pTatin3D: High-Performance Methods for Long-Term Lithospheric Dynamics

Dave A. May
Department of Earth Sciences
ETH Zürich, Zürich, Switzerland
dave.may@erdw.ethz.ch

Jed Brown
Mathematics and Computer Science Div.
Argonne National Laboratory
Department of Computer Science
University of Colorado Boulder
jedbrown@mcs.anl.gov

Laetitia Le Pourhiet
Univ. Nice Sophia Antipolis, CNRS, IRD,
Observatoire de la Côte d'Azur
Géoazur UMR 7329
250 rue Albert Einstein
Sophia Antipolis 06560 Valbonne, France
laetitia.le_pourhiet@upmc.fr

***Abstract*—Simulations of long-term lithospheric deformation involve post-failure analysis of high-contrast brittle materials driven by buoyancy and processes at the free surface. Geodynamic phenomena such as subduction and continental rifting take place over millions year time scales, thus require efficient solution methods. We present pTatin3D, a geodynamics modeling package utilising the material-point-method for tracking material composition, combined with a multigrid finite-element method to solve heterogeneous, incompressible visco-plastic Stokes problems. Here we analyze the performance and algorithmic tradeoffs of pTatin3D's multigrid preconditioner. Our matrix-free geometric multigrid preconditioner trades flops for memory bandwidth to produce a time-to-solution $> 2\times$ faster than the best available methods utilising stored matrices (plagued by memory bandwidth limitations), exploits local element structure to achieve weak scaling at 30% of FPU peak on Cray XC-30, has improved dynamic range due to smaller memory footprint, and has more consistent timing and better intra-node scalability due to reduced memory-bus and cache pressure.**

***Index Terms*—geodynamics, Stokes, variable viscosity, multilevel preconditioners, vectorization, matrix-free**

I. Introduction

The structures we observe in the present day Earth result from large-deformation processes that span million-year time scales. While rocks behave elastically on short time scales (e.g., seconds to thousands of years), over periods of 10^5–10^9 years, the dominant mode of deformation exhibited by rocks is ductile. Consequently, the evolution of rocks in geodynamic contexts is frequently described by the equations of stationary Stokes flow, which model the dynamics of an incompressible, highly viscous, creeping fluid. Simulation of geodynamic processes such as subduction and continental rifting place stringent requirements on numerical methods.

- It must be possible to track several rheologically distinct materials with nonlinear history-dependent rheology.
- Deformation tracking must remain accurate long after the material has experienced plastic failure.
- Models must be efficient for three-dimensional simulations at sufficiently high resolution to resolve brittle shear zones in the lithosphere (40–200 km thick) and crustal (5–50 km thick) layers.
- The methods must accommodate a deformable free surface.

Because of the presence of compositionally-distinct materials, brittle failure, and an inherent temperature dependence, the effective viscosity of rocks may be discontinuous, with jumps on the order of $10^9\,\mathrm{Pa\,s}$. The solution of such nonlinear heterogeneous Stokes problems typically accounts for $\sim$90% of the overall simulation time; thus solver robustness and efficiency are paramount. Indeed, the low performance of Stokes solvers represents a computational bottleneck which frequently impedes geodynamic and Earth science research efforts.

The marker-and-cell (MAC) method [1] and material-point method (MPM) [2] have been widely used in the geodynamics community because of their ability to accurately track post-failure deformation (e.g., [3]–[18]). The community is split between staggered-grid finite-difference methods [19] and mixed finite element methods for discretizing the Stokes problem. Finite element methods provide valuable geometric flexibility for tracking the deforming free surface using a boundary-fitted mesh (feasible since topographic variation is typically less than 10 km). Early use of finite element methods frequently involved low-order methods, such as the inf-sup (LBB condition) violating Q_1-P_0 element [6], [10], [14], [17], or the stabilized Q_1-Q_1 element [20], [21]. Both element types may exhibit pressure artifacts due to the unstable "checker-board mode" (Q_1-P_0, [22]) or due to the artificial compressibility introduced by polynomial projection stabilization [23]. As previously used by [15], [18], we use the inf-sup stable and locally conservative Q_2-P_1^{disc} element. Though robust, this element has more than twice as many nonzeros per row (cf. Q_1-Q_1 stab.), leading to significant cost for assembled matrix representations. Nonetheless, it is among the least expensive locally conservative and inf-sup stable elements which can accurately represent the hydrostatic mode present in models with a free surface.

Multigrid methods [24], [25] are the most successful and widely applicable scalable approach to solving elliptic problems. Within the geodynamics community, applying multigrid methods directly to the coupled Stokes problem, typically using Vanka smoothers [26], or splitting the system using approximate Schur complement techniques [27] have been explored, although there is no clear consensus as to which

SC14, November 16-21, 2014, New Orleans
978-1-4799-5500-8/14/$31.00

is universally superior. The former is more appropriate for staggered-grid finite-difference methods (see [12], [28]) while the latter has been more popular for use with finite element methods [29], [30].

This paper is organized as follows. § II presents the model and discretization, § III discusses design issues and performance of solver algorithms, § IV presents numerical results and performance analysis, § V presents a continental rifting simulation, and § VI draws conclusions and identifies remaining open problems.

II. Model and Discretization

A. Continuum equations

We consider the long-term dynamics of rocks to be described by the conservation of momentum of a creeping fluid with an isotropic viscosity in a domain Ω with boundary $\partial\Omega$:

$$\nabla \cdot \Big[2\eta(\mathbf{u},p)\mathbf{D}(\mathbf{u})\Big] - \nabla p = \mathbf{f}, \tag{1}$$

where $\mathbf{u}, p$ are the fluid velocity and pressure, respectively, η is the (nonlinear) effective shear viscosity; $\mathbf{f}$ is the body force; and the strain-rate operator $\mathbf{D}(\mathbf{u})$ is given by

$$\mathbf{D}(\mathbf{u}) := \tfrac{1}{2}\left(\nabla\mathbf{u}^T + \nabla\mathbf{u}\right). \tag{2}$$

The forcing term in Equation (1) is taken to be of the form $\mathbf{f} = \rho\mathbf{g}$, where ρ is the fluid density and $\mathbf{g}$ the gravity vector. We assume incompressibility (simplified mass conservation):

$$-\nabla \cdot \mathbf{u} = 0. \tag{3}$$

Equations (1) and (3) are closed with the following boundary conditions,

$$\mathbf{u} = \bar{\mathbf{u}} \qquad \mathbf{x} \in \Gamma_D \tag{4}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n} = \bar{\mathbf{t}} \qquad \mathbf{x} \in \Gamma_N, \tag{5}$$

where $\boldsymbol{\sigma}$ is the total stress, $\bar{\mathbf{t}}$ the traction vector, $\mathbf{n}$ the outward pointing normal to the boundary $\partial\Omega$. Γ_D and Γ_N denote the regions along the boundary $\partial\Omega$ where the Dirichlet and Neumann boundary conditions are applied, respectively, subject to the restrictions that $\Gamma_D \cap \Gamma_N = \emptyset$ and $\Gamma_D \cup \Gamma_N = \partial\Omega$.

In geodynamic models, typically multiple rock types (e.g., upper crust, lower crust, mantle) are represented. Associated with each rock type (which we denote by Φ) is a particular rheology that defines the flow law used to compute the effective nonlinear viscosity (η) and density (ρ). The effective viscosity accounts for both the parametrized creeping of viscous rocks and brittle behavior, while ρ accounts for compositional and thermal buoyancy variations. In addition to the conservation of mass and momentum, we evolve Φ according to

$$\frac{D\Phi}{Dt} = 0. \tag{6}$$

We note that while Equations (1) and (3) are independent of time, temporal dependence in the flow field is introduced through the evolution of rock type and temperature (when applicable, see § V).

B. Finite element discretization

We discretize Equations (1) and (3) using a standard mixed Q_2-P_1^{disc} finite element discretization [22]. The weak form of the linearized Stokes problem is as follows: Given a certain $(\mathbf{u}^*, p^*) \in V \times Q$, find $(\mathbf{u}, p) \in V \times Q$ such that

$$\mathcal{A}(\mathbf{u},\mathbf{w};\mathbf{u}^*,p^*) + \mathcal{B}(\mathbf{u},q) + \mathcal{B}(\mathbf{w},p) = \mathcal{F}(\mathbf{w}) \tag{7}$$

for all $(\mathbf{w}, q) \in V_0 \times Q$, The bilinear forms $\mathcal{A}(\cdot,\cdot;\mathbf{u}^*,p^*)$ and $\mathcal{B}(\cdot,\cdot)$ and the linear functional $\mathcal{F}(\cdot)$ in Equation (7) are defined as

$$\mathcal{A}(\mathbf{u},\mathbf{w};\mathbf{u}^*,p^*) = \int_\Omega 2\eta(\mathbf{u}^*,p^*)\mathbf{D}(\mathbf{u}) : \mathbf{D}(\mathbf{w})\, dV, \tag{8}$$

$$\mathcal{B}(\mathbf{w},q) = -\int_\Omega q\nabla\cdot\mathbf{w}\, dV, \tag{9}$$

$$\mathcal{F}(\mathbf{w}) = -\int_\Omega \mathbf{w}\cdot\mathbf{f}\, dV + \int_{\Gamma_N}\mathbf{w}\cdot\bar{\mathbf{t}}\, dS. \tag{10}$$

The spaces used are $V = H^1(\Omega)$ and the corresponding homogeneous Dirichlet space $V_0 = H^1_D(\Omega)$ for which $\mathbf{w} \in V_0$ vanishes along Γ_D (Dirichlet boundary), along with the pressure space $Q = \{q \in L_2(\Omega) : \int_\Omega q\, dV = 0\}$ if $\Gamma_N = \emptyset$, otherwise $Q = L_2(\Omega)$.

We partition Ω using a mesh of structured but deformed hexahedral elements. To preserve the order of accuracy of the Q_2-P_1^{disc} discretization, we define the pressure basis in the x, y, z coordinate system, as opposed to in the "mapped" coordinate system [31], [32]. The locally conservative (elementwise) properties of this element are highly desirable since many of the dynamically processes we wish to model utilize a free surface boundary condition ($\boldsymbol{\sigma} = \mathbf{0}$) and are driven entirely by buoyancy variations. Under these conditions, globally conservative elements (e.g. Q_2-Q_1), or low-order stabilized methods (Bochev stabilized Q_1-Q_1), tend to produce unacceptable numerical artifacts in the velocity and pressure field unless an extremely high mesh resolution is used [33].

C. Material-point method

The rock lithology Φ is discretized by using a set of Lagrangian material points. The flow law and forcing term associated with a given lithology is evaluated at the position of each material point $\mathbf{x}_p$, $p = 1, 2, \ldots, n_p$. The resulting effective viscosity η_p and density ρ_p is projected onto the quadrature points used to evaluate the relevant terms in the nonlinear residual in the following manner.

We assume that an arbitrary material point property f, is discretized via

$$f(\mathbf{x}) \approx \delta(\mathbf{x} - \mathbf{x}_p) f_p. \tag{11}$$

We then utilize an approximate local L_2 projection of f_p onto a continuous Q_1 finite element space. The corner vertices of each Q_2 finite element define the mesh f_p is projected onto. The local reconstruction for a node i is defined by

$$\hat{f}_i = \frac{\int_{\Omega_i} N_i(\mathbf{x}) f(\mathbf{x})}{\int_{\Omega_i} N_i(\mathbf{x})} \approx \frac{\sum_p N_i(\mathbf{x}_p) f_p}{\sum_p N_i(\mathbf{x}_p)}, \tag{12}$$

where the summation over p includes all material points contained within the support Ω_i of the trilinear interpolant N_i. Within each element, the projected material property, denoted by $\tilde{f}(\mathbf{x})$ is defined by interpolation:

$$\tilde{f}(\mathbf{x}) = \sum_i N_i(\mathbf{x}) \hat{f}_i. \tag{13}$$

D. Parallelism

Support for all parallel linear algebra, in the form of matrices, vectors, preconditioners, Krylov methods, and nonlinear solvers, is provided by PETSc [34]–[36].

Parallelism is achieved by spatially decomposing the structured Q_2 finite element mesh containing $M \times N \times P$ elements into structured subdomains containing $m \times n \times p$ Q_2 elements. The underlying mesh data structure utilizes the DMDA object defined within the PETSc library.

Parallelism associated with material points is defined by the spatial decomposition of the FE mesh. Material points located within a particular mesh subdomain associated with a processor k are managed by processor k. Following the advection of all material points, we apply a point location routine that simultaneously returns the local element index containing the material point and its local coordinate ξ_p. If the point location routine determines that the material point is not located on the current subdomain, the material point is inserted into a list $\mathcal{L}_s$. All material points in $\mathcal{L}_s$ are sent to all neighboring mesh subdomains, and the point location algorithm is reapplied to the newly received material points $\mathcal{L}_r$. Material points in $\mathcal{L}_r$ which are not contained within the current mesh subdomain are deleted. This simple strategy enables the communication of material points between processors and permits material points to leave the domain if any outflow type boundary conditions are prescribed.

III. Solver Design

An ideal solver would converge rapidly and reliably independent of resolution, coefficient contrast and structure, and nonlinearity, while exposing fine-grained parallelism and using little memory so that a broad range of problem sizes can be solved on a given machine. In practice, solver *efficiency* benefits from some degree of tuning to the problem, machine, and desired turnaround time. Retuning solvers for every scenario is not feasible, thus it is important that the solver design be simplified enough for the end user to make educated choices with predictable behavior. Since parallel solvers for nonlinear heterogeneous Stokes problems are relatively complicated, we present a unified approach reflecting the composition of components in PETSc [37], [38], and we demonstrate the most important trade-offs.

A. Nonlinearities

Since nonlinearities are evaluated at material points (§ II-C) and projected into a continuous space, we seek to amortize the cost of projection over a linear solve. The rheological models we consider typically define an effective viscosity $\eta(\cdot)$ which is a function of the strain-rate $\mathbf{D}(\mathbf{u})$, leading to the following Picard linearization of the weak form of Equation (7): Find $\mathbf{w}, p$ such that

$$\int_\Omega \mathbf{D}(\mathbf{v}) : \eta(\mathbf{D}(\mathbf{u}))\mathbb{I} : \mathbf{D}(\mathbf{w})\, dV + \mathcal{B}(\mathbf{v}, p) + \mathcal{B}(\mathbf{w}, q) - \mathcal{F}(\mathbf{v}) = 0, \quad \forall\, \mathbf{v}, q.$$

A Picard iteration involves successive solves with $\eta(\mathbf{D}(\mathbf{u}))$ taken from the previous iteration. For isotropic materials, this involves solves with a spatially varying scalar (isotropic) effective viscosity. Picard linearization is observed to stagnate in many plasticity models, so we turn to a Newton method which provides much faster convergence in the terminal phase. Newton linearization leads to a tensor-valued coefficient $\eta(\mathbf{D}(\mathbf{u}))\mathbb{I} + \eta' \otimes \mathbf{D}(\mathbf{u})$ that, when $\eta(\mathbf{D}(\mathbf{u})) = \hat{\eta}(\frac{1}{2}|\mathbf{D}(\mathbf{u})|^2)$ depends only on the second invariant of the strain-rate tensor, becomes $\eta(\mathbf{D}(\mathbf{u}))\mathbb{I} + \hat{\eta}'\mathbf{D}(\mathbf{u}) \otimes \mathbf{D}(\mathbf{u})$ (where $\hat{\eta}'$ is now a scalar). For yielding and shear-thinning materials, $\hat{\eta}' < 0$, so the viscosity tensor is flattening in the direction $\mathbf{D}(\mathbf{u})$. This anisotropy leads to difficulties with smoothers in multigrid; therefore, since it is transient and usually localized, we use the true Newton linearization only when applying the Krylov operator in the (approximate) solves at each Newton step. For the preconditioner, which is the primary cost, we use the Picard linearization.

Newton iterations are guarded by a backtracking line search, and tolerances for the linear solve are adaptively set by using the Eisenstat-Walker [39] method. As we will demonstrate, discrepancy between the sizes of residuals in velocity and pressure components can lead to slowed convergence, so it is useful for both diagnostic and reliability reasons to make the full residual available. GMRES defines the residual only via a recurrence; thus it is expensive to compute on each iteration. In such cases, we prefer to use GCR [40], which is a flexible method (can tolerate a nonlinear preconditioner) and provides the current iterate explicitly. Several of our solver configurations involve inner iterations that make the preconditioner nonlinear, in which case we must use either GCR or flexible GMRES [41]. For extremely ill-conditioned problems, (F)GMRES is preferred for its better numerical stability.

B. Field-split methods

Each Newton iteration on the Stokes problem requires solving a system of the form

$$\underbrace{\begin{bmatrix} \mathbf{J}_{uu} & \mathbf{J}_{up} \\ \mathbf{J}_{pu} & \mathbf{0} \end{bmatrix}}_{\mathbf{J}} \underbrace{\begin{bmatrix} \delta\mathbf{u} \\ \delta\mathbf{p} \end{bmatrix}}_{\delta\mathbf{X}} = - \underbrace{\begin{bmatrix} \mathbf{F}_u \\ \mathbf{F}_p \end{bmatrix}}_{\mathbf{F}}. \tag{14}$$

The viscous subproblem $\mathbf{J}_{uu}$ is SPD and analogous to an elasticity problem, with coefficient variation caused by material composition and dependence on strain-rate and temperature. Although a Picard linearization is typically used for the preconditioner, we do not distinguish in the following discussion. The convergence of splitting methods for Equation (14) will depend crucially on approximations to the Schur complement

$$\mathbf{S} = -\mathbf{J}_{pu}\mathbf{J}_{uu}^{-1}\mathbf{J}_{up}. \tag{15}$$

Several ways exist for approximating $\mathbf{S}$ [27], [29]. We will use a pressure mass matrix scaled by the inverse of effective viscosity $\eta(\mathbf{D}(\mathbf{u}))$. This preconditioner is simple and effective when used with discontinuous pressure spaces and is spectrally equivalent under some smoothness assumptions [42]. With such a simple preconditioner for $\mathbf{S}$, the (approximate) action of $\mathbf{J}_{uu}^{-1}$ using multigrid methods will be the leading cost in our preconditioner. Before going into detail, we consider the two approaches for constructing a preconditioner for the coupled problem in Equation (14).

Equation (14) admits the block factorization

$$\begin{bmatrix} \mathbf{J}_{uu} & \mathbf{J}_{up} \\ \mathbf{J}_{pu} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{uu} & 0 \\ \mathbf{J}_{pu} & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{1} & \mathbf{J}_{uu}^{-1}\mathbf{J}_{up} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{16}$$

(the non-unit diagonal can equivalently be grouped with the upper factor). Application of this block factorization using accurate iterative solves to apply $\mathbf{J}_{uu}^{-1}$ and $\mathbf{S}^{-1}$ is Schur complement reduction (SCR). The Uzawa method [43] is a well-known stationary iteration in the SCR family. These methods tend to be reliable, but each application of the Schur complement $\mathbf{S}$ involves an accurate solve with $\mathbf{J}_{uu}$, so they tend to be expensive. The other approach is to iterate on the full space problem (e.g. solve Equation (14) for $\delta\mathbf{u}, \delta\mathbf{p}$ simultaneously) and adopt a preconditioner defined by an approximate form of the factorization in Equation (16). The lesser triangular factor is typically dropped in this approach, leading to the block-triangular preconditioner

$$\mathbf{P} = \begin{bmatrix} \tilde{\mathbf{J}}_{uu} & 0 \\ \mathbf{J}_{pu} & \tilde{\mathbf{S}} \end{bmatrix}, \tag{17}$$

where the tilde indicates a spectrally equivalent approximation (multigrid for $\tilde{\mathbf{J}}_{uu}$ and the viscosity-scaled mass matrix for $\tilde{\mathbf{S}}$). Dropping the triangular factor is justified because, with exact $\tilde{\mathbf{J}}_{uu} = \mathbf{J}_{uu}$ and $\tilde{\mathbf{S}} = \mathbf{S}$, the left- or right-preconditioned operators $\mathbf{P}^{-1}\mathbf{J}$ and $\mathbf{J}\mathbf{P}^{-1}$, satisfy the minimal polynomial $(\lambda - 1)^2 = 0$ so that a suitable Krylov method will converge in at most two iterations [44], [45]. Unfortunately, the preconditioned matrix is non-normal and often leads to increased dependence on coefficient structure, a phenomenon discussed further in § IV-A.

C. Multigrid methods

We apply the action of $\tilde{\mathbf{J}}_{uu}$ using a multigrid V-cycle containing both geometric and algebraic parts. This is the most performance-critical part of the simulation, so it is important to balance implementation efficiency with robust convergence. We are interested in mixing matrix-free and assembled matrices and would like to expose fine-grained parallelism. Multiplicative smoothers are difficult to implement efficiently in parallel [46], have poor memory locality properties, and are especially ill-suited for use with finite element methods. To see this last effect, consider that computing a pointwise residual at a vertex involves visiting all quadrature points of all elements adjacent to that vertex. When computing a global residual, each quadrature point is visited only once; but when applying a multiplicative smoother, each quadrature point is visited once for each basis function with support on that quadrature point. Naively, this leads to an overhead of $(k+1)^d$ for Q_k finite elements in d-dimensions, much larger than can be recovered with intricate optimizations. Fortunately, extensive testing reinforces the results of [47] showing that polynomial smoothers for problems such as ours and elasticity attain efficiency similar to that of a multiplicative smoother, even for assembled matrices. For production use and simplicity of presentation in this paper, unless stated otherwise, we fix the smoother as Jacobi-preconditioned Chebyshev iterations targeting the interval $[0.2\lambda_{\max}, 1.1\lambda_{\max}]$, where $\lambda_{\max}$ is an estimate of the largest eigenvalue of the Jacobi-preconditioned operator, computed by a few iterations of a Krylov method.

Structured meshes with an IJK topology are employed in this work, however nodal coordinates are not required to be parallel to the x, y, z coordinate system. We utilize nodally nested mesh hierarchies, thereby allowing the geometry (node coordinates) of the coarse mesh to be trivially defined via injection. The prolongation $\mathbf{P}_k^{k+1}$ of the velocity field from level k (coarse), to $k+1$ (fine), uses trilinear interpolation (i.e., associated with an embedded Q_1 finite element space on the nodes of the Q_2 discretization). Restriction is then defined by $\mathbf{R}_{k+1}^k = (\mathbf{P}_k^{k+1})^T$. Coarse level operators are defined by either rediscretization of $\mathcal{A}$ on the coarse level mesh, or via the Galerkin approximation $\mathbf{A}_{\text{coarse}} = (\mathbf{P}_k^{k+1})^T \mathbf{A}_{\text{fine}} \mathbf{P}_k^{k+1}$.

Galerkin coarsening is more robust but is expensive to compute and requires assembled matrices, thereby requiring additional storage and thus limiting the range of problem sizes which are tractable on a given machine. Consequently, we prefer to perform at least one level of geometric coarsening with the finest level applied matrix-free, followed by an assembled level used either for further geometric coarsening with Galerkin, or to enable switching to algebraic multigrid. Since our implementation does not support distributed geometric coarsening (to reduced process sets), we use GAMG, a smoothed aggregation method available in PETSc to perform further distributed coarsening. We provide the six rigid-body modes and set a strength threshold of 0.01. The same smoother configuration is used in the geometric and algebraic parts of the multigrid cycle.

D. Matrix-free operators

Iterative solves using assembled sparse matrices are overwhelmingly limited by memory bandwidth, an increasingly precious commodity on modern hardware [48]. Each row of a matrix assembled by using a Q_2 discretization of the viscous problem has between 81 and 375 nonzeros, with an average of 192 (corners, edges, faces, and interior). This is a significant memory overhead and must be streamed through cache every time the operator is applied. Each nonzero matrix entry requires loading the scalar value and column index, only to perform one multiply and one add. If we assume that the vector reuses cache perfectly and count only the bandwidth cost for matrix entries and column indices, we observe 85% of STREAM Triad peak in sparse matrix multiplication on Edison, a Cray XC-30 at NERSC. The arithmetic intensity of

this operation is less than 1 flop for every 4 bytes (depending on column index optimization; $1/5.33$ as implemented for the test above). Current hardware delivers between 4 and 9 flops per byte (5.17 on Edison—460.8 GF/s at 89 GB/s Triad; 7.9 on Blue Gene/Q—204.8 GF/s at 26 GB/s Triad; similar for GPUs and Xeon Phi). Good sparse matrix implementations typically realize a high fraction of STREAM bandwidth; thus, little improvement can come from further sparse matrix optimizations. Moreover, memory bandwidth is a shared resource, so bandwidth-limited applications see poor intranode scalability and increased performance variability compared to compute-limited applications.

Finite element methods can be implemented without assembled matrices. Consider the application of the discretized scalar operator $A\mathbf{u}$ representing $-\nabla(\kappa\nabla\cdot u)$. The global residual is computed as

$$A\mathbf{u} = \sum_{e \in N_{\text{el}}} \mathcal{E}_e^T \mathcal{D}_{\mathbf{x}}^T \Lambda(\omega\kappa)\mathcal{D}_{\mathbf{x}}\mathcal{E}_e\mathbf{u}, \tag{18}$$

where $N_{\text{el}} = M \times N \times P$ is the number of elements in the domain, $\mathcal{E}_e$ gathers the values within the support of element e, $\mathcal{D}_{\mathbf{x}} = \{\mathcal{D}_i | i \in \{x, y, z\}\}$ is the physical gradient matrix on the element, and $\Lambda(\omega\kappa)$ is the "diagonal" operation of multiplying by the quadrature weight and coefficient at each quadrature point. To compute the physical gradient matrices on isoparametrically mapped elements, one computes the coordinate gradient $\nabla_\xi \mathbf{x} = (\mathcal{D}_\xi \otimes I_3)(\mathcal{E}_e \otimes I_3)\mathbf{x}$, where $\mathbf{x}$ is the vector of coordinates at finite element nodes and $\mathcal{D}_\xi$ is the derivative with respect to reference coordinates. For notational simplicity, we now elide tensorization with the 3×3 identity I_3 when operating on vector-valued quantities. The quantity $\nabla_\xi \mathbf{x}$ can be interpreted as 3×3 matrices at each quadrature point. Inverting these and then taking determinants produces the gradients $\nabla_{\mathbf{x}}\xi$ and quadrature weighting for physical elements. For Q_2 elements, the precomputed reference gradient matrix $\mathcal{D}_\xi$ is 81×27 and is mapped to physical space via the block-diagonal operation $\mathcal{D}_{\mathbf{x}} = \Lambda(\nabla_{\mathbf{x}}\xi)\mathcal{D}_\xi$.

We now count data motion and floating-point operations to compute the element action applied to a three-component velocity. Visiting an element requires $8 \cdot 3$ scalars for coordinates, $2 \cdot 8 \cdot 3$ scalars for state and residual, plus 27 scalars for the scalar-valued coefficient η. We use an explicit integer representation for $\mathcal{E}_e$, which requires an additional 27 values, for a total of 1008 bytes that will never be reused. With limited cache or a poor ordering of elements, this increases to 2376 bytes. Computing the metric term $\nabla_{\mathbf{x}}\xi$ requires $2 \cdot 81 \cdot 27 \cdot 3 + 42 \cdot 27 = 14256$ flops, constructing $\mathcal{D}_{\mathbf{x}}$ is $2{\cdot}81{\cdot}27{\cdot}3 = 13122$, and application of $\mathcal{D}_{\mathbf{x}}$ and $\mathcal{D}_{\mathbf{x}}^T$ each costs 13122 flops, for a total of 53622 flops. The arithmetic intensity is thus between 22.5 (pessimal cache) and 53 (perfect cache) flops/byte. This is well above that provided by hardware, so we expect this operation to be compute-limited. By comparison, an assembled matrix has 4608 nonzero entries per element, leading to a data requirement of 37248 bytes with perfect cache reuse for the vectors and implicit column indices. The results are summarized in Table I. Evidently, any machine that can perform 53622 flops (in the matrix-free element kernel) in less time than it can stream 37248 bytes will exceed the theoretical peak attainable using assembled sparse matrices.

TABLE I
MEMORY AND COMPUTE DEMANDS PER ELEMENT FOR DIFFERENT OPERATOR APPLICATION METHODS. ARITHMETIC INTENSITY IS REPORTED AS FLOPS/BYTE, COUNTING ADDS, MULTIPLIES, AND DIVISION (RARE) ALL AS 1 FLOP. "TENSOR" IS A MATRIX-FREE IMPLEMENTATION THAT USES THE TENSOR PRODUCT STRUCTURE PRESENT ON THE REFERENCE ELEMENT. "TENSOR C" ABSORBS THE METRIC TERMS INTO A TENSOR-VALUED COEFFICIENT. AVERAGE TIME (MILLISECONDS) AND GF/S ARE REPORTED FOR (PARALLEL) OPERATOR APPLICATION ON 8 NODES OF EDISON (3686 GF/S PEAK).

Operator	Flops	Pessimal Cache		Perfect Cache		Time	GF/s
		Bytes	F/B	Bytes	F/B	(ms)	
Assembled	9216	—	—	37248	0.247	42	113
Matrix-free	53622	2376	22.5	1008	53	22	651
Tensor	15228	2376	6.4	1008	15	4.2	1072
Tensor C	14214	5832	2.4	4920	2.9	—	—

Q_2 elements have tensor-product structure that can be exploited to reduce the number of operations by approximately $3\times$ relative to the dense representation. The reference derivative matrix $\mathcal{D}_\xi$ can be split into three pieces, $\hat{D} \otimes \hat{B} \otimes \hat{B}$, $\hat{B} \otimes \hat{D} \otimes \hat{B}$, and $\hat{B} \otimes \hat{B} \otimes \hat{D}$, where $\hat{B}$ and $\hat{D}$ are the $3 \otimes 3$ basis evaluation and derivative evaluation matrices in one dimension. With this tensor product structure, $\mathcal{D}_\xi$ can be applied in $2 \cdot 3^7 = 4374$ flops, one third of that required to apply it as an assembled operator. Since we cannot explicitly form $\mathcal{D}_{\mathbf{x}}$, we must restructure our element kernel in Equation (18) to take advantage of this savings:

$$A\mathbf{u} = \sum_{e \in N_{\text{el}}} \mathcal{E}_e^T \mathcal{D}_\xi^T \Lambda\Big((\nabla_{\mathbf{x}}\xi)^T(\omega\eta)(\nabla_{\mathbf{x}}\xi)\Big)\mathcal{D}_\xi\mathcal{E}_e\mathbf{u}. \tag{19}$$

Counting flops, we apply $\mathcal{D}_\xi$ or its transpose a total of three times ($3 \cdot 4374$ flops), compute metric terms at quadrature points ($42 \cdot 27$ flops), and perform $3 \cdot 12 \cdot 27$ flops in the quadrature loop, for a total of 15228 flops. Note that the metric terms are now applied as part of the quadrature loop and that we have removed the need to store an 81×27 matrix (17 kB). By removing the $\mathcal{D}_{\mathbf{x}}$ matrix (which is different on each element), multiple elements can be processed simultaneously without spilling out of L1 cache. These low memory requirements enable simple vectorization over elements, for which we consistently achieve greater than 30% of peak on AVX (Sandy Bridge) and AVX+FMA (Haswell) CPUs. The efficiency of matrix-free tensor-product formulations in linear solves has been demonstrated for Q_3 and higher orders [49] and forms the basis for spectral-element methods [50], [51], but non-tensor techniques are overwhelmingly common for Q_2 and lower-order elements. Spectral element methods typically perform a further optimization of choosing Gauss-Lobatto quadrature, for which $\hat{B}$ is the identity. This reduces the flops in $\mathcal{D}_\xi$ by a factor of 3 but is not sufficiently accurate for our deformed meshes with variable coefficients.

We note that a further rearrangement of Equation (19) is possible and desirable in the case of anisotropic coefficients.

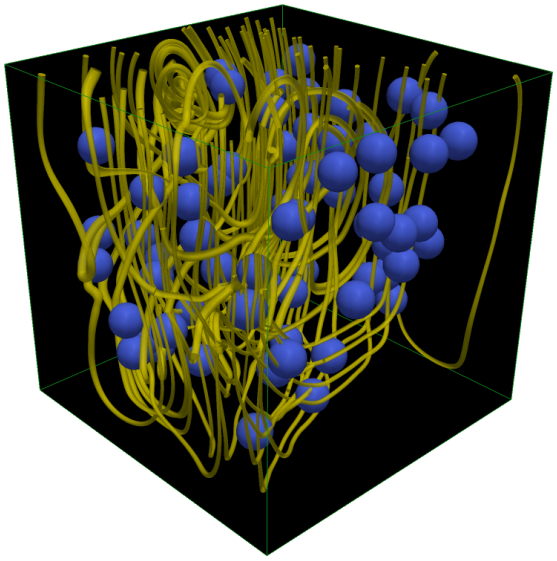

Fig. 1. Streamlines for a sedimentation example with $N_c = 75$ and $R_c = 0.05$. The streamline diameter is scaled according to the magnitude of the fluid velocity. The viscosity contrast between the inclusions (blue spheres) and the background material is $\Delta\eta = 10^4$.

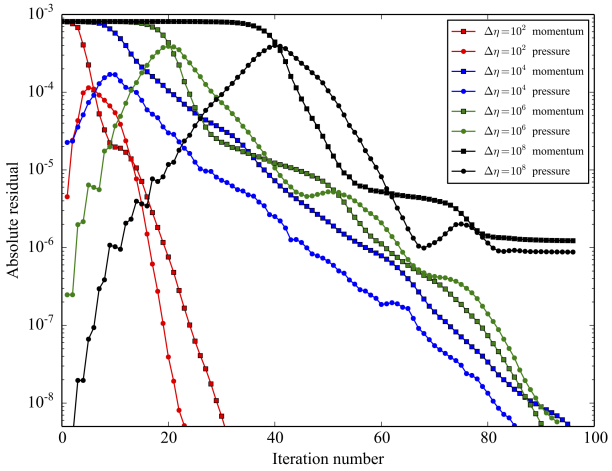

Fig. 2. Convergence of full-space iteration in varying viscosity contrast $\Delta\eta$.

The product $(\nabla_{\mathbf{x}}\xi)^T(\omega\eta)(\nabla_{\mathbf{x}}\xi)$, even if η is tensor-valued, is a rank-4 tensor containing 21 distinct entries (due to symmetry). If this is stored, the data requirements for an element increase to $2\cdot 8\cdot 3 + 21\cdot 27 = 4920$ bytes (perfect cache) or $2\cdot 27\cdot 3 + 21\cdot 27 = 5832$ bytes (pessimal cache), but the flops are reduced to $2\cdot 4920 + 2\cdot 81\cdot 27 = 14214$. This is little benefit for the present problem, and we do not pursue it further; but it is justified if η is anisotropic or for scalar problems (for which coordinate transformations are disproportionately expensive and anisotropy is only a rank-2 tensor).

IV. Numerical Results

A. Robustness

To investigate solver robustness, we consider a sedimentation problem chosen as a more demanding variant of the "sinker" problem in [29]. We populate the cubic domain $[0,1]^3$ with N_c randomly-placed nonintersecting spheres of radius R_c. Flow is driven by density variations between the spheres and background material. In Equation (1) the gravity vector is taken as $\mathbf{g} = (0,0,-9.8)$ (positive z-direction pointing towards the free surface). The ambient fluid has viscosity $(\Delta\eta)^{-1}$ and density 1, while the spheres have viscosity 1 and density 1.2. Slip boundary conditions are imposed at the walls and a free surface at the top ($z = 1$). As shown in Figure 1 via the streamlines, the flow pattern is complicated and nonlocal. Unlike the case $N_c = 1$, the presence of many inclusions prevents Krylov methods from accelerating convergence on the test problem beyond that seen in realistic scenarios.

In the performance tests presented, we chose $N_c = 8$ and $R_c = 0.1$ and ran the solver over three time steps (scientifically relevant sedimentation experiments would be run for many steps). To explore robustness with respect to coefficient contrast, we fix the spatial discretization at 64^3 Q_2 elements and vary the coefficient contrast. We employ the lower-triangular preconditioner (see Equation (17)) iterating in the full space and use a single application of a $V(2,2)$ cycle (one V-cycle employing two pre- and post- applications of the smoother) of our geometric multigrid preconditioner to define the action of $\tilde{\mathbf{J}}_{uu}^{-1}$. The geometric hierarchy contained three levels, with the coarsest operator defined via Galerkin projection. A single $V(2,2)$ cycle of a smoothed aggregation based algebraic multigrid preconditioner (GAMG) is used as the coarse grid solver. Both the geometric and algebraic multigrid utilize a Chebyshev iteration, preconditioned with Jacobi as the smoother on every level. At the coarsest level within the algebraic multigrid preconditioner, the coarse level solver was defined via a block Jacobi preconditioner, with an exact LU factorization applied on each of the subdomains. (see § III-C for further details). All Stokes problems are solved to an unpreconditioned relative tolerance of 10^{-5}. Unless otherwise stated, all numerical results reported have been computed using 64-bit indices.

Convergence for the vertical momentum residual and pressure (incompressibility) residual is shown in Figure 2. As is typical with buoyancy-driven flows, the iteration starts with a large vertical momentum residual and the pressure residual must increase to the same order as the momentum residual before the momentum begins to converge. As the contrast $\Delta\eta$ increases, these components take longer to equilibrate, at which point relatively steady convergence is observed. Replacing the momentum V-cycle with an accurate solve does not significantly change this behavior.

Since the Schur complement preconditioner $\tilde{\mathbf{S}}$ is spectrally equivalent, we attribute the slow convergence to non-normality of the preconditioned operator. Non-normality can be avoided by using Schur complement reduction (SCR), at the expense of accurate inner solves. This is usually significantly more expensive but is more robust to extreme coefficient contrasts.

TABLE II
ALGORITHMIC SCALABILITY FOR DIFFERENT PROBLEM AND PARTITION SIZES ON EDISON. THE '-' SYMBOL INDICATES THE JOB COULD NOT BE EXECUTED DUE TO MEMORY LIMITATIONS.

Grid	Cores	Its.	Coarse solve		Stokes solve (s)		
			Setup (s)	Apply (s)	Asmb	MF	Tens
64^3	192	112	0.5	1.2	40.1	26.9	14.8
96^3	192	110	1.3	4.0	-	90.6	50.6
96^3	1536	95	1.5	1.1	-	12.1	8.2
192^3	1536	141	2.7	5.8	-	120.0	45.1
192^3	12288	170	4.1	6.1	-	30.1	17.1

TABLE III
EFFICIENCY FOR DIFFERENT PROBLEM AND PARTITION SIZES ON EDISON. SEE TEXT FOR DEFINITIONS OF THE EFFICIENCY METRICS USED AND EVENTS PROFILED.

SpMV type	Grid (E)	Cores (C)	MG res TF/s	Stokes solve E/C/s	GF/C/s	GF/s
Asmb	64^3	192	0.1	46	0.9	173
MF	64^3	192	0.7	69	2.6	502
Tens	64^3	192	1.1	128	2.4	464
MF	96^3	192	0.8	58	2.6	499
Tens	96^3	192	1.0	103	2.6	447
MF	96^3	1536	5.0	47	2.2	3198
Tens	96^3	1536	6.6	72	2.2	2378
MF	192^3	1536	5.3	46	2.5	3839
Tens	192^3	1536	7.8	79	2.2	3303
MF	192^3	12288	36.1	19	1.5	18499
Tens	192^3	12288	35.3	26	1.1	12891

B. Scalability and efficiency

To investigate algorithmic scalability, in Table II we examine the number of iterations and CPU time (seconds) required for convergence as the mesh size and core count are varied. These experiments utilize for the same preconditioner and test problem defined in § IV-A. The CPU time reported highlights the time spent within the geometric multigrid coarse grid solver (setup and application), as well as the time required for a complete Stokes solve (time-to-solution) using SpMVs defined via assembled matrices (Asmb), our reference non-tensor matrix-free implementation (MF) and our tensor product based SpMV (Tens). In general, the iteration counts are found to slightly increase as either the number of elements in the mesh, or the number of cores used was increased. This is attributed to using a fixed number of levels (three) within the geometric multigrid for all element resolutions considered. Consequently, the coarse grid problem increases in size as the mesh is refined, thus placing a higher demand on the quality of the algebraically constructed coarse level operators. As we will show in § IV-C, purely algebraic multi-level preconditioners tend to require a higher number of iterations to convergence. The overall speed increase obtained from the Q_2 tensor product implementation is a factor of 2.7 cf. the equivalent preconditioner using assembled operators, and a factor of 1.8 compared with our reference SpMV implementation. Importantly we note that the setup cost for the algebraic preconditioner used as our coarse grid solver is small (less than 5 seconds on 12k cores) and exhibits only an increase in cost of $1.5\times$, over an $8\times$ increase in the number of processors.

Given a FE simulation containing E elements and executed on C cores, we have chosen to measure the efficiency (or computational scalability) in terms of elements per core, per second (E/C/s). This combines both algorithmic scalability (number of iterations independent of resolution and parallelism) and implementation efficiency. Additionally, we also report as performance measures GF/s and GF/s per core (GF/C/s). In Table III, we summarize the performance results around the threshold number of elements/core at which communication becomes significant for two events within our preconditioner. "MG res" represents the residual evaluation performed at the finest level within the MG hierarchy and reflects the performance of the SpMV implementation. "Stokes solve" represents the solution of the entire Stokes problem, which includes application of the Krylov method and multiple applications of the geometric multigrid preconditioner. We observe that the matrix-free implementation (MF) is uniformly faster than that of the assembled matrices (Asmb) and the tensor-product formulation (Tens) is uniformly faster than the (non-tensor) matrix-free implementation. Although the GF/s for operator application is always higher for the tensor-product formulation (not shown, see Table I), the tensor-product formulation does fewer flops so the end-to-end solve GF/C/s is lower. We find that at scale, both implementations of the matrix-free SpMV which we utilize as the kernel within our multigrid smoother yield a sustained performance greater than 35 teraflops per second. However, we note that the application of the entire Krylov method results in a performance between 13 and 18.5 teraflops per second.

C. Performance comparison

To assess the performance of our matrix-free multigrid implementation, in Table IV we compared the solve time with several robust multi-level implementations which utilize fully assembled matrices. The test problem consists of the sinker model defined in § IV-A, with a mesh of 96^3 elements. As before, the Stokes solution was deemed converged when the initial unpreconditioned residual was reduced by a factor of 10^5. Our reference matrix-free preconditioner (`GMG-i`) consisted of a three level hierarchy with the coarsest operator defined via Galerkin projection (see § IV-A for a full description). The following four preconditioner configurations which exploit assembled operators were considered for comparison;

- `GMG-ii` The operator on the finest level was assembled, and all coarse grid operators were defined via Galerkin projection. The smoother and coarse grid solver used were identical to `GMG-i`.
- `SA-i` The fine operator was assembled and GAMG was used as the preconditioner. A thresholding value of 0.01 was used for constructing the aggregation graph.

The smoother used was identical to GMG−i. The coarse level solver consisted of block Jacobi, with an exact LU factorization applied on each of the subdomains.

- SAML−i Identical to SA−i except the smoothed aggregation algebraic multi-level preconditioner ML [52] was employed. A drop tolerance of 0.01 was employed in constructing the prolongation operator. The maximum size of the coarse level problem was set to 100, and a repartitioning of the coarse level problem (defined via ParMETIS) occurred if the number of equations per subdomain was less than 512.
- SAML−ii Identical to SAML−i except a stronger smoother and coarse level solver was employed. The smoother consisted of FGMRES(2) preconditioned with block Jacobi-ILU(0), and the coarse level solver was an inexact Krylov solve (FGMRES) which was terminated when the relative residual was reduced by a factor of 10^2. The coarse level solver was block Jacobi, with an exact LU factorization applied on each of the subdomains.

In Table IV we report the number of iterations and CPU time (seconds) required for specific operations executed during the solution of a Stokes problem. The operation "MatMult" refers to the total CPU time spend performing SpMV, "PC setup" represents the total time associated with setting up all components of the Stokes preconditioner $\mathbf{P}$, "PC apply" represents the total time spent evaluating the action of $\mathbf{P}^{-1}$ (see Equation (17)) and "Solve" represents the complete time-to-solution of the variable viscosity Stokes problem. Iteration counts were found to generally be lower for the geometric MG preconditioners in comparison to the pure algebraic preconditioners. The GMG−ii preconditioner which utilized Galerkin coarse grid operators on all levels (except the finest) exhibited the lowest iteration count. However, despite requiring 23% less Krylov iterations, the preconditioner which employed a matrix-free tensor product SpMV yielded an overall time-to-solution which was $1.7\times$ faster. Compared to the algebraic multi-level preconditioner configurations we considered here, GMG−i was found to be between $3.3\times$ - $12.4\times$ faster. In addition to the time reported under "PC setup", the geometric multigrid preconditioners required an additional 1.3 seconds (GMG−i) and 3.2 seconds (GMG−ii) for the symbolic and numeric phases associated with evaluating $\mathbf{R}^T\mathbf{AR}$. Hence, the setup cost of GMG−ii was comparable to the pure algebraic approaches for the model resolution considered.

TABLE IV
PROFILING OF DIFFERENT MULTI-LEVEL PRECONDITIONERS IN TERMS OF THE NUMBER OF ITERATIONS AND CPU TIME (S). ALL SIMULATIONS EMPLOYED A MESH OF 96^3 Q_2 ELEMENTS AND USED 1536 CORES ON EDISON. SEE TEXT FOR THE DEFINITION OF THE MULTI-LEVEL PRECONDITIONERS AND OPERATIONS. TO ENABLE COMPARISON WITH ML, ALL CALCULATIONS WERE PERFORMED USING 32-BIT INDICES.

Precon. type	Its.	Operation (s) MatMult	PC setup	PC apply	Solve
GMG−i	95	1.6	1.1	5.3	6.9
GMG−ii	72	1.6	1.4	10.4	11.7
SA−i	129	15.2	5.5	20.1	22.7
SAML−i	454	68.9	4.8	71.9	85.5
SAML−ii	239	40.0	6.0	62.3	68.5

V. CONTINENTAL RIFTING AND BREAKUP

The breakup of continents occurs after a phase of continental rifting, in which the crust thins either over wide regions or over narrow localized ones, thereby participating in the diversity of passive continental margins. The processes that lead to continental breakup are important factors influencing the formation, maturation, and storage of hydrocarbons along passive margins and thus are of economic interest.

Geologic records have revealed that when continents separate, the propagation of breakup is highly time dependent [53]. Periods of quiescence propagation are usually associated with large fracture zones and the formation of oblique margins. Oblique structures cannot be simulated using 2D models. Because of the computational challenges associated with performing high resolution, 3D, nonlinear large-deformation modeling of the lithosphere and crust, few such transient simulations of continental breakup have been performed [54]–[60]

A. Model description

The model domain we consider spans 1200 km $\times$ 600 km in the horizontal direction (x-z) and 200 km in the vertical direction (y). Together with the Stokes problem in Equation (1) and (3), we additionally solve the energy equation

$$\frac{\partial T}{\partial t} + \boldsymbol{u}\cdot\nabla T = \nabla\cdot(\kappa\nabla T)\,, \tag{20}$$

where T is the temperature and κ the thermal diffusion. Equation (20) is solved using Q_1 finite elements, stabilized via the SUPG method [61].

Initially, the model domain is divided into three regions (lithologies) which we refer to as "mantle" (lower 160 km), "weak crust" (20 km thick), and "strong crust" (20 km thick). Material points are used to identify the different regions. Associated with each lithology is a unique set of material parameters. The flow used in each lithology consists of a a temperature, pressure, and strain-rate-dependent viscosity defined by an Arrhenius type law. The effective viscosity involves a Drucker-Prager stress limiter that parametrizes the brittle behavior of rocks in the two crustal layers near the surface. All lithologies are assumed to have buoyancy variations defined by the Boussinesq equations.

To initiate rifting, we introduce a small random material heterogeneity, which can be thought of as a zone of predefined "damage" (see Figure 3 - central zone along back face). We utilized two types of boundary conditions: (i) purely cylindrical extension of 2 cm/yr applied symmetrically in the x-direction, and (ii) symmetric extension of 2 cm/yr applied in the x-direction, together with a slight component of shortening (2 mm/yr) in the z-direction, which is applied on the opposite side of the damaged zone.

The simulation results presented were performed using 512 cores on "Rostand", an SGI ICE 8200 with compute nodes

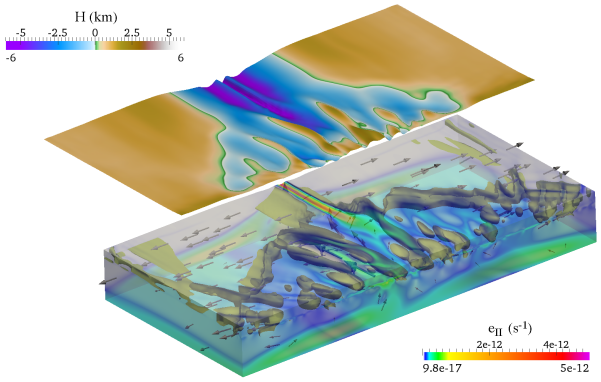

Fig. 3. Evolution of topography (H) and basin structures from a rifting experiment $\sim$10 Myr after breakup has occurred. Upper panel outlines the passive margins (topographic lows), which are accurately simulated via the deformed free surface. Lower panel shows the second invariant of the strain-rate tensor (e_{II}), which highlights the complex geometry of cross-cutting faults that intersect the isolated basin (front face). This simulation employed 2 mm/yr shortening which resulted in the development of oblique active structures (denoted by yellow isosurfaces).

consisting of 2×6 Intel Xeon (Series 5600) cores running at 2.8 GHz. Each model utilized a mesh resolution of $256 \times 32 \times 128$ Q_2 elements. Convergence of the nonlinear Stokes problem was defined to have occurred when $\|\mathbf{F}\| < 10^{-2}$, or when the initial nonlinear residual (redefined at each time step) was reduced by a factor of 10^4. We restricted the nonlinear solver to perform a maximum of five iterations. For the non dimensional scaling we adopted, these nonlinear stopping conditions proved effective for the transient rift problems considered here (see Figure 4). We defined the action of $\tilde{\mathbf{J}}_{uu}^{-1}$ in Equation (17) via a $V(3,3)$ cycle. The multigrid preconditioner was configured with three levels, using the following element hierarchy (from the coarsest to the finest level): $32 \times 16 \times 16$, $128 \times 32 \times 64$, $256 \times 32 \times 128$.

We used a coarse grid solver consisting of an inexact Krylov method (CG), preconditioned with an algebraically defined additive Schwarz method (ASM). The ASM preonditioner employed an overlap of 4, with subdomain solves defined via a single application of ILU(0). The coarse grid solver was terminated after 25 iterations, or if the initial residual was reduced by a factor of 10^4. From our experience, using ASM preconditioners for variable viscosity Stokes problems in which the total number of cores (subdomains) is $<$ 2-3$\times 10^3$, will result in an efficient coarse grid solver. However, when the numbers of cores (subdomains) becomes larger than $>$ 4×10^3, we find ASM is inefficient, leading to coarse grid solve times which are larger than the total time spent applying the smoother on the finest level. This is in part associated to (i) the poor algorithmic scalability of ASM and (ii) the number of global reductions required by the Krylov method which is applied to a fully distributed coarse grid operator which possesses as many subdomains as the fine grid operator, thereby exposing network latency. In such situations, where the core count exceeds 2k, we find that coarse grid preconditioners which are both computational and algorithmically scalable (e.g. the smoothed aggregation based GAMG implementation in PETSc) are essential.

In Figure 4 we summarize the performance characteristics of the nonlinear solver and preconditioner configuration adopted for the rifting experiments. As a function of each model time step, we show the total number of nonlinear iterations ("Total Newton" - green dots) required to ensure that $\|\mathbf{F}\| < 10^{-2}$, and the total number of Krylov iterations required to solve the linearized Stokes problem ("Total Krylov" - grey bars). The average number of Krylov iterations per time step is shown via the blue line. In the early stages of the simulation (first five time steps), we observe failure of our nonlinear solver, with more than five iterations being required. This is attributed to rapid variations in the free surface (topography) which occur due to an initial buoyancy structure that is out of equilibrium with the initially horizontal topography. Once this nonzero and dynamically consistent topographic surface has been established, enforcing $\|\mathbf{F}\| < 10^{-2}$ is possible with 1-2 Newton iterations. As time advances, damage accumulates in the central region of the domain and shear zones, represented via rheological nonlinearities, become highly localized. We note that despite the yielding condition (associated with strain-rate and pressure dependent viscosity) being activated throughout the entire simulation, we observe convergence of the nonlinear Stokes problem in typically less than three iterations.

Each model required approximately 1500-2000 time steps, with the average CPU time per time step being $\sim$160-200 seconds. This average reflects the time required to: solve the nonlinear Stokes problem; perform all nonlinear residual evaluations; interpolate between material points and the quadrature points; the update of material point history variables (plastic strain) and the coordinates; perform all mesh updates associated with the ALE formulation; solve the conservation of energy (Equation (20)) and to write any requested data to disk.

Our models on continental rifting confirm that a weak lower crust favors wider passive margins and that a quiescent period of propagation of continental rifting can be induced by shortening in the direction normal to the ridge. These models highlight that a very small amount of axial shortening also induces obliquity and that the accommodation of this obliquity by the mid-oceanic ridge or by the continental structures is a strong function of the viscosity of the lower crust. A weak lower crust favors margins that are oblique to spreading, while a strong lower crust favors ridge jumps and transform margins.

VI. Outlook

We have presented a practical geodynamics package using a material-point method and stable, locally conservative, mixed finite element discretization. By eschewing assembled sparse matrices in favor of a matrix-free evaluation that exploits tensor-product structure, the cost of applying the operator in multigrid smoothing and residual evaluation was reduced by an order of magnitude. With regards to time-to-solution, our

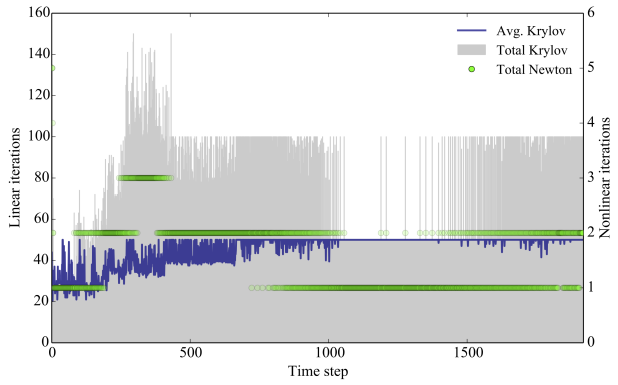

Fig. 4. Convergence behavior of the nonlinear and linear solvers used for the rift model as a function of model time step. Green markers indicate the number of nonlinear iterations required for convergence. The grey and blue lines represent the total and average number of Krylov iterations (applied to the Stokes operator) which were performed at each time step. Refer to text for details regarding the solver/preconditioner and stopping conditions used.

combined Chebyshev tensor-product based SpMV smoother is $2.7\times$ faster than the equivalent algorithm using assembled matrices. Compared to several purely algebraic multilevel preconditioner implementations, our matrix-free approach was found to be between $3.3\times$ - $12.4\times$ faster.

The performance of our nonlinear Stokes solver is now compute-bound, rather than memory-bound. Thus we expect the benefits to improve further if the hardware balance continues to skew toward flops over memory bandwidth. Avoiding assembled matrices also reduces memory requirements, thus increasing the maximum problem sizes that can be solved and increasing the utility of low-memory energy-efficient architectures.

We believe that algorithmic restructuring similar to the present work will improve performance and efficiency for many applications that currently rely on assembled sparse matrices. The greatest challenges will be in (i) finding methods that are amenable to matrix-free implementation *and* converge as rapidly and reliably as the best matrix-based algorithms and (ii) finding good interfaces to minimize coding effort required for applications to experiment with these methods.

ACKNOWLEDGMENT

NERSC is thanked for compute time on the Cray XC-30 "Edison". Philippe de Clarens at TOTAL is acknowledged for time on the SGI-ICE "Rostand". Caroline Baldassari (SGI) is thanked for helping set up the code on "Rostand". JB was supported by Swiss National Science Foundation Grant 200021-113503/1 and the U.S. Department of Energy's Office of Science under Contract DE-AC02-06CH11357. The authors thank the four anonymous reviewers for their constructive feedback.

REFERENCES

[1] F. H. Harlow and E. Welch, "Numerical calculation of time-dependent viscous flow of fluid with free surface," *Physics of Fluids*, vol. 8, no. 12, pp. 2182–2189, 1965.

[2] D. Sulsky, Z. Chen, and H. L. Schreyer, "A particle method for history-dependent materials," *Numerical Methods in Applied Mechanics and Engineering*, vol. 118, pp. 179–196, 1994.

[3] R. F. Weinberg and H. Schmeling, "Polydiapirs: Multiwave length gravity structures," *Journal of Structural Geology*, vol. 14, pp. 425–436, 1992.

[4] S. Zaleski and P. Julien, "Numerical simulation of rayleigh-taylor instability for single and multiple salt diapirs," *Tectonophysics*, vol. 206, no. 1-2, pp. 55–69, 1992.

[5] A. Poliakov and Y. Podladchikov, "Diapirism and topography," *Geophysical Journal International*, vol. 109, pp. 553–564, 1992.

[6] P. Fullsack, "An arbitrary Lagrangian-Eulerian formulation for creeping flows and its application in tectonic models," *Geophysical Journal International*, vol. 120, pp. 1–23, 1995.

[7] P. E. van Keken, S. D. King, H. Schmeling, U. R. Christensen, D. Neumeister, and M.-P. Doin, "A comparison of methods for the modeling of thermochemical convection," *Journal of Geophysical Research*, vol. 102, no. B10, pp. 22 477–22 496, 1997.

[8] A. Y. Babeyko, S. V. Sobolev, R. B. Trumbull, O. Oncken, and L. L. Lavier, "Numerical models of crustal-scale convection and partial melting beneath the Altiplano-Puna plateau," *Earth and Planetary Science Letters*, vol. 199, pp. 373–388, 2002.

[9] T. V. Gerya and D. A. Yuen, "Characterictics-based marker method with conservative finite-difference schemes for modeling geological flows with strongly variable transport properties," *Physics of the Earth and Planetary Interiors*, vol. 140, no. 4, pp. 293–318, 2003.

[10] L. Moresi, F. Dufour., and H.-B. Mühlhaus, "A Lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials," *Journal of Computational Physics*, vol. 184, pp. 476–497, 2003.

[11] S. J. H. Buiter, A. Y. Babeyko, S. Ellis, T. V. Gerya, B. J. P. Kaus, A. Kellner, G. Schreurs, and Y. Yamada, "The numerical sandbox: comparison of model results for a shortening and an extension experiment," in *Analogue and numerical modelling of crustal-scale processes*, ser. Special Publication - Geological Society of London, vol. 253. Geological Society of London, London, 2006, pp. 29–64.

[12] T. V. Gerya and D. A. Yuen, "Robust characteristics method for modelling multiphase visco-elasto-plastic thermo-mechanical problems," *Physics of the Earth and Planetary Interiors*, vol. 163, pp. 83–105, 2007.

[13] L. Moresi, S. Quenette, V. Lemiale, C. Meriaux, B. Appelbe, and H. B. Mühlhaus, "Computational approaches to studying non-linear dynamics of the crust and mantle," *Physics of the Earth and Planetary Interiors*, vol. 163, pp. 69–82, 2007.

[14] A. A. Popov and S. V. Sobolev, "SLIM3D: A tool for three-dimensional thermomechanical modeling of lithospheric deformation with elasto-visco-plastic rheology," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1-4, pp. 55–75, 2008, Recent Advances in Computational Geodynamics: Theory, Numerics and Applications.

[15] H. Schmeling, A. Y. Babeyko, A. Enns, C. Faccenna, F. Funiciello, T. V. Gerya, G. J. Golabek, S. Grigull, B. J. P. Kaus, G. Morra, S. M. Schmalholz, and J. van Hunen, "A benchmark comparison of spontaneous subduction models - Towards a free surface," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1-4, pp. 198–223, 2008.

[16] M. E. T. Quinquis, S. J. H. Buiter, and S. Ellis, "The role of boundary conditions in numerical models of subduction zone dynamics," *Tectonophysics*, vol. 497, no. 1-4, pp. 57–70, 2011.

[17] C. Thieulot, "FANTOM: Two- and three-dimensional numerical modelling of creeping flows for the solution of geological problems," *Physics of the Earth and Planetary Interiors*, vol. 188, no. 1–2, pp. 47–68, 2011.

[18] S. M. Lechmann, D. A. May, B. J. P. Kaus, and S. M. Schmalholz, "Comparing thin-sheet models with 3-D multilayer models for continental collision," *Geophysical Journal International*, vol. 187, no. 1, pp. 10–33, 2011.

[19] T. Gerya, *Introduction to numerical geodynamic modelling*. Cambridge University Press, 2010.

[20] W. Landry, L. Hodkinson, and S. Kientz, "GALE User Manual," Computational Infrastructure for Geodynamics, Tech. Rep. Version 2.0.1, 2012. [Online]. Available: http://www.geodynamics.org/cig/software/gale/gale.pdf

[21] C. Burstedde, O. Ghattas, G. Stadler, T. Tu, and L. C. Wilcox, "Parallel scalable adjoint-based adaptive solution of variable-viscosity Stokes flow problems," *Computer Methods in Applied Mechanics and Engineering*, vol. 198, pp. 1691–1700, 2009.

[22] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*. New York, NY, USA: Springer-Verlag New York, Inc., 1991.

[23] P. B. Bochev, C. R. Dohrmann, and M. D. Gunzburger, "Stabilization of low-order mixed finite elements for the Stokes equations," *SIAM Journal on Numerical Analysis*, vol. 44, no. 1, pp. 82–101, 2006.

[24] A. Brandt and O. Livne, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, Revised Edition*. SIAM, 2011.

[25] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*. Academic Press, 2001.

[26] S. Vanka, "Block-implicit multigrid solution of Navier-Stokes equations in primitive variables," *Journal of Computational Physics*, vol. 65, no. 1, pp. 138–158, 1986.

[27] H. Elman, V. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, "A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations," *Journal of Computational Physics*, vol. 227, no. 1, pp. 1790–1808, 2008.

[28] P. Tackley, "Modelling compressible mantle convection with large viscosity contrasts in a three-dimensional spherical shell using the yin-yang grid," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1-4, pp. 7–18, 2008.

[29] D. A. May and L. Moresi, "Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1, pp. 33–47, 2008.

[30] C. Burstedde, O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. Wilcox, and S. Zhong, "Scalable adaptive mantle convection simulation on petascale supercomputers," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. IEEE Press, 2008, p. 62.

[31] D. Boffi and L. Gastaldi, "On the quadrilateral Q_2-P_1 element for the Stokes problem," *International Journal for Numerical Methods in Fluids*, vol. 39, no. 11, pp. 1001–1011, 2002.

[32] G. Matthies and L. Tobiska, "The Inf-Sup condition for the mapped Q_k-P_{k-1}^{disc} element in arbitrary space dimensions," *Computing*, vol. 69, no. 2, pp. 119–139, 2002.

[33] W. Landry, "Robust, scalable methods for incompressible Stokes flow with yielding rheologies." Abstract #DI31A-1597, San Francisco, Calif.: Presented at 2009 Fall Meeting, AGU, 14-18 Dec. 2009.

[34] S. Balay, M. F. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, and H. Zhang, "PETSc Web page," http://www.mcs.anl.gov/petsc, 2014. [Online]. Available: http://www.mcs.anl.gov/petsc

[35] ——, "PETSc users manual," Argonne National Laboratory, Tech. Rep. ANL-95/11 - Revision 3.4, 2013. [Online]. Available: http://www.mcs.anl.gov/petsc

[36] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, "Efficient management of parallelism in object oriented numerical software libraries," in *Modern Software Tools in Scientific Computing*, E. Arge, A. M. Bruaset, and H. P. Langtangen, Eds. Birkhäuser Press, 1997, pp. 163–202.

[37] J. Brown, M. G. Knepley, D. A. May, L. C. McInnes, and B. F. Smith, "Composable linear solvers for multiphysics," in *Proceeedings of the 11th International Symposium on Parallel and Distributed Computing (ISPDC 2012)*. IEEE Computer Society, 2012, pp. 55–62. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/ISPDC.2012.16

[38] P. Brune, M. Knepley, B. Smith, and X. Tu, "Composing scalable nonlinear algebraic solvers," Argonne National Laboratory, Preprint ANL/MCS-P2010-0112, 2013.

[39] S. C. Eisenstat and H. F. Walker, "Choosing the forcing terms in an inexact Newton method," *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 16–32, 1996.

[40] S. Eisenstat, H. Elman, and M. Schultz, "Variational iterative methods for nonsymmetric systems of linear equations," *SIAM Journal on Numerical Analysis*, vol. 20, no. 2, pp. 345–357, 1983.

[41] Y. Saad, "A flexible inner-outer preconditioned GMRES algorithm," *SIAM Journal on Scientific Computing*, vol. 14, pp. 461–461, 1993.

[42] M. Olshanskii and A. Reusken, "Analysis of a Stokes interface problem," *Numerische Mathematik*, vol. 103, no. 1, pp. 129–149, 2006.

[43] H. Uzawa, "Iterative methods for concave programming," *Studies in Linear and Nonlinear Programming*, pp. 154–165, 1958.

[44] M. Murphy, G. Golub, and A. Wathen, "A note on preconditioning for indefinite linear systems," *SIAM Journal on Scientific Computing*, vol. 21, no. 6, pp. 1969–1972, 2000.

[45] I. C. F. Ipsen, "A note on preconditioning nonsymmetric matrices," *SIAM Journal on Scientific Computing*, vol. 23, pp. 1050–1051, 2001.

[46] M. Adams, "A distributed memory unstructured Gauss-Seidel algorithm for multigrid smoothers," in *Supercomputing, ACM/IEEE 2001 Conference*. IEEE, 2001, pp. 14–14.

[47] M. F. Adams, M. Brezina, J. J. Hu, and R. S. Tuminaro, "Parallel multigrid smoothing: polynomial versus Gauss–Seidel," *Journal of Computational Physics*, vol. 188, no. 2, pp. 593–610, 2003.

[48] J. D. McCalpin, "STREAM: Sustainable memory bandwidth in high performance computers," University of Virginia, Tech. Rep., 1995, http://www.cs.virginia.edu/stream.

[49] J. Brown, "Efficient nonlinear solvers for nodal high-order finite elements in 3D," *Journal of Scientific Computing*, vol. 45, pp. 48–63, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10915-010-9396-8

[50] P. Fischer, J. Kruse, J. Mullen, H. Tufo, J. Lottes, and S. Kerkemeier, "NEK5000–open source spectral element CFD solver," 2008. [Online]. Available: http://nek5000.mcs.anl.gov

[51] D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp, "A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the earth simulator," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. ACM, 2003, p. 4.

[52] M. Gee, C. Siefert, J. Hu, R. Tuminaro, and M. Sala, "ML 5.0 smoothed aggregation user's guide," Sandia National Laboratories, Tech. Rep. SAND2006-2649, 2006.

[53] V. Courtillot, "Propagating rifts and continental breakup," *Tectonics*, vol. 1, no. 3, pp. 239–250, 1982.

[54] S. Brune and J. Autin, "The rift to break-up evolution of the Gulf of Aden: Insights from 3D numerical lithospheric-scale modelling," *Tectonophysics*, vol. 607, pp. 65–79, 2013.

[55] T. V. Gerya, "Initiation of transform faults at rifted continental margins: 3D petrological-thermomechanical modeling and comparison to the Woodlark Basin," *Petrology*, vol. 21, no. 6, pp. 550–560, 2013.

[56] S. Brune, A. A. Popov, and S. V. Sobolev, "Modeling suggests that oblique extension facilitates rifting and continental break-up," *Journal of Geophysical Research: Solid Earth*, vol. 117, no. B8, 2012.

[57] L. Le Pourhiet, B. Huet, D. A. May, L. Labrousse, and L. Jolivet, "Kinematic interpretation of the 3D shapes of metamorphic core complexes," *Geochemistry Geophysics Geosystems*, vol. 13, no. 9, p. Q09002, 2012.

[58] V. Allken, R. S. Huismans, and C. Thieulot, "Factors controlling the mode of rift interaction in brittle-ductile coupled systems: A 3D numerical study," *Geochemistry, Geophysics, Geosystems*, vol. 13, no. 5, 2012.

[59] E. Choi, L. Lavier, and M. Gurnis, "Thermomechanics of mid-ocean ridge segmentation," *Physics of the Earth and Planetary Interiors*, vol. 171, no. 1, pp. 374–386, 2008.

[60] T. Gerya, "Dynamical instability produces transform faults at mid-ocean ridges," *Science*, vol. 329, no. 5995, pp. 1047–1050, 2010.

[61] A. N. Brooks and T. J. R. Hughes, "Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 32, no. 1, pp. 199–259, 1982.