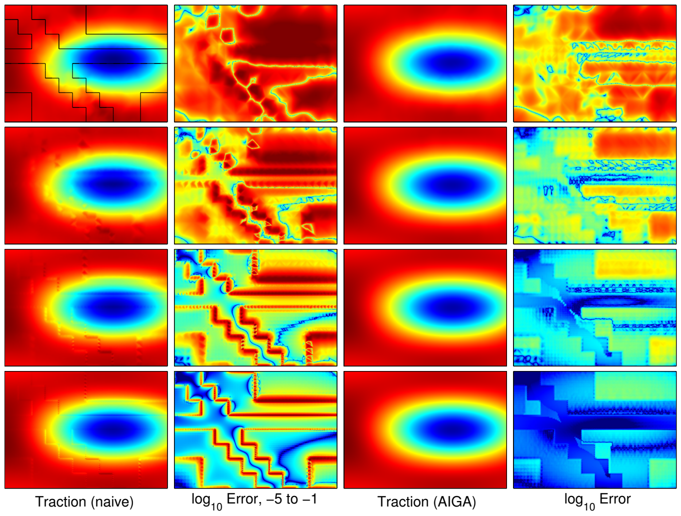


Figure 4: Tractions and pointwise absolute errors (relative to maximum traction magnitude) for DDMu(n) (left two columns) and AIGA methods (right) applied to the strike-strike GF for an example problem. From top to bottom, the level of refinement increases successively by 2 in each dimension. DDMu(n) causes errors where adjacent elements differ in size (outlined by black lines in the top-left image). Peak magnitude of the error stays approximately constant with refinement, causing a drop in order of accuracy from 2 to 1/2. Color scales are the same in respectively columns 1 and 3, and 2 and 4.

Let M_u be a uniform mesh. Each entry of DDMu's operator G_u corresponds to the output of one call of Okada's `DC3D`. Such an entry we call a *simple* GF.

Let M_n be a nonuniform mesh having the following property: Each element must tile each element larger than itself. If DDMu is applied to this mesh, we call the method DDMu(n). Let the smallest element in M_n have the same size as an element in M_u. By these two properties, every element $e \in M_n$ has associated subelements in M_u that tile e. In practice, we also choose M_n so that each element e's neighbors are no smaller than half or larger than twice e's length. M_n is constructed according to the *resolution function* f_r, which maps fault coordinate to maximum permissible element size.

Let $\mathcal{T}_n$ be a triangulation induced by M_n. Let $I_{n\to u}$ (sometimes written just I) be a linear operator mapping data on M_n to M_u. It implements smooth (C^1) interpolation; we use cubic Clough-Tocher interpolation over $\mathcal{T}_n$ (with certain choices made for the gradient estimates). This method has order of accuracy greater than 2.

Let $A_{u\to n}$ (or just A) be a linear operator mapping data on M_u to M_n. Let $e \in M_n$ be tiled by $E \subset M_u$. 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 A has OOA 2.

These three linear operators together implement exact IGA (EIGA): $G_n \equiv A_{u\to n} G_u I_{n\to u}$. To be clear, G_n is the output of the IGA method; A, G_u, and I are intermediate quantities.

Approximate IGA. EIGA has the undesirable property that its computational complexity is determined by the smallest element in M_n; this element induces the mesh M_u for which the three matrices A, G_u, and I must be computed. Approximate IGA (AIGA) uses an additional idea to solve this problem. Define a parameter δ_r that sets *receiver neighborhood size.* It is a number between 0 (no neighborhood; in fact, identical to DDMu(n)) and a problem-dependent value at which EIGA is reached.

Let $n_j \equiv \text{rbox}(e_j, \delta)$ be the set of elements such that each element $e \in n_j$ has distance from e_j no greater than δ length(e_j). Let e_j be a receiver element. The *initial neighborhood* for e_j is $n_j^i \equiv \text{rbox}(e_j, \delta_r)$. The *final neighborhood* for element e_j is $n_j^f \equiv \text{rbox}(e_j, \delta_r^f)$ for $\delta_r^f \geq \delta_r$. δ_r^f is the minimum value such that for any two elements e_j and e_k, if $e_j \in n_k^i$, then $e_k \in n_j^f$. Let e_j^S be the smallest element in n_j^f; $s = \text{length}(e_j^S)$ sets the subelement size that tiles every element in n_j^f and indexes the operators A^s, G_u^s, I^s.

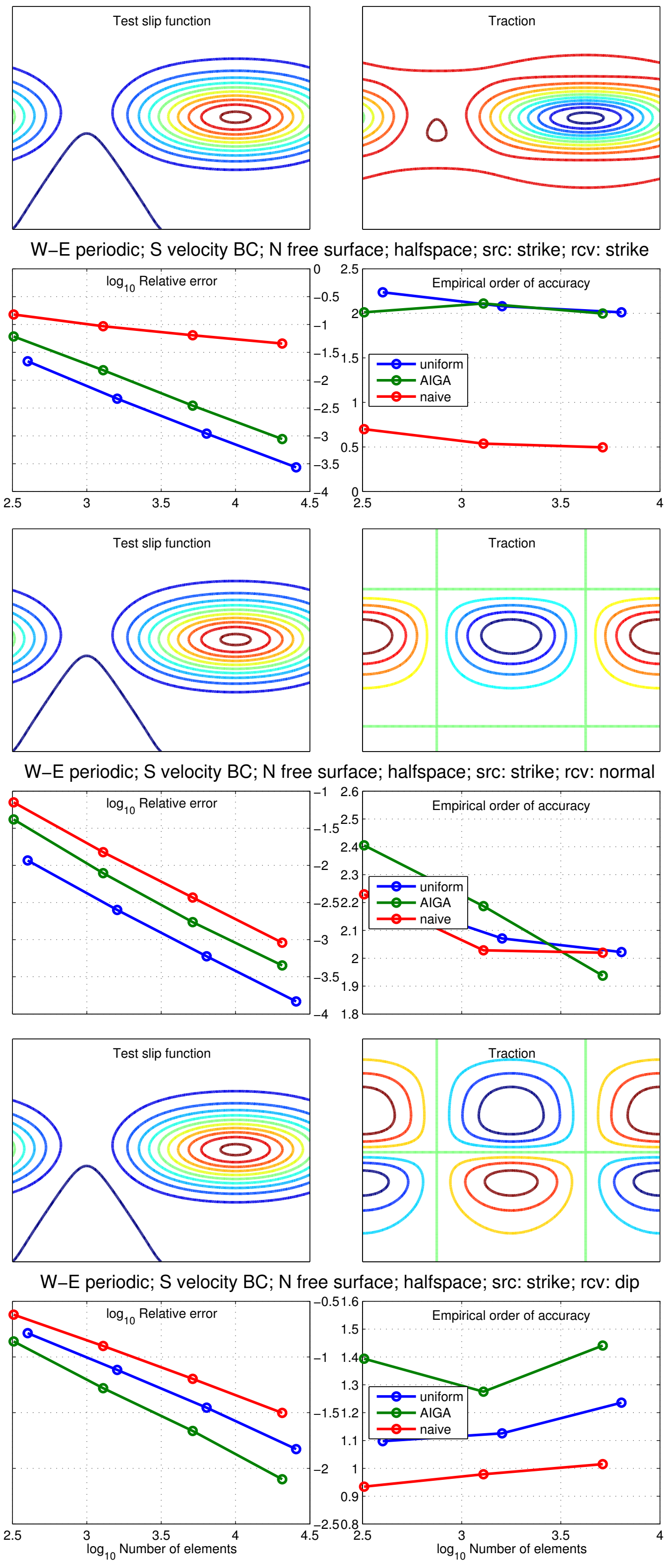


Figure 5: Convergence plots.

In the following, colors refer to Fig. 3. Full IGA source-receiver calculations are carried out for sources $e \in n_j^f$ (white inside of the red layer); these are type-1 source elements (w.r.t. receiver e_j (black)). A source e in a layer (red) around n_j^f contributes to IGA calculations for sources inside n_j^f (through interpolation), but e itself is not broken into subelements; these are type-2 source elements. All other source elements are type-3 (white outside of the red layer); associated with these are simple GFs.

`dc3dm` implements AIGA; DDMu is recovered with certain choices, so `dc3dm` can also be used simply as a convenient layer on top of `hmmvp` for problems involving planar rectangular faults. Matrices for all nine source-receiver dislocation-traction pairs and linear combinations of dislocations and tractions, respectively, can be calculated. Boundary conditions (BC) can be periodic in the surface-parallel direction (in both directions if the GF is for 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.

Convergence Analysis. Fig. 4 illustrates the application of DDMu(n) and AIGA to a sequence of increasingly refined nonuniform meshes. DDMu(n) produces quite visible artifacts where elements differ in size; AIGA does not. In this section we formalize this result.

Analysis of the DDMu shows that its OOA is 2 for the strike-strike, dip-dip, tensile-normal, strike-normal, dip-normal, tensile-strike, and tensile-dip source-receiver GFs and 1 for strike-dip and dip-strike. The lower OOA in the final two cases results from the self-interaction calculation for a term of the form $1/R$. Because the operators $I_{n\to u}$ and $A_{u\to n}$ have OOA at least 2 and the range of $I_{n\to u}$ is C^1, EIGA inherits the OOA of DDMu. If δ_r is chosen correctly, AIGA also does.

A suite of empirical convergence tests (ECT) is used as one test of `dc3dm`. The suite includes every corner combination of BCs and every source-receiver component. Fig. 5 shows results for a subset of the ECT for the most interesting combination of BCs and strike-(strike, normal, dip) source-receiver GFs. Relative error is with respect to the solution of DDMu on a very fine mesh.

A base mesh is created for DDMu and AIGA at refinement level 0. At level i, each base element is divided into 4^i elements. (In practice, an IGA mesh is not uniformly refined as in this test; rather, it is always made from scratch according to the resolution function. However, this practical method introduces additional and unnecessary complexity into the convergence analysis.)

Coarse solutions are mapped to the fine mesh using IGA's interpolant. Whether AIGA is more accurate than DDMu or the opposite is arbitrary, as the mesh and test slip function are chosen independently; only the order of accuracy matters. The mesh is chosen to be interesting, and the test slip function is chosen to permit converged results without refining the mesh too many levels and to respect the BCs.

δ_r^i, δ_r at level i, is chosen as $\delta_r^i = (2^i(2\delta_r^0 + 1) - 1)/2$. This choice implements the following rule. If $e \in E$, where E at level j is the set of elements that tile e at level $i < j$, then the area of the neighborhoods around e and $f \in E$ must be the same (in the limit of refinement; at finite refinement, areas are almost certainly slightly different).

Time-dependent simulations. In quasistatic (or quasidynamic) rate-state friction simulations, the resolution function f_r should be a function of fault properties. One such function is $f_r \equiv \alpha\mu' d_c/(b\sigma)$, where α is a constant $\lesssim 1/5$. The motivation for this particular f_r is that rupture tip length scales as $\mu' d_c/(b\sigma)$ for the aging evolution law and that quantity times one that depends on slip speed and background values for the slip law. Rupture tips must be well resolved in simulations.

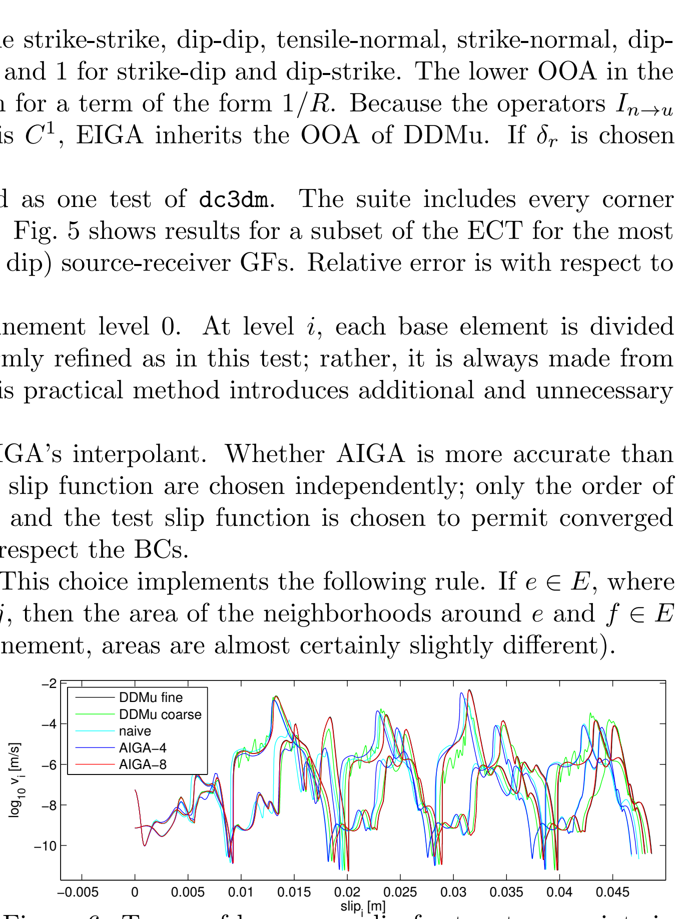


Figure 6: Traces of $\log_{10} v$ *vs.* slip for two trace points in the time-dependent simulation.

We ran two time-dependent simulations. Our AGU 2013 poster [Bradley, 2013] has comprehensive results. Here we focus on just one plot. Three meshes are used: (i) a uniform mesh with element size e_s, (ii) a nonuniform mesh with smallest element size e_s and N elements, and (iii) a uniform mesh with $\left\lceil\sqrt{N}\right\rceil^2$ elements. Multiple simulations are run: DDMu on (i) ('DDMu fine'), DDMu on (iii) ('DDMu coarse'), DDMu on (ii) (DDMu(n), 'naive'), and AIGA with $\delta_r = 4, 8$.

All simulations are run on a shared-memory computer having these specs: 16 cores, 2.6 GHz AMD Opteron 6212, 32 GB memory. Measurements are as follows: DDMu fine compression: 46 min, 2.8 GB (256 GB uncompressed); AIGA-4 compression: 12 min, 169 MB; AIGA-8 compression: 46 min (by chance), 213 MB; AIGA-8 simulation: 61 min, for a simulation speedup of 14.1× over DDMu resulting from a matrix size decrease factor of 12.7, a time step increase factor of 1.2 (when using an adaptive time step, time steps can be longer when the time-step-limiting activity is in a large-element region), and a slight loss in FLOPS because of more logic and memory movement relative to floating point operations. (DDMu using `hmmvp` is already $\gtrsim 90\times$ faster than it would be without `hmmvp`, so the overall speedup is approximately $14 \cdot 90 = 1260$.)

Fig. 6 shows $\log_{10} v$ *vs.* slip for two trace points on the fault over five cycles. The traces for DDMu on the fine mesh (black) are considered the correct solution. DDMu on the coarse mesh produces underresolved slip speed, resulting in a spiky trace. DDMu(n) produces smooth traces (cyan), but they are quite visibly different than the black ones. AIGA-8 (red) produces nearly identical traces to the black ones; red overlaps black almost everywhere. AIGA-4 has an inadequate value of δ_r.

References

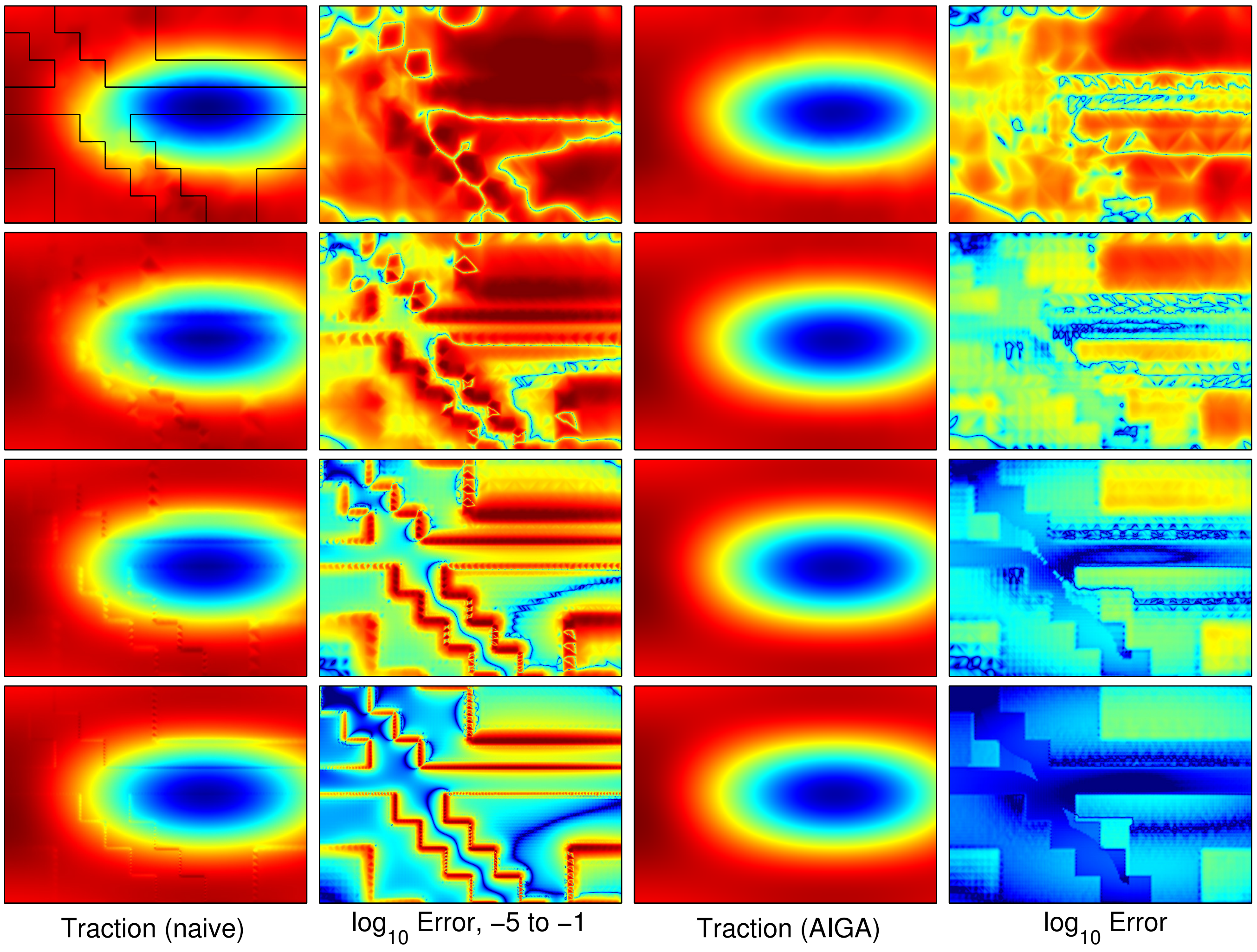
A. M. Bradley. dc3dm: Software for efficient quasistatic dislocation-traction operators on nonuniformly discretized rectangular faults. Abstract T51D-2490 presented at 2012 Fall Meeting, AGU, San Francisco, Calif., 3-7 Dec., 2013.

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

K. M. Johnson, D. Shelly, and A. M. Bradley. Numerical simulations of tremor-related creep reveal weak lower-crustal root of the San Andreas fault. *Geophys. Res. Lett.*, 2013.

Y. Okada. Internal deformation due to shear and tensile faults in a half-space. *Bull. Seism. Soc. Am.*, 82, 1992.

3 Exemplary Figure



Tractions and pointwise absolute errors (relative to maximum traction magnitude) for the naive (DDMu(n), left two columns) and AIGA (right) DDM methods applied to the strike-on-strike Green's function for an example problem. From top to bottom, the level of refinement increases successively by 2 in each dimension. DDMu(n) causes errors where adjacent elements differ in size (outlined by black lines in the top-left image). Peak magnitude of the error stays approximately constant with refinement, causing a drop in order of accuracy from 2 to 1/2. Color scales are the same in respectively columns 1 and 3, and 2 and 4. Images are zoomed to a region of interest.

4 Intellectual Merit and Broader Impacts

MERIT. hmmvp has a rigorous and demonstratively effective error control framework with a clear interpretation of error, thus allowing for approximations built using hmmvp also to have rigorous error control. IGA is a DDM that, for one useful class of nonuniform discretizations, has the same order of accuracy as the standard method on a uniform mesh. IGA and its implementation in dc3dm are supported by extensive order of accuracy analysis. The assessment methodology reveals potential errors researchers may make when using nonuniform constant-slip elements. It also provides a framework for our future work in developing DDMs for more complicated geometries.

IMPACTS. hmmvp enables researchers to run rate-state friction simulations on faults requiring about 100$\times$ more elements than the straightforward method permits, allowing more realistic rheology. It permits arbitrary geometry and (nonoscillatory) Green's function. hmmvp has been fully integrated into the simulators CFRAC (Mark Mclure, U. of Texas, Austin) and Unicycle (Sylvain Barbot, Nanyang Technological U.) and is being evaluated for use in another. It was used in Johnson et al. [2013]. dc3dm increases efficiency above that provided by hmmvp for a limited geometry, permitting even more efficient theoretical studies to be done. It also demonstrates that analyzing the convergence behavior of a DDM can lead to a more efficient method. Both hmmvp and dc3dm are free and open source software available at pangea.stanford.edu/research/CDFM/software.