PETSc's Software Strategy for the Design Space of Composable Extreme-Scale Solvers[☆]

Barry Smith[a], Lois Curfman McInnes[a], Emil Constantinescu[a], Mark Adams[b], Satish Balay[a], Jed Brown[a], Matthew Knepley[c], Hong Zhang[d]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory*
[b] *Applied Physics and Applied Mathematics Department, Columbia University*
[c] *Computation Institute, University of Chicago*
[d] *Department of Computer Science, Illinois Institute of Technology*

Abstract

Emerging extreme-scale architectures present new opportunities for broader scope of simulations as well as new challenges in algorithms and software to exploit unprecedented levels of parallelism. Composable, hierarchical solver algorithms and carefully designed portable software are crucial to the success of extreme-scale simulations, because solver phases often dominate overall simulation time. This paper presents the PETSc design philogophy and recent advances in the library that enable application scientists to investigate the design space of composable linear, nonlinear, and timestepping solvers. In particular, separation of the control logic of the algorithms from the computational kernels of the solvers is crucial to allow injecting new hardware-specific computational kernels without having to rewrite the entire solver software library. Progress in this direction has already begun in PETSc, with new support for pthreads, OpenMP, and GPUs as a first step toward hardware-independent, high-level control logic for computational kernels. This multipronged software strategy for composable extreme-scale solvers will help exploit unprecedented extreme-scale computational power in two important ways: by facilitating the injection of newly developed scalable algorithms and data structures into fundamental components, and by providing the underlying foundation for a paradigm shift that raises the level of abstraction from simulation of complex systems to the design and uncertainty quantification of these systems.

Keywords: design space exploration and interfaces, data movement and programming models

1. Introduction

A common theme emphasized in recent DOE reports [1–3] is the crucial importance of robust and scalable linear, nonlinear, and timestepping solvers for extreme-scale simulations based on partial differential equations (PDEs) and related modeling, for example, the power grid. The changing landscape of high-performance computing (HPC) systems (many-core node architectures, hybrid combinations of GPU and conventional processors, and high node counts) requires continued innovation in mathematical algorithms and software. Moreover, the unprecedented computing power of emerging architectures presents new opportunities for simulation of complex systems, including multiphysics [4], multiscale, and ensemble computations such as quantifying uncertainties and optimal design of these systems.

Power and manufacturing constraints are increasing the cost of memory access relative to floating point operations, as well as widening the performance gap between naive or unstructured floating point code and well-optimized code. Increasingly, the relevant metric is "science per watt," and algorithms must adapt to maximize this. Over the past decades, algorithmic innovations have provided as many orders of magnitude

[☆]This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

increase in science per watt as have hardware improvements. In order to continue this trend, it is crucial to adopt the best known algorithms to achieve high performance on cutting-edge architectures. And, in order to maximize the availability of these algorithms to science, the best implementations must be encapsulated in reusable libraries.

Recent advances in the Portable, Extensible Toolkit for Scientific computing (PETSc) [5] have substantially improved composability for multilevel, multidomain, multirate, and multiphysics algorithms [6]. These capabilities enable users to investigate the design space of composable linear, nonlinear, and timestepping solvers for these more complex simulations, without making premature choices about algorithms and data structures. The strong encapsulation of the PETSc design facilitates runtime composition of hierarchical methods without sacrificing the ability to customize problem-specific components [7, 8]. These capabilities are essential for application codes to evolve over time and to incorporate advances in algorithms for emerging extreme-scale architectures.

Just as crucial in the push toward extreme-scale computing are recent advances in the PETSc design that enable leveraging GPUs in all computational solver phases [9] and the hybrid MPI/pthread programming model [10]. These design advances mean that one does not have to forsake the most mathematically sophisticated, composable hierarchical solvers in order to utilize GPUs and multicore. Rather, the software logic that supports composable solvers is independent of the computational kernels running on the accelerator hardware, so that one can easily incorporate new kernels, tuned to a particular new hardware, without rewriting the application or high-level solver library. An overview article in the *Encyclopedia of Parallel Computing* [8] gives an introduction to some of the recent design decisions incorporated in PETSc.

We have already made important progress in separating the *control logic* of the PETSc software from the *computational kernels*. As the community transitions away from an MPI-only model for parallelism, this separation of concerns is crucial because we can avoid a total rewrite of our software base. That is, while good performance at the exascale will require a major overhaul of the code for computational kernels in PETSc, the high-level control logic is largely hardware-independent and requires only modest refactoring to adapt to new hardware. In other words, we will not need to reimplement from scratch the hundreds of linear, nonlinear, and timestepping solvers encapsulated in PETSc. Of course, an essential complement to new hardware-specific computational kernels is extending the solver libraries to incorporate new algorithms that reduce communication and synchronization, as well as new programming models that explicitly acknowledge data movement and hierarchies of locality. The key point is that our design enables a separation of concerns for these two fundamental aspects of extreme-scale solvers, thereby making tractable a potentially daunting transition process.

The remainder of this paper is organized as follows. Section 2 introduces overarching library design goals and provides a historical perspective. Section 3 explains our approach to composable extreme-scale linear, nonlinear, and timestepping solvers and provides highlights of recently added capabilities. Section 4 discusses future directions for exploiting new architectural features and addressing more complex simulations.

2. Historical Perspective

Our work in solver algorithms and software for the implicit solution of PDEs, as encapsulated in PETSc, began in the early 1990s and has always focused on the largest-scale parallel systems available at the time. The original goal of PETSc was to develop efficient algebraic solvers and provide the ability to use any appropriate solver for any suitable set of PDEs. That is, our **first goal** was to provide a simple and consistent way for the user to **specify the algebraic system** (in general, nonlinear and time-dependent) so that a wide variety of solvers could be explored, thereby enabling application scientists to experiment with diverse algorithms and implementations without requiring premature commitment to particular data structures and solvers. The system for specification goes far beyond simply requiring the user to provide the Jacobian of a nonlinear system in a particular sparse matrix format. Rather, we employ a set of specifications for how the user-provided code may provide information needed by implicit multilevel and Newton-based solvers. The specifications are layered, so that if the user code can provide more information or flexibility, more powerful solvers may then be employed. For example, if the user's code can evaluate the nonlinear

functions on a set of meshes, then geometric multigrid may be used to solve the Jacobian system. This approach is closely related to and inspired by the hypre team's *conceptual interfaces* [11].

Our **second goal** was to **provide scalable powerful parallel algebraic solvers** suitable for a variety of physical models. In the early 1990s, these goals were considered naive and impossible. Time has shown that the solution to the first goal has vastly simplified the solution to the second goal. With the mathematical, algorithmic, abstract framework that we developed, parallelism for algebraic solvers went from being a barrier for many scientific applications to being a mere nuisance. In fact, for most new single-physics PETSc application codes, linear, nonlinear, and timestepping solvers for PDEs "just work" in parallel, with little user effort.

A **third goal** of the PETSc design was its **extensibility to allow the use of powerful solvers developed by other groups**. Not only does PETSc provide a wide suite of linear, nonlinear, and timestepping solvers, the library can also directly use linear solvers from hypre [12], SuperLU [13], Trilinos [14], and over ten other solver packages. This capability has helped promote interoperability among high-performance numerical software packages and thereby helped end users explore the solver design space, where diverse and complementary aspects of algorithms naturally are the focus of different researchers throughout the HPC community. For example, the fusion codes M3D [15], XGC1 [16], and GTC-P [17] can switch among solvers from PETSc, hypre, and SuperLU_Dist at runtime, depending on which solver is most appropriate for a given simulation. Moreover, scientists can use PETSc infrastructure to introduce their own custom algorithmic variants that leverage application-specific knowledge.

3. Composable Linear, Nonlinear, and Timestepping Solvers

The main challenges posed by next-generation HPC platforms arise from the fact that future improvements in processor performance will come from multiple cores per processor rather than higher clock frequencies or just from more compute nodes. This circumstance poses a major challenge to the generation of simulation software written during the "all-MPI" golden age of parallel computing. The difficulties are compounded by latency and bandwidth effects of nonuniform memory access (NUMA) within a processing node. Thus, simple OpenMP pragmas for code running within a node will not provide satisfactory performance without attention to data affinity (ensuring that data is stored in physical memory that is "close" to where it will be accessed). In addition to increasing core counts, it is necessary to utilize packed vector registers and simultaneous threading.

As introduced in Section 1, in PETSc we are addressing these architectural changes with a multipronged approach of (1) *new composable algorithms* for linear, nonlinear, and timestepping solvers, which enable the construction of custom schemes tailored to exploit both hierarchical architectural features and increasingly complex modeling, and (2) *new computational kernels* that enable leveraging GPUs in all computational solver phases [9] and the hybrid MPI/pthread programming model [10]. Section 3.1 provides an overview of support for composable hierarchical linear solvers. Section 3.2 discusses our approach for nonlinear solvers and introduces new capabilities for variational inequalities. Section 3.3 discusses advances in integrators for ordinary differential equations (ODEs) and differential algebraic equations (DAEs).

3.1. Composable Sparse Linear Solvers

During recent years we have extended linear solvers in PETSc to better support coupling between two or more distinct PDE-based mathematical models for steady-state problems. This research builds on both the power of PETSc for solving individual PDE-based models and its extensibility, which supports the composition of multiple solvers. We provide multiphysics algebraic coupling primarily through the new `DM` object, which PETSc uses to define suitable decompositions and to provide the algebraic solvers with any necessary information that might depend on discretization, geometry, and physics, along with the `FieldSplit` preconditioner and `Nest` matrix format, which implement relaxation and factorization splittings with efficient and flexible storage [6]. Applications using these multiphysics features include lithosphere dynamics [18], subduction and mantle convection [19–21], ice sheet dynamics [22–24], subsurface reactive flow [25], tokamak fusion [26, 27], mesoscale materials modeling [28], and power networks [29, 30].

This recent work can be characterized as a framework for *physics-based preconditioners*. Thus, instead of simply handing a sparse matrix off to a black-box linear solver, the user describes the (usually logical block) structure of the matrix and then "tells" the solver how to compose the outer linear solver based on inner linear solvers associated with blocks and Schur complements [6]. The locations of the logical blocks of the matrix are efficiently provided by the PETSc abstract class of index sets (`IS`), and the composition of the solvers is handled by the PETSc preconditioner classes: `composite`, `Galerkin`, `FieldSplit`, and `multigrid`. These preconditioner classes are containers that manage the composition of simple solvers. For example, the multigrid container manages the calls to the smoother solver on each level and the calls to the restriction and interpolation operators. The fieldsplit container manages the calls to solvers on each or appropriate combinations of the separate fields. The preconditioners are composable, so that, for example, a fieldsplit solver can be used where each field is separately solved with multigrid when appropriate. Related numerical software includes pARMS [31] and the Teko package of Trilinos [14].

Index sets and logical blocks of "matrices" in PETSc also can be used with matrix-free linear operators, where a matrix data structure is not explicitly stored, but rather selected operations (typically matrix-vector products, as used in Jacobian-free Newton-Krylov methods [32]) are computed by user-provided or library-provided routines. In that case, "shell" matrices are returned; these can be manipulated and used in the composition of solvers just as if they were explicitly stored sparse matrices. This capability allows the same composition infrastructure to be used for both matrix-free and sparse matrix-based simulations, including probably the most common case, which is a combination of both matrix-free and partially stored matrices.

The innermost kernels of these composed preconditioners determine the overall floating point efficiency of the solvers. These routines, triangular solves used for ILU, LU, and SOR relaxation, are typically the most time-consuming parts of (time-dependent, nonlinear) simulations. We recently developed new data structures and associated software kernels for sparse triangular solves that achieved improvements ranging from 15 to 60 percent (that is, for some matrices, reducing the triangular solve times by more than half and the overall simulation time by up to 30 percent) [33].

We have recently begun developing capability for algebraic multigrid (AMG) with implementations of an unstructured geometric algorithm and a standard smoothed aggregation AMG solver as part of a multilevel library. Our approach enables new algorithms to be expressed with minimal new code. Fundamental building blocks, such as matrix-matrix products and a greedy maximal independent set aggregation algorithm, allow new development to focus specifically on algorithmic content of interest. We have observed good scaling in the solve phase up to 13K+ cores on the Cray XE6 at NERSC and very good strong scaling on the 2D Poisson solves from the XGC1 gyrokinetic fusion plasma code. This good strong scaling is achieved with process aggregation and repartitioning of the coarse grids, which become computational bottlenecks at scale.

Also notable is work with the UNIC project on neutronics for nuclear reactor simulation. The goal of this work was to reduce the uncertainties and biases in reactor design calculations by progressively replacing existing multilevel averaging techniques with methods based on highly accurate solutions of the governing equations. Scalable linear solvers in PETSc enabled the simulation of neutron transport in full reactor cores [34]. This combination, which has used well over half a trillion unknowns, runs on the latest petascale DOE machines, including 222,912 cores of Cray XT5 and 294,912 cores of Blue Gene/P. UNIC was a finalist in the 2009 Gordon Bell Competition at SC09 [35].

3.2. Flexible Nonlinear Solvers

The basic framework for fully implicit nonlinear solvers, including strong model coupling of multiple nonlinear systems, is (truncated) Newton's method (see, e.g., [36]), using (possibly matrix-free) Newton-(Krylov) techniques. The embodiment of Newton's method in PETSc is the `SNES` component [5], which incorporates enormous inherent flexibility in its design. Related software includes the NOX package of Trilinos [14] and the KINSOL component of SUNDIALS [37]. `SNES` uses the Newton approach of solving the nonlinear system using some "approximation" to the Jacobian of the nonlinear function. In PETSc, the approximation of the Jacobian can be computed in many ways, with default support for (1) provision of the analytic Jacobian or a simplified Jacobian, (2) efficient explicit computation of the Jacobian entries by finite differencing of $F(u^n)$ with coloring, (3) "matrix-free" matrix-vector product application [32] of $J(u^n)$ by either differencing of $F(u^n)$ or computer code, or (4) automatic generation of code that computes the

Jacobian or applies it to a vector via ADIC or ADIFOR [38]. The approximate solution of the linear system, in PETSc, can be computed in essentially unlimited ways, including the composable approaches introduced in Section 3.1, which are essential for coupled multiphysics problems. Of particular note is recent work on hydrodynamics [39, 40] and the subsurface flow code PFLOTRAN [41], which scales up to 40,000 cores on the Cray XT5 and IBM Blue Gene/P and is motivating new research on composable hierarchical solvers for extreme-scale systems.

Many applications communities are beginning to develop multiphysics applications that combine codes for two or more models [4]. The incorporation of matrix-free nonlinear solvers is particularly important for multiphysics scenarios because this approach eliminates the need to compute the fully coupled Jacobian and yet still enables Newton's method to achieve rapid quadratic convergence. While most applications cannot readily provide full Jacobians of coupled systems, approximations of the various Jacobians for the submodels are commonly available. Thus, in this context, the term *matrix-free* means that while there is no explicit storage of the entire sparse Jacobian matrix, there may be explicit storage of portions of the Jacobian related to particular subparts of the physics. PETSc also includes new capabilities for composing scalable nonlinear solvers such as nonlinear GMRES, nonlinear CG, quasi-Newton, and nonlinear multigrid [42].

An important new addition to `SNES` is comprehensive scalable support for algebraic variational inequalities (VIs), the discrete counterpart to differential variational inequalities (DVIs), which enable the consistent and efficient modeling of transitional phenomena, especially applications involving phase changes, free boundaries, and hybrid discrete-continuum behavior. DVIs contain two terms: a differential equation and a generalized algebraic equation represented by complementarity constraints or a box-constrained variational inequality, for example, $0 \leqslant u \leqslant 1$, that formalizes the concept of switches. Current support includes three scalable algorithmic approaches for the solution of box-constrained VIs: (1) a reduced space, active set, gradient projection Newton-based approach similar to that of TRON [43], (2) a semi-smooth solver similar to PATH [44], and (3) a variant of the reduced space active set method that introduces Lagrange multipliers to enforce the active constraints instead of removing them from the equations. As with all capabilities of PETSc, we have provided a suite of algorithms, since no single algorithm is best for all VIs. For example, if the original simulation has fast solvers (FFT for example) available, the Lagrange multiplier approach allows their use with VIs, while the other two do not. VI capabilities are an exciting new addition to the PETSc toolbox, because phase-field models and other problems with bounds on solutions can now be handled properly without resorting to "cut-off" or "smoothing" functions that damage the approximation properties of the models [28].

These VI solvers can be used in conjunction with any of the preconditioners that PETSc provides or can apply, including the composable preconditioning techniques introduced in Section 3.1. For example, for an Allen-Cahn variational inequality that arises in the phase-field approach to mesoscale modeling of irradiated materials [28], we can employ a block Schur complement preconditioner with multigrid as a block solver or invert the roles of the block preconditioner and multigrid and hence run multigrid on the entire problem using a Schur complement fieldsplit preconditioner as the smoother on each level; runtime options for dynamically constructing these different block preconditioners are explained in [6].

3.3. ODE and DAE Integrators

We have recently implemented a family of fully implicit general linear methods (GL), fully implicit generalized α methods, additive Runge-Kutta and Rosenbrock-W IMEX (implicit-explicit, also known as semi-implicit) methods, and explicit strong stability preserving Runge-Kutta methods. These complement the existing classical explicit Runge-Kutta and fully implicit methods in the PETSc scalable time integration component, `TS`. The user specifies the problem in the same way for all methods, by writing the ODE or DAE as $g(t, u, \dot{u}) = f(t, u)$, where g contains the "stiff" terms and f contains only nonstiff terms. Explicit methods will typically place all terms in f, leaving the default implementation, $g(t, u, \dot{u}) = \dot{u}$. The linear or nonlinear problems that arise at each timestep are solved by using the composable infrastructure discussed in Sections 3.1 and 3.2. If methods that require (approximate) Jacobian information are used, the user also provides a function to compute the Jacobian of the stiff part of the equation, $J_\alpha(t, u, \dot{u}) = \partial g/\partial u + \alpha \partial g/\partial \dot{u}$. Reformulating methods for this flexible interface has unified formerly special-purpose methods (such as the generalized α scheme that proved highly successful for isogeometric analysis of Cahn-Hilliard [45] and

Navier-Stokes-Korteweg [46]), thus allowing the authors to simplify their implementations for subsequent work and facilitating easy comparison with alternatives such as IMEX or GL.

Addressing PDE-based models in which different components evolve on different temporal scales is increasingly important [4]. Stiff components such as diffusion, fast reaction, and fast near-equilibrium waves typically require a stable implicit integrator, whereas for nonstiff terms such as slow advection, explicit integrators may be able to solve the problem with much less computational cost. IMEX methods offer the possibility of the best of both worlds: larger timesteps not limited by diffusive/dispersive or stiff wave stability constraints, and less global (or better overlap of) synchronization cost than required by fully implicit schemes. IMEX methods allow one to "dial-in" the amount of implicitness based on both the problem and machine characteristics, as well as to reduce the number of costly nonstiff function evaluations in the nonlinear solver and to avoid solving implicit systems with nonsmooth functions (like TVD limiters), which slow the convergence of nonlinear solvers. We have added two families of IMEX methods to PETSc. Additive Runge-Kutta methods use the stiff part g and nonstiff part f separately and can be nonlinearly implicit on the stiff part. Rosenbrock-W methods are linearly implicit, evaluate g and f together (thus allowing for more efficient/convenient specification), and support lagging the Jacobian of the stiff part across all the stages of a step without affecting order of accuracy. New schemes can be registered by simply providing the table of coefficients. PETSc includes many specific schemes from the literature as well as several new optimal methods with specific stability and accuracy properties.

Most additive Runge-Kutta, Rosenbrock-W, and general linear schemes provide embedded error estimates, which are used by our new adaptive error controllers. The controller logic is not dependent on the family of method, and new implementations can be registered by users. In addition to error control, most schemes provide dense output formulas that allow for high-order *consistent* interpolation within a step and stable extrapolation, which provides an initial guess for the nonlinear solve in the next timestep.

4. Future Directions

The previous sections demonstrate recent advances in the PETSc library that enable application scientists to investigate the design space of composable linear, nonlinear, and timestepping solvers. Separation of the control logic of the algorithms from the computational kernels of the solvers is crucial to allow injecting new hardware-specific computational kernels without having to rewrite the entire solver software library.

Future work on extreme-scale solvers will include attention to even more complex issues. Unlike many classical algorithms, the increasing cost of memory access and communication requires that next-generation libraries not treat their abstract building blocks as black boxes. Instead, it is increasingly important to have extensible components that enable mixing levels of abstraction. Examples include matrix-free methods for solving linear systems (trading the convenience of assembled matrices for lower memory requirements, perhaps exploiting special discretization properties), various forms of linear or nonlinear elimination to reduce communication costs [47, 48], nonlinear methods that do not rely on linear solves (potentially reducing communication and memory bandwidth requirements), and advanced analysis techniques that use components of a model instead of a complete full-accuracy model. A prototypical example of the latter is full-space PDE-constrained optimization in which the PDE is satisfied exactly only when the optimization converges, thus avoiding the need for high-accuracy PDE solves early in the iterative process (e.g., only a preconditioner for the PDE is needed at each optimization iteration) [49–51]. Multilevel methods for uncertainty quantification are another example, enabling accurate statistics to be obtained with a small number of realizations on the finest level [52].

It is important for software maintainability and support of different analysis types that the implementations of these *algorithmically* intrusive algorithms minimize the *software* intrusiveness. Because many problems of practical interest involve global dependence, it is essential for scalability to develop accurate coarse representations of problems. Ideally, these coarse representations apply not just to a linear or algebraic subproblem but also to "outer" analysis problems. Note that these coarse representations need not be restricted to the spatial domain; they may also apply to time, frequency, low rank structure, or combinations of these.

In the path to exascale, solver algorithms must become *more sophisticated* (built by composing a hierarchy of already highly complex algorithms), not less sophisticated. Thus, enhancing and refactoring existing scalable and flexible solver software libraries, such as PETSc, are the only way to achieve the long-term goal of exascale solvers. Starting from scratch would entail unnecessarily reproducing twenty years of previous work before the more sophisticated solvers could even be reasonably implemented. With the PETSc software, we are already embarking on next-generation algorithms and data structures. Building on this foundation of fundamental composable solver components will also facilitate a paradigm shift that raises the level of abstraction from simulation of complex systems to the design and uncertainty quantification of these systems.

References

[1] D. Brown (Chair), Applied Mathematics at the U.S. Department of Energy: Past, Present, and a View to the Future, Office of Science, U.S. Department of Energy (2008).

[2] D. Brown, P. Messina (Chairs), Scientific Grand Challenges: Crosscutting Technologies for Computing at the Exascale (2010).

[3] R. Rosner (Chair), The Opportunities and Challenges of Exascale Computing, Office of Science, U.S. Department of Energy (2010).

[4] D. E. Keyes, L. C. McInnes, C. Woodward, W. D. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, E. Constantinescu, D. Estep, K. Evans, C. Farhat, A. Hakim, G. Hammond, G. Hansen, J. Hill, T. Isaac, X. Jiao, K. Jordan, D. Kaushik, E. Kaxiras, A. Koniges, K. Lee, A. Lott, Q. Lu, J. Magerlein, R. Maxwell, M. McCourt, M. Mehl, R. Pawlowski, A. Peters, D. Reynolds, B. Riviere, U. Rüde, T. Scheibe, J. Shadid, B. Sheehan, M. Shephard, A. Siegel, B. Smith, X. Tang, C. Wilson, B. Wohlmuth, Multiphysics Simulations: Challenges and Opportunities, Tech. Rep. ANL/MCS-TM-321, Argonne National Laboratory, Report of workshop sponsored by the Institute for Computing in Science (ICiS), Park City, Utah, July 30 - August 6, 2011 (Dec 2011).

[5] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.2, Argonne National Laboratory (2011).

[6] J. Brown, M. G. Knepley, D. A. May, L. C. McInnes, B. F. Smith, Composable linear solvers for multiphysics, Preprint ANL/MCS-P2017-0112, Argonne National Laboratory, submitted to the 11th International Symposium on Parallel and Distributed Computing (ISPDC 2012) (2012).

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

[8] B. Smith, Encyclopedia of Parallel Computing, Springer, 2011, Ch. PETSc, the Portable, Extensible Toolkit for Scientific computing.

[9] V. Minden, B. F. Smith, M. Knepley, Preliminary implementation of PETSc using GPUs, in: Proceedings of the 2010 Workshop of GPU Solutions to Multiscale Problems in Science and Engineering, 2010.

[10] S. Abhyankar, B. Smith, K. Stevens, Preliminary implementation of hybrid MPI/pthread programming model in PETSc, Preprint ANL/MCS-P2011-0112, Argonne National Laboratory (2012).

[11] R. D. Falgout, J. E. Jones, U. M. Yang, Pursuing scalability for hypre's conceptual interfaces, ACM Trans. Math. Softw. 31 (2005) 326–350.

[12] R. Falgout, hypre users manual, Tech. Rep. Revision 2.0.0, Lawrence Livermore National Laboratory (2006).

[13] J. W. Demmel, J. R. Gilbert, X. S. Li, SuperLU users' guide, Tech. Rep. LBNL-44289, Lawrence Berkeley National Laboratory (2003).

[14] M. A. Heroux, J. M. Willenbring, Trilinos users guide, Tech. Rep. SAND2003-2952, Sandia National Laboratories (2003).

[15] S. Jardin et al., M3D Web page, `http://w3.pppl.gov/m3d`.

[16] M. Adams, S. Ku, P. Worley, E. D'Azevedo, J. Cummings, C. Chang, Scaling to 150k cores: Recent algorithm and performance engineering developments enabling XGC1 to run at scale, J. of Phys.: Conference Series 180.

[17] M. F. Adams, S. Ethier, N. Wichmann, Performance of particle and cell methods on highly concurrent computational architectures, J. of Phys.: Conference Series 78.

[18] B. Aagaard, S. Kientz, M. G. Knepley, S. Somala, L. Strand, C. Williams, Pylith user manual version 1.6.1 (2011).

[19] R. F. Katz, M. G. Knepley, B. Smith, M. Spiegelman, E. Coon, Numerical simulation of geodynamic processes with the Portable Extensible Toolkit for Scientific Computation, Phys. Earth Planet. In. 163 (2007) 52–68.

[20] D. A. May, L. Moresi, Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics, Physics of the Earth and Planetary Interiors 171 (1-4) (2008) 33–47, Recent Advances in Computational Geodynamics: Theory, Numerics and Applications. `doi:DOI:10.1016/j.pepi.2008.07.036`.

[21] R. F. Katz, M. Spiegelman, B. Holtzman, The dynamics of melt and shear localization in partially molten aggregates, Nature 442 (2006) 676–679.

[22] R. Katz, M. Worster, The stability of ice-sheet grounding lines, Proc. Roy. Soc. A 466 (2010) 1597–1620. `doi:10.1098/rspa.2009.0434`.

[23] T. Tautges et al., SISIPHUS: Scalable ice-sheet solvers and infrastructure for petascale, high-resolution, unstructured simulations, `http://trac.mcs.anl.gov/projects/sisiphus/wiki`.

[24] J. Brown, B. Smith, A. Ahmadia, Achieving textbook multigrid efficiency for hydrostatic ice sheet flow, submitted to SIAM Journal on Scientific Computing (2012).

[25] P. Lichtner et al., PFLOTRAN project, http://ees.lanl.gov/pflotran/.
[26] A. Hakim, J. Cary, J. Candy, J. Cobb, R. Cohen, T. Epperly, D. Estep, S. Krasheninnikov, A. Malony, D. McCune, L. McInnes, A. Pankin, S. B. J. Carlsson, M. Fahey, R. Groebner, S. Kruger, M. Miah, A. Pletzer, S. Shasharina, S. Vadlamani, D. Wade-Stein, T. Rognlein, A. Morris, S. Shende, G. Hammett, K. Indareshkumar, A. Pigarov, H. Zhang, Coupled whole device simulations of plasma transport in tokamaks with the FACETS code, in: Proceedings of SciDAC 2010 Conference, 2010.
[27] M. McCourt, T. D. Rognlien, L. C. McInnes, H. Zhang, Improving parallel scalability for edge plasma transport simulations with neutral gas species, Preprint ANL/MCS-P2018-0112, Argonne National Laboratory, submitted to the Proceedings of the Twenty Second International Conference on Numerical Simulations of Plasmas, Sept. 7-9, 2011, Long Branch, NJ (2012).
[28] L. Wang, J. Lee, M. Anitesu, A. E. Azab, L. C. McInnes, T. Munson, B. Smith, A differential variational inequality approach for the simulation of heterogeneous materials, in: Proceedings of SciDAC 2011 Conference, 2011.
[29] S. Abhyankar, Development of an implicitly coupled electromechanical and electromagnetic transients simulator for power systems, Ph.D. thesis, Illinois Institute of Technology (2011).
[30] S. Abhyankar, B. Smith, H. Zhang, A. Flueck, Using PETSc to develop scalable applications for next-generation power grid, in: Proceedings of the 1st International Workshop on High Performance Computing, Networking and Analytics for the Power Grid, ACM, 2011.
[31] Y. Saad, M. Sosonkina, pARMS: A package for the parallel iterative solution of general large sparse linear systems user's guide, Tech. rep., Minnesota Supercomputer Institute, University of Minnesota, http://www-users.cs.umn.edu/~saad/software/pARMS (2004).
[32] D. A. Knoll, D. E. Keyes, Jacobian-free Newton-Krylov methods: A survey of approaches and applications, J. Comp. Phys. 193 (2004) 357–397.
[33] B. Smith, H. Zhang, Sparse triangular solves for ILU revisited: Data layout crucial to better performance, International Journal of High Performance Computing Applications 25 (4) (2010) 386–391.
[34] M. A. Smith, C. Rabiti, D. Kaushik, B. Smith, W. S. Yang, G. Palmiotti, Fast reactor core simulations using the UNIC code, in: Proceedings of the International Conference on the Physics of Reactors, Nuclear Power: A Sustainable Resource, 2008.
[35] D. Kaushik, M. Smith, A. Wollaber, B. Smith, A. Siegel, W. S. Yang, Enabling high fidelity neutron transport simulations on petascale architectures, SC'09 Gordon Bell Prize Finalist (2009).
[36] J. Nocedal, S. J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.
[37] A. Hindmarsh, P. Brown, K. Grant, S. Lee, R. Serban, D. Shumaker, C. Woodward, SUNDIALS: suite of nonlinear and differential/algebraic equation solvers, ACM Transactions on Mathematical Software 31 (3) (2005) 363–396.
[38] P. Hovland, B. Norris, B. Smith, Making automatic differentiation truly automatic: Coupling PETSc with ADIC, Future Generation Computer Systems 21 (8) (2005) 1426–1438.
[39] P. M. Carrica, J. Huang, R. Noack, D. Kaushik, B. Smith, F. Stern, Toward large-scale computations of ship motions with dynamic overset curvilinear grids, in: Proceedings of the 27th Symposium on Naval Hydrodynamics, Seoul, Korea, 2008.
[40] P. M. Carrica, J. Huang, R. Noack, D. Kaushik, B. Smith, F. Stern, Large-scale DES computations of the forward speed diffraction and pitch and heave problems for a surface combatant, Computers and Fluids 39 (7) (2010) 1095–1111.
[41] R. T. Mills, V. Sripathi, G. Mahinthakumar, G. Hammond, P. C. Lichtner, B. F. Smith, Engineering PFLOTRAN for scalable performance on Cray XT and IBM BlueGene architectures, in: Proceedings of SciDAC 2010 Annual Meeting, 2010.
[42] P. Brune, M. Knepley, B. Smith, X. Tu, Composing scalable nonlinear solvers, Preprint ANL/MCS-P2010-0112, Argonne National Laboratory (2012).
[43] C.-J. Lin, J. Moré, Newton's method for large bound-constrained optimization problems, SIAM Journal on Optimization 9 (4) (1999) 1100–1127.
[44] S. Dirkse, M. Ferris, T. Munson, PATH Web page, http://pages.cs.wisc.edu/~ferris/path.html.
[45] H. Gómez, V. Calo, Y. Bazilevs, T. Hughes, Isogeometric analysis of the Cahn-Hilliard phase-field model, Computer Methods in Applied Mechanics and Engineering 197 (49-50) (2008) 4333–4352.
[46] H. Gomez, T. Hughes, X. Nogueira, V. Calo, Isogeometric analysis of the isothermal Navier-Stokes-Korteweg equations, Computer Methods in Applied Mechanics and Engineering 199 (25-28) (2010) 1828–1840.
[47] N. Nguyen, J. Peraire, B. Cockburn, Hybridizable discontinuous Galerkin methods, Spectral and High Order Methods for Partial Differential Equations (2011) 63–84.
[48] X.-C. Cai, D. E. Keyes, Nonlinearly preconditioned inexact Newton algorithms, SIAM J. Sci. Comput. 24 (2002) 183–200.
[49] G. Biros, O. Ghattas, Parallel Lagrange-Newton-Krylov-Schur methods for PDE constrained optimization. Part I: The Krylov-Schur solver, SIAM Journal on Scientific Computing 27 (2) (2005) 687–713.
[50] G. Biros, O. Ghattas, Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. part II: The Lagrange Newton solver, and its application to optimal control of steady viscous flows, SIAM Journal on Scientific Computing 27 (2) (2005) 714–739.
[51] E. Prudencio, R. Byrd, X. C. Cai, Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems, SIAM J. Sci. Comp. 27 (2006) 1305–1328.
[52] S. Mishra, C. Schwab, J. Šukys, Multi-level Monte Carlo finite volume methods for nonlinear systems of conservation laws in multi-dimensions, Journal of Computational Physics 231 (8) (2012) 3365–3388. doi:10.1016/j.jcp.2012.01.011.