NASA Contractor Report 189709
ICASE Report No. 92-44

ICASE

DOMAIN DECOMPOSITION: A BRIDGE BETWEEN NATURE AND PARALLEL COMPUTERS

David E. Keyes

Contract Nos. NAS1-18605 and NAS1-19480
September 1992

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665-5225

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

DOMAIN DECOMPOSITION: A BRIDGE BETWEEN NATURE AND PARALLEL COMPUTERS [*]

David E. Keyes
Department of Mechanical Engineering
Yale University
New Haven, CT 06520[†]

ABSTRACT

Domain decomposition is an intuitive organizing principle for a PDE computation, both physically and architecturally. However, its significance extends beyond the readily apparent issues of geometry and discretization, on one hand, and of modular software and distributed hardware, on the other. Engineering and computer science aspects are bridged by an old but recently enriched mathematical theory that offers the subject not only unity, but also tools for analysis and generalization. Domain decomposition induces function-space and operator decompositions with valuable properties. Function-space bases and operator splittings that are not derived from domain decompositions generally lack one or more of these properties. The evolution of domain decomposition methods for elliptically dominated problems has linked two major algorithmic developments of the last 15 years: multilevel and Krylov methods. Domain decomposition methods may be considered descendants of both classes with an inheritance from each: they are nearly optimal and at the same time efficiently parallelizable. Many computationally driven application areas are ripe for these developments. This paper progresses from a mathematically informal motivation for domain decomposition methods to a specific focus on fluid dynamics applications. Introductory rather than comprehensive, it employs simple examples, and leaves convergence proofs and algorithmic details to the original references; however, an attempt is made to convey their most salient features, especially where this leads to algorithmic insight.

[*]This paper is based on a series of tutorial lectures delivered at the invitation of the United Technologies Research Center during the spring of 1992.

[†]*Email:* keyes@cs.yale.edu. This work was supported in part by the NSF under contract ECS-8957475, by the United Technologies Research Center, East Hartford, CT, and by the National Aeronautics and Space Administration under NASA Contract Nos. NAS1-18605 and NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center Hampton, VA 23665.

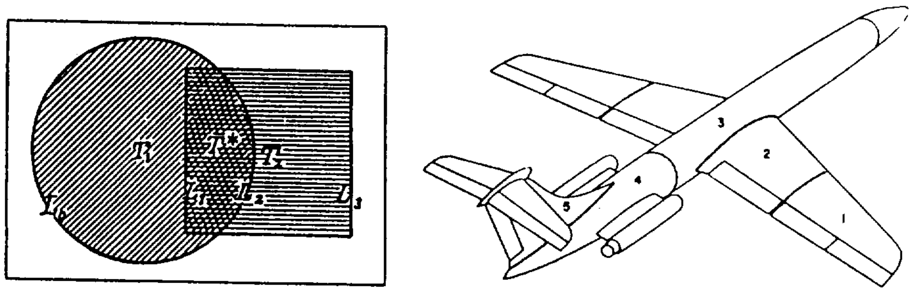

Figure 1 – Examples of domain decompositions into overlapping (left, due to Schwarz (Ref. 51)) and nonoverlapping (right, due to Przemieniecki (Ref. 49)) domains. These examples are of historical interest, having been employed in their original contexts to illustrate, respectively, an unaccelerated iterative substructuring method and a fully direct substructuring method.

INTRODUCTION

Domain decomposition extends the usefulness of numerical techniques for certain special partial differential equation problems to those of more general structure. Geometrical complexities or operator inhomogeneities that inhibit the global application of standard algorithms often suggest natural decompositions of problem domains into subdomains of simpler structure on which standard solvers are effective. An obvious hurdle to this approach is the specification of boundary conditions on the artificially introduced interfaces between subdomains, upon possession of which the subdomains would trivially decouple. Examples of such artificial interfaces, interior to the region on which physical boundary conditions are available, are shown in Fig. (1) ?

Rarely is either a stationary iteration or a direct derivation of the auxiliary conditions for the interfacial values from the differential or algebraic formulation of the problem the most economical way to proceed, though both have an extensive histories in the mathematical and engineering literature (see Refs. 3 and 41 for earlier annotated bibliographies). Instead, an acceleration procedure with a state vector that includes both interface and interior values may be set up, and local solvers may be employed as components of an approximate inverse, or preconditioner, for the overall system. For solvers whose complexity grows faster than linearly in the number of degrees of freedom, large problem size alone motivates decomposition and renders affordable the iteration required to enforce consistency at the artificial subdomain boundaries. (Though we confine attention to models of continuous media expressed as PDEs, we note that there are developments parallel to domain decomposition in the numerical analysis of integral equations ("multizone methods") and in the analysis of electrical and mechanical networks ("tearing methods"). In the latter context, Ref. 43 is of historical interest for its plot of Univac execution *days* versus decomposition granularity.)

Apart from distinctly domain-based approaches, there are at least two means of dividing a large or complex PDE problem into simpler, more tractable component problems. One is operator decomposition, in which the inverse of a different subset of terms of the PDE operator is approximated within each phase of an outer iteration that encompasses all terms. Alternating direction implicit (ADI) methods are classical examples. The other is function-space decomposition, in which a different component of the solution is computed within each phase. Spectral methods, generalizing classical Fourier analysis, are in this category. Though operator and function-space decomposition approaches have practical value, theoretical importance, and historical interest, they may be less optimal than domain decomposition from the perspective of parallel computation because they are not generally frugal in their data access requirements, as we illustrate below.

Parallel Computation in Space-Time

Domain decompositions are not fundamentally different from other types of decomposition, and may, in fact, be interpreted for analysis purposes as operator or function-space decompositions, but of special form that can be motivated by, among other things, the memory hierarchies of distributed-memory parallel computers. Each processor in such a system has rapid access to data stored in its own memory and slower access to data stored in the memory associated with other processors. The cost, in time, of accessing remote data may depend not only on the location of the data but also on the amount of data being simultaneously requested by all processors acting in parallel, and on the richness of the interprocessor communication network. It is possible, in principle, to construct for each processor-memory element a forward-pointing "data cone," in analogy to the "light cone" of physics, that indicates how far through the computer the data associated with that unit can propagate in a given time. Processors outside of a given cone at a given time level cannot be influenced by data originating at its apex. Similarly, it is possible to construct for each processor element a backward-pointing data cone that indicates the most recent possible age of remotely originating data of which it can be aware. The situation is represented schematically in Fig. , which is adapted from Ref. 47. (Of course, the analogy to the light cone of physics is not a very precise one, since there are many different data propagation speeds inside a multiprocessor computer, not just a single speed of light. Due to distance-insensitive software overheads, data propagation speeds are generally highly nonlinear functions of distance.) On many previous generations of computers it was unnecessary for scientific programmers to recognize data cones, since, in effect, there was only one cone and it had an apex angle that was effectively fully open, modulo memory cache effects. Data access rates were assumed to be insignificant compared to rates at which data was processed. Contemporary algorithm designers, in contrast, must recognize that there is a time cost associated with distance. More generally, there may be different time costs associated with different routings and packet sizes, and relative costs may vary greatly from machine to machine. As network technology is further pressed and machine diameters expanded, the importance of such differences must ultimately increase (unless padded by software, intentionally or unintentionally).

The solution of problems restricted to a mathematical subdomain that resides entirely within the domain of a processor requires communication with other processors only to set up provisional boundary conditions. In contiguity-preserving maps of grid points to processors this information is typically found through exchanges of an amount of data that is small compared with the amount stored locally. Furthermore, it occurs between a number of subdomains that is small compared with the total. Therefore, a decomposition that maps data onto processors in a geometrically contiguous manner leads potentially to a low ratio of time spent communicating to time spent computing, and thus to a high parallel efficiency. However, high efficiency is not an end in itself; the relative effectiveness of parallel numerical methods also depends on the total number of arithmetic operations they require, independent of communication costs. Examples of algorithms that may be efficiently parallelized but are nevertheless ineffective in an overall elapsed time sense abound. It is therefore important to note that decomposition by domain respects the

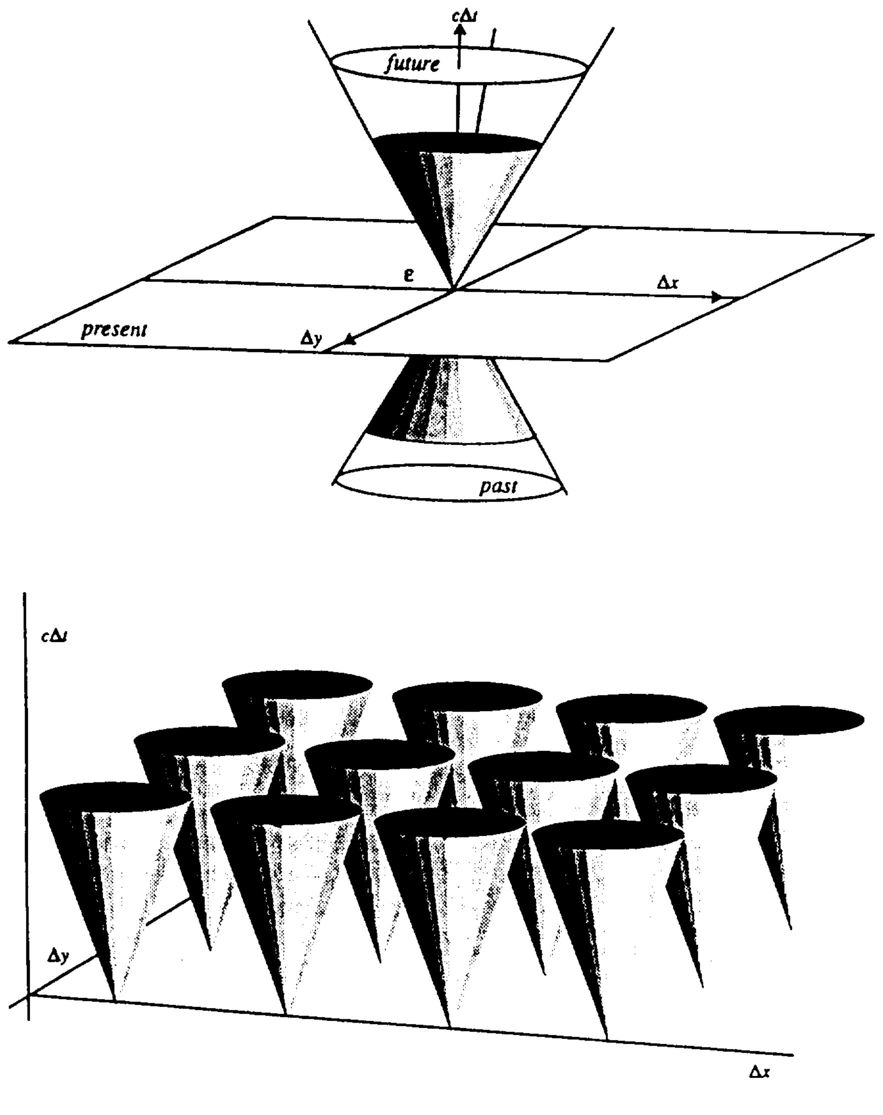

Figure 2 – Parallel computation viewed as a series of events in space-time. The past and future data cones of a single processor-memory element are shown above, and the interacting future cones of a planar array of processor elements below.

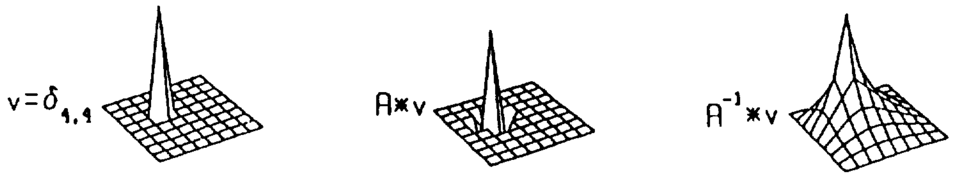

Figure 3 – The sparsity of A (discrete Poisson operator) and A^{-1} (discrete Green's function) contrasted by their action on a sample δ-function on a two-dimensional grid.

natural data dependencies of elliptic and time-parabolic problems.

Green's Functions as Data Dependency Graphs

The dependence of the solution at a given point on the data specified at another can be characterized for linear problems by the Green's function (Ref. 20) for the differential operator being inverted. A Green's function is a function of two variables, a field point and a source point, whose magnitude reflects the degree of influence of the source point on the field point. The strict evaluation of the solution at a field point x requires integration over all source points y for which the Green's function $G(x,y)$ is nonzero. If the solution is sought only to within a given tolerance, that tolerance defines the resolution required in the integration process. Regions in which the Green's function is small and smoothly varying can be resolved less accurately than those in which the Green's function is large or changing rapidly. For elliptic and parabolic problems, Green's functions decay monotonically with the distance between field and source point. Therefore, it is natural to place on or near the processor that will compute the value of the solution at x the forcing data at as many points "near" x as possible. ("Nearness" in this context may be measured in an anisotropic metric if the differential operator, and hence its Green's function, is anisotropic. An example will be furnished later.)

It has often been argued that PDEs are well-suited to parallel computation because differential operators can be approximated with local finite differences. Assuming that locality is preserved in the map from gridpoints to processors, this implies that the perfectly scalable, constant bandwidth nearest-neighbor mesh interconnect alone constitutes a sufficiently rich communication network for PDE computation. Such an argument is valid only if one is willing to iterate a number of times proportional to the discrete diameter of the grid on which the PDE is discretized. Though elliptic finite-difference operators are sparse, their inverses are dense, implying that the solution at every point is dependent upon the forcing at every other point. This elementary observation is illustrated in Fig. , which depicts a discrete δ-function forcing term at a point in a two-dimensional grid, along with the action of the five-point discrete Laplacian with homogeneous Dirichlet boundary conditions, and of its inverse on the same function. Thus, the rightmost plot is a sample single column of the matrix representing the discrete Green's function.

The figure illustrates both a nonzero influence of the data at each point on the solution at every other, and the rapid decay with distance of the *magnitude* of the influence. It is algorithmically unwise to ignore the nonlocal influence, but but also unwise to resolve it as finely as the local influence. The influence of distant points, in the tail of the Green's function, can be "coarse-grained" and communicated on an area-averaged basis. This role will be carried by coarse grids in the development below. On the other hand, for

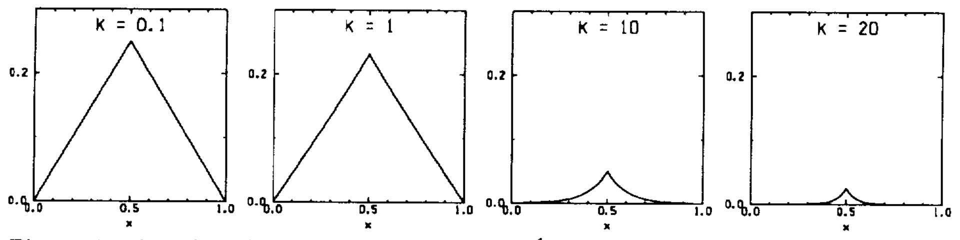

Figure 4 – A series of Green's functions, $G(x, \frac{1}{2})$, for a modified Helmholtz operator arising from implicit time discretization of parabolic problem for Eq. 2 for a source point at the midpoint of the interval and for several $k^2 \equiv 1/\Delta t$. As the limit of infinitesimal time step is approached, the Green's function is more and more concentrated.

commensurate accuracy, fine resolution is needed in the near field. The shape of the Green's function is the ultimate motivation for a two-level (or a multilevel) algorithm.

The efficient assimilation of distant data via a coarse grid becomes less important as the rate of decay of the discrete Green's function becomes more rapid. This phenomenon occurs in implicitly time-differenced parabolic problems as the time step is made smaller, as illustrated in Fig. , for the one-space-dimensional problem

$$-u'' + k^2 u = f, \tag{1}$$

arising from

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0 \tag{2}$$

on $x \in [0,1]$ with $u(0) = u(1) = 0$, when we make the replacement

$$\frac{\partial u}{\partial t} \approx \frac{u(x,t) - u(x, t - \Delta t)}{\Delta t} \tag{3}$$

for the transient term, where $k^2 \equiv 1/\Delta t$. The Green's function

$$G(x,y) = \frac{1}{k \sinh k} \times \left\{ \begin{array}{ll} \sinh(kx)\sinh(k(1-y)), & x < y \\ \sinh(ky)\sinh(k(1-x)), & x > y \end{array} \right\} \tag{4}$$

satisfies

$$-G''(x,y) + k^2 G(x,y) = \delta(x-y), \tag{5}$$

whence

$$u(x) = \int_0^1 G(x,y) f(y) dy. \tag{6}$$

This is an instance of the modified Helmholtz operator, in which the sign of the diagonal term agrees with the diagonal part of the diffusion term to strengthen the diagonal dominance of the matrix operator. For sufficiently small time step the tail is

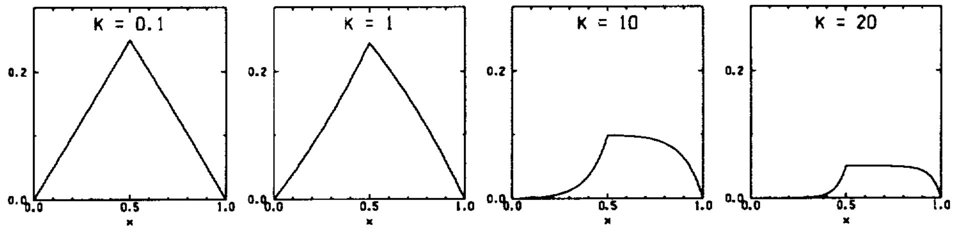

Figure 5 – A series of adjoint Green's functions, $G^{\dagger}(x, \frac{1}{2})$, for a convection-diffusion operator, Eq. 7 for a source point at the midpoint of the interval and for several convection strengths k. As the limit of large convection is approached, the Green's function is more and more one-sided.

insignificant, and a coarse grid brings no added advantage, as pointed out in Ref. 8.

The opposite is true of the wave Helmholtz operator, in which the diagonal term detracts from the diagonal dominance of the diffusion operator, producing greater indefiniteness and more oscillation of the Green's function as the wavenumber increases. Convergence theorems (Ref. 12) indicate that the coarse grid must be made fine enough to resolve the oscillations of the Green's function in this case.

For the important case of large first-order terms, the Green's function becomes one sided, as illustrated in Fig. , again for a one-dimensional model problem. In this case, due to the nonselfadjointness of the differential operator, the relevant kernel is the adjoint Green's function. If $u(x)$ satisfies the one-dimensional convection-diffusion equation with constant convection coefficient k (right-to-left when $k > 0$),

$$- u'' - ku' = f, \tag{7}$$

the adjoint Green's function,

$$G^{\dagger}(x,y) = \frac{1}{k(1-e^{k})} \times \left\{ \begin{array}{ll} (e^{kx}-1)(1-e^{k(1-y)}), & x < y \\ (e^{kx}-e^{k})(1-e^{-ky}), & x > y \end{array} \right\}, \tag{8}$$

satisfies

$$- G^{\dagger\prime\prime}(x,y) + kG^{\dagger\prime}(x,y) = \delta(x-y), \tag{9}$$

whence

$$u(x) = \int_0^1 G^{\dagger}(x,y)f(y)dy. \tag{10}$$

In the limit of large $|k|$, the tail of the relevant kernel extends far upstream and insignificantly downstream. Lest this seem to render a theory based on elliptic operators irrelevant for computational fluid dynamics applications, recall that many if not most production CFD codes operate either in a time-accurate or in a steady-state defect correction manner. In the latter, an artificially diffusive left-hand side operator is used to drive a high accuracy right-hand side operator to a steady state. Hence, the left-hand side operators whose inverse action is required in most CFD codes have a parabolic or elliptic character.

In this paper, we argue that domain decomposition is a compelling algorithmic paradigm for bridging the gap between a universe of applications that are physically hierarchical in a continuous sense, and a universe of parallel computers that are architecturally hierarchical in a discrete sense. When the hierarchies of the application and the architecture are well matched, the appropriate hierarchy of the algorithm that must bridge them is obvious. Even when this is not the case, algorithms that somehow take into account those aspects of the modeling process over which the user typically has no control, namely the Green's function of the physics and the data cones of the computer, are destined to be more effective than algorithms that have no regard for their structure.

THREE SOURCES OF PARALLELISM

In the partial differential equation

$$\mathcal{L}u = f \text{ in } \Omega \tag{11}$$

there are three potential sources of parallelism, or more generally, of ways to "divide, conquer and combine," which is also a useful approach in sequential algorithms.

One can decompose the operator, $\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \cdots + \mathcal{L}_N$, and attempt to make use of separate inversions of the different pieces. There is a variety of reasonable decompositions of this class, of which the traditional do not use a very large N and are sequential. Nevertheless, this type of decomposition can create phases with significant parallelism *within* each phase.

Alternatively, one can decompose the space of functions in which the solution is sought, and represent the solution as a sum of components from each subspace, $u = u_1 + u_2 + \cdots + u_N$. N is typically quite large in this case.

Finally, one can decompose the domain $\Omega = \Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_N$, leaving a set of problems coupled at their boundaries, each of which is of the same form as the original.

We illustrate these decompositions on the following model PDE computation. We consider the parabolic PDE

$$\frac{\partial u}{\partial t} + (\mathcal{L}_x + \mathcal{L}_y)u = f(x,y) \text{ in } \Omega \text{ with } u = 0 \text{ on } \partial\Omega \tag{12}$$

where

$$\mathcal{L}_x \equiv -\frac{\partial}{\partial x} a_x(x,y) \frac{\partial}{\partial x} + b_x(x,y)\frac{\partial}{\partial x}, \quad (a_x > 0), \tag{13}$$

and

$$\mathcal{L}_y \equiv -\frac{\partial}{\partial y} a_y(x,y) \frac{\partial}{\partial y} + b_y(x,y)\frac{\partial}{\partial y}, \quad (a_y > 0). \tag{14}$$

If a fully implicit time discretization is used, elliptic-looking problems for spatial discretization are generated at each time step, namely,

$$\left(\frac{I}{\Delta t} + \mathcal{L}_x + \mathcal{L}_y\right) u^{(\ell+1)} = \tilde{f} \equiv \left(\frac{I}{\Delta t}\right) u^{(\ell)} + f. \tag{15}$$

Algorithms Requiring Multiple Mappings: ADI and Spectral

One method for attacking this problem is an operator decomposition of Alternating Direction Implicit (ADI) type (Ref. 24). The compound iteration is written as

$$\begin{aligned} \left(\frac{I}{\Delta t/2} + \mathcal{L}_x\right) u^{(\ell+1/2)} &= \left(\frac{I}{\Delta t/2} - \mathcal{L}_y\right) u^{(\ell)} + f \\ \left(\frac{I}{\Delta t/2} + \mathcal{L}_y\right) u^{(\ell+1)} &= \left(\frac{I}{\Delta t/2} - \mathcal{L}_x\right) u^{(\ell+1/2)} + f \end{aligned} \tag{16}$$

with the iteration operator

$$\left(I + \frac{\Delta t}{2}\mathcal{L}_y\right)^{-1} \times \left(I - \frac{\Delta t}{2}\mathcal{L}_x\right) \times \left(I + \frac{\Delta t}{2}\mathcal{L}_x\right)^{-1} \times \left(I - \frac{\Delta t}{2}\mathcal{L}_y\right). \tag{17}$$

Upon spatial discretization, there are four sequential substeps per timestep, consisting of two sparse matrix multiplications represented by $\left(I - \frac{\Delta t}{2}\mathcal{L}_y\right)$ and $\left(I - \frac{\Delta t}{2}\mathcal{L}_x\right)$, and two sets of completely independent unidirectional tridiagonal solves in $\left(I + \frac{\Delta t}{2}\mathcal{L}_x\right)^{-1}$ and $\left(I + \frac{\Delta t}{2}\mathcal{L}_y\right)^{-1}$. The action of each term in $\mathcal{L}_x$ can be computed concurrently at different y-coordinates, and vice versa. Hence there is significant parallelism *within* each substep.

Another method for attacking this problem is a spectral method function-space decomposition (Ref. 33). One defines a set of global basis functions $\phi_j(x,y)$ and sets

$$u(x,y,t) = \sum_{j=1}^{N} a_j(t)\phi_j(x,y), \tag{18}$$

to get, using Galerkin's method, where $(\cdot,\cdot)$ is the ordinary inner product,

$$\frac{d}{dt}(\phi_i, u) + (\phi_i, \mathcal{L}u) = (\phi_i, f), \quad i = 1, \ldots, N. \tag{19}$$

Upon substitution of the expansion for u,

$$\sum_{j=1}^{N}(\phi_i,\phi_j)\frac{da_j}{dt} = -\sum_{j=1}^{N}(\phi_i, \mathcal{L}\phi_j)a_j + (\phi_i, f), \quad i = 1, \ldots, N, \tag{20}$$

which leads immediately to the system of ordinary differential equations in time:

$$\dot{a} = -M^{-1}Ka + M^{-1}g, \tag{21}$$

where the mass matrix M and stiffness matrix K are defined by

$$M \equiv [(\phi_i,\phi_j)], \quad K \equiv [(\phi_i, \mathcal{L}\phi_j)]. \tag{22}$$

Note that M is diagonal if the ϕ_j are orthogonal and, further, K is diagonal if the ϕ_j are eigenfunctions of $\mathcal{L}$. In this limit, the equations for the $a_j(t)$ completely decouple. More generally, M and K are perhaps dense. In this opposite limit, there is still significant parallelism available in the marching of the system of N ODEs for the coefficients of each wavenumber $a_j(t)$, since the computation to communication ratio is high *within* the spectral domain.

The ADI and spectral methods are straightforward to define and implement when the domain Ω has an exploitable tensor-product character. Irregular or non-simply-connected domains make it more difficult to apply methods based on a single global discretization. However, the main problem with these algorithms is not lack of geometrical flexibility. From the point of view of parallel computing, the main problem is in the data exchange costs when they are embedded in a complete code. These and most operator and function-space decomposition methods require frequent global exchanges of amounts of data proportional to the discrete dimension of the problem *between* the parallelizable phases.

In the case of ADI, the favorable mapping of data onto processors for the solution of the tridiagonal systems with constant y coordinate is the transpose of the favorable mapping for the constant x coordinate phase. Therefore, either one the phases will be computed with poor parallel efficiency or a pair of global transpositions of data must take place within every iteration. It is well understood how to carry out this transposition in a recursive way that very effectively exploits the communication channels of a hypercube network (Ref. 39). However, the amount of data required to be exchanged and the frequency with which the transpose must be performed maintain pressure on the architecture of a general parallel machine as it is scaled upwards. This situation becomes worse in three dimensions even though three-dimensional Green's functions are more rapidly decaying than their two-dimensional counterparts. Constant-index sets of global span may be ideal implicit aggregates in sequential FORTRAN, but not necessarily on parallel computers.

In the case of spectral methods, the partitioning of wavenumber space that leads to efficient parallelism (possibly including complete decoupling) in the solution for the spectral coefficients is altogether different from the domain-based partitioning required to assemble the physical solution from a sum over wavenumber. The physical solution may be required periodically for convergence checks or display; moreover, in the operator-split pseudo-spectral method as applied to convection-diffusion problems such as Navier-Stokes, the physical solution is required at each time step for the explicit advancement of the nonlinear convective terms. Thus, as with ADI, global exchanges of amounts of data proportional to the overall problem size are frequently required.

It is frequently proposed that heterogeneous architectures be assembled so that each arithmetic subtask of an overall computation can be executed on a node (or set of nodes) most appropriate for it. For instance, matrix element assembly might be a massive SIMD application, banded factorization and solution a vector processor application, and ODE-integration of chemical kinetics source terms in different regions an intermediate-granularity MIMD application. Like the ADI and spectral methods, an overall implementation composed of tasks such as these would require frequent transfers between processors of the full data of the problem, ultimately making the interprocessor network the bottleneck to further scalability. The domain decomposition paradigm is entirely different. In domain-decomposed parallelism, each processing element performs all of the operations on a subset of the data, rather than a subset of the operations on all of the data.

An Algorithm Requiring a Single Mapping: Additive Schwarz

As an example of an algorithm for the same model problem, Eq. 15, that requires only small to moderate degrees of global data exchange, we consider a small-overlap version of the Additive Schwarz method of Dryja and Widlund (Ref. 22). To expose the parallel properties of this prototype domain decomposition method, we need a modest degree of mathematical machinery, which we keep as small as possible by considering only piecewise linear nested discretizations.

We introduce a coarse grid on Ω (whose diameter, properly nondimensionalized, is $\mathcal{O}(1)$) by cutting it into nonoverlapping subdomains of quasi-uniform size H, Ω_k. Each of these subdomains is further divided into mesh cells, $\omega_{k,i}$, of quasi-uniform size h. Following the finite element formalism, a global coarse grid function space, V^H, is introduced:

$$V^H = \left\{ v^H \text{ continuous on } \Omega, \text{ linear on each } \Omega_k, \text{ vanishing on } \partial\Omega \right\}, \tag{23}$$

together with a global fine grid function space, V^h:

$$V^h = \left\{ v^h \text{ continuous on } \Omega, \text{ linear on each } \omega_{k,i}, \text{ vanishing on } \partial\Omega \right\}. \tag{24}$$

The accuracy of the discrete solution will be controlled by h and the granularity of the parallel decomposition by H. Part of the beauty of the Additive Schwarz method is that its asymptotic convergence rate depends only weakly, if at all, on h and H, leaving these parameters at the disposal of the physics and of the parallel computing environment.

One means of obtaining such highly convergent algorithms is to introduce some overlap between the subdomains. For this purpose, each Ω_k is extended to a subdomain Ω'_k by bordering it with fine grid cells of neighboring processors. (In practice, as little as $(\mathcal{O}(h))$ of overlap is often sufficient for good convergence performance, though convergence proofs for difficult operators or geometries may require more.) Figure illustrates a pair of nonoverlapping subdomains sharing a common interface Γ and an overlapped subdomain whose boundary points lie within the interior of neighboring subdomains. Subdomain extensions that lie outside of the physical boundary, $\partial\Omega$ are cut off, by definition. We then define another set of function spaces, this time local:

$$V_k^h = \left\{ v^h \in V^h, \text{ vanishing on and outside of } \partial\Omega'_k \right\}. \tag{25}$$

We define projection operators onto the coarse space, V^H, and onto each of the fine extended spaces, V_k^h, by means of the same Galerkin procedure used in the spectral method above, but with a different test space for each projection. Thus, let

$$A(u,v) \equiv (\frac{u}{\Delta t}, v) + (\mathcal{L}u, v) \tag{26}$$

and define P_k^h as the projection associated with subspace V_k^h, i.e.,

$$A(P_k^h u, v) = A(u,v), \forall v \in V_k^h, \tag{27}$$

and P^H as the projection associated with V^H, i.e.,

$$A(P^H u, v) = A(u,v), \forall v \in V^H. \tag{28}$$

Note that computing $P_k^h u$, for any u, requires solving a local Galerkin problem on Ω'_k with homogeneous Dirichlet boundary conditions. The discrete dimension of this problem is the number of points of the fine grid interior to Ω'_k. Note as well that all of the $P_k^h u$, $k =$

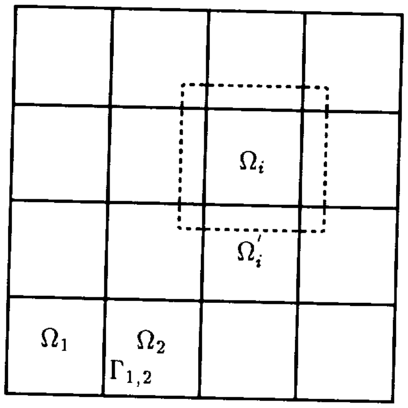

Figure 6 – Decomposition of a square domain into square subdomains of nonoverlapping and overlapping type (from Ref. 10).

$1, 2, \ldots, K$, can be found independently even though their domains overlap. Computing $P^H u$ requires solving a coarse global problem on Ω itself. The discrete dimension of this problem is the number of coarse grid vertices inside $\partial\Omega$. Hence this problem is small, though it does require some global data exchange. Though the communication required to form $P^H u$ is nontrivial, it need not be a sequential bottleneck like the larger communication phases of ADI or spectral methods, since it can be performed concurrently with the arithmetic steps of the fine mesh projections. The overall solution of the original implicitly time-discretized system at each time step has the form

$$Pu = b \tag{29}$$

where $P \equiv P^H + P_1^h + P_2^h + \cdots + P_K^h$, and where b is the sum of the projections of $\tilde{f}$ from Eq. 15.

To describe P in terms of matrix algebra, let the total number of degrees of freedom in the solution vector be n, and let A be the square nonsingular $n \times n$ matrix with elements $a_{ij} = A(\phi_i, \phi_j)$, where the ϕ_i constitute a nodal basis for V^h. Let n_k be the number of degrees of freedom interior to the subdomain Ω_k' and let R_k be the $n_k \times n$ matrix of zeros and ones that restricts the global solution vector to the interior of Ω_k'. Then R_k^T is an $n \times n_k$ matrix that extends by zero a solution in Ω_k' to the global domain. Next, let A_k be the $n_k \times n_k$ matrix with ijth element $A(\phi_i, \phi_j)$, where the ϕ_i constitute a nodal basis for V_k^h. Then $P_k^h = R_k^T (A_k)^{-1} R_k A$, $k = 1, \ldots, K$. Note that R_k and R_k^T involve no arithmetic operations, but are gather/scatter or communication routines. This defines all but one of the terms of P. In similar notation, P^H may be defined as $R_0^T (A_0)^{-1} R_0 A$, where the ijth element of A_0 is $A(\Phi_i, \Phi_j)$, where the Φ_i constitute a basis for V^H. R_0 and R_0^T are weighted restriction and prolongation operators based on multivariate interpolation. These operators involve floating point work in addition to communication. Similarly, b may be defined in matrix algebraic notation as $\sum_{k=0}^{K} R_k^T (A_k)^{-1} R_k f$.

The transformed problem, $Pu = b$, is solved by a Krylov method, such as GMRES (Ref. 50). Because of their importance in domain decomposition algorithms, Krylov methods are described in greater detail in a subsequent section. It is sufficient here to be reminded that Krylov methods operate by applying P to a series of vectors at each time step. Each application of P has significant parallelism, by construction. There is a set of concurrent subdomain problems to solve, which require some nearest-neighbor data exchange (proportional to the size of the overlap regions), and there is a single small global problem to solve. If the data of the problem is initially mapped onto parallel processors in accordance with the domain decomposition, no global remapping is ever needed.

Observations and Remarks

Of the three types of decomposition, only domain decomposition obviates the need to move large amounts of data globally. We have considered a parabolic problem. The fully elliptic case is included if $\Delta t \rightarrow \infty$ in Eq. 15. In this case, operator decomposition still needs to proceed in time-like relaxation steps. The function-space and domain decomposition algorithms can be applied directly to the elliptic case. Through the finite element subspaces of subdomain-scale support, V_k^h, and the global coarse space, V^H, domain decomposition has a function-space decomposition interpretation. Through the projection operators, P_k^h and P^H, domain decomposition has an operator decomposition interpretation. For problems with complex operators or geometry, a hybrid method, such as domain decomposition within each substep of a split-operator, or operator splitting within a domain-decomposed framework might be the best parallel technique. The spectral element method (Ref. 48) is a hybrid of domain and function-space decomposition that is further hybridized with operator splitting when applied to the Navier-Stokes equations. It has been parallelized with great success (Ref. 28).

APPROXIMATING THE INVERSE OF A SUM

It has been argued above that algorithmic insight derives from thinking of the process of solving a linearized PDE as the application of a formal inverse of the PDE operator to the right-hand side data. In this section, we further exploit this formalism.

The bugaboo in parallelizing any implicit method is that the inverse of the sum is *not* the sum of the inverses:

$$(A_1 + A_2)^{-1}v \neq A_1^{-1}v + A_2^{-1}v. \tag{30}$$

This contrasts with the situation of explicit methods, in which the ability to distribute the application of operators over addition is the source of embarrassing parallelism. The product of the sum *is* the sum of the products:

$$(A_1 + A_2)v = A_1 v + A_2 v. \tag{31}$$

Explicit methods are equivalent to matrix-vector multiplications, which are readily parallelized. Implicit methods are equivalent to matrix inversions (solves), which are generally highly sequential.

Modern domain decomposition algorithms find a compromise. They are preconditioned iterative methods with semi-implicit, divide-and-conquer preconditioners. They work by combining Krylov methods (requiring only matrix-vector multiplications) with approximate, parallelizable inverses. A profitable question is: *When* is the inverse of the sum well approximated by a sum of "partial inverses"? The inverse of a sum is *identical* to the sum of partial inverses for special decompositions. For example, given

$$Au = \mathbf{f}, \tag{32}$$

to be solved for $\mathbf{u}$, let A have a complete set of orthonormal eigenvectors:

$$A\mathbf{v}_k = \lambda_k \mathbf{v}_k. \tag{33}$$

Then A can be expressed as the sum of rank-one matrices:

$$A = \sum_k \lambda_k \mathbf{v}_k \mathbf{v}_k^T. \tag{34}$$

The solution $\mathbf{u} = A^{-1}\mathbf{f}$ can be expressed as a linear combination of the eigenvectors by multiplication of Eq. 32 by $\mathbf{v}_l$ and use of the orthonormality property. The result is

$$\mathbf{u} = \sum_k u_k \mathbf{v}_k, \quad \text{where } u_k = \frac{\mathbf{v}_k^T \mathbf{f}}{\lambda_k}. \tag{35}$$

Now define as "partial inverses" (and we use the "-1" notation loosely) the k rank-one matrices

$$A_k^{-1} \equiv \frac{1}{\lambda_k} \mathbf{v}_k \mathbf{v}_k^T. \tag{36}$$

Then,

$$\mathbf{u} = A^{-1}\mathbf{f} = \sum_k A_k^{-1}\mathbf{f}, \tag{37}$$

as desired. Once the eigendecomposition of A is known, each term of the right-hand member of Eq. 37 can be found independently. Abstracting, the key steps are the decomposition of the solution $\mathbf{u}$ into orthogonal subspaces, and the expression of A^{-1} as the sum of projections onto these subspaces.

Note that the partial inverses do not have the full rank of the original problem. Since the inverse action of a PDE operator is generally a computationally complex operation to apply (polynomial in the problem dimension), any useful decomposition of the inverse will consist of pieces that operate in subspaces of restricted dimension.

In practice, the subspaces spanned by eigenvectors in this example must be replaced with something else. Eigenvectors are expensive to compute and, having global support, they require too much storage. An inexpensive way to form a set of orthogonal subspaces whose total storage requirements are the same as that of one copy of the solution vector is to make each spanning vector zero over most of the domain. Unfortunately, subspaces that achieve strict mutual orthogonality in this way cannot span the entire space. Functions that are not zero at subdomain interfaces cannot be represented. A small subspace can supply what is missing, possibly at the sacrifice of not being orthogonal to the rest. This turns out to be a practical trade-off.

Consider a one-dimensional piecewise linear finite element example. The function $u(x)$ sketched in Fig. can be decomposed into the sum of the five functions in Fig. . Piecewise linear finite element bases for approximation of the five functions are shown

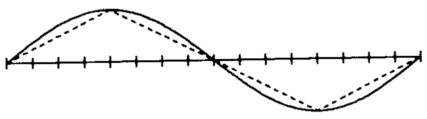

Figure 7 – A function to be decomposed into components in subspaces primarily of local support.

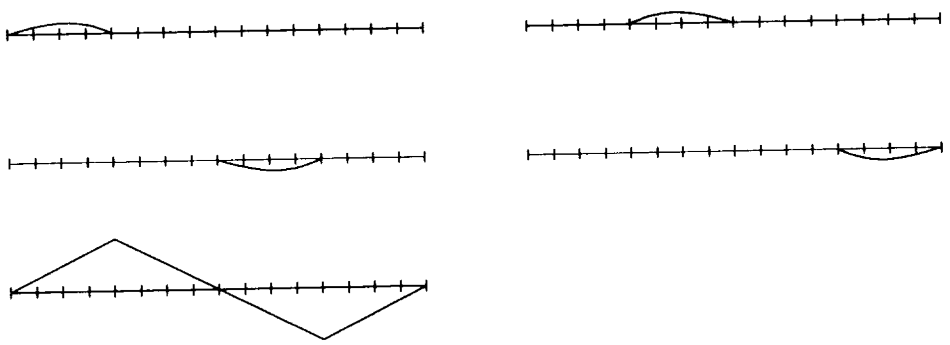

Figure 8 – Decomposition of the function in Fig.

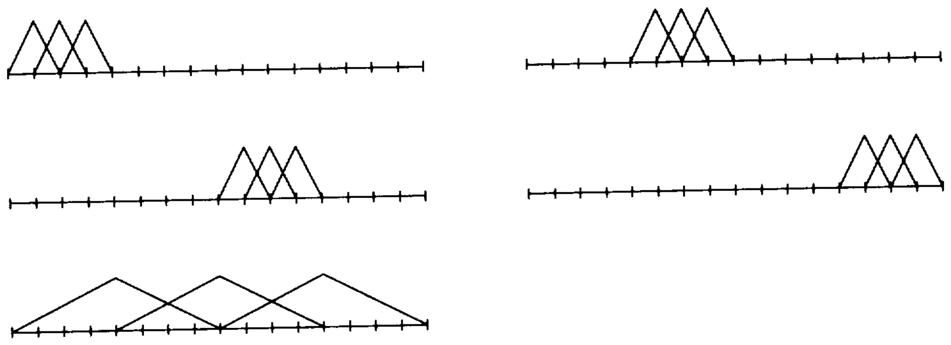

Figure 9 – Bases for the subspaces underlying the decomposition in Fig.

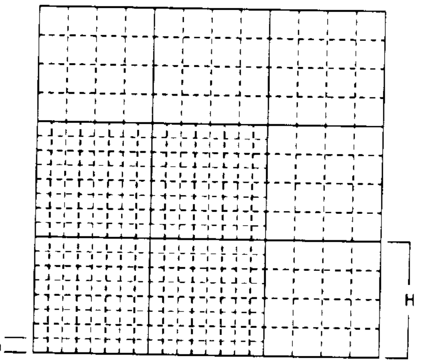

Figure 10 – Illustration (by "footprint") of a hierarchical decomposition in two dimensions, with locally uniform adaptive refinement.

in Fig. . The first four of these subspaces are mutually orthogonal. The last one is not orthogonal to any of the others. The union of the five subspaces is a two-level hierarchical basis that spans the same space as the piecewise linear basis for the original, global problem. Projection onto the first four subspaces consists of solving four small independent problems together accounting for 80% of the data in the problem. The projection onto the last subspace consists of solving one problem involving 20% of the data. This two-level hierarchical discretization can be extended to two or three dimensions, and can be generalized to incorporate adaptive local uniform refinement. In large, well-resolved problems, the percentage of fully concurrent work improves. A sketch of a hierarchical decomposition in two dimensions with locally uniform adaptive refinement in a corner is shown in Fig. .

The precise notion of orthogonality that is of greatest importance in convergence results for domain-decomposed solution of elliptic problems is not the ordinary one, $\mathbf{u}_k^T\mathbf{u}_j = 0, j \neq k$, but orthogonality with respect to A. This is the notion of A-conjugacy, $\mathbf{u}_k^T A\mathbf{u}_j = 0, j \neq k$. Orthogonality based on a decomposition of local support type only does not carry over into orthogonality with respect to A. A quantifiable measure of the mutual orthogonality with respect to A of domain decomposition-induced subspaces is furnished later.

Compatibility of Hierarchies of Scales

A practical degree of near-orthogonality is achieved by limiting the support of basis vectors to individual subdomains. The resulting segregation of coarse and fine spaces is a first step in the direction of multigrid, but domain decomposition algorithms often stop after just one or a small number of such steps, for physical, architectural, and algorithmic reasons. Roughly speaking, the number of intermediate scales that are desirable in the algorithm is governed by the number of intermediate scales in the physics and in the computer architecture. Domain decomposition algorithms have one or a few intermediate hierarchical scales, just like the physical systems they solve and the hardware they run on.

The scales present in a physical problem typically include

- the system diameter, L,
- the smallest scale of variation, $\ell \equiv \min_x |u(x)|/|\nabla u(x)|$, requiring resolution, and

- one or more intermediate scales natural to functional or geometrical description of the system.

In aerodynamics, for instance, L might be the wingspan of a vehicle, ℓ might be a fraction of the boundary layer thickness on the wing, and the wing chord or nacelle diameter would be intermediate scales on which it would be natural to develop a set of blocked grids.

The scales present in the memory hierarchy of a supercomputer include

- the global aggregate memory,
- a single floating point arithmetic register, and
- the cache size, the vector length, or the amount of local memory particular to a processor.

It is widely appreciated that the proper adaptation of computational task size to the intermediate scales of a supercomputer is critical to performance.

It is natural to attempt to mimic these scales in a domain decomposition algorithm, which distinguishes

- the integral scale of the domain, $\mathcal{O}(1)$,
- the finest resolved scale, h, and
- an intermediate scale of subdomain diameter, H.

From the point of view of software engineering, maintaining clean data interfaces between these scales encourages modular, adaptable code that has an object-oriented "feel" and maintainability, whether or not it is strictly object-oriented. Algorithms that ignore the natural hierarchies of scale may mismatch the frequency or ease of access of data to its actual local influence. For instance, ADI methods contain implicit aggregates (tridiagonal solves) that cluster together points at opposite ends of a domain, in the tails of each other's Green's functions, simply because they share a common index value. The excellent operation count efficiency of tridiagonal solves justifies this splitting on a computer with flat memory access costs. The efficiency of ADI must be reexamined on a machine with nonflat memory.

From a mathematical point of view, it is natural to recur on the idea of a two-level hierarchical decomposition. After reducing the global fine-grid problem to the union of many local fine-grid problems and a new global coarse-grid problem, one can regard the global coarse grid as the fine grid of a new problem, an approach that leads to multigrid. That domain decomposition and multigrid share a number of special cases is well known, which sets the stage for a consideration of the optimal number of levels of recursion for a practical problem on a real piece of parallel hardware. Stopping at *any* number of levels short of full multigrid can be shown to be suboptimal on theoretical complexity grounds on serial computers. On the other hand, a study of the optimal depth of recursion carried out on an early version of the MasPar MP-1, a massively parallel (8K) SIMD multiprocessor concludes that the optimal number of levels is two (Ref. 2).

TRANSFORMING TO A REDUCED KRYLOV BASIS

It is not convenient to form in parallel the action of an exact inverse to a typical PDE operator, but we have seen that computing an approximate inverse based on domain

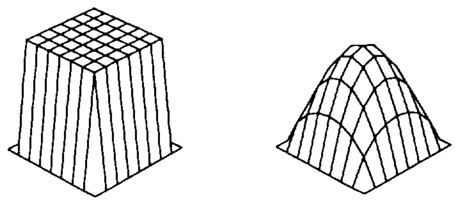

Figure 11 – Surface plots of gridfunctions of the right-hand side (left) and the solution (right) for a model Poisson problem.

decomposition is a reasonably parallelizable task. As is widely appreciated, approximate inverses can be strategically employed as preconditioners for iterative methods. Krylov or "conjugate gradient-like" iterative methods have received lots of attention in recent years because of a key parallel property: their ability to rely on matrix-vector products and vector dot products alone to drive a linear system towards convergence. Krylov methods are algorithms for solving nonsingular linear systems (or singular systems with a known null space) that terminate with the exact solution (modulo precision effects) in a finite number of operations, like direct methods, but typically get "sufficiently close" to the exact solution in fewer operations, like iterative methods. Krylov methods can compete with or surpass most other linear solvers on serial computers and can substantially outperform most other linear solvers on parallel computers. They exemplify a widespread algorithmic trend of "transformation to a reduced basis." Reduced basis methods attempt to represent complex systems (large number of degrees of freedom) with low to moderate numbers of degrees of freedom, without compromising essential phenomena. They often provide for intelligent "user" input and/or problem-specific adaptation. In the case of Krylov methods, this comes through preconditioning.

With respect to Eq. 32, in which A may now be nonsymmetric or even indefinite, the Krylov method of Generalized Minimal Residuals (GMRES) finds an approximate solution

$$\mathbf{u} \approx u_1\mathbf{v}_1 + u_2\mathbf{v}_2 + \cdots + u_m\mathbf{v}_m \tag{38}$$

by choosing the $\mathbf{v}_k$ (generally *not* eigenvectors) so that they: are extensible to the full space, attempt to capture the "most important" components of the solution in the lowest indices, and are convenient to generate and operate with. The "convenient generation" consists of matrix-vector multiplication; the first m Krylov vectors $\{\mathbf{v}_k\}_{k=1}^{m}$ span the Krylov space

$$K_m(A, \mathbf{r}_0) \equiv \left\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\right\}, \tag{39}$$

where $\mathbf{r}_0$ is the residual based on an initial guess, $\mathbf{r}_0 \equiv \mathbf{f} - A\mathbf{u}_0$. "Convenient to operate with" means "orthonormal" in the context of GMRES; the $\mathbf{v}_k$ are formed progressively by a modified Gram-Schmidt orthogonalization of the components of $K_m(A, \mathbf{r}_0)$. GMRES chooses the u_k to get the best possible fit of $\mathbf{u}$ in the span of $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ while keeping m as small as possible, for a given residual tolerance.

Scope does not permit a detailed exposition of GMRES or other comparably successful methods (with somewhat different proportions of matrix and vector operations), such as

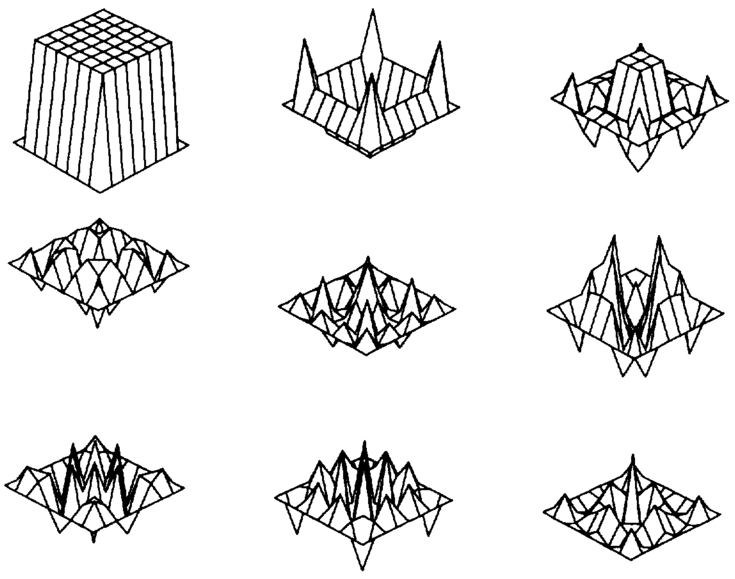

Figure 12 – Surface plots of gridfunctions of the Krylov vectors generated for the GMRES solution of the Poisson problem in Fig. .

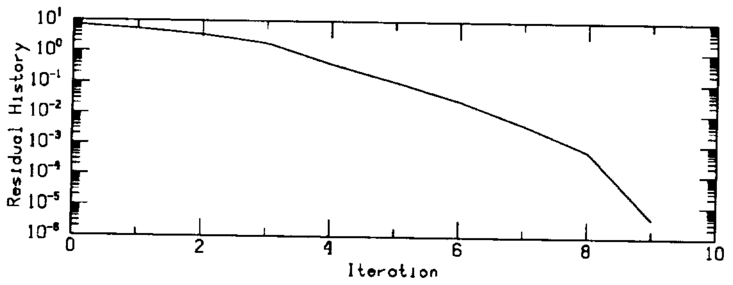

Figure 13 – Iteration history for Poisson equation example, Euclidean norm of residual.

Bi-CGSTAB (Ref. 55) or QMR (Ref. 29), but a simple illustration is effective. For ease of visualization, we use an artificially small example of unpreconditioned GMRES on Poisson's equation on the unit square subdivided into 8×8 uniform grid cells. The right-hand side and the solution are shown in Fig. .

From an initial guess of zero, GMRES forms the solution (to within one part in 10^5 in Euclidean norm of the residual) as a linear combination of nine Krylov vectors, shown in order of generation in Fig. . Each of these gridfunctions is orthogonal to the rest. Note that the first is a simple scaling of the right-hand side. This is the steepest descent direction from the zero initial guess. In this example, it is clear that this step picks up the principal "DC" component of the ultimate solution. The second Krylov vector is taken with opposite sign relative to the first, and corrects for the poor representation of the solution in the corners of the first, and so on.

The iteration history of the Euclidean norm of the residual is shown in Fig. . The converged magnitudes of the coefficients are shown in Fig. ; and Fig. shows the iteration

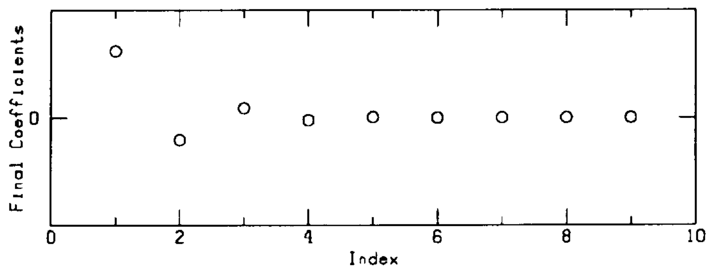

Figure 14 – Converged coefficients of the Krylov vectors of Fig. in the solution of Fig. .

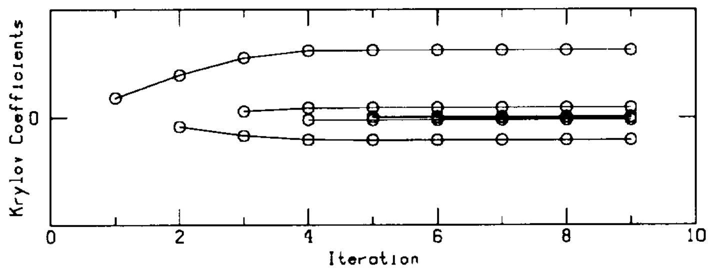

Figure 15 – Iteration history of coefficients for Poisson equation example.

history of each individual coefficient, which appears for the first time during the iteration at which the corresponding vector component is defined. Note that only 9 vectors were needed in this problem of 81 degrees of freedom (boundary points included). Of course, this problem has considerable symmetry, which GMRES exploited. Applying Gaussian elimination to this problem would not have exploited any of its symmetry, and a solution of any utility would require the specification of all 81 coefficients. Gaussian elimination uses as its basis vectors in this problem the 81 unit vectors, rather than the problem-adapted basis of GMRES.

One Krylov vector yields the exact solution to Eq. 32 if A is the identity matrix. Less trivially, it may be proved that the closer A is to the identity matrix in any of several senses, the fewer Krylov iterations will be required. For instance, properties of A that can be exploited in convergence proofs include A having a small number of clusters of eigenvalues and A being a low-rank perturbation of a multiple of the identity matrix (a multiple of the identity having perfect eigenvalue clustering). A more refined understanding of the rate of convergence of Krylov methods based on the details of the eigenvalue distribution is possible; see, for instance, Ref. 54. Preconditioning may be used to transform A towards more rapidly convergent forms. Left preconditioning solves $\tilde{A}x = \tilde{f}$, where $\tilde{A} \equiv B^{-1}A$ and $\tilde{f} \equiv B^{-1}f$. Right preconditioning solves $\tilde{A}\tilde{x} = f$ for $\tilde{x}$, where $\tilde{A} \equiv AB^{-1}$; then $x = B^{-1}\tilde{x}$.

The model problem above was solved again with the popular incomplete LU (ILU) preconditioning (Ref. 46) applied on the right. An incomplete factorization of A into the product of lower and upper factors is a factorization with fill-in limited, so that the union of the indices corresponding to nonzero entries in L and U is the same as the set of indices corresponding to nonzero entries in A. Generally $LU = A + E$, where E is a nonzero remainder matrix. In the notation of the previous paragraph, we take the preconditioner B equal to LU, so $B^{-1} = U^{-1}L^{-1}$. The formation and application

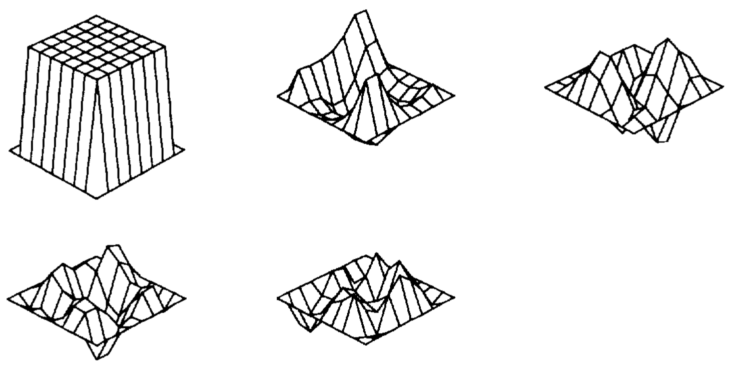
Figure 16 – Surface plots of gridfunctions of the Krylov vectors generated for the GMRES solution of the ILU-preconditioned Poisson problem in Fig. .

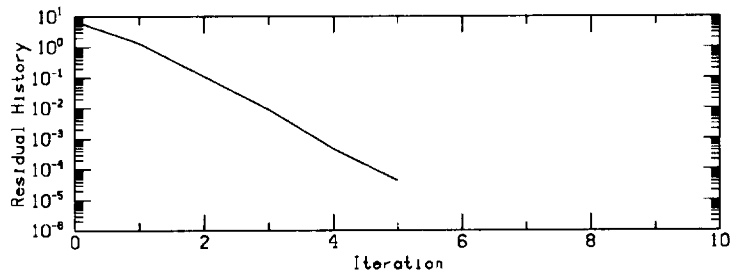

Figure 17 – Iteration history for ILU-preconditioned Poisson equation example, Euclidean norm of residual.

of B^{-1} are inexpensive on serial computers — of the same order as a matrix-vector multiplication when A is sparse. However, they lack the parallelism of a matrix-vector multiplication, and are typically unattractive global preconditioners in massively parallel applications (Ref. 17).

Figs. and depict information for the preconditioned case analogous to that of Figs. and above. Notice that the geometrical symmetry of the individual Krylov vectors is destroyed by the nonsymmetric ILU preconditioning. (We could have used incomplete Cholesky to preserve symmetry, though GMRES does not require it.) Nevertheless, since the new Krylov matrix, AB^{-1} is closer to the identity, the preconditioned iteration converges in roughly one-half as many steps.

Typical elliptic operators do not have well clustered spectra (except for degeneracies of symmetry), but smoothly spaced gaps between eigenvalues. Hence the overall condition number, the ratio of largest to smallest eigenvalue, is a useful if sometimes pessimistic characterization of the overall clustering. In a typical unpreconditioned discrete elliptic operator, the condition number $\kappa(A)$ grows as the square of the ratio of the subdomain diameter to the smallest resolved scale, $\kappa(A) \sim h^{-2}$. From a condition number point of view, the effect of preconditioning with exact subdomain solves is to replace this severe ratio of scales by the much more acceptable ratio of domain diameter to subdomain diameter, $\kappa(B^{-1}A) \sim H^{-2}$. Solving a coarse grid problem in addition to subdomain problems compresses this ratio closer to $\mathcal{O}(1)$.

TAXONOMY OF DOMAIN DECOMPOSITION METHODS

Recent years have witnessed a rapid evolution of algorithms having in common the paradigm of decomposition by domain (Refs. 13, 14, 30, 31, and 40), and in some cases little else. It is beyond our scope to review all that has taken place under the rubric of "domain decomposition." Much of the early research is devoted to the case of a small number of subdomains in which there is no need for a hierarchical formulation, since every subdomain has good access to physical boundary data. This case has limited applicability to parallelism, though it is a model environment for the study of interfacial treatments that possess wider applicability (Ref. 16). There is, however, a core of development for the case of many subdomains in which mathematical theory and computational experiment continue to leapfrog one another in a healthy interplay of consolidation and generalization that we briefly review herein.

The presence or absence of a coarse grid problem is a fundamental criterion by which to classify domain decomposition methods. Algorithms lacking a coarse problem, like the *invertebrates*, are invented with amazing variety, whenever a mathematically hostile niche becomes ripe for parallel computation. Hierarchical algorithms are the *vertebrates* of the domain decomposition kingdom, hardly without variety, but possessing certain common features to which their success can be attributed.

Block diagonally preconditioned iterative methods are classical coarse-grid-free algorithms. Nearly perfectly parallel if load-balanced, they function best when the coupling between degrees of freedom within a block dominates the interblock coupling that is ignored in the preconditioner and left for the outer acceleration method to account for. For small numbers of subdomains and thus blocks, the iteration matrices of such methods fit the desirable category mentioned above of low-rank perturbations to the identity operator. On the other hand, for elliptically dominated problems with roughly equilibrated subdomain sizes, the condition number of the block diagonally preconditioned problem still grows as H^{-2} (Refs. 6 and 56); such methods do not scale well in the fine granularity regime targeted by emerging parallel supercomputers. Multiblock codes with sequential local updating of overlapped regions are commonly used in aerodynamics and likewise fall into this category. With many domains and multicoloring, they may be parallelized, but the concomitant deterioration in convergence rate partially undoes the parallel gains. We should also provide pointers to the significant literature based on the use of Lagrange multipliers to enforce varying degrees of continuity between subdomain solution patches. This is described, for instance, in Refs. 21 and 27, and effectively illustrated on a distributed memory parallel machine in the latter.

After the number of levels, two additional major taxonomical criteria for domain decomposition algorithms are the manner in which the boundary data of the subdomains is updated and the order of the solution of the subproblems within a single iteration. They may be overlapping or nonoverlapping, and additive or multiplicative. Variations abound beyond these classifications as domain decomposition preconditioners are accelerated by different methods and as tuning parameters are introduced in pursuit of practical compromises, such as replacing exact subdomain solves with approximate solves.

Overlapping algorithms, such as the Additive Schwarz described above, make use of extended boundaries that are updated by Dirichlet data. Both fine and coarse grid problems are like the original undecomposed problem in terms of their physical dimension and discrete connectivity. Nonoverlapping algorithms require special treatment for the lower dimensional objects (one-dimensional interfacial edges and two-dimensional interfacial planes in three-dimensional problems) that are distinguished wherever subdomains abut. Rapidly convergent algorithms are known in which these interfaces are updated by approximations to pseudodifferential operators. An equivalence between overlapping and nonoverlapping formulations of domain-decomposed iteration can be established for certain problems (Refs. 4 and 15).

In additive (Jacobi-like) algorithms all subdomains can be updated simultaneously. Multiplicative (Gauss-Seidel-like) algorithms accept partial sequentiality in updating subdomains (or interfacial point sets thereof) in the interest of obtaining improved convergence. Multicolorings can be applied to subdomains, just as they have classically been applied to individual gridpoints, to produce algorithms with graded degrees of additivity and multiplicativity, and correspondingly graded parallel granularity. Nonoverlapping decompositions tend to produce algorithms with an outer multiplicative framework (of which an example, the "tile algorithm," is furnished below) inasmuch as the interfacial data need to be determined first to provide boundary data for the subdomain problems. Most of the work both on interfaces and in subdomains within the appropriate multiplicative phase has concurrency of the same granularity as the decomposition itself. A notable exception is the coarse grid solve for subdomain vertex values. Considerable unity has been brought to the additive/multiplicative classification of domain decomposition methods by the operator polynomial framework of Cai (Ref. 9). Ref. 1 contains a collection of fundamental results on the spectra of sums and products of orthogonal projection operators.

Two-dimensional Algorithms

In the past six years, there has been gratifying progress in the theory of domain decomposition-preconditioned Krylov algorithms for symmetric elliptic problems, and a number of fast methods have been designed for which the condition number of the iteration matrix is uniformly bounded or grows only in proportion to a power of $(1+\log(H/h))$, where H is the diameter of a typical subdomain and h is the diameter of a typical element into which the subdomains are divided, so that the ratio that appears in the bound is roughly the number of discrete degrees of freedom along a subdomain interface. Such algorithms are often called "optimal" or "nearly optimal" algorithms, respectively, though we note that these adjectives pertain to the convergence rate only, and not to the overall computational complexity. The nearly optimal algorithms may still retain terms that are polynomial in $1/H$ or in H/h, depending upon how the component problems (coarse grid, interfaces, subdomains) are solved. For nonsymmetric and indefinite problems, the theory to date is less satisfactory but similar convergence results can be achieved by applying pressure to the coarse grid solve, namely, by making it sufficiently fine.

A seminal result for the case of many subdomains is that of Bramble, Pasciak and Schatz in Ref. 6. For a self-adjoint problem in a two-dimensional region divided into nonoverlapping subdomains, a conjugate gradient-accelerated multiplicative algorithm consisting of two sets of subdomain solves, one set of interface solves, and one coarse grid solve per iteration converges in $\mathcal{O}(1+\log(H/h))$ iterations (the condition number being the square of this quantity). A two-level hierarchical basis plays a crucial role in the BPS-I preconditioner.

The BPS-I preconditioner has a matrix interpretation that we develop below with reference to a blocked version of the original stiffness matrix of the system. Denote by $Au = f$ a piecewise linear finite element discretization of Eq. 11 created by a Galerkin procedure using a non-hierarchical h-level basis. The degrees of freedom in u and f may be permuted so that the coarse grid vertices are ordered last, the interior points first,

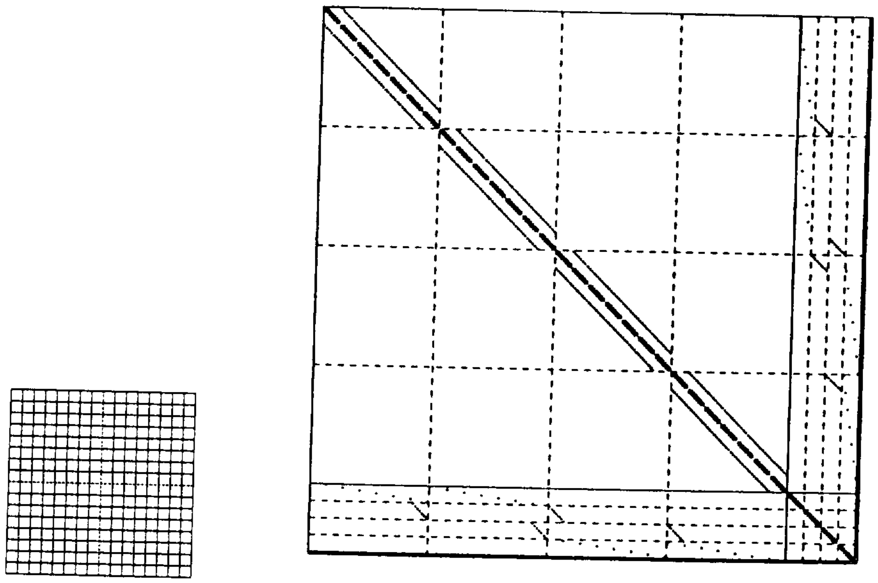

Figure 18 – Decomposition of a uniformly gridded square domain into four subdomains and the corresponding sparsity plot of the blocked matrix A (from Ref. 35).

and the interfaces in between, inducing on A the block structure:

$$A = \begin{pmatrix} A_I & A_{IB} & A_{IC} \\ A_{BI} & A_B & A_{BC} \\ A_{CI} & A_{CB} & A_C \end{pmatrix} \tag{40}$$

(Because of assumed selfadjointness, $A_{BI} = A_{IB}^T$, etc., but we prefer to keep the notation general to accommodate nonselfadjoint operators below.) Note that the partitions vary dramatically in size. If H is a quasi-uniform subdomain diameter and h a quasi-uniform fine mesh width, the discrete dimensions of A_I, A_B, and A_C are $\mathcal{O}(h^{-2})$, $\mathcal{O}(H^{-1}h^{-1})$, and $\mathcal{O}(H^{-2})$, respectively. For the case of a uniformly gridded square domain subdivided into four subsquares, the sparsity pattern of a sample A matrix is sketched in Fig. , assuming natural ordering within each subdomain and suppressing the homogeneous Dirichlet boundary degrees of freedom. For this decomposition, there is a single interior coarse grid vertex, so A_C is a scalar.

Consider, in addition, an alternative hierarchical basis with the usual elements of local $\mathcal{O}(h)$-diameter support for all degrees of freedom *except* for coarse grid vertices, and with special $\mathcal{O}(H)$-diameter elements replacing the local elements at the vertices. Each special element is 1 at one vertex, 0 on all other vertices, 0 at all nodes interior to any subdomain, and extends linearly along the edge connecting any adjacent pair of vertices. (For a uniform decomposition into square subdomains, the special elements are cross-shaped.) With respect to this modified basis, we have a modified stiffness matrix:

$$\tilde{A} = \begin{pmatrix} A_I & A_{IB} & \tilde{A}_{IC} \\ A_{BI} & A_B & \tilde{A}_{BC} \\ \tilde{A}_{CI} & \tilde{A}_{CB} & \tilde{A}_C \end{pmatrix} \tag{41}$$

The BPS-I preconditioner can now be written in factored form

$$\begin{pmatrix} I & -A_I^{-1}A_{IB} & -A_I^{-1}\tilde{A}_{IC} \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A_I & 0 & 0 \\ 0 & B_B & 0 \\ 0 & 0 & B_C \end{pmatrix}^{-1} \begin{pmatrix} I & 0 & 0 \\ -A_{BI}A_I^{-1} & I & 0 \\ -\tilde{A}_{CI}A_I^{-1} & 0 & I \end{pmatrix}, \quad (42)$$

where it remains to define the new matrix blocks B_B and B_C. We write the preconditioner in this form to enable comparisons with other factored forms below. In an actual implementation, the same action as B^{-1} defined above is carried out through a process involving two solves on each subdomain per iteration, one of which satisfies homogeneous boundary conditions with an inhomogeneous interior and the other inhomogeneous boundary conditions with a homogeneous interior. Note that A_I^{-1} is an operation that can be performed independently on each subdomain, since (see Fig.) A_I is itself block-diagonal. B_C is, to within a scaling, simply a global coarse grid discretization of the original operator. B_B is a block diagonal matrix with one block for each interface. Each interfacial block is the discretization of a pseudodifferential operator — the square root of the Laplace operator — on the interface. See Ref. 3 for a discussion as to why the square root of the Laplacian is a good preconditioner for the interfaces, and Ref. 6 for a complete convergence proof. B_B is computationally convenient, since it can be implemented in $\mathcal{O}((H/h)\log(H/h))$ operations via fast Fourier transforms. Note that in implementing the preconditioner the $\mathcal{O}(H^{-2})\times\mathcal{O}(h^{-2})$ matrix $\tilde{A}_{CI}$ forms ramp-weighted averages of the interior values which serve as the right-hand side of the solve with B_C. Its transpose $\tilde{A}_{IC}$ linearly interpolates the coarse-grid vertex values along the edges to serve as boundary values for the final set of interior solves with A_I. In the original algorithm, Bramble, Pasciak & Schatz already proposed replacing A_I^{-1} in the preconditioner with another approximate block B_I^{-1}, where, for instance fast Poisson solvers might be employed as spectrally equivalent replacements for non-constant-coefficient operators, or, as in Ref. 53, a small number of multigrid cycles might be employed as an approximate inverse. A theoretical discussion of this practice and its effect on convergence may be found in Ref. 5.

To appreciate the low computational complexity and power of the BPS-I preconditioner, note that the inverse of the full stiffness matrix, Eq. 40, from the nonhierarchical basis can be written in analogous partially factored form with more complex intermediate factors (Ref. 53):

$$\begin{pmatrix} I & -A_I^{-1}A_{IB} & -A_I^{-1}A_{IC} \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & -S_B^{-1}S_{BC} \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A_I & 0 & 0 \\ 0 & S_B & 0 \\ 0 & 0 & T_C \end{pmatrix}^{-1} \times$$

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & -S_{CB}S_B^{-1} & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ -A_{BI}A_I^{-1} & I & 0 \\ -A_{CI}A_I^{-1} & 0 & I \end{pmatrix}. \quad (43)$$

The various first-level Schur complements are defined as $S_B \equiv A_B - A_{BI}A_I^{-1}A_{IB}$, $S_{BC} \equiv A_{BC} - A_{BI}A_I^{-1}A_{IC}$, and $S_{CB} \equiv A_{CB} - A_{CI}A_I^{-1}A_{IB}$. T_C is a second-level Schur complement defined in terms of $S_C \equiv A_C - A_{CI}A_I^{-1}A_{IC}$ by $T_C \equiv S_C - S_{CB}S_B^{-1}S_{BC}$. These Schur complements are symmetric positive definite if A is (Ref. 19), hence guaranteed to be invertible, but they are dense, hence impractical for large problems. Because the inverses of the Schur complements are just Green's functions for the statically condensed problems, with favorable decay properties similar to the full dimensional Green's functions, an industry of "probing" them to form replacements of imposed sparsity has been developed (see Ref. 18 for a recent systematic review). In addition to adapting to the coefficient structure of the problem, probing has the philosophical attraction of being purely algebraic and can lead to substantial advantages in some applications. However, a probing technique of fixed sparsity pattern generally does not scale well as the resolution of the problem is increased. Without assembling a single Schur complement, the BPS-I preconditioner comes within a polylogarithm factor of being spectrally equivalent to the full stiffness matrix in the original basis. BPS-I is also a symmetrizable preconditioner, so that when A is itself symmetric, the preconditioned system may be accelerated with the conjugate gradient method (Ref. 32).

The "tile algorithm" of Refs. 10 and 37 is an attempt to package *some* of the power of the hierarchical BPS-I preconditioner into a more general form that does not depend on nor exploit symmetry of A and does not require two subdomain solves per iteration. Such generality is required before domain decomposition algorithms can be directly applied to convection-diffusion problems. Much experimentation yields an effective form for the preconditioner that can be written for comparison with the foregoing as follows:

$$\begin{pmatrix} I & -A_I^{-1}A_{IB} & -A_I^{-1}A_{IC} \\ 0 & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} A_I & 0 & 0 \\ 0 & T_B & 0 \\ 0 & 0 & B_C \end{pmatrix}^{-1} \times$$

$$\begin{pmatrix} I & 0 & 0 \\ 0 & I & -(A_{BC} - N_{BC})B_C^{-1} \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & W_{CB} & W_C \end{pmatrix} \tag{44}$$

The new notation includes T_B, a block diagonal matrix whose blocks are the discrete approximations to the terms of the original differential whose derivatives are tangential with the interface, and N_{BC}, an approximation to the leftover terms whose derivatives are normal to the interface. These normal terms are approximately interpolated from the coarse grid solution. The nonnegative weight matrices W_C and W_{CB} assemble a ramp-weighted average of the function values along the interfaces for use as the right-hand side of the coarse grid system. The row sums of W_{CB} and W_C are unity. As in the BPS-I method, it is tempting to replace the subdomain solves with approximate solves.

No theoretical convergence results have been given for the tile algorithm, but a closely related nonoverlapping multiplicative algorithm was proved in Ref. 11 to possess a condition number of $\mathcal{O}\left((1+\log(H/h))^3\right)$ for nonselfadjoint and/or indefinite problems. The proof requires that the coarse grid be taken "sufficiently fine," i.e., that the subdomain Reynolds number is sufficiently small in nonsymmetric problems, and that the oscillatory behavior of the Green's function is resolved by the coarse grid in indefinite problems. How "good" is convergence in $\mathcal{O}\left((1+\log(H/h))^p\right)$ iterations, where p is near unity? Recall that for a self-adjoint elliptic problem with smooth coefficients, point Jacobi takes $\mathcal{O}(1/h^2)$ iterations, point SSOR or conjugate gradients separately each take $\mathcal{O}(1/h)$, the combination of conjugate gradients preconditioned with point SSOR takes $\mathcal{O}(1/\sqrt{h})$, and multigrid takes $\mathcal{O}(1)$. The nonhierarchical methods all exhibit deteriorating convergence with the demand for greater resolution. By scaling the decomposition granularity parameter H to the resolution parameter h, domain decomposition methods put the burden of increasing resolution on an increasingly fine coarse grid. This is a "one-time" gain. By recurring on the coarse grid, multigrid maintains optimal algebraic convergence and simultaneously permits a small coarsest grid. Increasing communication-to-computation ratios on the coarser grids eventually impose a practical bound on the depth of this recursion. Multigrid depends on the ability to obtain discretizations of the operator at each of many scales; two-level domain decomposition at only one scale in addition to the finest. This intermediate scale H should most often be chosen with regard to architectural limits.

The parallel complexity of domain decomposition, including local and global communication costs per iteration has been considered in Refs. 34 and 35 with a particular emphasis on the best way to carry out the coarse grid solve. This solve becomes the parallel bottleneck when the number of subdomains is large. For moderate numbers of processors, globally broadcasting the weighted right-hand side and solving redundantly in parallel may be the optimal strategy. For large numbers of processors, precomputing the local influence functions for the coarse grid degrees of freedom and storing them on appropriate processors may be optimal (Ref. 37).

Results analogous to the polylogarithm convergence theorems above for nonoverlapping preconditioners have also been proved for additive overlapping preconditioners. A typical proof for selfadjoint problems employing either nonoverlapping or overlapping decompositions is based on bounding the smallest eigenvalue of the preconditioned system from below and largest eigenvalue from above. As the ratio these extreme eigenvalues approaches unity, the spectrum clusters, and a conjugate gradient-like method converges rapidly. By Rayleigh quotient arguments, the relevant convergence parameter is the ratio of the smallest $\hat{c}$ to the largest $\check{c}$ for which the double inequality

$$\check{c}(u^T Bu) \leq (u^T Au) \leq \hat{c}(u^T Bu) \tag{45}$$

holds for all u, where A is the discrete stiffness matrix and B the discrete preconditioner. If this ratio is independent of discretization and decomposition parameters, B is said to be "spectrally equivalent" to A, and a rapidly convergent method obtains. Proofs generally proceed by decomposing a general u into subspaces and bounding individual pieces of the quadratic forms separately.

The logarithms in the nonoverlapping convergence bounds can be traced in the proofs to the application of trace theorems to bound interfacial quantities, and can be removed altogether in overlapping methods. Hence, we have the famous Additive Schwarz result of Dryja & Widlund (Ref. 22) establishing an optimal, multigrid-like $\mathcal{O}(1)$ condition number for selfadjoint overlapping problems, and its extension to nonselfadjoint, possibly indefinite problems by Cai (Ref. 8) for the same $\mathcal{O}(1)$, given some requirements on the fineness of the coarse grid. Early Additive Schwarz proofs were restricted to smooth coefficients and fixed overlap of $\mathcal{O}(H)$. Numerical experiments revealed these restrictions to be pessimistic in many cases, and they are gradually being removed from the theory through refined analysis. Proofs for nonselfadjoint problems depend on hypotheses that render the symmetric part of A dominant over the skewsymmetric part. Again, numerical experiments have frequently shown such hypotheses to be pessimistic; however, more general theorems seem evasive.

To give an example of the convergence performance of the methods described in this

	$H = 1/4$			$H = 1/8$			$H = 1/16$	
$h =$	1/32	1/64	1/128	1/32	1/64	1/128	1/64	1/128
tile	21	23	31	27	35	50	21	34
CGK	38	37	37	33	30	33	22	24
ASM	33	35	35	29	26	25	19	18
MSM	15	15	14	16	15	15	10	10

	$H=1$		
ILU(0)	44	78	312
ILU(1)	28	44	99
ILU(2)	22	36	76

Table 1 – Iteration counts for solving the variable-coefficient, nonsymmetric indefinite problem.

section on a difficult model problem, we consider a nonsymmetric, indefinite problem on the uniformly gridded unit square from the test suite in Ref. 10:

$$\begin{aligned} \mathcal{L}u &\equiv -\tfrac{\partial}{\partial x}((1+\tfrac{1}{2}\sin(50\pi x)\tfrac{\partial u}{\partial x}) - \tfrac{\partial}{\partial y}((1+\tfrac{1}{2}\sin(50\pi x)\sin(50\pi y)\tfrac{\partial u}{\partial y}) \\ &\quad +20\sin(10\pi x)\cos(10\pi y)\tfrac{\partial u}{\partial x} - 20\cos(10\pi x)\sin(10\pi y)\tfrac{\partial u}{\partial y} - 70u \\ &= f, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{46}$$

The coefficients of the second-order terms oscillate but do not change sign. The coefficients of the first-order terms represent physically a ten-by-ten array of closed convection cells, with no convective transport between cells. However, the interfaces of the decomposition do not in general align with the convection cell boundaries, so this zero-convective-flux property is not exploited. The operator $\mathcal{L}$ is discretized by the five-point central-difference method. Table shows the number of iterations required to obtain a relative reduction in the Euclidean norm of the residual of 10^5 for problem sizes of up to 16,129 degrees of freedom.

Compared are the tile algorithm, the theoretically amenable version of the tile algorithm in Ref. 11 (denoted "CGK"), the Additive Schwarz method ("ASM") and a multiplicative version of the Schwarz method ("MSM") obtained by a multicoloring of the subdomains of Additive Schwarz. A fixed overlapping factor of $H/4$ in both x and y directions is employed both Schwarz methods. For comparison's sake, the performances of global ILU preconditioners with three successively greater levels of fill-in are also shown. All methods are accelerated by GMRES without restarting. The global ILU preconditioners are overwhelmingly outperformed by domain decomposition preconditioners at fine mesh sizes, even apart from parallelism.

This problem is difficult for all of the methods, but the iteration count for MSM is smaller than that of others by almost a factor of 2, or more. For a fixed coarse mesh size H, some methods tend to require fewer iterations when the fine mesh is refined; others require more. This behavior is believed related to the oscillatory coefficients in the second-order terms of $\mathcal{L}$. The discretization becomes more stable when h gets smaller relative to the wavelength of the oscillatory coefficients. The asymptotic mesh- and decomposition-independence of the Schwarz methods is evident even at rather modest problem size.

Three-dimensional Algorithms

In three dimensions, it is useful to distinguish between two types of hierarchical algorithms: vertex-based and wirebasket. In the latter, more degrees of freedom than simply the subdomain vertices participate in the nonlocal part of the preconditioner, such as a collection of degrees of freedom shared by several subdomains along an edge. For overlapping algorithms, convergence bounds and proofs carry over from two-dimensional to three-dimensional vertex-based schemes with little new difficulty. Nonoverlapping vertex-based algorithms encounter difficulty in three dimensions. The two-dimensional bound of $\mathcal{O}\left((1+\log(H/h))^2\right)$ on the condition number must be replaced with a sharp $\mathcal{O}\left((H/h)(1+\log(H/h))\right)$. It is not difficult to trace the origin of the linear factor in the discrete subdomain size, H/h, in estimates based on the form of Eq. 45; see Ref. 52. The problem is inherent in estimating terms arising from vertex degrees of freedom in the local basis by the corresponding terms in the hierarchical basis. The situation is less one of special problems in three dimensions than it is one of fortuitous cancellation in two dimensions: the two orders of differentiation in the dominant elliptic operator balance the two length dimensions in the differential area element of the stiffness matrix integrals, making the Galerkin inner product scale-invariant. In three dimensions, there is a power of length left over in the differential volume element. There are different solutions to this problem of linearly interpolating from a single vertex point over an entire subdomain. One is to employ vertex elements of higher polynomial degree. Another is to involve more points. Either way, a richer basis for the global problem is needed to allow better approximation.

Significant extensions of nonoverlapping methods in three dimensions, eliminating the linear growth in the condition number, are the wirebasket-based methods of Bramble, Pasciak & Schatz (Ref. 7), Mandel (Refs. 45 and 44), and Smith (Ref. 52). The last yields the most practical parallel algorithm, in that local and global subproblems of the preconditioner can be solved concurrently. The bottleneck of a global problem solution in three-dimensional problems — even a problem whose size relative to total number of degrees of freedom is very small — can be very significant. The intricacy of wirebasket preconditioners is beyond the scope of this overview to present. A key idea in all three approaches is that the null spaces of the elemental contributions to the stiffness matrix A and to the preconditioner B arising from each subdomain must be the same. For problems of few subdomains in which each touches a segment of the physical Dirichlet boundary, this condition is automatically satisfied by any reasonable preconditioner, since the null spaces are trivial. However, the local stiffness matrix of an interior subdomain of a scalar elliptic problem — not tied down by any boundary values — has as its null space the constant functions, and multicomponent problems may have richer null spaces. Preserving the null space in a preconditioner whose nonlocal component is more complex than a piecewise linear vertex space complicates both the construction and the proof of convergence.

Dryja and Widlund (Ref. 23) have proposed a unifying abstract framework for analyzing the quality of elliptic preconditioners that work by decomposing the solution space into subspaces that is elegant in its compactness. Let u be written as a sum of its projections into subspaces V_k, $k = 0, \ldots, K$, whose union is the full space:

$$\text{for all } u \in V, u = \sum_{k=0}^{K} u_k, \text{ where } u_k \in V_k, \text{ and } V_0 + V_1 + \cdots + V_K = V. \qquad (47)$$

Let $a(u, v)$ be a symmetric positive definite bilinear form and a solution $u \in V$ be sought such that

$$a(u, v) = (f, v) \tag{48}$$

for all $v \in V$, where $(\cdot, \cdot)$ is the ordinary inner product. In matrix notation, this is just Eq. 32. For each subspace V_k, let a symmetric positive definite bilinear form $b_k(u, v)$ be defined that approximates $a(u, v)$ on V_k. We are willing to solve subspace problems using the $b_k(\cdot, \cdot)$ to precondition $a(\cdot, \cdot)$. There are three key questions that need to be analyzed in any prospective choice of the $b_k(\cdot, \cdot)$ (Ref. 23):

- How much do the subspaces overlap? Mathematically, how small can C_1 be taken such that, for all $u \in V$, the sum of the energies of the subdomain preconditioner inner products is bounded by C_1 times the global energy,

$$\sum_k b_k(u_k, u_k) \leq C_1 a(u, u)\ ? \tag{49}$$

- How well does $b_k(\cdot, \cdot)$ approximate $a(\cdot, \cdot)$ in its own subspace? Mathematically, how small can C_2 be taken such that, for all $u_k \in V_k$, and all k, C_2 times the energy of each subdomain preconditioner inner product bounds the energy of the corresponding local component,

$$a(u_k, u_k) \leq C_2 b_k(u_k, u_k)\ ? \tag{50}$$

- How orthogonal are the subspaces for $k > 0$? Mathematically, what is the smallest spectral radius $\rho(\epsilon)$ of the $K \times K$ matrix ϵ whose elements ϵ_{ij} satisfy for all $u_i \in V_i$, $u_j \in V_j$, $i, j = 1, \ldots, K$

$$a(u_i, u_j) \leq \epsilon_{ij} \sqrt{a(u_i, u_i) a(u_j, u_j)}\ ? \tag{51}$$

Given C_1, C_2, and $\rho(\epsilon)$, the condition number of the Additive Schwarz-transformed system can be bounded by

$$\kappa \leq (1 + \rho(\epsilon)) C_1 C_2. \tag{52}$$

Equation 51 is the quantitative measure of orthogonality between subspaces promised in the discussion following Eq. 37. If subspaces V_i and V_j do not overlap, ϵ_{ij} is the cosine of an angle between them. In this case, Eq. 51 is known as the strengthened Cauchy-Buniakowskii-Schwarz (CBS) inequality. For a recent review of the role of the CBS inequality in multilevel methods, see Ref. 25. If there is no distinguished space, V_0, that is exempt from the orthogonality test of Eq. 51, then the bound is instead

$$\kappa \leq \rho(\epsilon) C_1 C_2. \tag{53}$$

We illustrate this formalism for a familiar ideal case: the eigendecomposition introduced earlier following Eq. 32. K is the dimension of the matrix A and each V_k is the ray spanned by the eigenvector $\mathbf{v}_k$. The solution $\mathbf{u}$ has the decomposition $\mathbf{u} = \mathbf{u}_1 + \mathbf{u}_2 + \cdots + \mathbf{u}_K = u_1\mathbf{v}_1 + u_2\mathbf{v}_2 + \cdots + u_K\mathbf{v}_K$. $a(\mathbf{u}, \mathbf{w})$ is the scalar $\mathbf{w}^T A \mathbf{u}$ and $b_k(\mathbf{u}, \mathbf{w})$ is the scalar $\mathbf{w}^T \lambda_k \mathbf{u}$ in each subspace k.

- Notice that, in this example, the subspaces do not overlap at all. We may take C_1 to be unity:

$$\sum_k b_k(\mathbf{u}_k, \mathbf{u}_k) = \sum_k \lambda_k u_k^2 = \sum_k \mathbf{u}_k^T A \mathbf{u}_k = a(\mathbf{u}, \mathbf{u}). \tag{54}$$

- Notice that, in this example, $b_k(\cdot,\cdot)$ perfectly approximates $a(\cdot,\cdot)$ in its own subspace. We may take C_2 to be unity:

$$a(\mathbf{u}_k, \mathbf{u}_k) = \mathbf{u}_k^T A \mathbf{u}_k = \lambda_k u_k^2 = b_k(\mathbf{u}_k, \mathbf{u}_k). \tag{55}$$

- Notice that, in this example, the subspaces are perfectly orthogonal. We may take ϵ to be the identity matrix:

$$\begin{array}{rcll} a(\mathbf{u}_i, \mathbf{u}_j) & = & (u_j \mathbf{v}_j)^T \lambda_i (u_i \mathbf{v}_i) = \lambda_i u_j u_i \mathbf{v}_j^T \mathbf{v}_i = 0 & \text{for } i \neq j, \\ a(\mathbf{u}_i, \mathbf{u}_i) & = & \lambda_i u_i^2 = \sqrt{a(\mathbf{u}_i, \mathbf{u}_i) \cdot a(\mathbf{u}_i, \mathbf{u}_i)} & \text{otherwise.} \end{array} \tag{56}$$

The spectral radius of ϵ is 1. Hence, we have for this trivial example that

$$\kappa \leq \rho(I) \cdot 1 \cdot 1 = 1. \tag{57}$$

The condition number of this perfectly preconditioned system is unity.

Parallel Promise and Generalizations

Hierarchical domain decomposition algorithms can be effectively implemented on distributed memory multiprocessors. In 1990 (as reported in Ref. 38 using the tile algorithm), a 103,201-unknown 2D elliptic problem partitioned into 768 subdomains was solved in the sense of a 10^5-fold reduction in Euclidean norm of the residual on a 64-processor Intel iPSC/860 in 12 iterations requiring 1.8 wall-clock seconds. Load balancing was complicated by local uniform mesh refinement in the vicinity of a reentrant corner in an L-shaped domain, preventing still better timing. Aggregate megaflop ratings across different phases of the computation varied from a bottlenecked 2.0 Mflops on the solution of the global coarse problem of $\mathcal{O}(10^3)$ degrees of freedom (one per subdomain, modulo edge effects) to 362 Mflops on the purely parallel factorization of the interior blocks of the preconditioner. (The 5.7 Mflops per node represented by the latter is recognized to be near the realizable peak for compiled FORTRAN on the i860.) Between these extreme phases of either extensive non-local communication or no communication whatsoever lies the phase of the matrix-vector multiplication with the unpreconditioned stiffness matrix, requiring local communication only. This ran at about 210 Mflops aggregate, or about 58% of the peak. The local parts of the preconditioning ran at an aggregate 271 Mflops.

In 1991, as reported in Ref. 53, a 1,030,512-unknown 3D elliptic problem on highly non-convex domain partitioned into 244 subdomains was solved on a 32-processor Intel iPSC/860 in 35 iterations requiring 88.5 seconds of wall-clock time. Single multigrid V-cycles were employed for the local subdomain solves. It is interesting to note that a simple diagonally preconditioned conjugate gradient solver required only about 2.7 times as much wall-clock time to converge to an answer of the same quality, although the condition number of the preconditioned system was more than a thousand times worse (approximately 7.85×10^4 for the diagonal preconditioning versus approximately 74.1 for the wirebasket-based preconditioning) and the number of iterations was 617 instead of just 35.

Another interesting benchmark reported in 1991 (Ref. 2) was the solution of an elliptic problem in two dimensions containing 1,979,649 unknowns divided into 16,384 subdomains with jumps of up to three orders of magnitude between piecewise constant coefficients between adjacent subdomains. This required 42 wall-clock seconds and 27 iterations on an 8,192-processor MasPar MP-1. In this SIMD environment, a hybrid Additive Schwarz/multigrid algorithm was employed with the role for multigrid reversed relative to the MIMD implementation above. Three multigrid V-cycles were employed as an approximate solver for the coarse grid, rather than one V-cycle as an approximate solver for the subdomains.

Results such as these indicate both the impressive scaling of convergence rates to truly large problems for hierarchical methods, and their vulnerability to the large latency of a distributed memory parallel computer in the execution of the hierarchical part of the preconditioner. This is a tradeoff that must be carefully watched as high performance fine granularity parallel supercomputers continue to evolve. As computer architectures evolve, two-level domain decomposition algorithms can evolve flexibly and modularly with them, replacing vulnerable components of the preconditioner with mathematically nearby but computationally better adapted components. These replacements trade numerical efficiency for implementation efficiency, but do not compromise the accuracy of the solution since the A matrix is unaffected.

EXAMPLES OF PARALLEL COMPUTATIONAL FLUID DYNAMICS

In this section, two model computational fluid dynamics problems are culled from earlier parallel domain decomposition papers co-authored with W. D. Gropp (Refs. 36 and 42) as illustrations of the technique. We do not represent that these examples define the current state of the art in any particular aspect, whether of modeling, or of convergence rate, or of parallel performance. Rather, they are presented as intermediate points along a path designed to lead from the linear, scalar problems most amenable to theoretical analysis and predominantly discussed above to more realistic challenges of computational fluid dynamics.

For the solution of steady flow problems, robust variations of Newton's method, assisted as necessary by parameter continuation such as artificial time or artificial viscosity, are often preferable in terms of overall execution time to less fully coupled iterative methods or associated explicit time-marching methods. The overall system written in the form

$$F(\phi) = 0, \tag{58}$$

where ϕ represents a column vector of all of the unknowns, may be solved efficiently by a damped modified Newton method provided that an initial iterate $\phi^{(0)}$ sufficiently close to the solution is supplied. The iteration is given by

$$\phi^{(k+1)} = \phi^{(k)} + \lambda^{(k)}\delta\phi^{(k)}, \tag{59}$$

where

$$\delta\phi^{(k)} = -(\tilde{J}^{(k)})^{-1}F(\phi^{(k)}), \tag{60}$$

where the matrix $\tilde{J}^{(k)}$ is an approximation to the actual Jacobian matrix evaluated at the k^{th} iterate. We refer to $\delta\phi^{(k)}$ as the k^{th} update. When $\lambda^{(k)} = 1$ and $\tilde{J}^{(k)} = J^{(k)} \equiv \frac{\partial F}{\partial \phi}(\phi^{(k)})$, for all k, a pure Newton method is obtained. The iteration terminates when

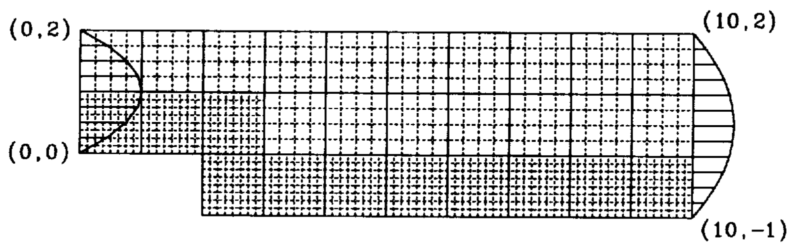

Figure 19 – Schematic of a composite grid for the backstep flow problem, with well-developed inflow (left) and outflow (right) velocity profiles superposed. The upper and lower surfaces are rigid walls. Refinement is employed near the step and in the recirculation region. (The composite grids actually used to generate the data in the following section are finer than shown here.)

some (scaled) 2-norm of $\delta\phi^{(k)}$ drops below a given tolerance. In well-conditioned systems this will, of course, also be true of the norm of $F(\phi^{(k)})$.

From the discussion of Eqs. 59 and 60 we identify the five basic tasks that together account for almost all of the execution time required by the Newton algorithm: (1) DAXPY vector arithmetic, (2) the evaluation of residual vectors, (3) the evaluation of Jacobians, (4) the evaluation of norms, and (5) the solution of linear equations involving the Jacobian matrix. The DAXPY requires no data exchanges between neighboring points. The residual and Jacobian evaluation (performed analytically here) require only nearest-neighbor data exchanges. The evaluation of norms and the linear system solution require global data exchanges and are hence the proper focus of a parallel implementation. We concentrate on the linear system solution alone, employing the nonoverlapping multiplicative tile algorithm in a block $n_c \times n_c$ sense, where there are n_c degrees of freedom at each gridpoint.

Incompressible Flow Over a Backstep

The flow over a backstep is a classic model problem from computational fluid dynamics. Our channel has a flat no-slip wall opposite the step, a fully developed inlet profile (located two step heights upstream), and a channel expansion ratio of 2 to 3 occurring abruptly at the step (see Fig.).

Inasmuch as the flow is well characterized as laminar, steady, and two-dimensional in the Reynolds number range we model, we use the streamfunction-vorticity formulation of the incompressible Navier-Stokes equations, in which velocity components (u, v) are replaced with (ψ, ω) through

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}, \text{ and } \omega = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x}. \tag{61}$$

The streamfunction satisfies the Poisson equation

$$-\nabla^2 \psi + \omega = 0, \tag{62}$$

and the vorticity the convection-diffusion equation (not in divergence form)

$$\frac{\partial \psi}{\partial y}\frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x}\frac{\partial \omega}{\partial y} - \nu\nabla^2 \omega = 0. \tag{63}$$

$h_{\text{eff}}^{-1} = 10$		$h_{\text{eff}}^{-1} = 20$				$h_{\text{eff}}^{-1} = 40$			
Global		Local		Global		Local		Global	
$N = 5,862$		$N = 10,422$		$N = 22,922$		$N = 40,742$		$N = 90,642$	
$p = 2$		$p = 4$		$p = 8$		$p = 16$		$p = 32$	
I_1	T	I_1	T	I_1	T	I_1	T	I_1	T
8	16.3	11	17.2	11	20.6	13	29.9	14	49.3

Table 2 – Number of GMRES iterations in the first Newton step, I_1, and time, T (in sec) for the full nonlinear solution, over a roughly linearly scaled range of numbers of processors, p, and numbers of degrees of freedom in the discretization, N.

Thus, n_c is 2 in this example. Note that both governing equations are elliptically dominated at sufficiently small cell Reynolds number, Uh_{eff}/ν. In Ref. 36 this problem is solved using a pure Newton method directly from an initial guess consisting of the inlet profile extrapolated unchanged downstream, patched to an initially stagnant region behind the step. Local uniform mesh refinement "h-type" refinement and second-order upwinding "p-type" refinement are employed in the A matrix and the accuracy of the solutions verified over a modest range of Reynolds numbers for which essentially two-dimensional steady laminar solutions are known. Among the various tables and graphs in Ref. 36, we focus on Table , obtained on a 32-processor Intel iPSC/860 using the tile algorithm described above. The computational grid is circumscribed by a rectangular domain spanned by 20 (square) tiles in the main flow direction and 6 tiles across the channel. The step height is two tile edges high. Because the 4×2 tiles occupied by the step itself are not included in the computational domain, the total number of tiles in the decomposition is $120 - 8 = 112$. The Reynolds number based on the step height and the centerline inlet velocity is 100. The effective resolution parameter of the mesh, h_{eff}, is the number of mesh intervals per unit step height. Labels "global" and "local" in the table refer to the span of the refined regions; "global" is for a globally uniformly refined grid based on the the listed h_{eff} and "local" is for a grid refined to the given h_{eff} only on tiles abutting the step sidewall and in a triangular region downstream of the step, with resolution twice as coarse elsewhere.

From the table, we observe first the logarithmic growth of I_1 in H/h_{eff}. As the latter goes through a pair of doublings, the number of Krylov iterations in the first Newton step increases linearly from 8 to 11 to 14. The three global grids are geometrically self-similar and a common initial guess for the first Newton step makes the problems algebraically similar as well, so the comparison is meaningful. We also observe that the parallel scaling of the algorithm is far from perfect at present, though it contains grounds for optimism. In a well-scaled algorithm, we would expect the execution time to remain constant as problem size and processor force are increased in proportion. The relatively large jump in execution time going from 16 to 32 processors can be attributed, in large part, to poor load balance. The 112 equally-sized tiles cannot be evenly distributed over 32 processors; until that point in the table, load imbalance is either nonexistent (for the globally refined cases of 2 and 8 processors) or small (for the locally refined cases of 4 and 16 processors, with accommodation for different discrete tile sizes). The gradual deterioration of execution time over the first four columns is partly attributable to the growing discrete dimension of the subdomain problems, which are handled with

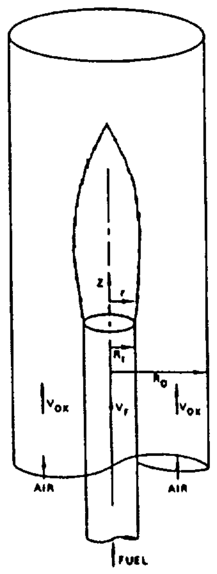

Figure 20 – Schematic of a confined non-premixed laminar jet flame.

polynomially complex exact solves, partly to a coarse grid system of constant size that must be assembled with the coordination of an increasing number of processors, and partly to the logarithmic convergence degradation. The first of these problems can be addressed with multigrid subdomain solves. The raw performance of the algorithm-machine combination is encouraging: full steady-state convergence of a well-resolved (90,642-unknown) nonlinear problem in less than one minute.

Variable Density Chemically Reacting Flow

The second example is physically more complex but algorithmically simpler. From Ref. 42, we report on a stripwise domain decomposition algorithm applied to a set of four Jacobians drawn from an axisymmetric, buoyant laminar methane-air diffusion flame in a high-aspect ratio geometry. Through an asymptotic model known as the "flamesheet," a meaningful model containing only three degrees of freedom per point may be derived. (In more detailed kinetics models, several dozens of chemical species may need representation at gridpoints in high-temperature regions.) A schematic is given in Fig. .

We again use a streamfunction-vorticity formulation, this time accommodating the axisymmetric geometry and the variable density through the more general definitions:

$$\rho r v_r = -\frac{\partial \psi}{\partial z}, \quad \rho r v_z = \frac{\partial \psi}{\partial r}, \quad \omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r} \tag{64}$$

(For historical reasons, the sign convention of the streamfunction is reversed relative to Eq. 61.)

The streamfunction equation, representing overall mass conservation, becomes

$$\frac{\partial}{\partial z}\left(\frac{1}{r\rho}\frac{\partial \psi}{\partial z}\right) + \frac{\partial}{\partial r}\left(\frac{1}{r\rho}\frac{\partial \psi}{\partial r}\right) + \omega = 0. \tag{65}$$

	J31-1			J31-2			J31-3			J63-1		
p	I_1	T	e	I_1	T	e	I_1	T	e	I_1	T	e
1	6	25.1	1.00	14	56.1	1.00	14	56.2	1.00	18	309.	1.00
2	6	13.1	.96	15	31.6	.89	16	33.7	.83	18	158.	.98
4	6	6.9	.91	16	17.2	.82	17	18.3	.77	18	80.8	.96
8	5	3.0	1.05	17	9.5	.74	20	11.3	.62	18	40.7	.95
16										23	30.7	.63

Table 3 – Number of linear (GMRES) iterations in solving the for Newton step with the given Jacobian, I, time, T (in sec) for the Newton step, and overall efficiency (relative to undecomposed algorithm on one processor) as the number of processors/subdomains p increases.

The vorticity equation, representing momentum conservation and written in divergence form, is

$$r^2\left[\frac{\partial}{\partial z}\left(\frac{\omega}{r}\frac{\partial\psi}{\partial r}\right)-\frac{\partial}{\partial r}\left(\frac{\omega}{r}\frac{\partial\psi}{\partial z}\right)\right]-\frac{\partial}{\partial r}\left(r^3\frac{\partial}{\partial r}\left(\frac{\mu}{r}\omega\right)\right)-\frac{\partial}{\partial z}\left(r^3\frac{\partial}{\partial z}\left(\frac{\mu}{r}\omega\right)\right)+ \quad (66)$$

$$r^2g\frac{\partial\rho}{\partial r}+r^2\left[\frac{\partial}{\partial r}\left(\frac{v_r^2+v_z^2}{2}\right)\frac{\partial\rho}{\partial z}-\frac{\partial}{\partial z}\left(\frac{v_r^2+v_z^2}{2}\right)\frac{\partial\rho}{\partial r}\right]=0. \quad (67)$$

The entire second line of this equation vanishes in constant density flows, but variable-density source terms actually dominate the upstream influx of momentum in this example. Density varies by nearly a factor of ten in room-temperature methane-air flames.

The equation for the flamesheet conserved scalar, representing the conservation of internal energy and chemical species and written in divergence form, is

$$\frac{\partial}{\partial z}\left(S\frac{\partial\psi}{\partial r}\right)-\frac{\partial}{\partial r}\left(S\frac{\partial\psi}{\partial z}\right)-\frac{\partial}{\partial r}\left(r\rho D\frac{\partial S}{\partial r}\right)-\frac{\partial}{\partial z}\left(r\rho D\frac{\partial S}{\partial z}\right)=0. \quad (68)$$

The three governing equations all possess the form "Laplacian in a dominant variable plus first-order and/or source terms." Algebraic state relations giving the density ρ and the variable transport properties μ and D in terms of S supplement this system. (See Ref. 42, and references therein.)

Solution of the system of governing equations employs pseudo-transient continuation, a form of implicit time differencing with an adaptively increased Δt that transforms the mathematical character of the system from parabolic to elliptic over the course of the nonlinear evolution. We have recently argued that polyalgorithmic linear solvers are appropriate in such problems, to exploit the changing character of the linear systems to be solved for the Newton corrections (Ref. 26). In this example, we employ a simple stripwise tile decomposition with no coarse grid, and with the tangential interface preconditioner T_B replaced with an interface probe technique (Ref. 16) known as projected-IP(1). Four sets of Jacobian matrices and residual vectors from a serial computer run of the flamesheet code are dumped to disk for analysis by a separate parallel domain decomposition code that was run on a 16-processor shared memory Encore

Multimax. A stripwise decomposition was produced with cuts normal to the dominant flow direction, across the narrow dimension of the high-aspect ratio domain. Studies of model problems in Ref. 37 show this to be more effective for relatively small numbers of subdomains than a decomposition with interior vertices. Heuristically, the discrete Green's functions are anisotropic, due to the high aspect ratio of the grid. Their tails decay more rapidly in the axial direction, z, than in the radial direction, r. As implicit aggregates of the parallel preconditioner, we therefore group together points of nearby z coordinate.

Rather than scaled examples, we present fixed-size problems with increasing numbers of subdomain strips, one per processor. Results for four different Jacobians are listed, three from a highly nonuniform 31×31 grid and one from a highly nonuniform 63×63 grid. The latter system possesses 11,907 degrees of freedom. The first three Jacobians come from different stages of the pseudo-transient continuation process, with very small, moderate, and very large Δt, respectively. The third Jacobian ("J31-3") is by far the worst conditioned of the set. We present iteration counts, execution times, and unscaled parallel efficiencies.

Apart from superlinear speedup in the first problem (due to an improvement of conditioning with increased decomposition that is generally unexpected with invertebrate decompositions such as this one), terminal efficiencies are in the 60% range, implying a ten-fold reduction in execution time on 16 processors. This sort of performance is entirely acceptable on large shared memory machines of the Cray class, where a combustion problem with realistic chemical kinetics must be shoehorned into memory, and any processors not engaged in the combustion computation are idle anyway, for want of memory.

CONCLUSIONS AND FUTURE PROSPECTS

The examples of the previous section demonstrate that domain decomposition algorithms are already valuable on today's architectures. The introductory sections argue that domain decomposition algorithms will be essential to scale PDE computations to the next generation of massively parallel supercomputers, such as the Intel Paragon and CM-5, in which fast processing elements and large networks will place a premium on accessing off-processor data. High-order discretizations harmonize with the theme of hierarchical solvers on such supercomputers, as algorithms must be recast to do as much useful work as possible *per point*, as well as *per iteration*, within memory and communication bandwidth constraints.

From the first five international conferences on domain decomposition (Refs. 13, 14, 30, 31, and 40), it is clear that the theory has outstripped both practice and parallel implementation of even model problems. The linear selfadjoint theory gives the appearance of being nearly complete, as efforts now concentrate on such features as mesh angle dependencies, overlap dependencies, and discontinuous coefficients. The nonselfadjoint, indefinite theory is hung off of the selfadjoint theory by first assuring that the symmetric part of the preconditioned operator is positive definite and dominant. This would be unsatisfactory from the point of view of computational fluid dynamics, except for the fact that many practical CFD codes are driven by a defect correction outer loop that employs left-hand side operators of precisely these characteristics. In some CFD applications, only an elliptic pressure equation or a viscous fractional step will be directly amenable to hierarchical domain decomposition analysis, but in such applications, these steps are typically the only ones that defy conventional parallelism, and the remaining steps can easily employ data-to-processor mappings compatible with those required by domain decomposition. Multicomponent theory is only nascent. Given a system of equations, each of which contains the Laplacian of one of the unknowns in which it is dominant, the multicomponent problem decouples in the asymptotic limit of a fine mesh. The formal tools of Yavneh (Ref. 57) help uncover such structure when it may be hidden in the system of PDEs as posed, but theoretical stimulation for algorithmic development of the strongly coupled case is needed.

In the applications community, domain decomposition is still being driven by gridding and modeling issues, not yet by convergence rates or parallelization. As problem sizes continue to expand, the advantages hierarchical domain decomposition offers in these latter two respects will assume a prominence equal to the first two.

Progress from linear, selfadjoint, uniformly refined scalar problems on a geometrically simple two-dimensional region to nonlinear, nonselfadjoint, generally refined multicomponent problems in geometrically complex three-dimensional regions are occurring one generalization at a time. We may hope that theory and numerical experiment will continue to be mutual stimuli along this path, the former providing actual and heuristic guidance on how to compute; the latter providing evidence of what may be provable.

If anything has become clear so far, it is that it is not sufficient to be a computer scientist to do parallel computation of PDEs. PDE problems possess fundamental hierarchies of data dependencies not reflected in absolute sparsity maps that must be understood mathematically before an appropriate algorithm can be selected or even an appropriately proportioned parallel computer designed. The parallel computational destiny of PDEs is not in the languages, environments, or architectures of the day, but in the physics of the problems they are modeling. The latter is unchanging, and all of the others are in the process of converging to it, not necessarily monotonically.

ACKNOWLEDGEMENTS

It is a pleasure to acknowledge many years of technical collaboration with Bill Gropp of the Mathematics and Computer Science Division at Argonne National Laboratory and with Xiao-Chuan Cai of the Mathematics Department at the University of Kentucky, with whom most of the numerical results reported above were obtained. The support of George Lea of the National Science Foundation helped bring this collaboration about. The constructive feedback of various members of the Computational Science Unit at the United Technologies Research Center and of Barry Smith, Xiao-Chuan Cai and Charbel Farhat on early versions of this paper have improved it considerably. The encouragement of Yousuff Hussaini, the Director of the Institute for Computer Applications in Science and Engineering at the NASA-Langley Research Center, to put a set of tutorial lectures on domain decomposition into this form is also gratefully acknowledged.

REFERENCES

1. Bjorstad, P. E., and Mandel, J., "On the Spectra of Sums of Orthogonal Projections with Applications to Parallel Computing," *BIT*, Vol. 31, 1991, pp. 76-88.
2. Bjorstad, P. E., and Skogen, M. D., "Domain Decomposition Algorithms of Schwarz Type, Designed for Massively Parallel Computers," in *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., and Voigt, R. G., (eds.), SIAM, Philadelphia, 1992.

3. Bjorstad, P. E., and Widlund, O. B., "Iterative Methods for the Solution of Elliptic Problems on Regions Partitioned into Substructures," *SIAM Journal of Numerical Analysis*, Vol. 23, 1986, pp. 1097–1120.

4. Bjorstad, P. E., and Widlund, O. B., "To Overlap or not to Overlap: A Note on a Domain Decomposition Method for Elliptic Problems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 10, 1989, pp. 1053–1061.

5. Börgers, C., "The Neumann-Dirichlet Domain Decomposition Method with Inexact Solvers on the Subdomains," *Numerische Mathematik*, Vol. 55, 1989, pp. 123–136.

6. Bramble, J. H., Pasciak, J. E., and Schatz, A. H., "The Construction of Preconditioners for Elliptic Problems by Substructuring, I," *Mathematics of Computation*, Vol. 47, 1986, pp. 103–134.

7. Bramble, J. H., Pasciak, J. E., and Schatz, A. H., "The Construction of Preconditioners for Elliptic Problems by Substructuring, IV," *Mathematics of Computation*, Vol. 53, 1989, pp. 1–24.

8. Cai, X.-C., *Some Domain Decomposition Algorithms for Nonselfadjoint Elliptic and Parabolic Partial Differential Equations*, Tech. Rep. 461, Courant Institute, NYU, September 1989.

9. Cai, X.-C., "An Optimal Two-level Overlapping Domain Decomposition Method for Elliptic Problems in Two and Three Dimensions," *SIAM Journal of Scientific and Statistical Computing* (to appear), 1992.

10. Cai, X.-C., Gropp, W. D., and Keyes, D. E., "A Comparison of Some Domain Decomposition and ILU Preconditioned Iterative Methods for Nonsymmetric Elliptic Problems," Submitted to *Journal of Numerical Linear Algebra and Applications*, 1992.

11. Cai, X.-C., Gropp, W. D., and Keyes, D. E., "Convergence Estimate for a Domain Decomposition Method," *Numerische Mathematik*, Vol. 61, 1992, pp. 153–169.

12. Cai, X.-C., and Widlund, O. B., "Domain Decomposition Algorithms for Indefinite Elliptic Problems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 13, 1992, pp. 243–258.

13. Chan, T. F., Glowinski, R., Périaux, J., and Widlund, O. B., (eds.), *Domain Decomposition Methods*, SIAM, Philadelphia, 1989. Proceedings of the Second International Symposium on Domain Decomposition Methods, Los Angeles, California, January 1988.

14. Chan, T. F., Glowinski, R., Périaux, J., and Widlund, O. B., (eds.), *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1990. Proceedings of a symposium held in Houston, Texas, March 1989.

15. Chan, T. F., and Goovaerts, D., "On the Relationship between Overlapping and Nonoverlapping Domain Decomposition Methods," *SIAM Journal of Matrix Analysis and Applications*, Vol. 13, 1992, pp. 663–670.

16. Chan, T. F., and Keyes, D. E., "Interface Preconditionings for Domain-Decomposed Convection-Diffusion Operators," in *Third International Symposium on Domain Decomposition Methods*, Chan, T. F., Glowinski, R., Périaux, J., and Widlund, O. B., (eds.), SIAM, Philadelphia, 1990, pp. 245–262.

17. Chan, T. F., Kuo, C.-C. J., and Tong, C., "Parallel Elliptic Preconditioners: Fourier Analysis and Performance on the Connection Machine," *Computer Physics Communications*, Vol. 53, 1989, pp. 237–252.

18. Chan, T. F., and Mathew, T. F., *The Boundary Probing Technique in Domain Decomposition*, Tech. Rep. 91-02, UCLA Comp. and App. Math., 1991.

19. Cottle, R. W., "Manifestations of the Schur Complement," *Linear Algebra and Applications*, Vol. 8, 1974, pp. 189–211.

20. Courant, R., and Hilbert, D., *Methods of Mathematical Physics*. Wiley, New York, 1953.

21. Dinh, Q. V., Glowinski, R., and Périaux, J., "Solving Elliptic Problems by Domain Decomposition Methods with Applications," in *Elliptic Problem Solvers II*, Birkhoff, G., and Schoenstadt, A., (eds.), 1984, pp. 395–426.

22. Dryja, M., and Widlund, O. B., *An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions*, Tech. Rep. 339, Courant Institute, NYU, December 1987.

23. Dryja, M., and Widlund, O. B., "Additive Schwarz Methods for Elliptic Finite Element Problems in Three Dimensions," in *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., and Voigt, R. G., (eds.), SIAM, Philadelphia, 1991.

24. Dupont, T., Kendall, R., and Rachford, H. H., "An Approximate Factorization Procedure for Solving Self-Adjoint Elliptic Difference Equations," *SIAM Journal of Numerical Analysis*, Vol. 5, 1968, pp. 559–573.

25. Eijkhout, V., and Vassilevski, P., "The Role of the Strengthened Cauchy-Buniakowskii-Schwarz Inequality in Multilevel Methods," *SIAM Review*, Vol. 33, 1991, pp. 405–420.

26. Ern, A., Giovangigli, V., Keyes, D. E., and Smooke, M. D., "Towards Polyalgorithmic Linear System Solvers for Nonlinear Elliptic Problems," in *Copper Mountain Conference on Iterative Methods*, University of Colorado, Denver, 1992. Submitted to *SIAM Journal of Scientific and Statistical Computing*.

27. Farhat, C., "A Saddle-Point Principle Domain Decomposition Method for the Solution of Solid Mechanics Problems," in *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., and Voigt, R. G., (eds.), SIAM, Philadelphia, 1992.

28. Fischer, P. F., and Patera, A. T., "Parallel Spectral Element Solution of the Stokes Problem," *Journal of Computational Physics*, Vol. 92, 1991, pp. 380–421.

29. Freund, R. W., and Nachtigal, N. M., "QMR: a Quasi-minimal Residual Method for Non-Hermitian Linear Systems," *Numerische Mathematik*, Vol. 60, 1991, pp. 315–339.

30. Glowinski, R., Golub, G. H., Meurant, G. A., and Périaux, J., (eds.), *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1988. Proceedings of a symposium held in Paris, France, January 1987.

31. Glowinski, R., Kuznetsov, Y. A., Meurant, G. A., Périaux, J., and Widlund, O. B., (eds.), *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1991. Proceedings of a symposium held in Moscow, USSR, May 1990.

32. Golub, G. H., and Loan, C. F. V., *Matrix Computations.* Johns Hopkins, Baltimore, 1989.

33. Gottlieb, D., and Orszag, S. A., *Numerical Analysis of Spectral Methods: Theory and Applications.* SIAM, Philadelphia, 1977.

34. Gropp, W. D., "Parallel Computing and Domain Decomposition," in *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., and Voigt, R. G., (eds.), SIAM, Philadelphia, 1992.

35. Gropp, W. D., and Keyes, D. E., "Domain Decomposition on Parallel Computers," *Impact of Computing in Science and Engineering*, Vol. 1, 1989, pp. 421–439.

36. Gropp, W. D., and Keyes, D. E., "Domain Decomposition Methods in Computational Fluid Dynamics," *International Journal for Numerical Methods in Fluids*, Vol. 14, 1992, pp. 147–165.

37. Gropp, W. D., and Keyes, D. E., "Domain Decomposition with Local Mesh Refinement," *SIAM Journal of Scientific and Statistical Computing*, Vol. 13, 1992, pp. 967–993.

38. Gropp, W. D., and Keyes, D. E., "Parallel Performance of Domain-decomposed Preconditioned Krylov Methods for PDEs with Locally Uniform Refinement," *SIAM Journal of Scientific and Statistical Computing*, Vol. 13, 1992, pp. 128–145.

39. Johnsson, S. L., "Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures," *Journal of Parallel and Distributed Computing*, Vol. 4, 1985, pp. 133–172.

40. Keyes, D. E., Chan, T. F., Meurant, G. A., Scroggs, J. S., and Voigt, R. G., (eds.), *Fifth Conference on Domain Decomposition Methods for Partial Differential Equations*, SIAM, Philadelphia, 1992. Proceedings of a symposium held in Norfolk, Virginia, May 1991.

41. Keyes, D. E., and Gropp, W. D., "A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and Their Parallel Implementation," *SIAM Journal of Scientific and Statistical Computing*, Vol. 8, 1987, pp. s166–s202.

42. Keyes, D. E., and Gropp, W. D., "Domain Decomposition Techniques for the Parallel Solution of Nonsymmetric Systems of Elliptic Boundary Value Problems," *Applied Numerical Mathematics*, Vol. 6, 1990, pp. 281–301.

43. Kron, G., "A set of Principles to Interconnect the Solutions of Physical Systems," *Journal of Applied Physics*, Vol. 24, 1953, pp. 965–980.

44. Mandel, J., "Iterative Solvers by Substructuring for the p-version Finite Element Method," *Computer Methods in Applied Mechanics and Engineering*, Vol. 80, 1990, pp. 117–128.

45. Mandel, J., "Two-level Domain Decomposition Preconditioning for the p-version Finite Element Method in Three Dimensions," *International Journal for Numerical Methods in Engineering*, Vol. 29, 1990, pp. 1095–1108.

46. Meijerink, J. A., and der Vorst, H. A. V., "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they Occur in Practical Problems," *Journal of Computational Physics*, Vol. 44, 1981, pp. 134–155.

47. Omtzigt, E. T. L., *Domain Flow and Streaming Architectures: A Paradigm for Efficient Parallel Computation*. PhD thesis, Yale University, 1992.

48. Patera, A. T., "A Spectral Element Method for Fluid Dynamics: Laminar Flow in a Channel Expansion," *Journal of Computational Physics*, Vol. 54, 1984, pp. 468–488.

49. Przemieniecki, J. S., "Matrix Structural Analysis of Substructures," *AIAA Journal*, Vol. 1, 1963, pp. 138–147.

50. Saad, Y., and Schultz, M. H., "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 7, 1986, pp. 856–869.

51. Schwarz, H., *Gesammelte Mathematische Abhandlungen*, Vol. 2. Springer, Berlin, 1890.

52. Smith, B. F., *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Tech. Rep. 517, Courant Institute, NYU, September 1990.

53. Smith, B. F., *A Parallel Implementation of an Iterative Substructuring Algorithm for Problems in Three Dimensions*, Tech. Rep. MCS-P249-0791, Mathematics and Computer Science Division, Argonne National Laboratory, 1991. *SIAM Journal of Scientific and Statistical Computing* (to appear).

54. Van der Sluis, A., and Van der Vorst, H. A., "The Rate of Convergence of Conjugate Gradients," *Numerische Mathematik*, Vol. 48, 1986, pp. 543–560.

55. Van der Vorst, H. A., "Bi-CGSTAB: a More Smoothly Converging Variant of CG-S for the Solution of Nonsymmetric Linear Systems," *SIAM Journal of Scientific and Statistical Computing*, Vol. 13, 1992, pp. 631–644.

56. Widlund, O. B., "Iterative Substructuring Methods: Algorithms and Theory for Elliptic Problems in the Plane," in *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, Glowinski, R., Golub, G. H., Meurant, G. A., and Périaux, J., (eds.), SIAM, Philadelphia, 1988, pp. 113–128.

57. Yavneh, I., "A Method for Devising Efficient Multigrid Smoothers for Complicated PDE Systems," in *Copper Mountain Conference on Iterative Methods*, University of Colorado, Denver, 1992.

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1992	3. REPORT TYPE AND DATES COVERED Contractor Report

4. TITLE AND SUBTITLE DOMAIN DECOMPOSITION: A BRIDGE BETWEEN NATURE AND PARALLEL COMPUTERS	5. FUNDING NUMBERS C NAS1-18605 C NAS1-19480 WU 505-90-52-01
6. AUTHOR(S) David E. Keyes	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23681-0001	8. PERFORMING ORGANIZATION REPORT NUMBER ICASE Report No. 92-44
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-189709 ICASE Report No. 92-44

11. SUPPLEMENTARY NOTES

Langley Technical Monitor: Michael F. Card
Final Report

To appear in Proc. of Symp. on Adaptive, Multilevel & Hierarchical Comp. Strategies, Nov. '92

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 64	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Domain decomposition is an intuitive organizing principle for a PDE computation, both physically and architecturally. However, its significance extends beyond the readily apparent issues of geometry and discretization, on one hand, and of modular software and distributed hardware, on the other. Engineering and computer science aspects are bridged by an old but recently enriched mathematical theory that offers the subject not only unity, but also tools for analysis and generalization. Domain decomposition induces function-space and operator decompositions with valuable properties. Function-space bases and operator splittings that are not derived from domain decompositions generally lack one or more of these properties. The evolution of domain decomposition methods for elliptically dominated problems has linked two major algorithmic developments of the last 15 years: multilevel and Krylov methods. Domain decomposition methods may be considered descendants of both classes with an inheritance from each: they are nearly optimal and at the same time efficiently parallelizable. Many computationally driven application areas are ripe for these developments. This paper progresses from a mathematically informal motivation for domain decomposition methods to a specific focus on fluid dynamics applications. Introductory rather than comprehensive, it employs simple examples, and leaves convergence proofs and algorithmic details to the original references; however, an attempt is made to convey their most salient features, especially where this leads to algorithmic insight.

14. SUBJECT TERMS domain decomposition; preconditioning; Krylov methods; computational fluid dynamics	15. NUMBER OF PAGES 43
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT