Fourteenth International Conference on Domain Decomposition Methods
Editors: Ismael Herrera , David E. Keyes, Olof B. Widlund, Robert Yates

7. Domain Decomposition in the Mainstream of Computational Science

D. E. Keyes[1] [2]

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 26.6 Tflop/s on a climate application [27]. 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.

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.

In this context, optimal parallel algorithms for PDE simulations of Jacobian-free Newton-Krylov type, preconditioned with Schwarz and Schur domain decompositions, including multilevel generalizations of Schwarz, are coming into prominence. 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 a rather high level. For large systems with strong nonlinearities robustification techniques have been developed, including pseudo-transient continuation, parameter continuation, grid sequencing, model sequencing,

[1]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, keyes@icase.edu

[2]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 by NASA under contract NAS1-19480 to ICASE.

and nonlinear preconditioning. These improvements in nonlinear rootfinding have made it possible for large-scale PDE-constrained optimization problems (e.g., design, control, parameter identification — usually the ultimate problems behind the proximate PDEs) to be placed into the same domain-decomposed algorithmic framework as the PDE, itself.

The architecture of the terascale systems available in the United States, built around hierarchical distributed memory, appears hostile to conventional sequential optimal PDE algorithms, but is ultimately suitable apart from reservations about limited 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: valuable extent code designed for a sequential PDE analysis can often be "componentized" and made part of an effective domain-decomposed, operator-split preconditioner.

For a pair of web-downloadable full-scale reviews documenting these themes more fully, see [17, 18]. This page-limited chapter skims these reviews at a high level, emphasizing the importance of domain decomposition to large-scale scientific computing.

2. The Newton-Krylov-Schwarz Family of Algorithms. Many problems in engineering and applied physics can be written in the form

$$V\frac{\partial \mathbf{u}}{\partial t} + \mathcal{F}(\mathbf{u}) = 0, \tag{2.1}$$

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 strictly 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. Steady-state equations often arise from *a priori* equilibrium 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:

$$(\frac{V}{\tau^\ell})\mathbf{u}^\ell + \mathbf{f}(\mathbf{u}^\ell) = (\frac{V}{\tau^\ell})\mathbf{u}^{\ell-1}. \tag{2.2}$$

Higher-order temporal schemes are easily put into this framework with the incorporation of additional history vectors with appropriate weights on the right-hand side. We

are not directly concerned with discretization or the adaptivity of the discretization to the solution in this chapter. However, the achievement of nonlinear consistency by Newton's method on each time step is motivated by a desire to go to higher order than the pervasive standard of no better than first-order in time 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 [16].

Whether discretized time accurately or not, we are left at each time step with a system of nonlinear algebraic equations (2.2), written abstractly as $\mathbf{F}^\ell(\mathbf{u}^\ell) = 0$. We solve these systems in sequence for each set of discretized spatial grid functions, $\mathbf{u}^\ell$, using an inexact Newton method. 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 $\mathbf{u}^{\ell,k} \rightarrow \mathbf{u}^\ell$, as $k \rightarrow \infty$ 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.

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

$$\mathbf{F}'(\mathbf{u}^{k-1})\delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^{k-1}), \tag{2.3}$$

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 $\mathbf{F}'(\mathbf{u}^{k-1})$.

Newton-Krylov Methods. A Newton-Krylov (NK) method uses a Krylov method, such as GMRES [26], to solve (2.3) 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.

Newton-Krylov-Schwarz Methods. A Newton-Krylov-Schwarz (NKS) method combines a Newton-Krylov method, such as Newton-GMRES [6], with a Krylov-Schwarz (KS) method, such as restricted additive Schwarz [9]. 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, they are preconditioned with domain-decomposed additive Schwarz methods [28]. The system is 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. Schwarz methods [7, 11, 28, 32] 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 additive Schwarz approximate inverse is defined as

$$B_{ASM}^{-1} = \sum_i {R_i}^T A_i^{-1} R_i. \tag{2.4}$$

From the PDE perspective, subspace decomposition is domain decomposition. B^{-1} is formed out of (approximate) local solves on (possibly overlapping) subdomains.

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": $B_{ASM}^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i>0} R_i^T A_i^{-1} R_i$. 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, such as a multigrid V-cycle. The exact forward matrix-vector action of A in $B^{-1}A$ is still required, even if inexact solves are employed in the preconditioner.

Table 2.1: *Theoretical condition number estimates $\kappa(B^{-1}A)$, for self-adjoint positive-definite elliptic problems [28] 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	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$

Condition number estimates for $B^{-1}A$ are given in the first column of Table 1 in terms of the quasi-uniform mesh parameter h, and subdomain parameter H. 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 tradeoff 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.

The *restricted* additive Schwarz Method (RASM) eliminates interprocess communication during the interpolation phase of the additive Schwarz technique [9]. In particular, if we decompose a problem into a set of overlapping subdomains Ω_i, the conventional additive Schwarz method is a three-phase process consisting 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 performs a complete restriction operation but does not use any communication during the interpolation phase, denoted instead as ${R'_i}^T$. 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 [9], for reasons that are beginning to be understood [8].

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 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.

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 $B^{-1}A = \sum_i P_i$. 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 the partitioning 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 rapidly 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 [4, 5, 14, 30]. 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 [31].

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. A preconditioner may be defined 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. This leads to the method denoted "Hybrid II" in [28]:

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

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 representations of A and A^{-1} as sums over subspaces,

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

The latter is nothing but a special case of the Schwarz formula! In practice, invariant subspaces are far too expensive to obtain for practical use in Schwarz, 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.

Contrast of Domain Decomposition with Other Decompositions. It is worthwhile to emphasize the architectural advantages of Schwarz-type domain decomposition methods *vis-à-vis* other mathematically useful decompositions.

Given the operator equation $\mathcal{L}u = f$ in Ω, 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 $u = \sum_j u_j \phi_j$, or domain decompositions $\Omega = \cup_j \Omega_j$.

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

$$\frac{\partial u}{\partial t} + [\mathcal{L}_x + \mathcal{L}_y]u = f(x, y, t) \text{ in } \Omega, \tag{2.5}$$

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

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

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 overall iteration matrix mapping $u^{(\ell)}$ to $u^{(\ell+1)}$ is factored into 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 (2.5) 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 already. We replace $Au = f$ by $B_{ASM}^{-1}Au = B_{ASM}^{-1}f$ and solve by a Krylov method. There are several Krylov steps per time step, each requiring a matrix-vector multiplies with $B_{ASM}^{-1}A$. 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.

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

$$J(\mathbf{u})B_{split}^{-1}\mathbf{v} \approx \frac{\mathbf{F}(\mathbf{u}+\epsilon B_{split}^{-1}\mathbf{v}) - \mathbf{F}(\mathbf{u})}{\epsilon}, \tag{2.6}$$

where "*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. 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.

The essential insight of physics-based preconditioning is that preconditioner in a Newton-Krylov method maps a nonlinear residual to an approximate state-vector correction, namely, the Newton update. Such a map implicitly resides in most interative procedures of computational physics. 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 [24], time-dependent MHD equations [10], steady state incompressible Navier-Stokes equations [19, 25], and time-dependent incompressible Navier-Stokes equations [20, 25]. Also in [20], 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.

3. 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) [1, 2], 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. Among other important packages implementing Newton-Krylov in parallel, we mention Aztec [15], KINSOL [29], NITSOL [23], and the Iterative Template Library (ITL) [21].

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.

When well tuned, large-scale PDE codes spend almost all of their time in two phases: flux computations to evaluate conservation law residuals, 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 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 has a distinct proportion of work to datasize to communication requirements and each stresses a different subsystem of contemporary high-performance computers. In the language of a vertex-centered code, in which the data is

- 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.

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.

Analysis of a parallel aerodynamics code reimplemented in PETSc [13] shows that, after tuning, as expected, 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.

4. Terascale Optimal PDE Simulations (TOPS). Under the Scientific Discovery through Advanced Computing (SciDAC) initiative of the U.S. Department of Energy (`http://www.science.doe.gov/scidac/`), 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,

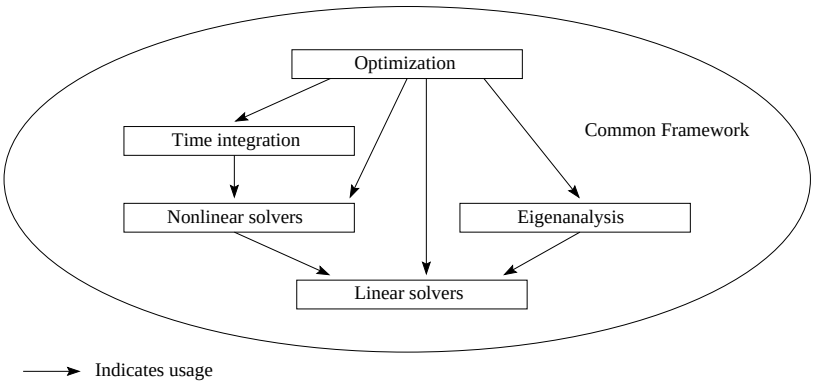

Figure 4.1: *An arrow from A to B indicates that A typically uses B. Optimization of systems governed by PDEs requires repeated access to a PDE solver. The PDE system may be steady-state or time-dependent. Time-dependent PDEs are typically solved with implicit temporal differencing. After choice of the 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.*

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.

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 4.1. In addition, TOPS emphasizes two cross-cutting topics: software integration (or interoperability) and high-performance coding techniques for PDE applications.

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.

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 [12], PETSc [1], ScaLAPACK [3], SuperLU [22]), 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.

The motivation behind TOPS 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.

5. 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. Domain decomposition is desirable for possessing a communication cost that is subdominant to computation – even optimal order computation, linear in the problem size – and fixed in ratio, as problem size and processor count are scaled in proportion.

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.

REFERENCES

[1] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. Efficient management of parallelism in object-oriented numerical software libraries. In *Modern Software Tools in Scientific Computing*, pages 163–201. Birkhauser, 1997.

[2] S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith. The Portable, Extensible Toolkit for Scientific Computing, version 2.3.1. `http://www.mcs.anl.gov/petsc/`, 2002.

[3] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.

[4] A. Brandt. Multi-level adaptive solutions to boundary value problems. *Math. Comp.*, 31:333, 1977.

[5] A. Brandt. Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics. Technical report, von Karman Institute, 1984.

[6] P. N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM J. Sci. Stat. Comput.*, 11:450–481, 1990.

[7] X.-C. Cai. Some domain decomposition algorithms for nonselfadjoint elliptic and parabolic partial differential equations. Technical Report 461, Courant Institute, 1989.

[8] X.-C. Cai, M. Dryja, and M. Sarkis. RASHO: A restricted additive Schwarz preconditioner with harmonic overlap. In *Proceedings of the 13th International Conference on Domain Decomposition Methods*. Domain Decomposition Press, 2002.

[9] X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM J. Sci. Comput.*, 21:792–797, 1999.

[10] L. Chacon, D. A. Knoll, and J. M. Finn. An implicit nonlinear reduced resistive MHD solver. *J. Comput. Phys.*, 178:15–36, 2002.

[11] M. Dryja and O. B. Widlund. An additive variant of the Schwarz alternating method for the case of many subregions. Technical Report 339, Department of Computer Science, Courant Institute, 1987.

[12] R. D. Falgout and U. M. Yang. Hypre: a library of high performance preconditioners. In *Lecture Notes in Computer Science, vol. 2331*, pages 632–641. Springer-Verlag, 2002.

[13] W. D. Gropp, D. K. K. D. E. Keyes, and B. F. Smith. High performance parallel implicit CFD. *Parallel Computing*, 27:337–362, 2001.

[14] W. Hackbusch. *Iterative Methods for Large Sparse Linear Systems*. Springer, 1993.

[15] S. A. Hutchinson, J. N. Shadid, and R. S. Tuminaro. Aztec user's guide: Version 1.1. Technical Report SAND95-1559, Sandia National Laboratories, October 1995.

[16] C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. *SIAM J. Numer. Anal.*, 35:508–523, 1998.

[17] D. E. Keyes. Terascale implicit methods for partial differential equations. In *Recent Advances in Numerical Methods for Partial Differential Equations and Applications*. AMS, 2002.

[18] D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. submitted to J. Comput. Phys., 2002.

[19] D. A. Knoll and V. Mousseau. On Newton-Krylov multigrid methods for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 163:262–267, 2000.

[20] 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.*, 23:381–397, 2001.

[21] L.-Q. Lee and A. Lumsdaine. The iterative template library. submitted to ACM Transactions on Mathematical Software, 2002.

[22] X. S. Li and J. W. Demmel. SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. submitted to ACM Transactions on Mathematical Software; also available as Lawrence Berkeley National Laboratory tech report LBNL-49388, 2002.

[23] H. F. W. Michael Pernice. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM J. Sci. Stat. Comput.*, 19:302–318, 1998.

[24] V. Mousseau, D. A. Knoll, and W. Rider. Physics-based preconditioning and the Newton-Krylov method for non-equilibrium radiation diffusion. *J. Comput. Phys.*, 160:743–765, 2000.

[25] M. Pernice and M. D. Tocci. A multigrid-preconditioned Newton-Krylov method for the incompressible Navier-Stokes equations. *SIAM J. Sci. Comput.*, 23:398–418, 2001.

[26] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, July 1986.

[27] S. Shingu, H. Takahara, H. Fuchigami, M. Yamada, Y. Tsuda, W. Ohfuchi, Y. Sasaki, K. Kobayashi, T. Hagiwara, S.-I. Habata, M. Yokokawa, H. Itoh, and K. Otsuka. A 26.58 Tflop/s global atmospheric simulation with the spectral transform method on the earth simulator. Proceedings of SC2002, 2002.

[28] B. F. Smith, P. Bjørstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.

[29] A. G. Taylor and A. C. Hindmarsh. User documentation for KINSOL: a nonlinear solver for sequential and parallel computers. Technical Report UCRL-ID-131185, Lawrence Livermore National Laboratory, July 1998.

[30] U. Trottenberg, A. Schuller, and C. Oosterlee. *Multigrid*. Academic Press, 2000.

[31] H. M. Tufo and P. Fischer. Fast parallel direct solvers for coarse grid problems. *J. Par. Dist. Comput.*, 61:151–177, 2001.

[32] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1991.