Contemporary Mathematics

Terascale Implicit Methods for Partial Differential Equations

David E. Keyes

1. Introduction

Computational peak performance on full-scale scientific applications, as tracked by the Gordon Bell prize, has increased by four orders of magnitude since the prize was first awarded in 1988 — twenty-five times faster than can accounted for by Moore's Law alone. The extra factor comes from process concurrency, which is as much as 8,192-fold on the $100M "ASCI White" machine at Lawrence Livermore, currently ranked as the world's second most powerful, after the new Japanese "Earth Simulator". The latter was recently clocked at more than 35 trillion floating point operations per second (Tflop/s) on the LINPACK benchmark and at more than 25 Tflop/s on a climate application. Though architectural concurrency is easy to achieve, algorithmic concurrency to match is less so in scientific codes. Intuitively, this is due to global domains of influence in many problems presented to the computer as implicitly discretized operator equations — implicitness being all but legislated for the multiscale systems of global climate, transonic airliners, petroleum reservoirs, tokamaks, etc., the simulation of which justifies expenditures for the highest-end machines.

For many years, we have been pursuing optimal parallel algorithms for PDE simulations through the Jacobian-free Newton-Krylov methodology, preconditioned with Schwarz and Schur decompositions, including multilevel generalizations of the former. One of the main benefits of the Jacobian-free Newton-Krylov approach is the exploitation of multiple discrete representations of the underlying continuous operator, the idea being to converge fully to a representation of high fidelity through a series of inexpensive and stable steps based on representations of lower fidelity. Simultaneous advances in object-oriented software engineering have enabled the construction of internally complex software systems in which these algorithmic elements can be combined modularly, recursively, and relatively efficiently in parallel, while presenting a programming environment that allows the user to function at

1991 *Mathematics Subject Classification.* Primary 65N55; Secondary 65Y05, 65N30.

Supported in part by the U.S. Department of Energy under SciDAC subcontract FC02-01ER25476 to Old Dominion University, by Lawrence Livermore National Laboratory under ASCI Level-2 subcontract B347882 to Old Dominion University, and in part by NASA under contract NAS1-19480 to ICASE.

a rather high level. Since our work is applications driven, we have also been concerned with the development of robustification techniques for large systems with strong nonlinearities, including pseudo-transient continuation, parameter continuation, grid sequencing, model sequencing, and nonlinear preconditioning. Recently, we have helped show how some large-scale PDE-constrained optimization problems (e.g., design, control, parameter identification — usually the ultimate problems behind the proximate PDEs) can be placed into the same rootfinding algorithmic framework as the PDE, itself.

Terascale simulation of systems based on PDEs is a technology in demand by a variety of federal and industrial organizations. It has been well said (by R. W. Hamming) that, "The purpose of computing is insight, not numbers." Increasingly, however, the purpose of computing is *numbers* — numbers accurate enough to base policy decisions and hardware investments upon! Analytical investigations are limited in applicability to relatively simple systems. Experimental approaches are often controversial (e.g., in biology), dangerous (e.g., in global climate), prohibited (e.g., in nuclear weapons techology), impossible to perform (e.g., in cosmology), difficult to instrument (e.g., in automotive safety), or simply expensive (e.g., in high energy physics). In these, and in many other areas, computational simulation has become a critical companion to experiment. However, simulation is far from a reliable investigative methodology in problems of multiple spatial or temporal scales. It is not yet routinely possible to work with systems that are represented by a number of degrees of freedom in excess of billions. Large computational platforms have been provided, capable of multiple teraflop/s, but these systems often perform at a small fraction (less than 10%; sometimes closer to 1%) of their peak on production scientific workloads.

The architecture of the terascale systems available in the United States, built around hierarchical distributed memory, appears hostile to conventional sequential optimal PDE algorithms in some respects, but is ultimately suitable apart from reservations about memory bandwidth. The distributed aspects must be overcome with judicious combinations of message-passing and/or shared memory program models. The hierarchical aspects must be overcome with register blocking, cache blocking, and prefetching. Algorithms for these PDE-based simulations must be highly concurrent, straightforward to load balance, latency tolerant, cache friendly (with strong temporal and spatial locality of reference), and highly scalable (in the sense of convergence rate) as problem size and processor number are increased in proportion. The goal for algorithmic scalability is to fill up the memory of arbitrarily large machines while preserving constant (or at most logarithmically growing) running times with respect to a proportionally smaller problem on one processor. Domain-decomposed multilevel methods are natural for all of these *desiderata*. Domain decomposition is also natural for the software engineering of simulation codes, although it is not within the scope of this chapter to examine this last issue in deserved detail. Valuable extent code designed for a sequential PDE analysis can often be "componentized" and made part of an effective domain-decomposed, operator-split preconditioner.

A key requirement of candidate solution algorithms is mathematical optimality. This means a convergence rate as independent as possible of discretization parameters. In practice, linear systems require a hierarchical, multilevel approach

to obtain rapid linear convergence. Nonlinear systems require a Newton-like approach to obtain asymptotically quadratic convergence. The concept of optimality can also be extended into the physical modeling regime to include continuation schemes and physics-informed preconditioning, so that multiple scale problems are attacked with a manageable number of scales visible to the numerics at any given stage. Ill-conditioning is often directly related to the presence of a wide range of scales.

In this chapter, we review some of the basic algorithmic progress mentioned above for scalable nonlinearly implicit solution of PDEs. Our approach is nonrigorous, this being justified on one hand by limitations of space and scope, and on the other by its recent success in many PDE-based "Grand Challenge" problems — "the proof of the pudding is in the eating." We hope that computational scientists and engineers will find helpful paradigms for their own work and that mathematicians will find gaps in the presentation that are worthy of theoretical examination.

Often, practicing computational modelers bring considerable insight from the physics of a problem into the numerics, through various types of operator splitting, in which phenomena arising from balances between subsets of the terms in the full model are isolated. It behooves the designers of practical algorithms and software implementations thereof to note and seek to accommodate such "tricks". Similarly, computer scientists possess architectural knowledge that ought to influence the design of algorithms. Floating point operations on operands that are locally cached are extremely cheap, whereas summoning data values from remote memories can be the principal bottleneck of the computation. We sometimes bow to the constraints of the architecture without mathematical justification, and let the overall execution time, a combination of convergence rate and cost per step, be the arbiter of what works best.

The organization of this chapter is as follows. Section 2 describes the algorithmic core, Newton-Krylov-Schwarz (NKS), as well as Schur preconditioning, a domain-decomposed alternative to Schwarz. Various extensions to this core, related to the large-scale numerical analysis of nonlinear PDEs, are described in Section 3. Section 4 makes a brief foray into the field of optimization subject to large-scale PDE-based constraints. This is built upon Schur preconditioning applied algebraically, with Schwarz applied inside of Schur in the usual subdomain-by-subdomain sense. Section 5 briefly highlights a Gordon Bell prize-winning computation based on Newton-Krylov-Schwarz algorithmics. This has been thoroughly documented elsewhere especially in its aspect of high performance, but is summarized here to emphasize the practical rewards of an NKS approach. Finally, in Section 6 we describe the major goals of a five-year, nine-institution project, "Terascale Optimal PDE Simulations," which is one of the seven "Integrated Software Infrastructure Centers" of the DOE's "Scientific Discovery through Advanced Computing" (SciDAC) initiative, launched in 2001. This project is built on top of a significant base of existing software, including the PETSc library from Argonne National Laboratory and the Hypre library from Lawrence Livermore National Laboratory. To the extent that this chapter serves as a readable bridge to some of the functions of these freely available packages, which are becoming increasingly interoperable and extensible, it will serve its main purpose. It is, in effect, a second installment of an evolving "metapaper," the first installment of which [**61**] it liberally recycles for background.

2. The Newton-Krylov-Schwarz Family of Algorithms

To illustrate the algorithmic methodologies of this section, we consider a system of partial differential equations that can be written in the form

$$V\frac{\partial \mathbf{u}}{\partial t} + \mathcal{F}(\mathbf{u}) = 0, \tag{1}$$

where $\mathbf{u}$ is a vector of functions depending upon spatial variables $\mathbf{x}$ and t, $\mathcal{F}$ is a vector of spatial differential operators acting on $\mathbf{u}$, and V is a diagonal scaling matrix with nonnegative diagonal entries. If all of the equations are "prognostic" then V has all positive diagonal entries; but we may also accommodate the case of entirely steady-state equations, $V = 0$, or some combination of positive and zero diagonal entries, corresponding to prognostic equations for some variables and steady-state constraints for others. Many problems in engineering and applied physics are so formulated, where the steady-state equations often arise from *a priori* assumptions designed to suppress timescales faster than those of dynamical interest, e.g., acoustic waves in aerodynamics, gravity waves in geophysics, Alfvén waves in magnetohydrodynamics, etc.

Semidiscretizing in space to approximate $\mathcal{F}(\mathbf{u})$ with $\mathbf{f}(\mathbf{u})$, and in time with implicit Euler, we get the algebraic system:

$$(\frac{V}{\tau^\ell})\mathbf{u}^\ell + \mathbf{f}(\mathbf{u}^\ell) = (\frac{V}{\tau^\ell})\mathbf{u}^{\ell-1}. \tag{2}$$

Higher-order temporal schemes are easily put into this framework with the incorporation of addition history vectors with appropriate weights on the right-hand side. We are not much directly concerned with discretization or the adaptivity of the discretization to the solution this chapter. However, much of what we pursue algebraically is motivated by a desire to go to higher order than the pervasive standard of no better than first-order in time (when dealing with unsteady phenomena) and second-order in space.

Because $\mathbf{f}$ may be highly nonlinear, even a steady-state numerical analysis is often made to follow a pseudo-transient continuation until the ball of convergence for Newton's method for the steady-state problem is reached. In this case, time accuracy is not an issue, and τ^ℓ becomes a parameter to be placed at the service of the algorithm. Generally, a sequence of problems is solved with $\tau^\ell \to \infty$ as $\ell \to \infty$, $\ell = 1, 2, \ldots$. In this exposition, we lay aside all formal questions of existence and uniqueness of such steady states, with the recognition that practicing engineers and scientists are already "solving" such problems and know for which of the (possibly many) solutions they are looking. For a more careful look at the dangers attendant to well-posedness of pseudo-transient continuation, see [**47**].

Whether discretized time accurately or not, we are left at each time step with a system of nonlinear algebraic equations (2), written abstractly as $\mathbf{F}^\ell(\mathbf{u}^\ell) = 0$. We solve these systems in sequence for each set of discretized spatial gridfunctions, $\mathbf{u}^\ell$, using an inexact Newton method. (Methods that generalize this lockstep approach for advancing all the gridfunctions in time, or that possess parallelism spanning multiple time steps are not uninteresting, but beyond the scope of this chapter.) The resulting linear systems for the Newton corrections involving the Jacobian of $\mathbf{F}^\ell$ with respect to instantaneous or lagged iterates $\mathbf{u}^{\ell,k}$, are solved with a Krylov method, relying only on Jacobian-vector multiplications. (Here, $\mathbf{u}^{\ell,0} \equiv \mathbf{u}^{\ell-1}$, and $\mathbf{u}^{\ell,k} \to \mathbf{u}^\ell$, as $k \to \infty$ in a Newton iteration loop on inner index k.) The Krylov

method needs to be preconditioned for acceptable inner iteration convergence rates, and the preconditioning is the "make-or-break" aspect of an implicit code. The other phases possess high concurrency and parallelize well already, if properly load balanced, being made up of vector updates, inner products, and sparse matrix-vector products.

The job of the preconditioner is to approximate the action of the Jacobian inverse in a way that does not make it the dominant consumer of memory or cycles in the overall algorithm and (most importantly) does not introduce idleness through chained data dependencies, as in Gaussian elimination. The true inverse of the Jacobian is usually dense, reflecting the global Green's function of the continuous linearized PDE operator it approximates, and it is not obvious that a good preconditioner approximating this inverse action can avoid extensive global communication. A good preconditioner saves time and space by permitting fewer iterations in the Krylov loop and smaller storage for the Krylov subspace than would be required in its absence. An additive Schwarz preconditioner accomplishes this in a localized manner, with an approximate solve in each subdomain of a partitioning of the global PDE domain. Applying any subdomain preconditioner within an additive Schwarz framework tends to increases floating point rates over the same preconditioner applied globally, since the smaller subdomain blocks maintain better cache residency. Combining a Schwarz preconditioner with a Krylov iteration method inside an inexact Newton method leads to a synergistic parallelizable nonlinear boundary value problem solver with a classical name: Newton-Krylov-Schwarz (NKS). In the remainder of this section, we build up NKS from the outside inwards.

2.1. Inexact Newton Methods. We use the term "inexact Newton method" to denote any nonlinear iterative method for solving $\mathbf{F}(\mathbf{u}) = 0$ through a sequence $\mathbf{u}^k = \mathbf{u}^{k-1} + \lambda^k \delta\mathbf{u}^k$, where $\delta\mathbf{u}^k$ approximately satisfies the true Newton correction equation

$$\mathbf{F}'(\mathbf{u}^{k-1})\delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^{k-1}), \tag{3}$$

in the sense that the linear residual norm $||\mathbf{F}'(\mathbf{u}^{k-1})\delta\mathbf{u}^k + \mathbf{F}(\mathbf{u}^{k-1})||$ is sufficiently small. Typically the right-hand side of the linear Newton correction equation, which is the nonlinear residual $\mathbf{F}(\mathbf{u}^{k-1})$, is evaluated to full precision, so the inexactness arises from incomplete convergence employing the true Jacobian, freshly evaluated at $\mathbf{u}^{k-1}$, or from the employment of an inexact Jacobian for $\mathbf{F}'(\mathbf{u}^{k-1})$. Typical choices would be: (1) an implicitly defined operator whose action on a vector is constructed from multivariate Taylor expansions of $\mathbf{F}$ or from automatic differentiation software; (2) a matrix whose elements are constructed from finite differences of $\mathbf{F}$; (3) a matrix that is lagged (or some of whose elemental terms are lagged) from the evaluation at some previous state $\mathbf{u}^{k'}$, $k' < k-1$; (4) a matrix derived from a discretization related to, but not the same as, that used for $\mathbf{F}$, itself; or (5) a matrix that has been simplified by omission of elements that are inconvenient to a particular storage scheme or approximate parallelizable inversion process.

The first choice arises in the context of Krylov methods and leads to the nomenclature "Jacobian-free Newton-Krylov" [**82**] since the matrix elements themselves are never formed. The latter four possibilities for economizing on the Jacobian are common in the construction of a preconditioner for this matrix-free forward Jacobian action. In other contexts, the third possibility above is referred to as a

"modified Newton method" [**118**]. The latter two we regard as so inexact that they are demoted to "defect correction methods" [**77**].

In the the first method above, the Jacobian-vector action may be approximated by two or more function evaluations (including that of $\mathbf{F}(\mathbf{u}^{k-1})$, itself) in a discrete analog of Fréchet derivatives of smooth functions, for example, $\mathbf{F}'(\mathbf{u})\mathbf{v} \approx \frac{1}{h}[\mathbf{F}(\mathbf{u} + h\mathbf{v}) - \mathbf{F}(\mathbf{u})]$. For this first-order approximation, the cost for both the residual and the Jacobian-vector product is two residual evaluations. The parameter h is chosen to balance approximation error and subtractive floating point cancellation error.

Another approximate Jacobian-vector product derived from the same multivariate Taylor expansion that underlies the finite-difference approximations above, which is, however, free of subtractive cancellation error, has recently been rediscovered and popularized [**133**]. It features an imaginary perturbation,

$$\mathbf{F}(\mathbf{u} + ih\mathbf{v}) = \mathbf{F}(\mathbf{u}) + ih\mathbf{F}'(\mathbf{u})\mathbf{v} + c_2\mathcal{O}(h^2) + ic_3\mathcal{O}(h^3),$$

about any point $\mathbf{u}$, where $\mathbf{F}$, interpreted as a complex function of a complex-valued argument, is assumed analytic. Here, $\mathbf{u}$ and $\mathbf{v}$ are real. When $\mathbf{F}$ is real for real argument — the usual case for physical situations, except when time-harmonicity has been exploited in reducing the unsteady problem — all quantities except for i in the expansion above are also real; therefore, by extracting real and imaginary parts, we can identify $\mathbf{F}(\mathbf{u}) = \mathrm{Re}[\mathbf{F}(\mathbf{u} + ih\mathbf{v})] + \mathcal{O}(h^2)$ and $\mathbf{F}'(\mathbf{u})\mathbf{v} = \mathrm{Im}[\mathbf{F}(\mathbf{u} + ih\mathbf{v})]/h + \mathcal{O}(h^2)$. Special care is needed for discretizations with flux limiters and any other nondifferentiable features of $\mathbf{F}(\mathbf{u})$, but with minor code alterations, both $\mathbf{F}(\mathbf{u})$ and $\mathbf{F}'(\mathbf{u})\mathbf{v}$ are available without subtraction from a single complex evaluation of $\mathbf{F}(\mathbf{u})$. The cost of complex arithmetic in the residual evaluation is approximately four times that of real arithmetic. (This assumes an equal number of multiplies and adds in the evaluation of $\mathbf{F}(\mathbf{u})$ and equal cost for each. Complex multiplies require six real floating operations and adds two.) Implications for evaluation of sensitivity derivatives by this technique are explored in [**101**].

Alternatively, using a package such as ADIFOR [**13**] or ADIC [**14**], the Jacobian vector product can be evaluated simultaneously with $\mathbf{F}$ at a cost approximately equal to 2.5 times the cost of the residual, itself. This ratio has been improving with the sophistication of automatic differentiation (AD) software technology to the point where AD now provides poses the best trade-off between accuracy and cost for Jacobian-vector products.

In method (2) above, the approximate Jacobian is explicitly constructed, element-by-element, from a sequence of finite differences (usually chosen with the aid of graph coloring on the discrete stencil), each of which supplies one or more columns of $\mathbf{F}'(\mathbf{u})$, for example, $[\mathbf{F}'(\mathbf{u})]_{ij} = \frac{\partial \mathbf{F}_i}{\partial \mathbf{u}_j}(\mathbf{u}) \approx \frac{1}{h}[\mathbf{F}_i(\mathbf{u} + h\mathbf{e}_j) - \mathbf{F}_i(\mathbf{u})]$, where h is a differencing parameter and $\mathbf{e}_j$ is the j^{th} unit vector.

Inexact Newton methods require a strategy for terminating the inner linear iterations, in effect choosing tolerance η_k, in

$$||\mathbf{F}(\mathbf{u}^{k-1}) + \mathbf{F}'(\mathbf{u}^{k-1})(\mathbf{u} - \mathbf{u}^{k-1})|| \leq \eta_k ||\mathbf{F}(\mathbf{u}^{k-1})|| \ . \tag{4}$$

One of the Eisenstat-Walker [**50**] criteria is

$$\eta_k = \frac{\left|\ ||\mathbf{F}(\mathbf{u}^{k-1})|| - ||\mathbf{F}(\mathbf{u}^{k-1}) + \mathbf{F}'(\mathbf{u}^{k-1})(\mathbf{u} - \mathbf{u}^{k-1})||\ \right|}{||\mathbf{F}(\mathbf{u}^{k-1})||}. \tag{5}$$

Ajmani et al. [**1**] adopt: $\eta_k = \frac{c}{\log \tau^k}$. Venkatakrishnan & Mavriplis [**130**] adaptively choose η_k so that work and storage per Newton step are bounded at some fixed expenditure.

The first strategy is theoretically elegant and appears to assure the minimum linear iteration work consistent with a guaranteed asymptotically superlinear convergence. We are satisfied with it in some contexts. However, as η_k contracts with convergence according to (3) the resulting stringent linear convergence requirements may be difficult to meet in large, ill-conditioned problems. Moreover, superlinear nonlinear convergence may be too expensive a goal in practice, where the objective is to minimize execution time rather than number of inexact Newton steps.

The strategy of bounding work and storage per Newton step seems common in highly resolved problems whose memory requirements approach the maximum available, since the size of linear algebra workspaces may grow with k.

Experience shows that a hybrid approach is the often most cost-effective. Such an approach involves an initially loose convergence criterion (to avoid "oversolving," in the sense of [**50**]) evolving to a tighter criterion, but subject to a linear-work-per-step bound (in the sense of [**130**]). For problems sufficiently small and well conditioned (linearly) to apply (5), overall cost is not as important.

Historical remarks. The pioneering work of Dembo, Eisenstat, & Steihaug [**44**] showed that properly tuned inexact Newton methods can save enormous amounts of work (through approximating the Newton corrections, which can in turn permit approximation of the Jacobian matrix) over a sequence of Newton iterations, while still converging quadratically. This theory was revisited to provide inexpensive, constructive formulae for the sequence of inexact tolerances by Eisenstat & Walker [**50**]. Smooke [**118**] and Schreiber & Keller [**115**] devised Newton-chord methods with models for cost-effective frequency of Jacobian reevaluation. The use of various approximate Newton methods in computational fluid dynamics (CFD) emerged independently in various regimes. Vanka [**128**] implemented Newton solvers in primitive-variable Navier-Stokes problems. Venkatakrishnan [**129**], Orkwis [**105**], and Whitfield & Taylor [**132**] established Newton-like methods in difficult transonic problems. These works employed various direct or stationary iterative methods for the linear Newton correction equation, and were based on explicit matrix representations of the Jacobian operator.

2.2. Newton-Krylov Methods. A Newton-Krylov (NK) method uses a Krylov method, such as GMRES [**113**], to solve (3) for $\delta \mathbf{u}^\ell$. From a computational point of view, one of the most important characteristics of a Krylov method for the linear system $Ax = b$ is that information about the matrix A needs to be accessed only in the form of matrix-vector products in a relatively small number of carefully chosen directions. When the matrix A represents the Jacobian of a discretized system of PDEs, each of these matrix-vector products is similar in computational and communication cost to a stencil update phase (or "global flux balance") of an explicit method applied to the same set of discrete conservation equations or to a single finest-grid "work unit" in a multigrid method. NK methods are suited for nonlinear problems in which it is unreasonable to compute or store a true full Jacobian, where the action of A can be approximated by discrete directional derivatives.

Some Krylov methods for nonsymmetric problems require matrix-vector products with A^T as well as A [**112**]. It does not seem possible to approximate the action of A^T from finite differences of the original function evaluation, though it *is*

possible from the latest versions of automatic differentiation packages, employing the so-called "reverse mode" [**55**]. The reverse mode computation of Jacobian-transpose-vector products is generally more expensive than the forward mode for the Jacobian-vector product, but is practically affordable in some PDE contexts. Other nonsymmetric Krylov solvers, such as CGS [**120**], BiCGSTAB [**127**], and TFQMR [**53**] could be substituted for GMRES and converge about as well in terms of the total number of matrix-vector products. In our experience with model problems (see, e.g., [**76**]), most such methods employ two matrix-vector products per step and converge in about half as many steps. It should be borne in mind, however, that their behaviors can differ wildly, and in nonuniformly rankable ways, for specially chosen problems [**100**]. Our experience with such solvers in the Jacobian-free NK context is less favorable than that with GMRES. They have the advantage of requiring less memory, and the important potential of requiring fewer global reductions (for inner products), but the disadvantage of nonmonotonic and, in some cases, wildly oscillating residual norm histories, leading to decreased numerical stability. Another advantage of GMRES is its use of matrix-vector products with unit-norm vectors $\mathbf{v}$, which tend to be well suited for finite-difference approximations involving scale-sensitive perturbations, for example, $\mathbf{F}(\mathbf{u} + h\mathbf{v})$ [**93**].

We advocate using NK in a split-discretization formulation, in which economizations are taken in the left-hand side preconditioner blocks of J relative to the more accurate, physical discretization-dictated right-hand operator for J. Examples of such economizations include: sacrificed coupling for process concurrency [**78**], segregation of physics into successive phases with simple structure (operator-splitting) [**7**], the Jacobian of a lower-order discretization for fewer nonzeros and fewer colors in a minimal coloring [**71**], the Jacobian of a related discretization allowing "fast" solves [**20**], a Jacobian with lagged values for any terms that are expensive to compute or small or both, and a Jacobian stored in half precision for superior (nearly doubled) memory bandwidth, as measured in words per second, in the bandwidth-limited linear algebra routines of a sparse, unstructured PDE solver [**60**].

Historical remarks. The advent of Krylov iterative methods (see, e.g., [**112**] for a survey) inside of inexact Newton iterations in a matrix-free context can be traced to the ODE-oriented papers of Gear & Saad [**54**], Chan & Jackson [**41**], and Brown & Hindmarsh [**19**] and the PDE-oriented work of Brown & Saad [**20**]. (The term "Newton-Krylov" seems first to have been applied to such problems in [**20**].) The GMRES [**113**] method was firmly established in CFD following the work of Wigton, Yu, & Young [**134**] and Johann, Hughes, & Shakib [**67, 116**]. Venkatakrishnan & Mavriplis showed in [**130**] that NK methods (preconditioned with a global incomplete factorization) were competitive with multigrid methods for large-scale CFD problems; a similar comparison for the matrix-free form of such methods was given by Keyes in [**77**]. Various practical aspects of NK methods in CFD were explored in [**1, 8, 66, 93, 92, 102, 111, 121, 123**]. Newton-Krylov has been demonstrated to be an effective implicit solver for large-scale nonlinear problems derived from PDEs (see, e.g., P. Brown and collaborators at LLNL [**21, 68**] and D. Knoll and collaborators at LANL [**88, 94**]). It has been applied to problems in aerodynamics, radiation transport, porous media, semiconductors, geophysics, astrophysical

MHD, population dynamics, and other fields. It has been implemented in a parallel matrix-free object-oriented framework, including both FD and AD distributed matvecs, in PETSc software from Argonne [**4**].

2.3. Newton-Krylov-Schwarz Methods. A Newton-Krylov-Schwarz (NKS) method combines a Newton-Krylov method, such as nonlinear GMRES [**134**], with a Krylov-Schwarz (KS) method, such as restricted additive Schwarz [**36**]. If the Jacobian A is ill-conditioned, the Krylov method will require an unacceptably large number of iterations. In order to control the number of Krylov iterations, while obtaining concurrency proportional to the number of processors, we precondition them with domain-decomposed additive Schwarz methods [**117**]. The system can be transformed into the equivalent form $B^{-1}Ax = B^{-1}b$ through the action of a preconditioner, B, whose inverse action approximates that of A, but at smaller cost. It is in the choice of preconditioning that the battle for low computational cost and scalable parallelism is usually won or lost. In KS methods, the preconditioning is introduced on a subdomain-by-subdomain basis through a conveniently computable approximation to a local Jacobian. Such Schwarz-type preconditioning provides good data locality for parallel implementations over a range of parallel granularities, allowing significant architectural adaptability.

Schwarz methods [**23, 48, 117, 136**] create concurrency at a desired granularity algorithmically and explicitly through partitioning, without the necessity of any code dependence analysis or special compiler. Generically, in continuous or discrete settings, Schwarz partitions a solution space into n subspaces, possibly overlapping, whose union is the original space, and forms an approximate inverse of the operator in each subspace. Algebraically, to solve the discrete linear system, $Ax = f$, let Boolean rectangular matrix R_i extract the i^{th} subset of the elements of x defining an algebraic subspace: $x_i = R_i x$, and let $A_i \equiv R_i A R_i^T$ be invertible within the i^{th} subspace. Then the Schwarz approximate inverse, B^{-1}, is defined as $\sum_i R_i^T A_i^{-1} R_i$. From the PDE perspective, subspace decomposition is domain decomposition. We form $B^{-1} \approx A^{-1}$ out of (approximate) local solves on (possibly overlapping) subdomains, as in Figure 1. This can be used to iterate in a stationary way, as a splitting matrix: $x^k = (I - B^{-1}A)x^{k-1} + B^{-1}f$. However, since $\rho(I - B^{-1}A)$ may be greater than unity in general, this additive splitting may not converge as a stationary iteration. "Multiplicative" Schwarz methods (Gauss-Seidel-like, relative to the Jacobi-like "additive" above) can be proved convergent when A derives from an elliptic PDE, under certain partitionings.

In the grid-based context of a PDE, Boolean operators R_i and R_i^T, $i = 1, \ldots, n$, represent gather and scatter (communication) operations, mapping between a global vector and its i^{th} subdomain support. When A derives from an elliptic operator and R_i is the characteristic function of unknowns in a subdomain, optimal convergence (independent of $\dim(x)$ and the number of partitions) can be proved, with the addition of a coarse grid, which is denoted with subscript "0": $B^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i>0} R_i^T A_i^{-1} R_i$ [**23**]. Here, R_0 is a conventional geometrically based multilevel interpolation operator. It is an important freedom in practical implementations that the coarse grid space need not be related to the fine grid space or to the subdomain partitioning.

The $A_i^{-1}(i > 0)$ in B^{-1} are often replaced with inexact solves in practice. The exact forward matrix-vector action of A in $B^{-1}A$ is still required, even if inexact solves are employed in the preconditioner.

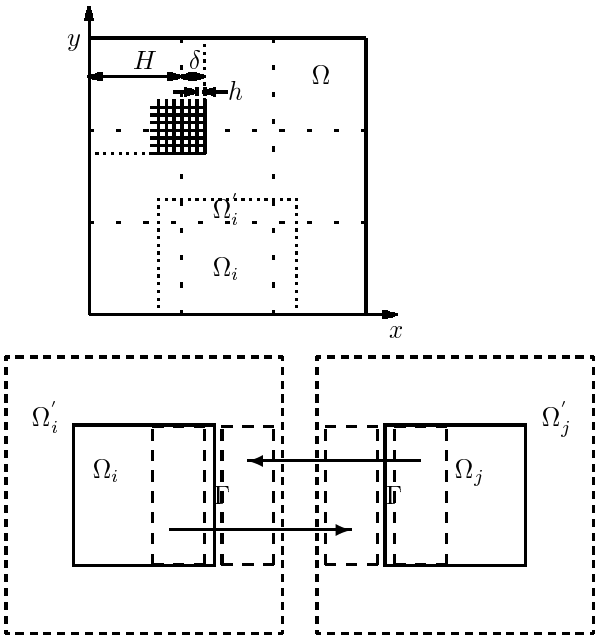

FIGURE 1. **Upper:** A domain Ω partitioned into nine overlapping subdomains, Ω_i, extended slightly by overlapping to subdomains Ω_i', showing the scales of the mesh spacing (h), the subdomain overlap (δ), and the subdomain diameter (H). **Lower:** Two adjacent subdomains with common edge Γ pulled apart to show overlap regions as separate buffers, which are implemented in the local data structures of each.

Condition number estimates for $B^{-1}A$ are given in the first column of Table 1. The two-level Schwarz method with generous overlap has a condition number that is independent of the fineness of the discretization and the granularity of the decomposition, which implies perfect algorithmic scalability. However, there is an increasing implementation overhead in the coarse-grid solution required in the two-level method that offsets this perfect algorithmic scalability. In practice, a one-level method is often used, since it is amenable to a perfectly scalable implementation. Alternatively, a two-level method is used but the coarse level is solved only approximately, in a trade-off that depends upon the application and the architecture. These condition number results are extensible to nonself-adjointness, mild indefiniteness, and inexact subdomain solvers. The theory requires a "sufficiently fine" coarse mesh, H, for the first two of these extensions, but computational experience shows that the theory is often pessimistic.

We use the *restricted* additive Schwarz Method (RASM), which eliminates interprocess communication during the interpolation phase of the additive Schwarz technique [**36**]. In particular, if we decompose a problem into a set of possibly overlapping subdomains Ω_i, the conventional additive Schwarz method can be expressed as

$$M_{ASM}^{-1} = \sum_i {R_i}^T A_i^{-1} R_i, \tag{6}$$

where the three-phase preconditioning process consists of first collecting data from neighboring subdomains via global-to-local restriction operators R_i, then performing a local linear solve on each subdomain A_i^{-1}, and finally sending partial solutions to neighboring subdomains via the local-to-global prolongation operators ${R_i}^T$. The RASM preconditioner is expressed in operator notation as

$$M_{RASM}^{-1} = \sum_i {R_i'}^T A_i^{-1} R_i. \tag{7}$$

It performs a complete restriction operation but does not use any communication during the interpolation phase, ${R_i'}^T$. This provides the obvious benefit of a 50% reduction in nearest-neighbor communication overhead. In addition, experimentally, it preconditions better than the original additive Schwarz method over a broad class of problems [**36**], for reasons that are beginning to be understood [**26**].

Although the spectral radius, $\rho(I - B^{-1}A)$, may exceed unity, the spectrum, $\sigma(B^{-1}A)$, is profoundly clustered, so Krylov acceleration methods should work well on the preconditioned solution of $B^{-1}Ax = B^{-1}f$. Krylov-Schwarz methods typically converge in a number of iterations that scales as the square-root of the condition number of the Schwarz-preconditioned system. For convergence scalability estimates, assume one subdomain per processor in a d-dimensional isotropic problem, where $N = h^{-d}$ and $P = H^{-d}$. Then iteration counts may be estimated as in the last two columns of Table 1.

TABLE 1. Theoretical condition number estimates $\kappa(B^{-1}A)$, for self-adjoint positive-definite elliptic problems [**117**] and corresponding iteration count estimates for Krylov-Schwarz based on an idealized isotropic partitioning of the domain in dimensions 2 or 3.

Preconditioning	$\kappa(B^{-1}A)$	*2D Iter.*	*3D Iter.*
Point Jacobi	$\mathcal{O}(h^{-2})$	$\mathcal{O}(N^{1/2})$	$\mathcal{O}(N^{1/3})$
Domain Jacobi	$\mathcal{O}((hH)^{-1})$	$\mathcal{O}((NP)^{1/4})$	$\mathcal{O}((NP)^{1/6})$
1-level Additive Schwarz	$\mathcal{O}(H^{-2})$	$\mathcal{O}(P^{1/2})$	$\mathcal{O}(P^{1/3})$
2-level Additive Schwarz	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

The proof of these estimates is generally approached via an algebra of projection operators, $P_i \equiv {R_i}^T A_i^{-1} R_i A$. The ratio of upper bound to lower bound of the spectrum of the sum of the orthogonal projections P_i is an estimate of the condition number for $M^{-1}A = \sum_i P_i$. Since $||P_i|| \leq 1$, the upper bound follows easily from the geometry of the decomposition and is a generally a constant related to the number of colors required to color the subdomains. The lower bound depends crucially upon a good partition of the solution space. Without a coarse subspace to support the solution at subdomain boundaries, the fine space contributions must fall rapidly to zero from finite values in the subdomain interiors, resulting in high H^1 "energy" inversely proportional to the overlap distance over which the solutions must decay.

For simple intuition behind this table consider the following: errors propagate from the interior to the boundary in steps that are proportional to the largest implicit aggregate in the preconditioner, whether pointwise (in N) or subdomainwise (in P). The use of overlap in going from Domain Jacobi to Additive Schwarz avoids

the introduction of high energy at near discontinuities at subdomain boundaries. The two-level method projects out low-wavenumber errors at the price of solving a global problem.

Only the two-level method scales perfectly in convergence rate (constant, independent of N and P), like a traditional multilevel iterative method [**17, 18, 62, 125**]. However, the two-level method shares with multilevel methods a nonscalable cost-per-iteration from the necessity of solving a coarse-grid system of size $\mathcal{O}(P)$. Unlike recursive multilevel methods, a two-level Schwarz method may have a rather fine coarse grid, for example, $H = \mathcal{O}(h^{1/2})$, which potentially makes it less scalable overall. Parallelizing the coarse grid solve is necessary. Neither extreme of a fully distributed or a fully redundant coarse solve is optimal, but rather something in between. When reuse is possible, storing a parallel inverse can be cost-effective [**126**].

When it appears additively in the Schwarz preconditioner, the coarse grid injects some work that potentially spoils the "single-program, multiple data" (SPMD) parallel programming paradigm, in which each processor runs an identical image over local data. For instance, the SPMD model would not hold if one subset of processors worked on the coarse grid problem concurrently to the others each working on subdomains. Therefore, in two-level SPMD implementations, other Schwarz preconditioner polynomials than the purely additive are used in practice. We may define a preconditioner that solves the fine subdomains concurrently in the standard way, and then assembles a new residual and solves the coarse grid in a separate phase. The map in this case is

$$u \leftarrow u + \sum_{i=1}^{n} A_i^{-1}(f - Au)$$

followed by

$$u \leftarrow u + A_0^{-1}(f - Au).$$

This leads to the method denoted "Hybrid II" in [**117**]:

$$B^{-1} = A_0^{-1} + (I - A_0^{-1}A)(\sum_{i=1}^{n} A_i^{-1}).$$

Of course, the subspace inverses are typically done approximately, as in the purely additive case.

Readers uncomfortable with the appearance of the Schwarz formula $A^{-1} \approx \sum_i {R_i}^T A_i^{-1} R_i$, implying that the inverse of the sum is well approximated by the sum of the inverses in subspaces, may benefit from recalling an exact result from eigenanalysis. Let $\{r_i\}_{i=1}^N$ be a complete set of orthonormal row (left) eigenvectors for an SPD matrix A. Then $r_i A = a_i r_i$, or $a_i = r_i A r_i^T$, for corresponding eigenvalues a_i. Then, we have the representation of A as a sum over subspaces,

$$A = \sum_{i=1}^{N} r_i^T a_i r_i,$$

and similarly for A^{-1},

$$A^{-1} = \sum_{i=1}^{N} r_i^T a_i^{-1} r_i = \sum_{i=1}^{N} r_i^T (r_i A r_i^T)^{-1} r_i.$$

This is nothing but the Schwarz formula! In practice, invariant subspaces are far too expensive to obtain for practical use of the Schwarz formula, and their basis vectors are general globally dense, resulting in too much storage and communication in forming restrictions and prolongations. Characteristic subspaces of subdomains, in contrast, provide locality and sparsity, but are not invariant upon multiplication by A, since the stencils overlap subdomain boundaries. Choosing good decompositions is a balance between conditioning and parallel complexity, in practice. It is a mathematical art that has attracted many great minds, and resulted in many near optimal algorithms — enough to continue to feed an annual conference with 14 volumes to date and a community whose collective homepage can be found at `http://www.ddm.org/`.

Historical remarks. The application of domain decomposition-based preconditioners to nonlinearly implicit CFD algorithms has been our focus for over a decade [**80**]. Cai's doctoral dissertation and derivatives [**23, 24, 25, 37, 38, 29**] extended overlapping Schwarz theory to the nonselfadjoint operators of convection-diffusion problems and demonstrated their optimality — even without the benefit of a coarse grid component — in the parabolic case, and to indefinite operators.

The term "Newton-Krylov-Schwarz" was coined in [**31**]. NKS methods have been taken up by Cai and collaborators [**27, 28, 30, 34**], Knoll and collaborators [**83, 86, 87, 93**], Pernice and collaborators [**108**], and Tidriri [**122, 124**], among many others. Schwarz-type domain decomposition methods have been extensively developed for finite difference/element/volume PDE discretizations over the past decade, as reported in the annual proceedings of the international conferences on domain decomposition methods, of which the most recent volume is [**43**].

One of the main contemporary motivations for domain decomposition methods is divide-and-conquer concurrency. Scalability studies based on dimensionless ratios of communication and computation parameters for message-passing aspects of domain-decomposed iterative methods appeared in [**56, 57**]. Recently, the motivation for domain decomposition even on sequential computers with deep memory hierarchies has become very apparent [**131**]. Now that Jacobian-explicit and Jacobian-free flavors of NKS are available in the PETSc [**6**] and AZTEC [**65**], NKSOL [**20**], NITSOL [**109**], and KINSOL [**64**] toolkits, its use is widespread.

2.4. Newton-Krylov-Schur Methods. For subsequent use in the context of PDE-constrained optimization, we consider an alternative to Schwarz preconditioning, called Schur preconditioning, which derives from partitioned Gaussian elimination.

In the domain decomposition context, Gaussian elimination can be employed to construct, by explicit condensation of the degrees of freedom on the interior of subdomains, lower-dimensional systems for degrees of freedom that act as separators between subdomains. In the literature of continuous differential operators, this is the Poincare-Steklov operator; in linear algebra, it is the Schur complement or capacitance matrix. Przemieniecki [**110**] provided an early formalization in mechanics, called "substructuring," and Kron [**90**] did the same for electrical networks, coining the term "diakoptics" (meaning "method of tearing"). Przemieniecki was surprisingly prescient in writing, "From past experiences [...], it is evident that some form of structural partitioning is usually necessary either because different methods of analysis are used on different structural components or because of the limitations imposed by the capacity of digital computers. Even when the next

generation of faster and larger digital computers becomes a well-established tool [...], it seems rather doubtful, because of the large number of unknowns, that the substructure displacement method of analysis would be wholly superseded by an overall analysis carried out on the complete structure" [**110**].

An equivalence can be demonstrated between the Schur and Schwarz approaches in simple cases, in the sense that the Schwarz iterates are produced when the subdomain interior solutions are reconstructed from the Schur iterates on the separator [**15, 40**]. This equivalence requires exact subdomain solves on each subdomain, a price which is usually relaxed when Schwarz is used in practice, in the interest of optimal complexity.

Schur methods are based on a partitioning of the grid that orders the separator nodes last. Let the degrees of freedom associated with the separators be denoted u_B and the unknowns associated with the substructures thus created be denoted u_I. This partition induces a permutation of the discrete system $Au = f$, as follows:

$$\begin{bmatrix} A_I & A_{IB} \\ A_{BI} & A_B \end{bmatrix} \begin{pmatrix} u_I \\ u_B \end{pmatrix} = \begin{pmatrix} f_I \\ f_B \end{pmatrix}.$$

A_I is block diagonal, with a block for each substructure. Note that because the separator nodes correspond to a physically lower dimensional operator, the algebraic dimension of A_B is small in a relative sense. The Schur complement $S \equiv A_B - A_{BI}A_I^{-1}A_{IB}$, arises formally from the factorization:

$$A = \begin{bmatrix} A_I & A_{IB} \\ A_{BI} & A_B \end{bmatrix} = \tag{8}$$

$$\begin{bmatrix} I & 0 \\ A_{BI}A_I^{-1} & I \end{bmatrix} \begin{bmatrix} A_I & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A_I^{-1}A_{IB} \\ 0 & I \end{bmatrix}.$$

Its significance is that once $Su_B = \hat{f_B} \equiv f_B - A_{BI}A_I^{-1}f_I$ is solved for u_B, the block diagonal system $A_I u_I = \hat{f_I} \equiv f_I - A_{IB}u_B$ may be solved for u_I.

Because the Schur complement system itself is expensive to construct, though its action on individual vectors is relatively inexpensive, Krylov iteration (e.g., conjugate gradients) is preferred to direct Gaussian elimination, if a good preconditioner can be derived. For the case of domain decomposition on elliptic boundary value problems, considerable theory from the continuous case can be used to construct optimal preconditioners, in the sense that the condition number of the preconditioned systems is independent of the discretization parameter for a simple interface.

For complex interfaces, it is useful to build up a Schwarz preconditioner for the Schur complement by means of adding together (in appropriate subspaces) the action of the simple interface preconditioners on overlapping partitions of the composite interface. In analogy with the Boolean restriction operators defined above for extracting subdomains from the full domain, let us define Boolean restriction operators for extracting blocks of the interfacial Schur complement corresponding to (possibly overlapping) subsets of the interface, Q_i, and let $S_i \equiv Q_i S Q_i^T$. Then, a preconditioner for the Schur complement in a so-called "Neumann-Neumann" algorithm can tentatively be written in the Schwarz form

$$M^{-1} = \sum_i D_i Q_i^T S_i^{-1} Q_i D_i.$$

Here, D_i is a diagonal weighting matrix with a diagonal value that is inverse to the number of times a degree of freedom appears in different overlapping interfacial subsets. In general, however, the S_i^{-1} may be singular, so a coarse space must be introduced that takes care of components of the vector being multiplied in the null space. A hybrid multiplicative-additive Schwarz preconditioner for the Schur complement can thereby be constructed from the expression for M^{-1} above and a coarse space inversion, one famous symmetrized form of which was named the "balancing Neumann-Neumann" preconditioner by Mandel [**49, 91**]. A disadvantage of all methods that work directly with the Schur complement is the need to perform exact subdomain solves to form its action. In Neumann-Neumann algorithms, two such solves are performed per subdomain per iteration, in addition to the two coarse solves performed sequentially with respect to the subdomain solves. However, Neumann-Neumann methods, especially in the form of the Finite Element Tearing and Interconnection (FETI) algorithm [**10**] and the Primal-Dual Finite Element Tearing and Interconnection (FETI-DP) algorithm [**52**], have become the most important parallel methods in structural analysis today.

In [**79**], it was shown (for the symmetric, e.g., elliptic, case) that a Krylov iteration on the reduced Schur complement system, S, preconditioned with a given M^{-1}, is mathematically equivalent to the same Krylov iteration on the full system A, with M replacing S in the lower right corner of the diagonal matrix in (8), in the sense that iterates for u_B, suitably extended with backsolves for the eliminated variables u_I, are the same iterate-for-iterate as those of the original full system. This observation was used to motivate building a full system preconditioner from a good preconditioner for the Schur complement on the subdomain boundaries and *approximate* solvers in the subdomains. In this case, the inverse action of A_I never needs to be formed exactly, as it does when the interior degrees of freedom are explcitly eliminated. Since this action would otherwise be the term of leading computational complexity, a new class of preconditioners is created that remains in the full space but uses subspace components in an approximate block Gaussian elimination. This observation is at the heart of the methods discussed in Section 4.

A mathematical signature of the data-parallel Schwarz and Schur methods is a triple product of matrix operators consisting of the inverse of a square operator in the middle, with "rectangular" operators on either side that map between spaces of different dimensions. A Schur complement contains such triple products in which the middle term may be of higher dimension than the terms of the sum, itself. Roughly speaking, such a term is contributed for each subdomain adjacent to each separator segment, and the net action of all such terms on a vector defined over the "wirebasket" that remains after subdomain elimination can be evaluated with subdomain-scale concurrency. A Schwarz preconditioner contains such triple products in which the middle term is of lower dimension than the terms of the sum. Again, the net action of all such terms on a vector defined over the total domain can be evaluated with subdomain-scale concurrency. A key aspect of domain decomposition convergence theory is that, for many problem classes, the number of iterations required by the Krylov method (essentially, the number of times the preconditioned action has to be evaluated) depends only weakly on the granularity of the decomposition into subdomains. This leaves the parallel granularity at the disposal of architectural considerations.

2.5. Contrast of NKS with Defect Correction. A typical legacy nonlinear PDE code may employ a defect correction solver rather than Newton's method. To solve the sequence of nonlinear problems (2), a left-hand-side matrix (related to a Jacobian) is created in whose construction computational short-cuts are employed. For instance, in the context of convection, it may be stabilized by a degree of first-order upwinding that would not be acceptable in the discretization of the residual itself. We denote this generic distinction in the update equation (9) by subscripting the residual "high" and the left-hand-side matrix "low":

$$J_{low}\delta\mathbf{u} = -\mathbf{f}_{high}. \tag{9}$$

Often, J_{low} is based on a low-accuracy residual for $\mathbf{f}$:

$$J_{low} = \frac{V}{\tau} + \frac{\partial \mathbf{f}_{low}}{\partial \mathbf{u}}. \tag{10}$$

The inconsistency of the left- and right-hand sides prevents the use of large time steps, τ. Using J_{low} (or, more typically, some inexpensive approximation thereto, denoted $\tilde{J_{low}}$) as a preconditioner, we replace (9) with

$$(\tilde{J_{low}})^{-1}J_{high}\delta\mathbf{u} = -(\tilde{J_{low}})^{-1}\mathbf{f}_{high}, \tag{11}$$

in which the action of J_{high} on a vector is obtained through Fréchet or automatic differentiation using $\mathbf{f}_{high}$. Note that the operators on both sides of (11) are based on consistent high-order discretizations; hence, time steps can be advanced to arbitrarily large values, recovering a true Newton method in the limit.

From the viewpoint of linear convergence rate, it would seem ideal to use a preconditioner based on J_{high} in (11), but such a preconditioner can be much more expensive and memory-consumptive than one based on J_{low}. In (11), we have merely shifted the inconsistency from the nonlinear to the linear iteration. From the point of overall execution time, it is not obvious which is better: many inexpensive nonlinear iterations in which the inner linear problem (9) is preconditioned by J_{low}, or fewer more expensive nonlinear iterations containing the inner problem (11). The answer is strongly affected by the sequence of time steps employed in (10). When the parameter τ is small, J_{low} and J_{high} are both dominated by the same diagonal matrix, and the extra costs of working with J_{high} in the preconditioner may be unjustified.

The potential for (11) to outperform (9) is demonstrated in the CFD context in [**34, 77**]. In [**58**], the deterioration with advancing Courant-Friedrichs-Lewy (CFL) number of a solver based on approximate factorization of the operator in (10) is contrasted with an advancing CFL approach based on (11). Dimensionally split approximate factorization schemes also require low CFL number. In spite of this disadvantage in their rate of convergence to steady states, dimensionally split schemes continue to enjoy memory advantages over more implicit schemes, the fully matrix-free work of Qin, Ludlow, & Shaw [**111**] being a recent example. A range of options for J_{low} and J_{high} is explored in the context of CFD in [**66**]. The combination of choice for obtaining low-residual steady-state solutions (designated "ALLMUS" therein) corresponds to the use of J_{high} on the left-hand side, as well as the right, as in (11).

It should be borne in mind that the margin of superiority of (11) over less nonlinearly implicit schemes is very sensitive to the frequency of reevaluation (and refactorization) of J_{low} and to the intimate coupling of the optimal reevaluation

frequency with CFL advancement strategy and Krylov subspace size. Evaluation and refactorization of J_{low} are still expensive, comparable in arithmetic cost to the evaluation of f_{high} and typically more expensive in terms of communication. The frequency of J_{low} evaluation is a relatively neglected topic in the literature, since it is so problem dependent. An empirical sequential cost model is outlined in [**83**].

Other tuning parameters with a strong influence on the performance of (11) are those that define the difference between J_{low} and $\tilde{J_{low}}$. These include parameters defining incomplete factorization fill (whether position-based or threshold-based); relaxation or multilevel method parameters if the preconditioning is implemented by a number of sweeps of an iterative method; and, in the parallel context, the number of subdomains, subdomain overlap, the use of a coarse grid in the Schwarz method, and so forth. These algorithmic tuning choices are, in principle, amenable to systematic optimization with direct search methods [**46**] and should be explored before undertaking a series of "production" runs.

2.6. Contrast of Domain Decomposition with Other Decompositions. Before leaving this section, it is worthwhile to summarize the architectural advantages of Schwarz-type domain decomposition methods *vis-à-vis* other mathematically useful decompositions.

Given the operator equation

$$\mathcal{L}u = f \text{ in } \Omega$$

and a desire for either concurrent or sequential "divide-and-conquer," one can devise operator decompositions

$$\mathcal{L} = \sum_j \mathcal{L}_j,$$

function-space decompositions

$$f = \sum_j f_j \phi_j, \quad u = \sum_j u_j \phi_j,$$

or domain decompositions

$$\Omega = \cup_j \ \Omega_j.$$

Let us contrast an example of each on the parabolic PDE in two space dimensions

$$\frac{\partial u}{\partial t} + [\mathcal{L}_x + \mathcal{L}_y]u = f(x,y,t) \text{ in } \Omega, \tag{12}$$

with $u = 0$ on $\partial\Omega$, where $\mathcal{L}_x \equiv -\frac{\partial}{\partial x} a_x(x,y) \frac{\partial}{\partial x} + b_x(x,y) \frac{\partial}{\partial x}$, $(a_x > 0)$ and with a corresponding form for $\mathcal{L}_y$. Upon implicit time discretization

$$\begin{aligned} \left[\tfrac{I}{\Delta t} + \mathcal{L}_x + \mathcal{L}_y\right] u^{(\ell+1)} &= \left[\tfrac{I}{\Delta t}\right] u^{(\ell)} + f \\ &\equiv \tilde{f}, \end{aligned}$$

we get an elliptic problem at each time step.

The Alternating Direction Implicit (ADI) method is an example of operator decomposition. Proceeding in half-steps, one each devoted to the x- and y-directions, we write

$$\left[\frac{I}{\Delta t/2} + \mathcal{L}_x\right] u^{(\ell+1/2)} = \left[\frac{I}{\Delta t/2} - \mathcal{L}_y\right] u^{(\ell)} + f$$

$$\left[\frac{I}{\Delta t/2} + \mathcal{L}_y\right] u^{(\ell+1)} = \left[\frac{I}{\Delta t/2} - \mathcal{L}_x\right] u^{(\ell+1/2)} + f.$$

The iteration matrix,

$$\left[I + \frac{\Delta t}{2}\mathcal{L}_y\right]^{-1} \times \left[I - \frac{\Delta t}{2}\mathcal{L}_x\right] \times \left[I + \frac{\Delta t}{2}\mathcal{L}_x\right]^{-1} \times \left[I - \frac{\Delta t}{2}\mathcal{L}_y\right],$$

implies four sequential substeps per time step, two sparse matrix-vector multiplies and two sets of unidirectional bandsolves. If the data is alternately laid out in unidirectional slabs on the processors, so as to allow each set of unidirectional bandsolves to be executed independently, then we have perfect parallelism *within* substeps, but, global data exchanges *between* substeps. In other words, computation and communication each scale with the bulk size of the data of the problem.

A Fourier or spectral method is an example of a function-space decomposition. We expand

$$u(x, y, t) = \sum_{j=1}^{N} a_j(t)\phi_j(x, y).$$

Enforcing Galerkin conditions on (12) with the basis functions ϕ_i, we get

$$\frac{d}{dt}(\phi_i, u) = (\phi_i, \mathcal{L}u) + (\phi_i, f), \quad i = 1, \ldots, N.$$

Plugging the expansion into the Galerkin form,

$$\sum_{j=1}^{N}(\phi_i, \phi_j)\frac{da_j}{dt} = \sum_{j=1}^{N}(\phi_i, \mathcal{L}\phi_j)a_j + (\phi_i, f), i = 1, \ldots, N.$$

Inverting the mass matrix, $M \equiv [(\phi_j, \phi_i)]$ on both sides, and denoting the stiffness matrix by $K \equiv [(\phi_j, \mathcal{L}\phi_i)]$, we get a set of ordinary differential equations for the expansion coefficients:

$$\dot{a} = M^{-1}Ka + M^{-1}g.$$

If the basis functions are orthogonal and diagonalize the operator, then M and K are diagonal, and these equations perfectly decouple, creating N-fold concurrency for the evolution of the spectral components. However, in applications, it is necessary to frequently reconstitute the physical variable u. This is true for interpreting or visualizing the model and also for handling possible additional terms of the PDE in physical space in a "pseudo-spectral" approach, since it is unlikely that practically arising operators readily lead to orthogonal eigenfunctions for which there are fast transforms. Transforming back and forth from physical to spectral space on each iteration leads, again, to an algorithm where the computation and the communication together scale with the problem size, and there is all-to-all communication.

An additive Schwarz domain decomposition method for this problem has been described in subsection 2.3. We define subdomain restriction operators, R_i, and extension operators, R_i^T, for subdomains $i = 1, \ldots, N$, and replace $Au = f$, by $B^{-1}Au = B^{-1}f$, where $B^{-1} \equiv \sum_i R_i^T(\tilde{A}_i)^{-1}R_i$ and solve by a Krylov method. There are several Krylov steps per time step, each requiring a matrix-vector multiplies with $B^{-1}A$. Due to the concurrency implied by the sum, there is parallelism on each subregion. However the dominant communication is nearest-neighbor data exchange, whose size scales as the perimeter (resp., surface in three dimensions), compared to the computation, whose size scales as the area (resp., volume). Therefore, domain decomposition possesses excellent scalability properties with respect to implementation on distributed memory computers. There is a need for a small

global sparse linear system solve in some problems, to obtain mathematical optimality. (This is not necessary for the parabolic problem considered above.) Though this small problem requires global communication (either to set up redundant instances, solved concurrently, or to carry out a collaborative solution) and demands analysis and extreme care to keep subdominant, it escapes the bulk communication burdens of the other approaches.

3. Extensions to Newton-Krylov-Schwarz

In this section, we consider transformations of various forms of the vanilla NKS method for the discretized rootfinding problem. Some of these methods create a sequence of modified problems and others work directly with the original discretization, through advanced preconditioning.

The lack of convergence robustness of Newton's method is frequently bemoaned. In practice, globalization strategies leading from a convenient initial iterate into the ball of convergence of Newton's method around the desired root are required. For problems arising from differential equations, there are many choices. The issue of globalization is a much more vexing for steady boundary value problems (BVPs) than for initial value problems (IVPs), where accurately following the physical transient often guarantees a good initial guess and a diagonally dominant Jacobian on each time step. Based on the robustness of IVP solvers, BVPs are often approached through a false time-stepping.

3.1. Pseudo-transient Continuation. Pseudo-transient continuation solves the steady-state problem $\mathbf{f}(\mathbf{u}) = 0$, for which a solution is presumed to exist, through a series of problems

$$\mathbf{F}^\ell(u) \equiv \frac{\mathbf{u} - \mathbf{u}^{\ell-1}}{\tau^\ell} + \mathbf{f}(\mathbf{u}) = 0, \quad \ell = 1, 2, \ldots, \tag{13}$$

which are derived from a method-of-lines model

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{f}(\mathbf{u}),$$

each of which is solved (approximately) for $\mathbf{u}^\ell$. The physical transient is followed when the time step τ^ℓ is sufficiently small, and the problems at each time step are well solved. leading the iterations through a physically feasible sequence of states. Furthermore, the Jacobians associated with $\mathbf{F}^\ell(\mathbf{u}) = 0$ are well conditioned when τ^ℓ is small. See [**51**] for an analysis of this effect based on the spectrum of the preconditioned operator in the case of the constant coefficient heat equation.

τ^ℓ is advanced from $\tau^0 \ll 1$ to $\tau^\ell \to \infty$ as $\ell \to \infty$, so that $\mathbf{u}^\ell$ approaches the root of $\mathbf{f}(\mathbf{u}) = 0$. We emphasize that pseudo-transient continuation does *not* require reduction in $||\mathbf{f}(\mathbf{u}^\ell)||$ at each step, as do typical linesearch or trust region globalization strategies [**45**]; it can "climb hills."

Strict Newton iteration applied to (13) yields

$$\mathbf{u}^{\ell,k} = \mathbf{u}^{\ell-1} - (\mathbf{I} + \tau^\ell \mathbf{f}'(\mathbf{u}^{\ell,k}))^{-1}(\mathbf{u}^{\ell,k} + \tau^\ell \mathbf{f}(\mathbf{u}^{\ell,k}) - \mathbf{u}^{\ell-1}), \quad k = 0, 1, \ldots. \tag{14}$$

If we take $\mathbf{u}^{\ell,0} = \mathbf{u}^{\ell-1}$ (the simplest initial iterate), then the first correction step is

$$\mathbf{u}^{\ell,1} = \mathbf{u}^{\ell-1} - (\frac{1}{\tau^\ell}\mathbf{I} + \mathbf{f}'(\mathbf{u}^{\ell-1}))^{-1}\mathbf{f}(\mathbf{u}^{\ell-1}). \tag{15}$$

In some problems, it may be required to iterate the Newton corrector (14) more than once [**66**] or until it converges ($\lim_{k\to\infty} \mathbf{u}^{\ell,k} \equiv \mathbf{u}^\ell$), thus leading in the limit

to following the transient implicitly. However, we generally prefer to advance in pseudo-time after just one Newton step (15).

A time-stepping scheme is required to complete the algorithm. One choice is successive evolution-relaxation (SER) [**99**], which lets the time step grow in inverse proportion to residual norm progress:

$$\tau^{\ell} = \tau^{\ell-1} \cdot \frac{||\mathbf{F}(\mathbf{u}^{\ell-2})||}{||\mathbf{F}(\mathbf{u}^{\ell-1})||}. \tag{16}$$

Alternatively, a temporal truncation error strategy bounds the maximum temporal truncation error in each individual component, based on a local estimate for the leading term of the the error. (The idea is not to control the error, *per se*, but to control the stepsize through its relationship to the error.) Another approach sets target maximum magnitudes for change in each component of the state vector and adjusts the time step so as to bring the last measured change to the target. All such devices are "clipped" into a range about the current time step in practice. Typically, the time step is not allowed to more than double in a favorably converging situation, or to be reduced by more than a factor of ten in an unfavorable one, unless feasibility is at stake, in which case the time step may be drastically cut [**74**].

The globalization theory of [**74**] employs a three-phase approach, whose phases in practice may or may not be cleanly demarcated in residual norm convergence plots. Initially, $||\mathbf{u}^0 - \mathbf{u}^*|| \gg 1$ and $\tau^0 \ll 1$. During an "induction phase" the solution is marched in a method-of-lines sense with relatively small time step until $||\mathbf{u} - \mathbf{u}^*||/||\mathbf{u}^0 - \mathbf{u}^*|| \ll 1$. Success of this phase is governed by stability and accuracy of the integration scheme (we simply use the backward Euler method) and by the choice of initial iterate. This theory has been extended to index-1 differential-algebraic systems in [**42**].

For problems in which a complex feature, such as a shock or a flame front, must arise from a structure-free initial condition, the induction phase is typically by far the longest. In a grid-sequenced problem, in which the initial iterate on a given fine grid is interpolated from a converged solution on a coarser grid, and in which solution features are correctly located (if not fully resolved), the induction phase on the finest grid can be relatively brief [**119**]. During a second "transition phase" the time step is built up in the neighborhood of the solution. The critical assumption is existence of a β such that $\|(\mathbf{I} + \tau \mathbf{f}'(\mathbf{u}))^{-1}\| \leq (1 + \beta\tau)^{-1}$ for all $\tau \geq 0$ if $\|\mathbf{u} - \mathbf{u}^*\| \leq \epsilon$. Finally comes a "root polishing phase," during which the the time steps approach infinity (or some user-imposed upper bound) and iterates approach the root with asymptotic Newton-like convergence. This phase is treated by a conventional local analysis, as in [**72**].

The main result of the theory is that there is either convergence from $\mathbf{u}^0$ to $\mathbf{u}^*$ or an easily detectable (undesirable) contraction of τ^{ℓ} toward 0, allowing recovery actions before blow-up or floating point faults from infeasible steps. (Robust recovery is particularly important in parallel applications.) The main hypotheses of the theory, including smooth differentiability of $\mathbf{f}(\mathbf{u})$, are difficult to verify in practice. They are also rarely respected in practice, since instantaneous analytical approximations of $\mathbf{f}'(\mathbf{u})$ are too expensive in memory and execution time.

When nested within a pseudo-transient continuation scheme to globalize the Newton method [**75**], the implicit framework (called ΨNKS) has four levels:

```
do l = 1, n_time
   SELECT TIME-STEP
```

```
   do k = 1, n_Newton
      compute nonlinear residual and Jacobian
      do j = 1, n_Krylov
         do i = 1, n_Precon
            solve subdomain problems concurrently
         enddo
         perform Jacobian-vector product
         ENFORCE KRYLOV BASIS CONDITIONS
         update optimal coefficients
         CHECK LINEAR CONVERGENCE
      enddo
      perform vector update
      CHECK NONLINEAR CONVERGENCE
   enddo
enddo
```

The operations written in uppercase customarily involve global synchronizations, about which we comment in Subsection 3.4.

Often in such large ill-conditioned problems relatively little progress is made in a given inner linear iteration, with the consequence that the Newton correction is effectively underdamped and the steady-state residual norm improves only slightly. This provides direct feedback limiting the increase of the time step (and possibly decreasing it), which maintains or improves the linear conditioning of the next step, rather than letting the conditioning deteriorate with increasing pseudo-time step. (See [**51**] for a polyalgorithmic method that exploits the effect of the pseudo-time step on the linear conditioning.)

Effective use of ΨNKS in PDEs requires attention to several details. We mention, in particular, the sensitivity of the methodology to the implicitness of the boundary conditions, the presence of limiters in the discretization, and numerical properties such as the scaling of the differencing parameter in the matrix-free application of the Jacobian and the convergence of the inner Krylov iterations.

Historical remarks. The use of pseudo-transience as a means of approaching steady states (typically in the form of time-parabolization of an elliptic boundary value problem) has been independently reinvented in contexts far too numerous to mention. We have been particularly influenced by two forms: the "successive evolution-relaxation" strategy of Mulder and Van Leer [**99**] and the temporal truncation error strategy described in [**81**], both of which smoothly adapt the aggressiveness of the timestepping to the progress of the iterations toward steady state, ultimately leading to Newton-like asymptotic superlinear or quadratic convergence rates. Some sufficient conditions for globalized convergence for such strategies are given in [**74**].

3.2. Other Continuation Methods. Besides pseudo-transient continuation, there are two other important types of continuation in the literature of numerical solutions for nonlinear BVPs, namely, continuation in a physical parameter of the problem, and mesh sequencing, which is continuation in a discretization parameter, namely the mesh spacing.

Physical parameters often provide "knobs" by which the nonlinearity in a problem can be varied. Perhaps the most important example is the Reynolds number, which directly multiplies the convective terms of Navier-Stokes, but there are many

other examples including body forcings and boundary forcings. The solution of $\mathbf{f}(\mathbf{u}, \pi^\ell) = 0$, where π^ℓ is such a parameter, can be implicitly defined as $\mathbf{u}(\pi^\ell)$.

We suppose that $\mathbf{f}(\mathbf{u}, \pi^0) = 0$ is "easy" to solve; for instance, it may be linear in $\mathbf{u}$, as when π is a Reynolds number and the governing equations reduce to the Stokes subset. Given $\mathbf{u}^{\ell-1}$ corresponding to $\pi^{\ell-1}$, we can posit a good initial guess for $\mathbf{u}^\ell$ at a nearby π^ℓ from the Taylor expansion

$$\mathbf{u}^{\ell,0} = \mathbf{u}(\pi^{\ell-1}) + \left(\frac{\partial \mathbf{u}}{\partial \pi}\right)^{\ell-1} (\pi^\ell - \pi^{\ell-1}) \ . \tag{17}$$

Implicitly differentiating $\mathbf{f}(\mathbf{u}, \pi) = 0$ with respect to π gives

$$\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)\left(\frac{\partial \mathbf{u}}{\partial \pi}\right) + \left(\frac{\partial \mathbf{f}}{\partial \pi}\right) = 0 \ , \tag{18}$$

or

$$\left(\frac{\partial \mathbf{u}}{\partial \pi}\right) = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)^{-1}\left(\frac{\partial \mathbf{f}}{\partial \pi}\right) \tag{19}$$

whence the right-hand side of (17) can be evaluated. This presumes that one is able to readily solve linear systems with the Jacobian, $\frac{\partial \mathbf{f}}{\partial \mathbf{u}}$; otherwise, poorer approximations are possible, including the simple "bootstrapping" procedure of using just $\mathbf{u}(\pi^{\ell-1})$, itself, for $\mathbf{u}^{\ell,0}$.

Mesh sequencing, a one-way form of multigrid, is useful when a nonlinear problem is easier to solve on a coarser grid than the one on which the solution is ultimately desired, either because the nonlinearity, itself, is milder or because the linear conditioning of the sequence of nonlinear correction problems is milder. An initial iterate for the next finer mesh is constructed by interpolation from the solution on the preceding coarser mesh. Asymptotically, under certain assumptions that are natural when the discretization ultimately becomes fine enough to accurate resolve the continuous statement of the BVP, it can be shown that the initial interpolant lies in the domain of convergence of Newton's method [**118**] on the finer grid. Unfortunately, it is usually not easy to determine when this asymptotic range is reached. Consequently, another continuation method, such as pseudo-transience, may be used to drive the initial interpolant towards the Newton domain on each mesh step. Such nested continuation methods are often required in practice on highly nonlinear problems, such as detailed kinetics combustion. Since a decreasing number of inner continuation steps are required on the finer meshes, the nested approach can be economical.

3.3. Other Robustification Techniques. Whatever the combination of continuation strategies that may be invoked to prepare for a full Newton iteration on the ultimate accurately discretized BVP, modified Newton-like systems need to be solved at each stage. Traditional physics-independent, discretization-independent algebraic robustification strategies can be employed on these systems and any code intended for general purpose by non-experts should default to some combination of the strategies of line search, trust region, and back-tracking, [**69, 73**] or the more crude, but often successful "damping on percentage change" [**85, 135**]. These highly developed arts are beyond the scope of this chapter; however, it is important to keep in mind that they are usually built into the software to complement the robustification techniques that we consider here that have their origins in the PDE system, itself.

3.4. Nonlinear Preconditioning. The lack of a global convergence theory for Newton's method is a severe drawback that has been met in practice with a variety of inventions. Some, generally those rooted in the physics known to lie behind particular discrete nonlinear systems, are applied outside of Newton's method and exercise their beneficial effect by changing the system or the initial iterate fed to Newton's method. Others, generally those rooted in mathematical assumptions about the behavior of $\mathbf{F}(\mathbf{u})$ near a root, are applied inside, and have their effect by modifying the strict Newton correction before it is accepted. In this section, we consider a new technique, additive Schwarz preconditioned inexact Newton (or "ASPIN"), which includes multiple nested applications of Newton's method. ASPIN involves a (generally nonlinear) transformation of the original rootfinding problem for $\mathbf{F}(\mathbf{u})$ to a new rootfinding problem, $\Phi(\mathbf{u}) = 0$, to which an outer Jacobian-free Newton method is applied. The formation of $\Phi(\mathbf{u})$ at a given point $\mathbf{u}$, which is required many times in the course of performing the outer Jacobian-free Newton-Krylov iteration, in turn involves the solution of possibly many smaller nonlinear systems by Newton's method.

Without such a transformation, Newton's method may stagnate for many iterations in problems with imbalanced nonlinearities. We informally refer to such problems as "nonlinearly stiff." An example is compressible flow with a shock. The size of the global Newton step may be limited in such problems by high curvature in the neglected terms of the multivariate expansion of $\mathbf{F}(\mathbf{u})$ coming from just a few degrees of freedom defined near the shock. Cai and collaborators [**32, 33, 35**] have devised ASPIN to concentrate nonlinear work at such strong nonlinearities, and produce a more balanced global nonlinear problem, on which Newton does not need as much robustification.

From an algebraic viewpoint, ASPIN is a generic transformation that requires only the unique solvability of subsystems of the original $\mathbf{F}(\mathbf{u})$ in the neighborhood of the root $\mathbf{u}_*$ to effect. From a physical viewpoint, ASPIN is a family of methods in which the subsystems may be chosen by domain decomposition, isolation of equations arising from different physical phenomena, identification of nonlinear stiffness, or still other criteria. As with all Schwarz methods, many flavors of nonlinear Schwarz preconditioning are possible — additive, multiplicative, or general polynomial combination of sub-operators; single-level or multi-level; overlapping or nonoverlapping. In this section, we discuss the additive, single-level case, with arbitrary overlap. We illustrate domain and equation partitioning.

Let

$$S = (1, \ldots, n)$$

be an index set for the degrees of freedom $\mathbf{u}_i$ and the corresponding residual components $\mathbf{F}_i$. Assume that $S_1, \ldots, S_N$ is a partition of S in the sense that

$$\bigcup_{i=1}^{N} S_i = S, \text{ and } S_i \subset S.$$

The subsets may overlap; if n_i be the dimension of S_i, then, in general,

$$\sum_{i=1}^{N} n_i \geq n.$$

Using the partition of S, we introduce subspaces of R^n and the corresponding restriction and extension matrices. For each S_i we define $V_i \subset R^n$ as

$$V_i = \{v | v = (v_1, \ldots, v_n)^T \in R^n, v_k = 0, \text{ if } k \notin S_i\} \ .$$

Elements of vectors V_i outside of S_i are trivial, but we consider what is formally a full-dimensional subspace, since each component of $\mathbf{F}$ formally depends on all components of $\mathbf{u}$. In a practical application of ASPIN, however, we assume that there is a dominant association of certain components of $\mathbf{F}$ with certain components of $\mathbf{u}$, and that, in fact, the square subblock of the full Jacobian $\mathbf{F}'(\mathbf{u})$ describing this dominant relationship is invertible near the desired root. This is major restriction for general nonlinear algebraic systems, but it is completely natural for systems arising from partial differential equations describing local conservation laws. Just as in additive Schwarz for linear problems, most of the computing in ASPIN is carried out within these local blocks.

For each S_i, let R^{n_i} be the space that contains the nontrivial part of V_i, and let R_i be the $n_i \times n$ Boolean restriction operator from a vector in R^n to a vector in R^{n_i}, with the corresponding extension operator defined by the transpose R_i^T.

Using the restriction operator, define the subdomain nonlinear function as

$$\mathbf{F}_i = R_i \mathbf{F}.$$

For any given $\mathbf{v} \in R^n$, define $\mathbf{T}_i(\mathbf{v}) \in V_i$ as the solution of the following subspace nonlinear system

$$\mathbf{F}_i(\mathbf{v} - \mathbf{T}_i(\mathbf{v})) = 0,$$

for all subspaces $i = 1, \ldots, N$. We now introduce the transformed function

$$\Phi(\mathbf{u}) = \sum_{i=1}^{N} \mathbf{T}_i(\mathbf{u}), \tag{20}$$

which we refer to as the nonlinearly preconditioned $\mathbf{F}(\mathbf{u})$. A single evaluation of the function $\Phi(\mathbf{v})$, for a given $\mathbf{v}$, involves the calculation of the $\mathbf{T}_i$, which in turn involves the solution of nonlinear systems involving each $\mathbf{F}_i$. If the overlap is zero, this is a block nonlinear Jacobi preconditioner.

In the linear case, this algorithm reduces to the additive Schwarz algorithm. Using the usual notation, if

$$\mathbf{F}(\mathbf{u}) = A\mathbf{u} - \mathbf{b},$$

then

$$\Phi(\mathbf{u}) = \left(\sum_{i=1}^{N} R_i^T A_i^{-1} R_i \right) (A\mathbf{u} - \mathbf{b}),$$

where A_i^{-1} is the inverse of $A_i = R_i A R_i^T$.

If $\Phi(\mathbf{u}) = 0$ is to be solved using a Newton type algorithm, then the Jacobian $\Phi'(\mathbf{u})$ is needed in one form or another. Since it is generally dense Jacobian-free methods are essential. It is shown in [**32**] that $\Phi'(\mathbf{u})$ is well approximated at a point $\mathbf{u}$ near a root $\mathbf{u}_*$ by $\mathcal{J}(\mathbf{u}) = \sum_{i=1}^{N} R_i^T J_i^{-1}(\mathbf{u}) R_i J(\mathbf{u})$. where $J(\mathbf{u})$ is the Jacobian of the original nonlinear system, $\mathbf{F}'(\mathbf{u})$ and J_i, is the Jacobian of the subdomain nonlinear system, $J_i(\mathbf{u}) = R_i^T J(\mathbf{u}) R_i$, for $i = 1, \ldots, N$. If $\mathbf{F}(\mathbf{u})$ is sparse nonlinear function of its arguments, then J is a sparse matrix and so are the J_i, so it is economical to apply $\mathcal{J}$ to an arbitrary vector.

From a software engineering viewpoint, it is convenient that the action required to apply the preconditioned Jacobian to an arbitrary vector is already present in

any Newton-Krylov-Schwarz code, since it is the action of the linearly Schwarz-preconditioned Jacobian of the original nonlinear function. The action of J on a vector can be approximated by the usual Fréchet derivative in a matrix-free manner, or with explicit elements. The actions of J_i^{-1} on subvectors corresponding to the nontrivial components in V_i can be performed within (possibly overlapping) partitions. Several techniques are available for computing the J_i, for example, by analytic formula, multi-colored finite differencing, or automatic differentiation. A triangular factorization of J_i may be performed, since the action is needed multiple times within a single outer Newton step on $\Phi(\mathbf{u}) = 0$.

Evaluation of $\Phi(\mathbf{u})$ itself is not a linear process with *a priori* deterministic complexity, but involves summation of local corrections that are the result of inner Newton iterations on $\mathbf{F}_i(\mathbf{u})$. In a parallel implementation, if a different processor is used for each partition S_i, communication to obtain nontrivial ghost values belonging to and updated in other processors may be necessary. We note, however, that these ghost values do not change during the solution of the subdomain nonlinear system. It is known in practice [**50**] that the linear systems of the main Newton iteration for $\Phi(\mathbf{u})$ do not need to be solved well (in fact, often *should* not be solved well) in early stages, in the sense that it is only necessary to enforce the norm of the Newton correction equation residual

$$||\mathcal{J}(\mathbf{u})\delta\mathbf{u} + \Phi(\mathbf{u})||_2$$

to be bounded by some tolerance (called "forcing term" in [**50**]) that may itself be a function of Φ and other by-products of the computation, approaching zero at a rate sufficient to guarantee convergence, superlinear convergence, or even quadratic convergence of the outer iteration. This provides some latitude in the degree of accuracy to which the subdomain nonlinear problems are solved in the early iterations.

It is shown in [**32**] that Newton's method applied to a nonlinearly preconditioned version of the velocity-vorticity driven cavity problem, based on domain decomposition converges rapidly (e.g., in 5–10 global Newton iterations) at Reynolds numbers far beyond those at which Newton's method applied to the original discretization of the problem hopelessly stagnates.

It is shown in [**35**] that Newton convergence of the nonlinearly transformed version of the problem of shocked flow in a variable cross-section duct is much less sensitive to mesh refinement than the original discretization.

The driven cavity problem can be ameliorated by continuation in Reynolds number and the shocked flow problem through mesh sequencing in other contexts. Nevertheless, it is interesting to see that a purely algebraic method, ASPIN, is effective at rebalancing nonlinearities so that Newton converges easily. We expect that it will have wide applicability in problems with complex nonlinearities as a means of increasing nonlinear robustness.

Unfortunately, it is difficult to obtain direct approximations to the dense Jacobian of the transformed system, $\mathcal{J}$, so as to improve the *linear conditioning* of the resulting Newton correction problems. Therefore, these problems are subject to linear ill-conditioning as mesh resolution increases. To conquer this linear ill-conditioning, multilevel methods of ASPIN have been devised. The straightforward FAS multigrid approach on $\Phi(\mathbf{u})$ may not be practical since the nonlinear correction to be computed at each coarse level requires an evaluation of the fine-grid residual, which is subsequently restricted to compute the coarse-grid defect that drives the

correction. Since each fine-grid residual involves a host of fine-grid nonlinear subproblems, this is expensive. An alternative multi-level method is investigated, with promising results, in [**33**].

A case of special interest in ASPIN is the case of relatively few subspaces, e.g., partitioning by equation type in a multicomponent problem, as opposed to by subdomain in a problem of millions of gridcells. In this case, it is natural to take the overlap to be zero; then the diagonal blocks of $\sum_{i=1}^{N} R_i^T J_{S_i}^{-1} R_i J$ are all identities, and do not involve any computations when multiplied with vectors. For two subdomains,

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix}$$

so

$$\sum_{i=1}^{2} R_i^T J_i^{-1} R_i J = \begin{pmatrix} I & J_{11}^{-1} J_{12} \\ J_{22}^{-1} J_{21} & I \end{pmatrix}.$$

For example, partition "1" could represent the fluid degrees of freedom, and partition "2" the structural degrees of freedom in a nonlinear fluid-structure interaction. The Jacobians of each need to be inverted only on the portion of each partition that couples directly to the other.

ASPIN has been motivated above primarily as a nonlinear robustification technique. However, in the long run of terascale simulation of PDEs its most important virtue may come to be understood as deferring synchonization from the outer Jacobian-free Newton-Krylov loop to the multiple inner Newton-Krylov loops that form the corrections that are summed to evaluate each Φ. The synchronizations, whether inner or outer, occur over all of the degrees of freedom in the respective Krylov spaces. The outer Newton-Krylov loop may have billions of degrees of freedom distributed over thousands of processors; hence each outer synchronization has an exhorbitant communication cost. In contrast, the inner synchronizations are over subsets of variables, which can be hosted naturally by subsets of processors. Eliminating some outer synchronizations in favor of some extra inner work is therefore a beneficial algorithmic adaptation to terascale architecture.

3.5. Physics-based Preconditioning. An important class of preconditioners for the Jacobian-free Newton-Krylov method is physics-based operator splitting. The operator notation for the right-preconditioned, matrix-free form of the method is:

$$J_{full}(\mathbf{u}) B_{split}^{-1} \mathbf{v} \approx \frac{\mathbf{F}_{full}(\mathbf{u} + \epsilon B_{split}^{-1} \mathbf{v}) - \mathbf{F}_{full}(\mathbf{u})}{\epsilon}. \tag{21}$$

Here, subscript "*full*" refers to the full nonlinear function, and "*split*" denotes a preconditioning process handled in an operator-split manner. Many operator-split time integration methods have been developed based on insight from the physics of the underlying system [**16, 63, 104, 106**]. Since these methods can be utilized as preconditioners, we refer to this approach as "physics-based preconditioning." It is well understood that operator-split methods have limitations as solvers, thus they most likely also have limitations as preconditioners. However, they still provide an interesting class of preconditioners for the Jacobian-free Newton-Krylov method.

Following Mousseau et al. [**96**], we illustrate the construction of an operator-split preconditioner for a stiff wave system, specifically, the one-dimensional shallow

water wave equations with a stiff gravity wave:

$$\frac{\partial h}{\partial t} + \frac{\partial uh}{\partial x} = 0, \tag{22}$$

$$\frac{\partial uh}{\partial t} + \frac{\partial u^2 h}{\partial x} = -gh\frac{\partial h}{\partial x}. \tag{23}$$

Here u is the fluid velocity, h is the hydrostatic pressure, x is the spatial coordinate, t is time, g is gravity, and $\sqrt{gh}$ is the fast wave speed. This is the time scale we wish to "step over" in evolving the mesoscale dynamics of interest, since typically $\sqrt{gh} \gg u$. A semi-implicit method is constructed by linearizing and implicitly discretizing only those terms that contribute to the stiff gravity wave. Thus, some insight from the physics is required to produce the implicit system. With ℓ as the current time and $\ell - 1$ as the previous time, and spatial discretization suppressed, we have

$$\frac{h^\ell - h^{\ell-1}}{\Delta t} + \frac{\partial (uh)^\ell}{\partial x} = 0, \tag{24}$$

$$\frac{(uh)^\ell - (uh)^{\ell-1}}{\Delta t} + \frac{\partial (u^2 h)^{\ell-1}}{\partial x} + gh^{\ell-1}\frac{\partial h^\ell}{\partial x} = 0. \tag{25}$$

Note that the nonlinear term in h on the right-hand side of (23) has been linearized by evaluating the square of the wave speed (gh) at the previous time. We evaluate $\frac{\partial (u^2 h)}{\partial x}$ at the previous time since it does not contribute to the linearized gravity wave.

The momentum equation (25) is rearranged as

$$(uh)^\ell = -\Delta t g h^{\ell-1}\frac{\partial h^\ell}{\partial x} + S^{\ell-1} \quad (S^{\ell-1} \equiv (uh)^{\ell-1} - \Delta t\frac{\partial (u^2 h)^{\ell-1}}{\partial x}). \tag{26}$$

Equation (26) is then substituted into (24) to give the following parabolic equation:

$$\frac{h^\ell - h^{\ell-1}}{\Delta t} - \frac{\partial}{\partial x}\left(\Delta t g h^{\ell-1}\frac{\partial (h)^\ell}{\partial x}\right) = \frac{\partial S^{\ell-1}}{\partial x} \tag{27}$$

Equation (27) is solved for h^ℓ, and then one can backsolve for $(uh)^\ell$ using (26). Using classical operator splitting, a system of two hyperbolic PDEs has been transformed into a single parabolic PDE and an explicit update. The price is temporal splitting error. For this simple problem, the source of the nonlinear inconsistency is the linearized wave speed (a time discretization error) and the fact that advection in (25) is at a different time level. This will be an issue when advection and wave propagation happen on the same time scale. The innovation of physics-based preconditioning is realizing that the nonlinear inconsistent solution of (26) and (27) can be used as the preconditioner to a nonlinearly consistent Newton-Krylov solution of (22)–(23).

The essential insight of [**96**] is that the function of a preconditioner in a Newton-Krylov method for the shallow water system is to map $[res_h, res_{uh}]$ to $[\delta h, \delta uh]$. Using the semi-implicit method in delta form (suppressing spatial discretization) the linearized equations are:

$$\frac{\delta h}{\Delta t} + \frac{\partial \delta (uh)}{\partial x} = -res_h, \tag{28}$$

$$\frac{\delta(uh)}{\Delta t} + gh^{\ell-1}\frac{\partial \delta h}{\partial x} = -res_{uh}. \tag{29}$$

Following the classical approach by substituting (29) into (28), and eliminating $\delta(uh)$, produces

$$\frac{\delta h}{\Delta t} + \frac{\partial}{\partial x}(\Delta t g h^n \frac{\partial \delta h}{\partial x}) = -res_h + \frac{\partial}{\partial x}(\Delta t\ res_{uh}). \tag{30}$$

This parabolic equation can be approximately solved for δh. Then $\delta(uh)$ can be evaluated:

$$\delta(uh) = -\Delta t g h^n \frac{\partial \delta h}{\partial x} - res_{uh}. \tag{31}$$

Thus, using the semi-implicit method, $[res_h, res_{uh}]$ has been mapped to $[\delta h, \delta uh]$ at the cost of just one approximate parabolic solve. The effectiveness of this preconditioning approach has been demonstrated in one dimension in [**95**], and more recently the concept has been extended to the two-dimensionsal shallow water equations including the Coriolis force [**96**].

The use of operator-split solvers as preconditioners for Jacobian-free Newton-Krylov appears not to have a long history, but is rapidly developing. See instances for time-independent reaction diffusion equations [**97**], time-dependent MHD equations [**39**], steady state incompressible Navier-Stokes equations [**84, 107**], and time-dependent incompressible Navier-Stokes equations [**89, 107**]. Also in [**89**], a standard approximate linearization method used for phase-change heat conduction problems, has been employed as a preconditioner for a JFNK solution of phase-change heat conduction problems.

4. PDE-constrained Optimization

As exemplified in Sections 2 and 3, years of two-sided (from architecture up, from applications down) algorithms research has made it possible to solve partial differential equation (PDE) problems implicitly with reasonable scalability. PDEs are equality constraints on the state variables in many optimization problems. Hardly auxiliary, the PDE system may contain millions of degrees of freedom. In problems of shape optimization and control, the number of optimization parameters is typically much smaller than the number of state variables. In problems of parameter identification, the number of parameters to be optimized may be comparable to the number of state variables, but few general-purpose optimization frameworks have been demonstrated at the scale required for three-dimensional problems. Since the PDE analysis generally dominates the computation, we propose that large-scale PDE-constrained optimization codes be constructed around the data structures and functional capabilities of the PDE solver.

Optimization is easily incorporated through the Lagrange saddle-point formulation into a Newton-like parallel PDE framework that accommodates Schur-type algebraic substructuring. Newton's method is a common element in the most rapidly convergent solvers and optimizers. Furthermore, a PDE solver that is not part of an optimization framework is probably short of what the client really wants. Hence, for both algorithmic and teleological reasons, analysis and optimization belong together.

4.1. Implications of NKS for Optimization. Equality constrained optimization leads, through the Lagrangian formulation, to a multivariate nonlinear rootfinding problem for the gradient (the first-order necessary conditions), which is amenable to treatment by Newton's method. To establish notation, consider the following canonical framework, in which we enforce equality constraints on the state variables only. (Design variable constraints require additional notation, and inequality constraints require additional algorithmics, but these generalizations are well understood.) Choose m design variables α to minimize the objective function, $\phi(u,\alpha)$, subject to n state constraints, $h(u,\alpha)=0$, where u is the vector of state variables. In the Lagrange framework, a stationary point of the Lagrangian function

$$\mathcal{L}(u,\alpha,\lambda) \equiv \phi(u,\alpha) + \lambda^T h(u,\alpha)$$

is sought. When Newton's method is applied to the first-order optimality conditions, a linear system known as the Karush-Kuhn-Tucker (KKT) system arises at each step. There is a natural "outer" partitioning: the vector of parameters is often of lower dimension than the vectors of states and multipliers. This suggests a Schur complement-like block elimination process at the outer level, not for concurrency, but for numerical robustness and conceptual clarity. Within the state-variable subproblem, which must be solved repeatedly in the Schur complement reduction, Schwarz provides a natural "inner" partitioning for concurrency.

A major choice to be made in the Newton approach to constrained optimization is between exact elimination of the states and multipliers by satisfying constraint feasibility at every step (reduced system), and progress in all variables simultaneously, possibly violating constraints on intermediate iterates (full system). An advantage of the former is the existence of high-quality, robust black box software for this reduced sequential quadratic programming (RSQP) [**103**] approach. The advantages of the latter are in reuse of high-quality parallel PDE software, the freedom to use inexact solves (since finely resolved PDE discretizations in 3D militate against exact elimination), and the ease of application of automatic differentiation software, without having to differentiate through the nonlinear subiterations that would be implied by repeated projection to the constraint manifold in RSQP.

We mention three classes of PDE-constrained optimization:

- **Design optimization** (especially shape optimization): α parametrizes the domain of the PDE (e.g., a lifting surface) and ϕ is a cost-to-benefit ratio of forces, energy expenditures, etc. Typically, m is small compared with n and does not scale directly with it. However, m may still be hundreds or thousands in industrial applications.
- **Optimal control**: α parametrizes a continuous control function acting in part of or on the boundary of the domain, and ϕ is the norm of the difference between desired and actual responses of the system. For boundary control, $m \propto n^{2/3}$; for control by body forces, $m \propto n$.
- **Parameter identification/data assimilation**: α parametrizes an unknown continuous constitutive function defined throughout the domain, and ϕ is the norm of the difference between measurements and simulation results. Typically, $m \propto n$.

Written out in partial detail, the optimality conditions are

$$\frac{\partial \mathcal{L}}{\partial u} \equiv \frac{\partial \phi}{\partial u} + \lambda^T \frac{\partial h}{\partial u} = 0 \ , \tag{32}$$

$$\frac{\partial \mathcal{L}}{\partial \alpha} \equiv \frac{\partial \phi}{\partial \alpha} + \lambda^T \frac{\partial h}{\partial \alpha} = 0 \ , \tag{33}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} \equiv h = 0 \ . \tag{34}$$

Newton's method iteratively seeks a correction,

$$\begin{pmatrix} \delta u \\ \delta \alpha \\ \delta \lambda \end{pmatrix} \quad \text{to the iterate} \quad \begin{pmatrix} u \\ \alpha \\ \lambda \end{pmatrix}.$$

With subscript notation for the partial derivatives, the Newton correction (KKT) equations are

$$\begin{bmatrix} (\phi_{,uu} + \lambda^T h_{,uu}) & (\phi_{,u\alpha} + \lambda^T h_{,u\alpha}) & h_{,u}^T \\ (\phi_{,\alpha u} + \lambda^T h_{,\alpha u}) & (\phi_{,\alpha\alpha} + \lambda^T h_{,\alpha\alpha}) & h_{,\alpha}^T \\ h_{,u} & h_{,\alpha} & 0 \end{bmatrix} \begin{pmatrix} \delta u \\ \delta \alpha \\ \delta \lambda \end{pmatrix} = - \begin{pmatrix} \phi_{,u} + \lambda^T h_{,u} \\ \phi_{,\alpha} + \lambda^T h_{,\alpha} \\ h \end{pmatrix},$$

or

$$\begin{bmatrix} W_{uu} & W_{\alpha u}^T & J_u^T \\ W_{\alpha u} & W_{\alpha\alpha} & J_\alpha^T \\ J_u & J_\alpha & 0 \end{bmatrix} \begin{pmatrix} \delta u \\ \delta \alpha \\ \lambda_+ \end{pmatrix} = - \begin{pmatrix} g_u \\ g_\alpha \\ h \end{pmatrix}, \tag{35}$$

where $W_{ab} \equiv \frac{\partial^2 \phi}{\partial a \partial b} + \lambda^T \frac{\partial^2 h}{\partial a \partial b}$, $J_a \equiv \frac{\partial h}{\partial a}$, and $g_a = \frac{\partial \phi}{\partial a}$, for $a, b \in \{u, \alpha\}$, and where $\lambda_+ = \lambda + \delta\lambda$.

4.2. Newton Reduced SQP. The RSQP method [**103**] consists of a three-stage iteration. We follow the language and practice of [**11, 12**] in this and the next subsection.

- **Design Step** (Schur complement for middle blockrow):

$$H \ \delta\alpha = f \ ,$$

  where H and f are the reduced Hessian and gradient, respectively:

$$H \equiv W_{\alpha\alpha} - J_\alpha^T J_u^{-T} W_{\alpha u}^T + \left(J_\alpha^T J_u^{-T} W_{uu} - W_{\alpha u} \right) J_u^{-1} J_\alpha$$
$$f \equiv -g_\alpha + J_\alpha^T J_u^{-T} g_u - \left(J_\alpha^T J_u^{-T} W_{uu} - W_{\alpha u} \right) J_u^{-1} h.$$

- **State Step** (last blockrow):

$$J_u \ \delta u = -h - J_\alpha \ \delta\alpha.$$

- **Adjoint Step** (first blockrow):

$$J_u^T \ \lambda_+ = -g_u - W_{uu} \ \delta u - W_{\alpha u}^T \ \delta\alpha.$$

In each overall iteration, we must form and solve with the reduced Hessian matrix H, and we must solve separately with J_u and J_u^T. The latter two solves are almost negligible compared with the cost of forming H, which is dominated by the cost of forming the sensitivity matrix $J_u^{-1} J_\alpha$. Because of the quadratic convergence of Newton, the number of overall iterations is few (asymptotically independent of

m). However, the cost of forming H at each design iteration is m solutions with J_u. These are potentially concurrent over independent columns of J_α, but prohibitive.

In order to avoid computing any Hessian blocks, the design step may be approached in a quasi-Newton (e.g., BFGS) manner [**103**]. Hessian terms are dropped from the adjoint step right-hand side.

- **Design Step** (severe approximation to middle blockrow):

$$Q\ \delta\alpha = -g_\alpha + J_\alpha^T J_u^{-T} g_u\ ,$$

  where Q is a quasi-Newton approximation to the reduced Hessian.
- **State Step** (last blockrow):

$$J_u\ \delta u = -h - J_\alpha\ \delta\alpha.$$

- **Adjoint Step** (approximate first blockrow):

$$J_u^T\ \lambda_+ = -g_u.$$

In each overall iteration of quasi-Newton RSQP, we must perform a low-rank update on Q or its inverse, and we must solve with J_u and J_u^T. This strategy vastly reduces the cost of an iteration; however, it is no longer a Newton method. The number of overall iterations is many. Since BFGS is equivalent to unpreconditioned CG for quadratic objective functions, $\mathcal{O}(m^p)$ sequential cycles ($p > 0$, $p \approx \frac{1}{2}$) may be anticipated. Hence, quasi-Newton RSQP is not scalable in the number of design variables, and no ready form of parallelism can address this convergence-related defect.

To summarize, conventional RSQP methods apply a (quasi-)Newton method to the optimality conditions: solving an approximate $m \times m$ system to update α, updating u and λ consistently (to eliminate them), and iterating. The unpalatable expense arises from the exact linearized analyses for updates to u and λ that appear in the inner loop. We therefore consider replacing the exact elimination steps of RSQP with preconditioning steps in an outer loop, as described in the next subsection.

4.3. Full Space Lagrange-NKS Method. The new philosophy is to apply a Krylov-Schwarz method directly to the $(2n+m)\times(2n+m)$ KKT system (35). For this purpose, we require the action of the full matrix on the full-space vector and a good full-system preconditioner, for algorithmic scalability. One Newton SQP iteration is a perfect preconditioner—a block factored solver, based on forming the reduced Hessian of the Lagrangian H—but, of course, far too expensive. Backing off wherever storage or computational expense becomes impractical for large-scale PDEs generates a family of attractive methods.

To precondition the full system, we need approximate inverses to the three left-hand side matrices in the first algorithm of §4.2, namely, H, J, and J^T. If a preconditioner is available for H, and exact solves are available for J, and J^T, then it may be shown that conjugate gradient Krylov iteration on the (assumed symmetrizable) reduced system and conjugate gradient iteration on the full system yield the same sequence of iterates. The iterates are identical in the sense that if one were to use the values of α arising from the iteration on the reduced system in the right-hand side of the block rows for u and λ, one would reconstruct the iterates of the full system, when the same preconditioner used for H in the reduced system is used for the $W_{\alpha\alpha}$ block in the full system. Moreover, the spectrum of

the full system is simply the spectrum of the reduced system supplemented with a large multiplicity of unit eigenvalues. If one retreats from exact solves with J and J^T, the equivalence no longer holds; however, if good preconditioners are used for these Jacobian blocks, then the cloud of eigenvalues around unity is still readily shepherded by a Krylov method, and convergence should be nearly as rapid as in the case of exact solves.

This Schur-complement-based preconditioning of the full system was proposed in this equality-constrained optimization context by Biros and Ghattas in 1998 [**11**] and earlier in a related context by Batterman and Heinkenschloss [**9**]. From a purely algebraic point of view, the same Schur-complement-based preconditioning was advocated by Keyes and Gropp [**79**] in the context of domain decomposition. There, the reduced system was a set of unknowns on the interface between subdomains, and the savings from the approximate solves on the subdomain interiors more than paid for the modest degradation in convergence rate relative to interface iteration on the Schur complement. The main advantage of the full system problem is that the Schur complement never needs to be formed. Its exact action is felt on the design variable block through the operations carried out on the full system.

Biros and Ghattas have demonstrated the large-scale parallel effectiveness of the full system algorithm on a 3D Navier-Stokes flow boundary control problem, where the objective is dissipation minimization of flow over a cylinder using suction and blowing over the back portion of the cylinder as the control variables [**12**]. They performed this optimization with domain-decomposed parallelism on 128 processors of a T3E, using an original optimization toolkit add-on to the PETSc [**4**] toolkit. To quote one result from [**12**], for 6×10^5 state constraints and 9×10^3 controls, full-space LNKS with approximate subdomain solves beat quasi-Newton RSQP by an order of magnitude (4.1 hours versus 53.1 hours).

Two names have evolved for the new algorithm: Lagrange-Newton-Krylov-Schwarz was proposed by Keyes at the 1999 SIAM Conference on Optimization, and Lagrange-Newton-Krylov-Schur by Biros and Ghattas in [**12**]. The former emphasizes the use of NKS to precondition the large Jacobian blocks, the latter the use of Schur complements to precondition the overall KKT matrix. Both preconditioner suffixes are appropriate in a nested fashion, so we propose Lagrange-Newton-Krylov-Schur-Schwarz (LNKSS) when both preconditioners are used.

Automatic differentiation has two potential roles in the new algorithm: formation of the action on a Krylov vector of the full KKT matrix, including the full second-order Hessian blocks, and supply of approximations to the elements of J (and J^T) for the preconditioner.

4.4. Example of LNKS Parameter Identification. We illustrate Schur preconditioning and automatic differentiation in an elementary parameter identification context. The governing constraint is the steady state version of the nonlinear conduction equation for material temperature:

$$\nabla \cdot (\beta(x) T^{\alpha} \nabla T) = 0, \tag{36}$$

subject to inhomogeneous Dirichlet conditions on opposite faces of a slab, with homogeneous (insulating) Neumann conditions on the transverse surfaces. The resulting BVP is discretized with centered finite differences. The state variables are the discrete temperatures at the mesh nodes, and the design variable is the parameter α. The cost function is temperature matching, $\phi(\alpha, T) = \frac{1}{2}||T(x) -$

$\bar{T}(x)||^2$, where $\bar{T}$ is synthetic data based on a given α-profile. This parameter is specified for the computation of $\bar{T}(x)$, and then "withheld," to be determined by the optimizer. More generally, $\bar{T}(x)$ would be a desired or experimentally measured profile, and the phenomenological law and material specification represented by α would be determined to fit. The Brisk-Spitzer form of the nonlinear dependence of the diffusivity on the temperature in optically thick radiation transport is $\alpha = 2.5$, and that is what we employ here. Very ideal quadratic convergence of α, characteristic of Newton's method for the overall full system residual norm, is seen in the case on the left, whereas on the right convergence slows. This example must be regarded as preliminary at the time of writing and this behavior is under further investigation.

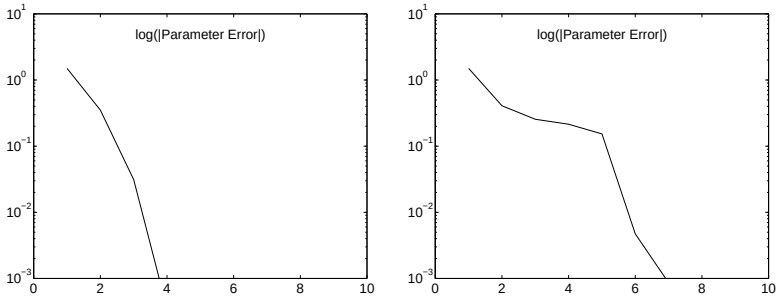

FIGURE 2. Newton convergence for the α parameter (vertical axis is $\log|\alpha - 2.5|$. **Left:** coarsely resolved 2D problem **Right:** finely resolved 1D problem

Our implementation of LNKS is in the software framework of PETSc [**4**] and ADIC [**14**].

4.5. Complexity of AD-based LNKS. Although our demonstration example is trivially low dimensional, LNKS will generally be applied to large problems of n state variables and m parameters. Upon surveying existing AD tools, we conclude that the preconditioned matrix-vector product can be formed in time linear in n and m. The shopping list of matrix actions in forming the preconditioned Jacobian-vector product of LNKS is $W_{uu}, W_{\alpha\alpha}, W_{\alpha u}$, $W_{\alpha u}^T$, J_α, J_α^T, J_u^{-1}, J_u^{-T}, and H^{-1}.

The first six are needed in the full-system matrix-vector multiplication. For this multiplication we require "working accuracy" comparable to the state of the art in numerical differentiation.

Accurate action of the last three is required in RSQP but not in the full system preconditioner. We recommend approximate factorizations of lower-quality approximations, including possibly just $W_{\alpha\alpha}$ for H, or a traditional quasi-Newton rank-updated approximation to the inverse.

We estimate the complexity of applying each block of the KKT Jacobian, assuming only that $h(u, \alpha)$ is available in subroutine call form and that all differentiated blocks are from AD tools, such as the ADIC [**14**] tool we are using in a parallel implementation of LNKSS. We assume that J_u is needed element by element, in order to factor it; hence, J_u^T is also available. Since these are just preconditioner blocks, we generally derive these elements from a different (cheaper) function call

TABLE 2. Complexity of formation of matrix objects or matrix-vector actions using forward or hybrid modes of modern automatic differentiation software. The asterisk signifies that the reverse mode consumes memory, in a carefully drawn time-space trade-off, so r is implementation-dependent.

Object	Cost: Forward Mode	Cost: Fastest (Hybrid) Mode
J_u, J_u^T	$p_u C_h$	$p_u C_h$
$J_\alpha v$	$2C_h$	$2C_h$
$J_\alpha^T v$	$p_\alpha C_h$	$r{C_h}^*$
$W_{uu}v, W_{\alpha u}^T v$	$p_u C_h + qC_\phi$	$r(C_h + C_\phi)^*$
$W_{\alpha\alpha}v, W_{\alpha u}v$	$p_\alpha C_h + qC_\phi$	$r(C_h + C_\phi)^*$

for the gradient of the Lagrangian than that used for the matvec. Define C_h, the cost of evaluating h; p_u, 1 + the chromatic number of $J_u \equiv h_{,u}$; and p_α, 1 + the chromatic number of $J_\alpha \equiv h_{,\alpha}$. Then the costs of the Jacobian objects are shown in the first three rows of Table 2.

For the Hessian arithmetic complexity, we estimate the cost of applying each forward block to a vector. Assume that $h(u, \alpha)$ and $\phi(u, \alpha)$ are available and that all differentiated blocks are results of AD tools. Define C_ϕ, the cost of evaluating ϕ; q, 1 + number of nonzero rows in ϕ''; and r, an implementation-dependent "constant," typically ranging from 3 to 100. Then the cost of the Hessian-vector products can be estimated from the last two rows of Table 2.

For the inverse blocks, we need only low-quality approximations or limited-memory updates [**22**] of the square systems J_u^{-1}, J_u^{-T}, and H^{-1}.

The complexities for *all* operations required to apply the full-system matrix-vector product and its preconditioner are at worst linear in n or m, with coefficients that depend upon chromatic numbers (affected by stencil connectivity and inter-component coupling of the PDE, and by separability structure of the objective function) and the implementation efficiency of AD tools.

4.6. Summary and Future Plans. As in domain decomposition algorithms for PDE analysis, partitioning in PDE-equality constrained optimization may be used to improve some combination of robustness, conditioning, and concurrency. Orders of magnitude of savings may be available by converging the state variables and the design variables within the same outer iterative process, rather than a conventional SQP process that exactly satisfies the auxiliary state constraints.

As with any Newton method, globalization strategies are important. These include parameter continuation (physical and algorithmic), mesh sequencing and multilevel iteration (for the PDE subsystem, at least; probably for controls, too), discretization order progression, and model fidelity progression. The KKT system appears to be a preconditioning challenge, but an exact factored preconditioner is known, and departures of preconditioned eigenvalues from unity can be quantified by comparisons of original blocks with blockwise substitutions in inexact models and solves. (For the full system, the preconditioned KKT matrix will be nonnormal, so its spectrum does not tell all.)

With the extra, but automatable, work of forming Jacobian transposes and Hessian blocks, but no extra work in Jacobian preconditioning, any parallel analysis

code may be converted into a parallel optimization code—and automatic differentiation tools make this relatively painless.

The gamut of PDE solvers based on partitioning should be mined for application to the KKT necessary conditions of constrained optimization and for direct use in inverting the state Jacobian blocks inside the optimizer.

5. Parallel Implementation of NKS Using PETSc

To implement NKS methods on distributed memory parallel computers, we employ the "Portable, Extensible Toolkit for Scientific Computing" (PETSc) [**5, 6**], a library that attempts to handle through a uniform interface, in a highly efficient way, the low-level details of the distributed memory hierarchy. Examples of such details include striking the right balance between buffering messages and minimizing buffer copies, overlapping communication and computation, organizing node code for strong cache locality, preallocating memory in sizable chunks rather than incrementally, and separating tasks into one-time and every-time subtasks using the inspector/executor paradigm. The benefits to be gained from these and from other numerically neutral but architecturally sensitive techniques are so significant that it is efficient in both the programmer-time and execution-time senses to express them in general purpose code.

PETSc is a large and versatile package integrating distributed vectors, distributed matrices in several sparse storage formats, Krylov subspace methods, preconditioners, and Newton-like nonlinear methods with built-in trust region or linesearch strategies and continuation for robustness. It has been designed to provide the numerical infrastructure for application codes involving the implicit numerical solution of PDEs, and it sits atop MPI for portability to most parallel machines. The PETSc library is written in C, but may be accessed from user codes written in C, FORTRAN, and C++. PETSc version 2, first released in June 1995, has been downloaded thousands of times by users worldwide. PETSc has many features relevant to PDE analysis, including matrix-free Krylov methods, blocked forms of parallel preconditioners, and various types of time-stepping.

A diagram of the calling tree of a typical NKS application appears in Figure 4. The arrows represent calls that cross the boundary between application-specific code and PETSc library code; all internal details of both are suppressed. The top-level user routines perform I/O related to initialization, restart, and post-processing and calls PETSc subroutines to create data structures for vectors and matrices and to initiate the nonlinear solver. PETSc calls user routines for function evaluations $\mathbf{f}(\mathbf{u})$ and (approximate) Jacobian evaluations $\mathbf{f}'(\mathbf{u})$ at given vectors $\mathbf{u}$ representing the discrete state of the flow. Auxiliary information required for the evaluation of $\mathbf{f}$ and $\mathbf{f}'(\mathbf{u})$ that is not carried as part of $\mathbf{u}$ is communicated through PETSc via a user-defined "context" that encapsulates application-specific data. (Such information typically includes dimensioning data, grid data, physical parameters, and quantities that could be derived from the state $\mathbf{u}$, but are most conveniently stored instead of recalculated, such as constitutive quantities.)

When well tuned, large-scale PDE codes spend almost all of their time in two phases: flux computations to evaluate conservation law residuals, called "function evaluations" in Figure 4, where one aims to have such codes spent almost *all* their time, and sparse linear algebraic kernels, which are a fact of life in implicit

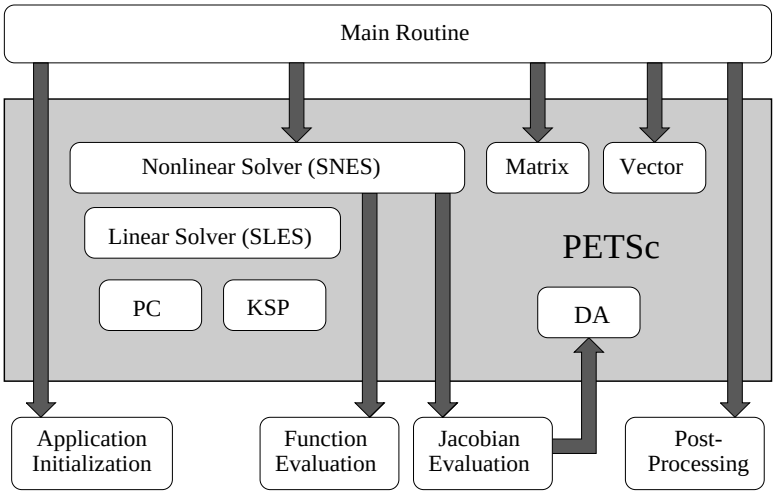

FIGURE 3. Coarsened calling tree of the FUN3D-PETSc code, showing Coarsened calling tree of the FUN3D-PETSc code, showing the user-supplied main program and callback routines for providing the initial nonlinear iterate, computing the nonlinear residual vector at a PETSc-requested state, and evaluating the Jacobian (preconditioner) matrix.

methods. Altogether, four basic groups of tasks can be identified based on the criteria of arithmetic concurrency, communication patterns, and the ratio of operation complexity to data size within the task. These four distinct phases, present in most implicit codes, are vertex-based loops, edge-based loops, recurrences, and global reductions. Each of these groups of tasks stresses a different subsystem of contemporary high-performance computers. Analysis of our demonstration code shows that, after tuning, the linear algebraic kernels run at close to the aggregate memory bandwidth limit on performance, the flux computations are bounded either by memory bandwidth or instruction scheduling (depending upon the ratio of load/store units to floating-point units in the CPU), and parallel efficiency is bounded primarily by slight load imbalances at synchronization points.

5.1. PDE Complexity Analysis. As mentioned above, there are four groups of tasks in a typical PDE solver, each with a distinct proportion of work to datasize to communication requirements. In the language of a vertex-centered code, in which the data is stored at cell vertices, these tasks are as follows:

- Vertex-based loops
  - state vector and auxiliary vector updates
- Edge-based "stencil op" loops
  - residual evaluation, Jacobian evaluation
  - Jacobian-vector product (often replaced with matrix-free form, involving residual evaluation)
  - interpolation between grid levels
- Sparse, narrow-band recurrences
  - (approximate) factorization, back substitution, relaxation/smoothing

- vector inner products and norms
  - orthogonalization/conjugation
  - convergence progress checks and stability heuristics

Vertex-based loops are characterized by work closely proportional to datasize, pointwise concurrency, and no communication.

Edge-based "stencil op" loops have a large ratio of work to datasize, since each vertex is used in many discrete stencil operations, and each degree of freedom at a point (momenta, energy, density, species concentration) generally interacts with all others in the conservation laws—through constitutive and state relationships or directly. There is concurrency at the level of the number of edges between vertices (or, at worst, the number of edges of a given "color" when write consistency needs to be protected through mesh coloring). There is local communication between processors sharing ownership of the vertices in a stencil.

Sparse, narrow-band recurrences involve work closely proportional to data size, the matrix being the largest data object and each of its elements typically being used once. Concurrency is at the level of the number of fronts in the recurrence, which may vary with the level of exactness of the recurrence. In a preconditioned iterative method, the recurrences are typically broken to deliver a prescribed process concurrency; only the quality of the preconditioning is thereby affected, not the final result. Depending upon whether one uses a pure decomposed Schwarz-type preconditioner, a truncated incomplete solve, or an exact solve, there may be no, local only, or global communication in this task.

Vector inner products and norms involve work closely proportional to data size, mostly pointwise concurrency, and global communication. Unfortunately, inner products and norms occur rather frequently in stable, robust linear and nonlinear methods (recall the pseudo-code example in Subsection 3.1).

Based on these characteristics, one anticipates that vertex-based loops, recurrences, and inner products will be *memory bandwidth-limited*, whereas edge-based loops are likely to be only *load/store-limited*. However, edge-based loops are vulnerable to *internode bandwidth* if the latter does not scale. Inner products are vulnerable to *internode latency* and *network diameter*. Recurrences can resemble some combination of edge-based loops and inner products in their communication characteristics if preconditioning fancier than simple Schwarz is employed. For instance, if incomplete factorization is employed globally or a coarse grid is used in a multilevel preconditioner, global recurrences ensue.

5.2. Test Problem. Our demonstration application code, FUN3D, is a tetrahedral, vertex-centered unstructured mesh code originally developed by W. K. Anderson of the NASA Langley Research Center for compressible and incompressible Euler and Navier-Stokes equations [**2, 3**]. FUN3D uses a control volume discretization with a variable-order Roe scheme for approximating the convective fluxes and a Galerkin discretization for the viscous terms. FUN3D has been used for design optimization of airplanes, automobiles, and submarines, with irregular meshes comprising several million mesh points. The optimization involves many analyses, typically sequential. Thus, reaching the steady-state solution in each analysis cycle in a reasonable amount of time is crucial to conducting the design optimization. We test the code on the ONERA M6 wing, a standard three-dimensional test case, for which extensive experimental data is given in [**114**]. A frequently studied parameter combination combines a freestream Mach number of 0.84 with an angle of

attack of 3.06^o. This transonic case gives rise to a characteristic λ-shock. Our best achievement to date for multimillion meshpoint simulations is about 15 μsec per degree of freedom for satisfaction of residuals close to machine precision.

We have ported FUN3D into the PETSc [**4**] framework using the single program multiple data (SPMD) message-passing programming model, supplemented by multithreading at the physically shared memory level. Thus far, our large-scale parallel experience with PETSc-FUN3D is with compressible or incompressible Euler flows, but nothing in the solution algorithms or software changes when additional physical phenomenology present in the original FUN3D is included. Of course, the convergence rate varies with conditioning, as determined by Mach and Reynolds numbers and the correspondingly induced mesh adaptivity. Robustness becomes an issue in problems that admit shocks or employ turbulence models. When nonlinear robustness is restored in the usual manner, through pseudo-transient continuation, the conditioning of the linear inner iterations is enhanced, and parallel scalability may be improved. In some sense, the subsonic Euler examples on which we concentrate, with their smaller number of flops per point per iteration and their aggressive pseudotransient buildup toward the steady-state limit, may be a *more* severe test of parallel performance than more physically complex cases.

Achieving high sustained performance, in terms of solutions per second, requires attention to three factors. The first is a scalable implementation, in the sense that time per iteration is reduced in inverse proportion to the number of processors, or that time per iteration is constant as problem size and processor number are scaled proportionally. The second is good per-processor performance on contemporary cache-based microprocessors. The third is algorithmic scalability, in the sense that the number of iterations to convergence does not grow with increased numbers of processors. This factor arises because the requirement of a scalable implementation generally forces parameterized changes in the algorithm as the number of processors grows. If the convergence is allowed to degrade, however, the overall execution is not scalable, and this must be countered algorithmically. These factors in the overall performance are considered in [**60**], from which some performance results are cited here.

PETSc features distributed data structures—index sets, vectors, and matrices—as fundamental objects. Iterative linear and nonlinear solvers are implemented within PETSc in a data structure-neutral manner, providing a uniform application programmer interf ace. We use MeTiS [**70**] to partition the unstructured mesh. The basic philosophy of any efficient parallel computation is "owner computes," with message merging and overlap of communication with computation where possible via split transactions. Each processor "ghosts" its stencil dependencies on its neighbors' data. Grid functions are mapped from a global (user) ordering into contiguous local orderings (which, in unstructured cases, are designed to maximize spatial locality for cache line reuse). Scatter/gather operations are created between local sequential vectors and global distributed vectors, based on runtime-deduced connectivity patterns.

5.3. Implementation Scalability. Domain-decomposed parallelism for PDEs is a natural means of overcoming Amdahl's law in the limit of fixed problem size per processor. As noted in Subsection 2.6, computational work on each evaluation of the conservation residuals scales as the volume of the (equal-sized) subdomains, whereas communication overhead scales only as the surface. This ratio is fixed when

TABLE 3. Scalability bottlenecks on ASCI Red for a fixed-size 2.8M vertex mesh. The preconditioner used in these results is block Jacobi with ILU(1) in each subdomain. We observe that the principal nonscaling factor is the implicit synchronization.

	Percentage of Time		
Number of Processors	Global Reductions	Implicit Synchronizations	Ghost Point Scatters
128	5	4	3
512	3	7	5
3072	5	14	10

problem size and processors are scaled in proportion, leaving only global reduction operations over all processors as an impediment to perfect performance scaling.

In [**78**], it is shown that on contemporary tightly coupled parallel architectures in which the number of connections between processors grows in proportion to the number of processors, such as meshes and tori, aggregate internode bandwidth is more than sufficient, and limits to scalability may be determined by a balance of work per node to synchronization frequency. On the other hand, if there is nearest-neighbor communication contention, as when a fixed resource like an ethernet switch is divided among all processors, the number of processors is allowed to grow only as the one-fourth power of the problem size (in three dimensions). This is a curse of typical Beowulf-type clusters with inexpensive networks; we do not discuss the problem here, although it is an important practical limitation.

When the load is perfectly balanced (which is easy to achieve for static meshes) and local communication is not an issue because the network is scalable, the optimal number of processors is related to the network diameter. For logarithmic networks, like a hypercube, the optimal number of processors, P, grows directly in proportion to the problem size, N. For a d-dimensional torus network, $P \propto N^{d/d+1}$. The proportionality constant is a ratio of work per subdomain to the product of synchronization frequency and internode communication latency. In Table 3, we present a closer look at the relative cost of computation for PETSc-FUN3D for a fixed-size problem of 2.8 million vertices on the ASCI Red machine, from 128 to 3072 nodes. The intent here is to identify the factors that retard the scalability. The overall parallel efficiency (denoted by $\eta_{overall}$) is broken into two factors: η_{alg} measures the degradation in the parallel efficiency due to the increased iteration count of this (non-coarse-grid-enhanced) NKS algorithm as the number of subdomains increases, while η_{impl} measures the degradation coming from all other nonscalable factors such as global reductions, load imbalance (implicit synchronizations), and hardware limitations.

From Table 3, we observe that the buffer-to-buffer time for global reductions for these runs is relatively small and does not grow on this excellent network. The primary factors responsible for the increased overhead of communication are the implicit synchronizations and the ghost point updates (interprocessor data scatters).

Interestingly, the increase in the percentage of time (3% to 10%) for the scatters results more from algorithmic issues than from hardware/software limitations. With

an increase in the number of subdomains, the percentage of grid point data that must be communicated also rises. For example, the total amount of nearest neighbor data that must be communicated per iteration for 128 subdomains is 2 gigabytes, while for 3072 subdomains it is 8 gigabytes. Although more network wires are available when more processors are employed, scatter time increases. If problem size and processor count are scaled together, we would expect scatter times to occupy a fixed percentage of the total and load imbalance to be reduced at high granularity.

Mesh partitioning has a dominant effect on parallel scalability for problems characterized by (almost) constant work per point. As shown above, poor load balance causes idleness at synchronization points, which are frequent in implicit methods (e.g., at every conjugation step in a Krylov solver). With NKS methods, then, it is natural to strive for a very well balanced load. The p-MeTiS algorithm in the MeTiS package [**70**], for example, provides almost perfect balancing of the number of mesh points per processor. However, balancing work alone is not sufficient. Communication must be balanced as well, and these objectives are not entirely compatible. Figure 4 shows the effect of data partitioning using p-MeTiS, which tries to balance the number of nodes and edges on each partition, and k-MeTiS, which tries to reduce the number of noncontiguous subdomains and connectivity of the subdomains. Better overall scalability is observed with k-MeTiS, despite the better load balance for the p-MeTiS partitions. This is due to the slightly poorer numerical convergence rate of the iterative NKS algorithm with the p-MeTiS partitions. The poorer convergence rate can be explained by the fact that the p-MeTiS partitioner generates disconnected pieces within a single "subdomain," effectively increasing the number of blocks in the block Jacobi or additive Schwarz algorithm and increasing the size of the interface. The convergence rates for one-level block iterative methods degrade with increasing number of blocks, as discussed in conjunction with Table 1.

5.4. Algorithmic Tuning for ΨNKS Solver. The following is an incomplete list of parameters that need to be tuned in various phases of a pseudo-transient Newton-Krylov-Schwarz algorithm.

- Nonlinear robustness continuation parameters: discretization order, initial time step, pseudo-time step evolution law
- Newton parameters: convergence tolerance on each time step, globalization strategy (line search or trust region parameters), refresh frequency for Jacobian preconditioner
- Krylov parameters: convergence tolerance for each Newton correction, restart dimension of Krylov subspace, overall Krylov iteration limit, orthogonalization mechanism
- Schwarz parameters: subdomain number, quality of subdomain solver (fill level, number of sweeps), amount of subdomain overlap, coarse grid usage
- Subproblem parameters: fill level, number of sweeps

5.4.1. *Parameters for Pseudo-Transient Continuation.* Although asymptotically superlinear, solution strategies based on Newton's method must often be approached through pseudo-time stepping. For robustness, pseudo-time stepping is often initiated with very small time steps and accelerated subsequently. However, this

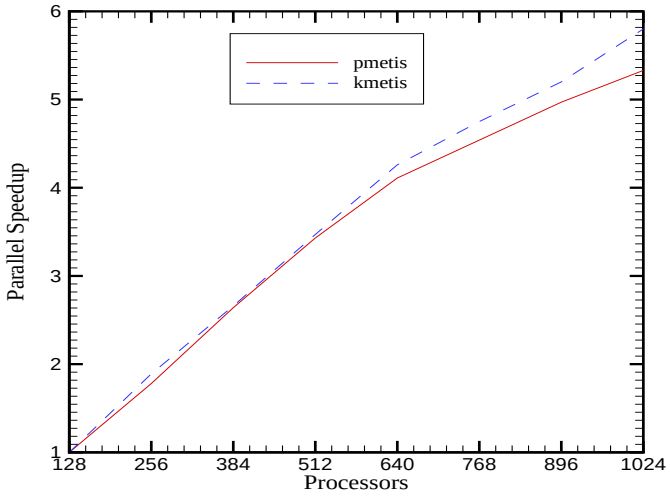

FIGURE 4. Parallel speedup relative to 128 processors on a 600 MHz Cray T3E for a 2.8M vertex case, showing the effect of partitioning algorithms k-MeTiS, and p-MeTiS.

conventional approach can lead to long "induction" periods that may be bypassed by a more aggressive strategy, especially for the smooth flow fields.

The time step is advanced toward infinity by a power-law variation of the switched evolution/relaxation (SER) heuristic of Van Leer and Mulder [**98**]. To be specific, within each residual reduction phase of computation, we adjust the time step according to

$$N_{CFL}^{\ell} = N_{CFL}^{0} \left(\frac{\|f(u^0)\|}{\|f(u^{\ell-1})\|} \right)^p ,$$

where p is a tunable exponent close to unity. Figure 5 shows the surprisingly sensitive effect of initial CFL number (a dimensionless measure of the time step size), N_{CFL}^0, on the convergence rate. In general, the best choice of initial CFL number is dependent on the grid size and Mach number. A small CFL adds nonlinear stability far from the solution but retards the approach to the domain of superlinear convergence of the steady state. For flows with near discontinuities, it is usually safer to start with small CFL numbers; however, this figure shows that such a strategy may not lead to the best execution time.

In flows with shocks, high-order (second or higher) discretization for the convection terms should be activated only after the shock position has settled down. We begin such simulations with a first-order upwind scheme and switch to second-order after a certain residual reduction. The exponent (p) in the power law above is damped to 0.75 for robustness when shocks are expected to appear in second-order discretizations. For first-order discretizations, this exponent may be as large as 1.5. A reasonable switchover point of the residual norm between first-order and second-order discretization phases has been determined empirically. In shock-free

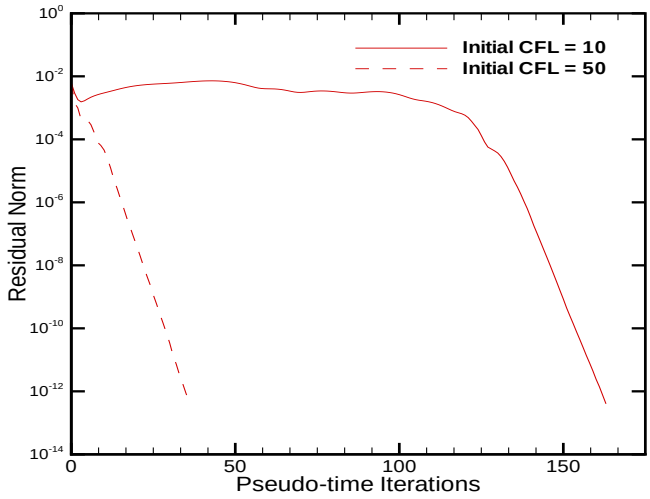

FIGURE 5. Residual norm versus iteration count for a 2.8M-vertex case, showing the effect of initial CFL number on convergence rate. The convergence tuning of nonlinear problems is notoriously case specific.

simulations we use second-order accuracy throughout. Otherwise, we normally reduce the first two to four orders of residual norm with the first-order discretization, then switch to second. This order of accuracy applies to the flux calculation. The preconditioner matrix is always built out of a first-order analytical Jacobian matrix.

5.4.2. *Parameters for Krylov Solver.* We use an inexact Newton method on each time step; that is, the linear system within each Newton iteration is solved only approximately. Especially in the beginning of the solution process, this saves a significant amount of execution time. We have considered the following three parameters in this phase of computation: convergence tolerance, the number of simultaneously storable Krylov vectors, and the total number of Krylov iterations. The typical range of variation for the inner convergence tolerance is 0.001–0.01. We have experimented with progressively tighter tolerances near convergence, and saved Newton iterations thereby, but did not save time relative to cases with loose and constant tolerance. The Krylov subspace dimension depends largely on the problem size and the available memory. We have used values in the range of 10–30 for most of the problems. The total number of linear iterations (within each nonlinear solve) has been varied from 10 for the smallest problem to 80 for the largest one. A typical number of fine-grid flux evaluations for achieving 10^{-10} residual reduction on a million-vertex Euler problem is a couple of thousand.

5.4.3. *Additive Schwarz Preconditioner.* Table 4 explores two quality parameters for the additive Schwarz preconditioner: subdomain overlap and quality of the subdomain solve using incomplete factorization. We exhibit execution time and iteration count data from runs of PETSc-FUN3D on the ASCI Red machine for a fixed-size problem with 357,900 grid points and 1,789,500 degrees of freedom.

These calculations were performed using GMRES(20), one subdomain per processor (without overlap for block Jacobi and with overlap for ASM), and ILU(k) where k varies from 0 to 2, and with the natural ordering in each subdomain block. The use of ILU(0) with natural ordering on the first-order Jacobian, while applying a second-order operator, allows the factorization to be done in place, with or without overlap. However, the overlap case does require forming an additional data structure on each processor to store matrix elements corresponding to the overlapped regions.

From Table 4, we see that larger overlap and more fill help in reducing the total number of linear iterations as the number of processors increases, as theory and intuition predict. However, both increases consume more memory, and both result in more work per iteration, ultimately driving up execution times in spite of faster convergence. Best execution times are obtained for any given number of processors for ILU(1), as the number of processors becomes large (subdomain size small), for zero overlap.

The additional computation/communication costs for additive Schwarz (as compared with block Jacobi) are the following:

(1) Calculation of the matrix couplings among processors. For block Jacobi, these need not be calculated.
(2) Communication of the "overlapped" matrix elements to the relevant processors.
(3) Factorization of the larger local submatrices.
(4) Communication of the ghost points in the application of the ASM preconditioner. We use restricted additive Schwarz method (RASM) [**36**], which communicates only when setting up the overlapped subdomain problems and ignores the updates coming from the overlapped regions. This saves a factor of two in local communication relative to standard ASM.
(5) Inversion of larger triangular factors in each iteration.

The execution times reported in Table 4 are highly dependent on the machine used, since each of the additional computation/communication costs listed above may shift the computation past a knee in the performance curve for memory bandwidth, communication network, and so on.

5.4.4. *Other Algorithmic Tuning Parameters.* In [**59**] we highlight some additional tunings that have yielded good results in our context. Some subsets of these parameters are not orthogonal but interact strongly with each other. In addition, optimal values of some of these parameters depend on the grid resolution.

We emphasize that the discussion in this section does not pertain to discretization parameters, which constitute another area of investigation — one that ultimately impacts performance at a higher level. The algorithmic parameters discussed in this section do not affect the accuracy of the discrete solution, but only the rate at which the solution is attained. In all of our experiments, the goal has been to minimize the overall execution time, not to maximize the floating-point operations per second. There are many tradeoffs that enhance Mflop/s rates but retard execution completion.

5.5. Large-Scale Demonstration Runs. We use PETSc's profiling and logging features to measure the parallel performance. PETSc logs many different types of events and provides valuable information about time spent, communications, load

TABLE 4. Execution times and linear iteration counts on the 333 MHz Pentium Pro ASCI Red machine for a 357,900-vertex case, showing the effect of subdomain overlap and incomplete factorization fill level in the additive Schwarz preconditioner. **The best execution times for each ILU fill level and number of processors are in boldface in each row.**

ILU(0) in Each Subdomain						
Number of Processors	Overlap					
	0		1		2	
	Time	Linear Its	Time	Linear Its	Time	Linear Its
32	688s	930	**661s**	816	696s	813
64	**371s**	993	374s	876	418s	887
128	**210s**	1052	230s	988	222s	872
ILU(1) in Each Subdomain						
Number of Processors	Overlap					
	0		1		2	
	Time	Linear Its	Time	Linear Its	Time	Linear Its
32	598s	674	**564s**	549	617s	532
64	**334s**	746	335s	617	359s	551
128	**177s**	807	178s	630	200s	555
ILU(2) in Each Subdomain						
Number of Processors	Overlap					
	0		1		2	
	Time	Linear Its	Time	Linear Its	Time	Linear Its
32	**688s**	527	786s	441	—	—
64	**386s**	608	441s	488	531s	448
128	**193s**	631	272s	540	313s	472

balance, and so forth for each logged event. PETSc uses manual counting of flops, which are afterwards aggregated over all the processors for parallel performance statistics. We have observed that the flops reported by PETSc are close to (within 10% of) the values statistically measured by hardware counters on the R10000 processor.

PETSc uses the best timers available at the user level in each processing environment. In our rate computations, we exclude the initialization time devoted to I/O and data partitioning. To suppress timing variations caused by paging in the executable from disk, we preload the code into memory with one nonlinear iteration, then flush, reload the initial iterate, and begin performance measurements.

Since we are solving large fixed-size problems on distributed-memory machines, it is not reasonable to base parallel scalability on a uniprocessor run, which would thrash the paging system on a single node, inflating the parallel efficiency of multiprocessor runs. Our base processor number is such that the problem has just fit into the local memory.

The same fixed-size problem is run on large ASCI Red configurations with sample scaling results shown in Figure 6. The implementation efficiency is 91% in going from 256 to 3072 nodes. For the data in Figure 6, we employed the `-procs 2` runtime option on ASCI Red. This option enables 2-processor-per-node multithreading during threadsafe, communication-free portions of the code. We have activated this

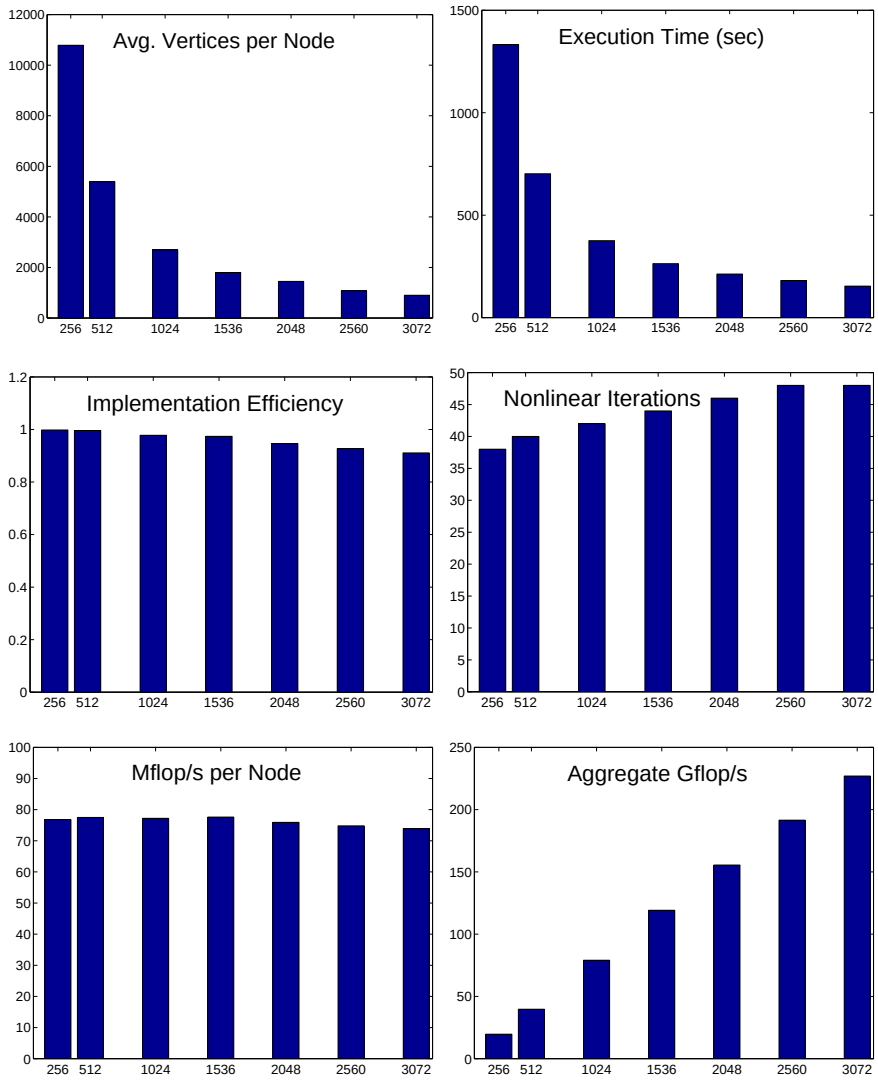

FIGURE 6. Parallel performance for a fixed size mesh of 2.8M vertices run on up to 3072 ASCI Red 333 MHz Pentium Pro processors.

feature for the floating-point-intensive flux computation subroutine alone. On 3072 nodes, the largest run we have been able to make on the unclassified side of the machine to date, the resulting Gflop/s rate is 227. Undoubtedly, further improvements to the algebraic solver portion of the code are also possible through multithreading, but the additional coding work does not seem justified at present.

Figure 8 shows aggregate flop/s performance and a log-log plot showing execution time for our largest case on the three most capable machines to which we have thus far had access. In both plots of this figure, the dashed lines indicate ideal behavior. Note that although the ASCI Red flop/s rate scales nearly linearly, a higher fraction of the work is redundant at higher parallel granularities, so the

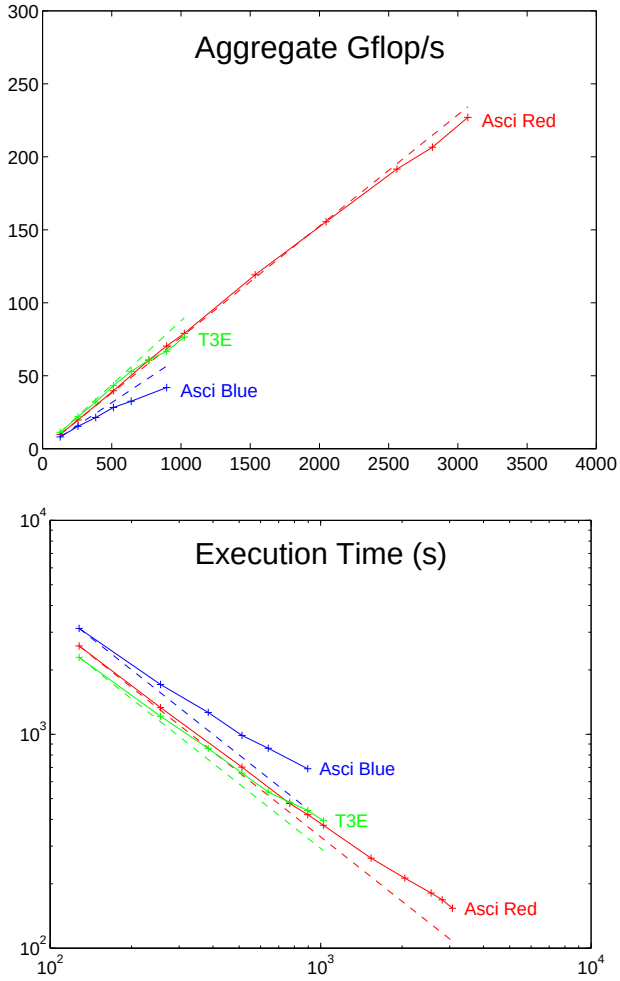

FIGURE 7. Gigaflop/s ratings and execution times on ASCI Red (up to 3072 2-processor nodes), ASCI Pacific Blue (up to 768 processors), and a Cray T3E (up to 1024 processors) for a 2.8M-vertex case, along with dashed lines indicating "perfect" scalings.

execution time does not drop in exact proportion to the increase in flop/s. The number of vertices per processor ranges from about 22,000 to fewer than 1,000 over the range shown. We point out that for just 1,000 vertices in a three-dimensional domain, about half are on the interface (e.g., 488 interface vertices on a $10 \times 10 \times 10$ cube).

6. Terascale Optimal PDE Simulations (TOPS)

In PDE simulations, infinite-dimensional continuous models are approximated with finite-dimensional models. To obtain the required accuracy and resolve the multiple scales of the underlying physics, the finite-dimensional models must often be extremely large, thus requiring terascale computers. Fortunately, continuous

problems provide a natural way to generate a hierarchy of approximate models, through which the required solution may be obtained efficiently by various forms of "bootstrapping." The most dramatic examples are multigrid methods, but other hierarchical representations are also exploitable.

Under the Scientific Discovery through Advanced Computing (SciDAC) initiative of the U.S. Department of Energy, a nine-institution team is building an integrated software infrastructure center (ISIC) that focuses on developing, implementing, and supporting optimal or near optimal schemes for PDE simulations and closely related tasks, including optimization of PDE-constrained systems, eigenanalysis, and adaptive time integration, as well as implicit linear and nonlinear solvers. The Terascale Optimal PDE Simulations (TOPS) Center is researching and developing and will deploy a toolkit of open source solvers for the nonlinear partial differential equations that arise in many application areas, including fusion, accelerator design, global climate change, and the collapse of supernovae. These algorithms — primarily multilevel methods — aim to reduce computational bottlenecks by one or more orders of magnitude on terascale computers, enabling scientific simulation on a scale heretofore impossible.

Along with usability, robustness, and algorithmic efficiency, an important goal of this ISIC is to attain the highest possible computational performance in its implementations by accommodating to the memory bandwidth limitations of hierarchical memory architectures.

PDE simulation codes require implicit solvers for multiscale, multiphase, multiphysics phenomena from hydrodynamics, electromagnetism, radiation transport, chemical kinetics, and quantum chemistry. Problem sizes are typically now in the millions of unknowns; and with emerging large-scale computing systems and inexpensive clusters, we expect this size to increase by a factor of a thousand over the next five years. Moreover, these simulations are increasingly used for design optimization, parameter identification, and process control applications that require many repeated, related simulations.

Unfortunately, the implicit solution algorithms currently used in many contemporary codes have far from optimal computational complexities and are invariably bottlenecks that limit the scalability of the entire application, independent of the quality of the implementation. For example, for an algorithm with a 3/2 exponent in the complexity an increase in problem size of a factor of 100 can easily result in an increase in work requirements of 1000. When such a simulation is run on a system with 100 times as many processors (to scale the problem size with the available memory), it requires at least ten times longer to run. In comparison, optimal complexity algorithms have work (and memory) requirements that grow only linearly or logarithmically with problem size. (Technically, there is at least a logarithm that comes into the per-iteration time for any application that performs global reductions, but the number of iterations should be constant.) Multilevel (or multigrid) methods make up a class of optimal complexity algorithms that have produced spectacular improvements in overall simulation time. However, current multilevel software tends to be problem-specific and is mature only for scalar (as opposed to multicomponent) PDEs. Because of the potential payoff, the TOPS ISIC will expend much of its effort on developing practical, usable multilevel methods for comprehensive aspects of PDE simulations.

The TOPS ISIC is concerned with five PDE simulation capabilities: adaptive time integrators for stiff systems, nonlinear implicit solvers, optimization, linear solvers, and eigenanalysis. The relationship between these areas is depicted in Figure 9. In addition, the ISIC will contain two cross-cutting topics: software integration (or interoperability) and high-performance coding techniques for PDE applications.

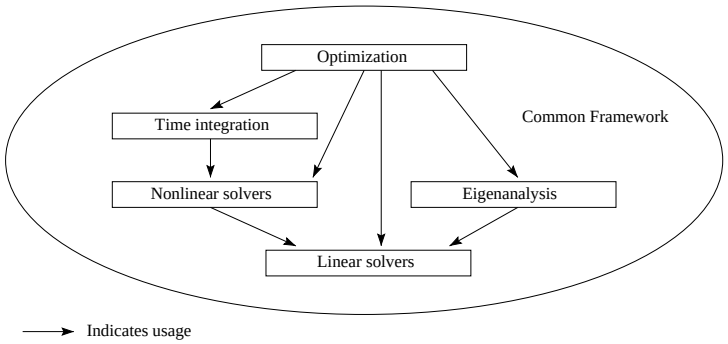

FIGURE 8. *An arrow from A to B indicates that A typically uses B. Optimization of systems governed by PDEs requires repeated access to a PDE solver. The PDE system may be steady-state or time-dependent. Time-dependent PDEs are typically solved with implicit temporal differencing. After choice of the time-integration scheme, they, in turn, require the same types of nonlinear solvers that are used to solve steady-state PDEs. Many algorithms for nonlinear problems of high dimension generate a sequence of linear problems, so linear solver capability is at the core. Eigenanalysis arises inside of or independently of optimization. Like direct PDE analysis, eigenanalysis generally depends upon solving a sequence of linear problems. All of these five classes of problems, in a PDE context, share grid-based data structures and considerable parallel software infrastructure. Therefore, it is compelling to undertake them together.*

Optimal (and nearly optimal) complexity numerical algorithms almost invariably depend upon a hierarchy of approximations to "bootstrap" to the required highly accurate final solution. Generally, an underlying continuum (infinite dimensional) high fidelity mathematical model of the physics is discretized to "high" order on a "fine" mesh to define the top level of the hierarchy of approximations. The representations of the problem at lower levels of the hierarchy may employ other models (possibly of lower physical fidelity), coarser meshes, lower order discretization schemes, inexact linearizations, and even lower floating-point precisions. The philosophy that underlies our algorithmics and software is the same as that of this chapter — to make the majority of progress towards the highly resolved result through possibly low-resolution stages that run well on high-end distributed hierarchical memory computers.

The ingredients for constructing hierarchy-of-approximations-based methods are remarkably similar, be it for solving linear systems, nonlinear problems, eigenvalue problems, or optimization problems, namely:

(1) A method for generating several discrete problems at different resolutions (for example on several grids),
(2) An inexpensive (requiring few floating point operations, loads, and stores per degree of freedom) method for iteratively improving an approximate solution at a particular resolution,
(3) A means of interpolating (discrete) functions at a particular resolution to the next finer resolution,
(4) A means of transferring (discrete) functions at a particular resolution to the next coarser resolution (often obtained trivially from interpolation).

Software should reflect the simplicity and uniformity of these ingredients over the five problem classes and over a wide range of applications. With experience we expect to achieve a reduction in the number of lines of code that need to be written and maintained, because the same code can be reused in many circumstances.

Algorithms and software for the solution of linear and nonlinear systems of equations, especially those arising from PDEs, have been principal emphases of the Department of Energy research portfolio for decades. This points both to the central importance of this project, and also to the historical difficulty of reconciling the conflicting objectives of solver software technology. Solvers are supposed to be of *general purpose*, since a great diversity of applications require them, but they are also supposed to be *highly performant*, since they are often the inner loops of such applications. However, high performance usually requires exploitation of special structure (e.g., symmetry, dense blocking, geometrical or coefficient regularity), which may be different in different applications. Then, too, solvers for PDEs are supposed to be *robust* across all regimes of use, since scientists trained in the application domain cannot also be required to be expert in tuning solvers, but they are also supposed to have *optimal complexity*, since desired discrete problem size is limited only by the validity of the continuum model. Once again, algorithmic optimality (work and memory requirements a small multiple of their information-theoretic minima) is generally achieved by exploitation of special structure that cannot be assumed in a robust code.

These conflicting objectives do not describe a hopeless situation, however. The opportunity for 21st century solver developers is to exploit advances in object-oriented programming to construct highly versatile and adaptive software that finds, creates, and exploits structure wherever possible, while automatically "falling back" to conservative approaches in the remaining (hopefully lower-dimensional) parts of a problem. The solver toolkit of the future will be a collection of objects with rich and recursive interconnections, rather than a collection of subroutines through which a relatively small number of calling sequences are predefined. Algorithmic theory, scientific software engineering, and understanding of architecturally-motivated performance optimizations have all advanced significantly since the last time many applications communities "fastened onto" their canonical solver technology. Advances along these three fronts must be packaged, refined, freshly promoted, and supported for the benefit of the user community.

The efforts defined for TOPS, the co-PIs joining to undertake them, and the alliances proposed with other groups have been chosen to exploit the present opportunity to revolutionize large-scale solver infrastructure, and lift the capabilities of dozens of DOE's computational science groups as an outcome. The co-PIs' current software (e.g., Hypre, PETSc, ScaLAPACK, SuperLU), though not algorithmically optimal in many cases, and not yet as interoperable as required, is in the hands of thousands of users, and has created a valuable experience base. Just as we expect the user community to drive research and development, we expect to significantly impact the scientific priorities of users by emphasizing optimization (inverse problems, optimal control, optimal design) and eigenanalysis as part of the solver toolkit.

Optimization subject to PDE-constraints is a particularly active subfield of optimization because the traditional means of handling constraints in black-box optimization codes — with a call to a PDE solver in the inner loop — is too expensive. We are emphasizing "simultaneous analysis and design" methods in which the cost of doing the optimization is a small multiple of doing the simulation and the simulation data structures are actually part of the optimization data structures.

Likewise, we expect that a convenient software path from PDE analysis to eigenanalysis will impact the scientific approach of users with complex applications. For instance, a PDE analysis can be pipelined into the scientific added-value tasks of stability analysis for small perturbations about a solution and reduced dimension representations (model reduction), with reuse of distributed data structures and solver components.

Our motivating belief is that most PDE simulation is ultimately a part of some larger scientific process that can be hosted by the same data structures and carried out with many of the same optimized kernels as the simulation, itself. We intend to make the connection to such processes explicit and inviting to users, and this will be a prime metric of our success. The organization of the effort flows directly from this program of "holistic simulation": Terascale software for PDEs should extend from the analysis to the scientifically important auxiliary processes of sensitivity analysis, modal analysis and the ultimate "prize" of optimization subject to conservation laws embodied by the PDE system.

7. Conclusions

The emergence of the nonlinearly implicit Jacobian-free Newton-Krylov-Schwarz family of methods has provided a pathway towards terascale simulation of PDE-based systems.

Large-scale implicit computations have matured to a point of practical use on distributed/shared memory architectures for static-grid problems. More sophisticated algorithms, including solution adaptivity, inherit the same features *within* static-grid phases, of course, but require extensive additional infrastructure for dynamic parallel adaptivity, rebalancing, and maintenance of efficient, consistent distributed data structures.

While mathematical theory has been crucial in the development of NKS methods, their most successful application also depends upon a more-than-superficial understanding of the underlying architecture and of the physics being modeled. In the future, as we head towards petascale simulation and greater integration of

complex physics codes in full system analysis and optimization, we expect that this interdisciplinary interdependence will only increase.

Acknowledgements

The author thanks Xiao-Chuan Cai, Omar Ghattas, Bill Gropp, Dana Knoll, and Barry Smith for long-term collaborations on parallel algorithms, and Satish Balay, Paul Hovland, Dinesh Kaushik, and Lois McInnes from the PETSc team at Argonne National Laboratory (along with Gropp and Smith and others) for their wizardry in implementation and making the applications described in this paper possible, and to Widodo Samyono for nascent collaborations on the LNKS algorithm.

References

1. K. Ajmani, W.-F. Ng, and M.-S. Liou, *Preconditioned conjugate gradient methods for the Navier-Stokes equations*, J. Comp. Phys. **110** (1994), 68–81.
2. W. K. Anderson and D. L. Bonhaus, *An implicit upwind algorithm for computing turbulent flows on unstructured grids*, Comp. Fluids **23** (1994), 1–21.
3. W. K. Anderson, R. D. Rausch, and D. L. Bonhaus, *Implicit/multigrid algorithms for incompressible turbulent flows on unstructured grids*, J. Comput. Phys. **128** (1996), 391–408.
4. S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, *PETSc 2.0 users manual*, Tech. Report ANL-95/11, Mathematics and Computer Science Division, Argonne National Laboratory, 1995, (see `http://www.mcs.anl.gov/petsc/`).
5. ———, *Efficient management of parallelism in object-oriented numerical software libraries*, Modern Software Tools in Scientific Computing, Birkhauser, 1997, pp. 163–201.
6. ———, *The Portable, Extensible Toolkit for Scientific Computing, version 2.3.1*, `http://www.mcs.anl.gov/petsc/`, 2002.
7. R.E. Bank, T.F. Chan, W.M. Coughran, and R.K. Smith, *The alternate block factorization procedure for systems of partial differential equations*, BIT **29** (1989), 938–954.
8. T. J. Barth and S. W. Linton, *An unstructured mesh Newton solver for fluid flow and its parallel implementation*, Tech. Report 95-0221, AIAA, 1995.
9. A. Battermann and M. Heinkenschloss, *Preconditioners for Karush-Kuhn-Tucker matrices arising in optimal control of distributed systems*, Tech. Report TR96-34, Dept. of Computational and Applied Mathematics, Rice University, 1996.
10. M. Bhardwaj, D. Day, C. Farhat, M. Lesoinne, K. Pierson, and D. Rixen, *Application of the FETI method to ASCI problems: Scalability results on one thousand processors and discussion of highly heterogeneous problems*, Int. J. Numer. Meths. Engineering **47** (2000), 513–536.
11. G. Biros and O. Ghattas, *Parallel Newton-Krylov methods for PDE-constrained optimization*, Proceedings of SC99, IEEE Computer Society, 1999.
12. ———, *A Lagrange-Newton-Krylov-Schur method for PDE-constrained optimization*, SIAG/OPT Views-and-News **11** (2000), no. 2, 1–6.
13. C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *ADIFOR - generating derivative codes from Fortran programs*, Scientific Programming **1** (1992), 1–29.
14. C. Bischof, L. Roh, and A. Mauer, *ADIC — An extensible automatic differentiation tool for ANSI-C*, Software–Practice and Experience **27** (1997), 1427–1456.
15. P. E. Bjorstad and O. B. Widlund, *To overlap or not to overlap: A note on a domain decomposition method for elliptic problems*, SIAM J. Sci. Stat. Comput. **10** (1989), 1053–1061.
16. R. L. Bowers and J. R. Wilson, *Numerical Modeling in Applied Physics and Astrophysics*, Jones and Bartlett, 1991.
17. A. Brandt, *Multi-level adaptive solutions to boundary value problems*, Math. Comp. **31** (1977), 333.
18. ———, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics*, Tech. report, von Karman Institute, 1984.
19. P. N. Brown and A. C. Hindmarsh, *Matrix-free methods for stiff systems of ODE's*, SIAM J. Numer. Anal. **23** (1986), 610–638.

20. P. N. Brown and Y. Saad, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput. **11** (1990), 450–481.
21. P. N. Brown and C. S. Woodward, *Preconditioning strategies for fully implicit radiation diffusion with material-energy transfer*, Tech. Report UCRL-JC-139087, Lawrence Livermore National Laboratory, 2000.
22. R. H. Byrd, J. Nocedal, and R. B. Schnabel, *Representations of quasi-Newton matrices and their use in limited-memory methods*, Math. Prog., Ser. A **63** (1994), 129–156.
23. X.-C. Cai, *Some domain decomposition algorithms for nonselfadjoint elliptic and parabolic partial differential equations*, Technical Report 461, Courant Institute, 1989.
24. ———, *An optimal two-level overlapping domain decomposition method for elliptic problems in two and three dimensions*, SIAM J. Sci. Comput. **14** (1993), 239–247.
25. ———, *Multiplicative schwarz methods for parabolic problems*, SIAM J. Sci. Comput. **15** (1994), 587–603.
26. X.-C. Cai, M. Dryja, and M. Sarkis, *RASHO: A restricted additive Schwarz preconditioner with harmonic overlap*, Proceedings of the 13th International Conference on Domain Decomposition Methods, Domain Decomposition Press, 2002.
27. X.-C. Cai, C. Farhat, and M. Sarkis, *Schwarz methods for the unsteady compressible Navier-Stokes equations on unstructured meshes*, Proceedings of the Eighth International Conference on Domain Decomposition Methods, Wiley, 1997, pp. 453–460.
28. ———, *A minimum overlap restricted additive Schwarz preconditioner and applications in 3D flow simulations*, Proceedings of the Tenth International Conference on Domain Decomposition Methods, AMS, 1998, pp. 238–244.
29. X.-C. Cai, W. D. Gropp, and D. E. Keyes, *An comparison of some domain decomposition and ilu preconditioned iterative methods for nonsymmetric elliptic problems*, Numer. Lin. Alg. Applics. **1** (1994), 477–504.
30. X.-C. Cai, W. D. Gropp, D. E. Keyes, R. G. Melvin, and D. P. Young, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput. **19** (1998), 246–265.
31. X.-C. Cai, W. D. Gropp, D. E. Keyes, and M. D. Tidriri, *Newton-Krylov-Schwarz methods in CFD*, Proceedings of the International Workshop on Numerical Methods for the Navier-Stokes Equations, Vieweg, 1995, pp. 17–30.
32. X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, to appear in SIAM J. Sci. Comput., 2002.
33. X.-C. Cai, D. E. Keyes, and L. Marcinkowski, *Nonlinear additive schwarz preconditioners and applications in computational fluid dynamics*, (to appear in Int. J. of Numerical Methods in Fluids), 2002.
34. X.-C. Cai, D. E. Keyes, and V. Venkatakrishnan, *Newton-Krylov-Schwarz: An implicit solver for CFD*, Proceedings of the Eighth International Conference on Domain Decomposition Methods, Wiley, 1997, pp. 387–400.
35. X.-C. Cai, D. E. Keyes, and D. P. Young, *A nonlinearly additive Schwarz preconditioned inexact newton method for shocked duct flow*, Proceedings of the 13th International Conference on Domain Decomposition Methods, Domain Decomposition Press, 2002.
36. X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput. **21** (1999), 792–797.
37. X.-C. Cai and O. B. Widlund, *Domain decomposition algorithms for indefinite elliptic problems*, SIAM J. Sci. Stat. Comput. **13** (1992), 243–258.
38. ———, *Multiplicative Schwarz algorithms for nonsymmetric and indefinite elliptic problems*, SIAM J. Numer. Anal. **30** (1993), 936–952.
39. L. Chacon, D. A. Knoll, and J. M. Finn, *An implicit nonlinear resistive and Hall MHD solver*, J. Comput. Phys. (2001), in review.
40. T. F. Chan and D.Goovaerts, *On the relationship between overlapping and nonoverlapping domain decomposition methods*, SIAM J. Matrix Anal. Applics. **13** (1992), 663–675.
41. T. F. Chan and K. R. Jackson, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Stat. Comput. **5** (1984), 535–542.
42. T. S. Coffey, C. T. Kelley, and D. E. Keyes, *Pseudo-transient continuation and differential-algebraic equations*, submitted to SIAM J. Sci. Comput., 2002.

43. N. Débit, M. Garbey, R. Hoppe, D. E. Keyes, Y. Kuznetsov, and J. Périaux, *Proceedings of the Thirteenth International Conference on Domain Decomposition Methods*, Domain Decomposition Press, 2002.
44. R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal. **19** (1982), 400–408.
45. J. E. Dennis and R. B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, Prentice-Hall, 1983.
46. J. E. Dennis and V. Torczon, *Direct search methods on parallel machines*, SIAM J. Optimization **1** (1991), 448–474.
47. P. Deuflhard, *Adaptive pseudo-transient continuation for nonlinear steady state problems*, Tech. Report ZIB-Report 02-14, Konrad-Zuse-Zentrum, March 2002.
48. M. Dryja and O. B. Widlund, *An additive variant of the Schwarz alternating method for the case of many subregions*, Tech. Report 339, Department of Computer Science, Courant Institute, 1987.
49. ———, *Schwarz methods of Neumann-Neumann type for three-dimensional elliptic finite element problems*, Commun. Pure Appl. Math. **48** (1995), 121–155.
50. S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput. **17** (1996), 16–32.
51. A. Ern, V. Giovangigli, D. E. Keyes, and M. D. Smooke, *Towards polyalgorithmic linear system solvers for nonlinear elliptic systems*, SIAM J. Sci. Comput. **15** (1994), 681–703.
52. C. Farhat, M. Lesoinne, P. LeTallec, K. Pierson, and D. Rixen, *FETI-DP: A dual-primal unified feti method - part i: A faster alternative to the two-level feti method*, Int. J. Numer. Meths. Engineering **50** (2001), 1523–1544.
53. R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems*, SIAM J. Sci. Stat. Comput. **14** (1993), 470–482.
54. C. W. Gear and Y. Saad, *Iterative solution of linear equations in ode codes*, SIAM J. Sci. Stat. Comput. **4** (1983), 583–601.
55. A. Griewank, *Evaluating derivatives: Principles and techniques of algorithmic differentiation*, SIAM, 2000.
56. W. D. Gropp and D. E. Keyes, *Complexity of parallel implementation of domain decomposition techniques for elliptic partial differential equations*, SIAM J. Sci. Stat. Comput. **9** (1988), 312–326.
57. ———, *Domain decomposition on parallel computers*, Impact of Computing in Science and Engineering **1** (1989), 421–439.
58. W. D. Gropp, D. E. Keyes, and J. S. Mounts, *Implicit domain decomposition algorithms for steady, compressible aerodynamics*, Sixth International Symposium on Domain Decomposition Methods, AMS, 1994, pp. 203–213.
59. W. D. Gropp, D. K. Kaushik D. E. Keyes, and B. F. Smith, *Performance modeling and tuning of an unstructured mesh CFD application*, Proceedings of SC2000, IEEE Computer Society, 2000.
60. ———, *High performance parallel implicit CFD*, Parallel Computing **27** (2001), 337–362.
61. W. D. Gropp, L. C. McInnes, M. D. Tidriri, and D. E. Keyes, *Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD*, Int. J. High Performance Computing Applications **14** (2000), 102–136.
62. W. Hackbusch, *Iterative methods for large sparse linear systems*, Springer, 1993.
63. F. H. Harlow and A. A. Amsden, *A numerical fluid dynamical calculation method for all flow speeds*, J. Comput. Phys. **8** (1971), 197–214.
64. A. C. Hindmarsh and A. G. Taylor, *PVODE and KINSOL: Parallel software for differential and nonlinear systems*, Tech. report, Lawrence Livermore National Laboratory, 1998.
65. S. A. Hutchinson, J. N. Shadid, and R. S. Tuminaro, *Aztec user's guide: Version 1.1*, Tech. Report SAND95-1559, Sandia National Laboratories, October 1995.
66. H. Jiang and P. A. Forsyth, *Robust linear and nonlinear strategies for solution of the transonic Euler equations*, Computers and Fluids **24** (1995), 753–770.
67. Z. Johann, T. J. R. Hughes, and F. Shakib, *A globally convergent matrix-free algorithm for implicit time-marching schemes arising in finite element analysis in fluids*, Computational Methods in Applied Mechanics and Engineering **87** (1991), 281–304.
68. J.E. Jones and C.S. Woodward, *Newton-Krylov-multigrid solvers for large scale, highly heterogeneous, variably saturated flow problems*, SIAM J. Sci. Stat. Comput. (submitted, 2001).

69. J. E. Dennis Jr. and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, 1983.
70. G. Karypis and V. Kumar, *A fast and high quality scheme for partitioning irregular graphs*, SIAM J. Scientific Computing **20** (1999), 359–392.
71. D. K. Kaushik, D. E. Keyes, and B. F. Smith, *Newton-Krylov-Schwarz methods for aerodynamic problems: Compressible and incompressible flows on unstructured grids*, Proceedings of the 11th International Conference on Domain Decomposition Methods, Domain Decomposition Press, 1999, pp. 513–520.
72. C. T. Kelley, *Iterative methods for linear and nonlinear equations*, SIAM, 1995.
73. ———, *Iterative methods for linear and nonlinear equations*, SIAM, 1995.
74. C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Numer. Anal. **35** (1998), 508–523.
75. ———, *Convergence analysis of pseudo-transient continuation*, SIAM J. Num. Anal. **35** (1998), 508–523.
76. D. E. Keyes, *Domain decomposition methods for the parallel computation of reacting flows*, Computer Physics Commun. **53** (1989), 181–200.
77. ———, *Aerodynamic applications of Newton-Krylov-Schwarz solvers*, Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics, Springer, 1995, pp. 1–20.
78. ———, *How scalable is domain decomposition in practice?*, Proceedings of the 11th International Conference on Domain Decomposition Methods, Domain Decomposition Press, 1999.
79. D. E. Keyes and W. D. Gropp, *A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation*, SIAM J. Sci. Stat. Comput. **8** (1987), no. 2, s166–s202.
80. ———, *Domain decomposition techniques for nonsymmetric systems of elliptic boundary value problems: Examples from CFD*, Second International Symposium on Domain Decomposition Methods (T. F. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds.), SIAM, 1989, pp. 321–339.
81. D. E. Keyes and M. D. Smooke, *A parallelized elliptic solver for reacting flows*, Parallel Computations and Their Impact on Mechanics (A. K. Noor, ed.), ASME, 1987, pp. 375–402.
82. D. A. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, submitted to J. Comput. Phys., 2002.
83. D. A. Knoll, P. R. McHugh, and D. E. Keyes, *Newton-Krylov methods for low Mach number compressible combustion*, AIAA J. **34** (1996), 961–967.
84. D. A. Knoll and V.A. Mousseau, *On Newton-Krylov multigrid methods for the incompressible Navier-Stokes equations*, J. Comput. Phys. **163** (2000), 262–267.
85. D. A. Knoll, A. K. Prinja, and R. B. Campbell, *A direct Newton solver for the two-dimensional tokamak edge plasma fluid equations*, J. Comput. Phys. **104** (1993), 418–426.
86. D. A. Knoll and W. J. Rider, *A multilevel Newton-Krylov method for nonsymmetric, nonlinear boundary value problems*, Tech. report, Los Alamos National Laboratory, 1997.
87. D. A. Knoll, W. J. Rider, and G. L. Olson, *Newton-Krylov methods applied to nonequilibrium radiation diffusion*, Tech. report, Los Alamos National Laboratory, 1998.
88. D. A. Knoll and W.J. Rider, *A multigrid preconditioned Newton-Krylov method*, SIAM J. Sci. Comput. **21** (2000), 691–710.
89. D. A. Knoll, W. B. VanderHeyden, V. A. Mousseau, and D. B. Kothe, *On preconditioning Newton-Krylov methods in solidifying flow applications*, SIAM J. Sci. Comput. (2001), in press.
90. G. Kron, *A set of principles to interconnect the solutions of physical systems*, J. Appl. Phys. **24** (1953), 965–980.
91. J. Mandel, *Balancing domain decomposition*, Commun. Numer. Meths. Engineering **9** (1992), 233–241.
92. D. J. Mavriplis, *On convergence acceleration techniques for unstructured meshes*, Tech. Report 98-2966, AIAA, 1998.
93. P. R. McHugh and D. A. Knoll, *Inexact Newton's method solutions to the incompressible Navier-Stokes and energy equations using standard and matrix-free implementations*, Proceedings of the AIAA Eleventh Annual Computational Fluid Dynamics Conference, 1993.

94. P.R. McHugh, D.A. Knoll, and D.E. Keyes, *Application of a Newton-Krylov-Schwarz algorithm to low Mach number combustion*, AIAA J. **36** (1998), 290–292.
95. V.A. Mousseau, *Newton-Krylov methods for time accurate solutions of multiphysics systems with advection*, Los Alamos National Laboratory Report, LA-UR-00-2524 (2000).
96. V.A. Mousseau, D. A. Knoll, and J. Reisner, *Nonlinearly consistent method for the shallow water equations*, Mon. Wea. Rev. (2002), to appear.
97. V.A. Mousseau, D. A. Knoll, and W.J. Rider, *Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion*, J. Comput. Phys. **160** (2000), 743–765.
98. W. Mulder and B. Van Leer, *Experiments with implicit upwind methods for the Euler equations*, J. Computational Physics **59** (1985), 232–246.
99. ———, *Experiments with implicit upwind methods for the Euler equations*, J. Comp. Phys. **59** (1995), 232–246.
100. N. M. Nachtigal, S. C. Reddy, and L. N. Trefethen, *How fast are nonsymmetric matrix iterations?*, SIAM J. Matrix Anal. Appl **13** (1992), 778–795.
101. J. C. Newton, W. K. Anderson, and D. L. Whitfield, *Multidisciplinary sensitivity derivatives using complex variables*, Tech. Report 98-08, Mississippi State University Engineering Research Center, July 1998.
102. E. J. Nielsen, R. W. Walters, W. K. Anderson, and D. E. Keyes, *Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code*, Tech. Report 95-1733, AIAA, 1995.
103. J. Nocedal and S. J. Wright, *Numerical optimization*, Springer, 1999.
104. E. S. Oran and J. P. Boris, *Numerical Simulation of Reactive Flow*, Elsevier, 1987.
105. P. D. Orkwis, *Comparison of Newton's and quasi-Newton's method solvers for the Navier-Stokes equations*, AIAA J. **31** (1993), 832–836.
106. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, 1980.
107. M. Pernice and M. D. Tocci, *A multigrid-preconditioned Newton-Krylov method for the incompressible Navier-Stokes equations*, SIAM J. Sci. Comput. **23** (2001), 398–418.
108. M. Pernice, L. Zhou, and H. F. Walker, *Parallel solution of nonlinear partial differential equations using a globalized inexact Newton-Krylov-Schwarz method*, Tech. Report 48, University of Utah Center for High Performance Computing, 1997.
109. Michael Pernice and Homer F. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput. **19** (1998), 302–318.
110. J. S. Przemieniecki, *Matrix structural analysis of substructures*, AIAA J. **1** (1963), 138–147.
111. N. Qin, D. K. Ludlow, and S. T. Shaw, *A matrix-free preconditioned Newton/GMRES method for Navier-Stokes equations*, Submitted to J. of Computational Physics, 1997.
112. G. H. Golub R. W. Freund and N. M. Nachtigal, *Iterative solution of linear systems*, Acta Numerica (1992), 57–100.
113. Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **7** (1986), 856–869.
114. V. Schmitt and F. Charpin, *Pressure distributions on the ONERA M6 wing at transonic Mach numbers*, Tech. Report AR-138, AGARD, May 1979.
115. R. Schreiber and H. B. Keller, *Driven cavity flows by efficient numerical techniques*, J. Comp. Phys. **49** (1983), 310–333.
116. F. Shakib, T. J. R. Hughes, and Z. Johan, *Element-by-element algorithms for nonsymmetric matrix problems arising in fluids*, Superlarge Problems in Computational Mechanics, Plenum, 1987, pp. 1–34.
117. B. F. Smith, P. Bjørstad, and W. D. Gropp, *Domain decomposition: Parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, 1996.
118. M. D. Smooke, *Solution of Burner-Stabilized Premixed Laminar Flames by Boundary Value Methods*, J. Comput. Phys. **48** (1982), 72–105.
119. M. D. Smooke and R. M. Mattheij, *On the solution of nonlinear two-point boundary value problems on successively refined grids*, Appl. Numer. Math. **1** (1985), 463–487.
120. P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **10** (1989), 36–52.
121. M. D. Tidriri, *Krylov methods for compressible flows*, Tech. Report 95-48, ICASE, June 1995.

122. ———, *Schwarz-based algorithms for compressible flows*, Tech. Report 96-4, ICASE, January 1996.
123. ———, *Efficient preconditioning of Newton-Krylov matrix-free algorithms for compressible flows*, J. Comp. Phys. **132** (1997), 51–61.
124. ———, *Hybrid Newton-Krylov/domain decomposition methods for compressible flows*, Proceedings of the Ninth International Conference on Domain Decomposition Methods in Sciences and Engineering, 1998, pp. 532–539.
125. U. Trottenberg, A. Schuller, and C. Oosterlee, *Multigrid*, Academic Press, 2000.
126. H. M. Tufo and P.F. Fischer, *Fast parallel direct solvers for coarse grid problems*, J. Par. Dist. Comput. **61** (2001), 151–177.
127. H. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging varient of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput. **13** (1992), 631–644.
128. S. P. Vanka, *Block-implicit calculation of steady turbulent recirculating flows*, Int. J. Heat Mass Transfer **28** (1985), 2093–2103.
129. V. Venkatakrishnan, *Newton solution of inviscid and viscous problems*, AIAA J. **27** (1989), 885–891.
130. V. Venkatakrishnan and D. J. Mavriplis, *Implicit solvers for unstructured meshes*, J. Comp. Phys. **105** (1993), 83–91.
131. G. Wang and D. K. Tafti, *Performance enhancements on microprocessors with hierarchical memory systems for solving large sparse linear systems*, Int. J. High Performance Computing Applications **13** (1999), 63–79.
132. D. L. Whitfield and L. K. Taylor, *Discretized Newton-relaxation solution of high resolution flux-difference split schemes*, Proceedings of the AIAA Tenth Annual Computational Fluid Dynamics Conference, 1991, pp. 134–145.
133. ———, *Variants of a two-level method for the approximate numerical solution of field simulation equations*, Tech. Report 98-09, Mississippi State University Engineering Research Center, July 1998.
134. L. B. Wigton, N. J. Yu, and D. P. Young, *GMRES acceleration of computational fluid dynamics codes*, Tech. Report 85-1494, AIAA, 1985.
135. K.A. Winkler, M.L. Norman, and D. Mihalas, *Implicit Adaptive-Grid Radiation Hydrodynamics*, Multiple Time Scales (J.U. Brackbill and B.I. Cohen, eds.), Academic Press, 1985.
136. J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Review **34** (1991), 581–613.

Mathematics & Statistics Department, Old Dominion University, Norfolk, VA 23529-0077 and ISCR, Lawrence Livermore Nat. Lab., Livermore, CA 94551-9989 and ICASE, NASA Langley Res. Ctr., Hampton, VA 23681-2199

E-mail address: keyes@icase.edu