NASA Contractor Report 181719

ICASE REPORT NO. 88-52

ICASE

DOMAIN DECOMPOSITION METHODS FOR THE
PARALLEL COMPUTATION OF REACTING FLOWS

David E. Keyes

Contract Nos. NAS1-18107 and AFOSR 88-0117
September 1988

INSTITUTE FOR COMPUTER APPLICATIONS IN SCIENCE AND ENGINEERING
NASA Langley Research Center; Hampton, Virginia 23665

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665

Domain Decomposition Methods for the Parallel Computation of Reacting Flows

David E. Keyes*
Department of Mechanical Engineering
Yale University
New Haven, CT 06520

Abstract

Domain decomposition is a natural route to parallel computing for partial differential equation solvers. In this procedure, subdomains of which the original domain of definition is comprised are assigned to independent processors at the price of periodic coordination between processors to compute global parameters and maintain the requisite degree of continuity of the solution at the subdomain interfaces. In the domain-decomposed solution of steady multidimensional systems of PDEs by finite difference methods using a pseudo-transient version of Newton iteration, the only portion of the computation which generally stands in the way of efficient parallelization is the solution of the large, sparse linear systems arising at each Newton step. For some Jacobian matrices drawn from an actual two-dimensional reacting flow problem, we make comparisons between relaxation-based linear solvers and also preconditioned iterative methods of Conjugate Gradient and Chebyshev type, focusing attention on both iteration count and global inner product count. The generalized minimum residual method with block-ILU preconditioning is judged the best serial method among those considered, and parallel numerical experiments on the Encore Multimax demonstrate for it approximately 10-fold speedup on 16 processors. The three special features of reacting flow models in relation to these linear systems are: the possibly large number of degrees of freedom per gridpoint, the dominance of dense *intra*-point source-term coupling over *inter*-point convective-diffusive coupling throughout significant portions of the flow-field, and strong nonlinearities which restrict the time-step to small values (independent of linear algebraic considerations) throughout significant portions of the iteration history. Though these features are exploited to advantage herein, many aspects of the paper are applicable to the modeling of general convective-diffusive systems.

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18107 and by the Air Force Office of Scientific Research under Contract No. AFOSR 88-0117 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23665.

1. Introduction

Computational scientists and engineers porting large-scale codes to most contemporary parallel computers are faced directly with the related issues of partitioning programs across processors and partitioning data across memories. For the large subset of scientific codes which are based on the discretization of partial differential equations in multidimensional domains, domain decomposition is a natural basis for the partitioning, at least in the coarse-to-medium granularity range. It is "natural" in the sense that both the data structures and the algorithms employed in a serial code can be carried over to the parallel case, changed only by the addition of new code to treat the artificially introduced boundaries and to mediate computation of global parameters. We shall not attempt a quantification of the granularity of the computation in terms of an absolute number of processors, since granularity is a relative concept in domain decomposition applications. Typically, we would like to require that the subdomains remain "chunky", in other words, that the number of boundary degrees of freedom remain small relative to the number in the subdomain interiors. In this respect, the subdomains will resemble the original domain, for which a good solver is presumed known. We note in passing that domain decomposition has other virtues apart from creating parallel threads of execution, such as operator adaptivity, graceful accommodation of complex geometry, and clean memory management of problems too large to fit in active core. Some instances of these have been described theoretically and illustratively in the collections [6, 12, 23, 24].

According to our parallelization paradigm, the domain of definition of the PDE is decomposed into subdomains and a one-to-one mapping of subdomain to processor is created.* For an explicit stage of a calculation, of which matrix multiplication is representative, the task of finding independent threads of execution ends with the decomposition. Parallelization consists of simply subdividing the ranges of DO-loops. For an implicit stage, of which matrix inversion is representative, a proven serial algorithm may be employed on each subdomain, but additional provisions must be made to enforce the original equations at the subdomain interfaces. In their simplest form, the interface conditions can take the form of interpolative updating of data in regions of subdomain overlap. Other forms include simultaneous updates of interior and interface data in a global preconditioned iterative process and incorporation of the interface constraints in a global variational principle.

There are potentially three penalties to be paid in distributing a solution algorithm over an array of independent processors: synchronization overhead, communication overhead, and degradation of convergence rate. The synchronization penalty arises if the processors have data dependencies which cause them to idle. This can happen at convergence checkpoints, for instance, or at any other point where an exchange of data is required if the processors have unequal amounts of work to do. The communication penalty is the time spent physically exchanging messages between processors which depend on common data (or granting controlled access to shared memory). This can be non-negligible even in perfectly load-balanced parallel configurations. In an effort to keep the synchronization and communication overhead down, natural tradeoffs may exist which degrade the convergence rate of the algorithm by substituting readily available data for the best possible data. These penalties are measured indirectly through the speedup and efficiency figures-of-merit of a parallel implementation. A common measure of speedup is the ratio of the uniprocessor execution time of a given algorithm to that of the multiprocessor execution of the same algorithm. The corresponding measure of efficiency is the speedup divided by the number of processors. Within the context of a specific algorithm, efficiency and speedup are functions of problem parameters,

*Mapping one subdomain to a sub-cluster of processors is also possible, but involves a different class of issues, since space-based decompositions lend themselves to lock-step homogeneous programming of the individual processors, while non-space-based decompositions might not. In a multigrid context, generalization in the other direction might be natural, namely mapping several subdomains from different multigrid levels onto the same processor.

such as problem size and decomposition granularity, as well as machine parameters, such as the ratio of local to remote memory access times or the ratio of floating point arithmetic time to the latency period of interprocessor messages. While theoretical computational complexity analyses can elucidate some general trends, the demonstration of parallel efficiency inevitably requires some experimental information in all but ideal problems.

The cost-effectiveness of obtaining parallelization through domain decomposition as opposed to other methods can depend on the relative weighting of three (to some extent mutually exclusive) summands in the cost objective: efficiency, wall-clock execution time, and software development time. Though efficiency is always important in the many-processor regime, our main interest is in algorithms that perform competitively over a range of granularity which specifically includes a single processor. In confining attention to algorithms which are "methods of choice" in serial, many which parallelize efficiently are ruled out. One near-term motivation for techniques which perform acceptably over a range of granularities is that jobs which use nearly all of the memory of a multiprocessor supercomputer, such as the Cray-XMP or the ETA-10, effectively tie up most of the processors sharing that memory, whether their cycles are actually employed or not. The ability to keep the processing force in rough proportion to memory usage over a range of problem sizes can be an economic advantage. Independent of economic considerations, there is obviously a premium on the ability to reduce wall-clock time by adding processors in "production mode" in many applications. Until the elusive day when native software facilities make parallelization automatic, domain decomposition will have a significant role to play in meeting these objectives.

The outline of this case study of the applicability of domain decomposition to the computation of reacting flows (which must be regarded as merely preliminary in several respects to be noted) is as follows. Section 2 describes a reacting flow model with features that stretch the supercomputer state-of-the-art, and also the relation to it of an approximate reduced model known as the "flame sheet", to which numerical consideration is confined in the sequel. A numerical discretization and a solution algorithm are presented in section 3. Section 4 provides a brief complexity analysis for the domain-decomposed parallel implementation and shows the pivotal role in the parallel efficiency of the sparse linear solves required by Newton's method. Section 5 discusses alternative linear algebraic methods and presents some serial results for sample Jacobians drawn from flame sheet simulations. Parallelization issues for the linear algebra itself are considered in section 6, and some numerical results obtained for one of the methods over a range of 1 to 16 processors on the Encore Multimax, a shared memory computer. In section 7 we draw some conclusions and state some open research questions.

2. Mathematical Modeling of Reacting Flows

The goal of this section is to convey the structure and dimension of the reacting flow system, and not to dwell on the underlying physics. Our attention is confined to two space dimensions, because a "third dimension", namely, the number of components defined at each point, already appears in the full model, and because many combustion applications are well approximated by axisymmetry. Under this restriction, it is natural to adopt a streamfunction-vorticity formulation of the fluid dynamics rather than a velocity-pressure formulation. The number of unknowns is thereby reduced by one, and more conveniently, each PDE in the streamfunction/vorticity system is formally of elliptic type.

For definiteness, we cite without derivation the following set of elliptic boundary value problems (see, *e.g.*, [13]). Though typical of many reacting flow models, they do not include effects which will in some contexts be essential (*e.g.*, radiation, turbulence), and which could alter the algebraic connectivity of the discrete governing system. The problem motivating the present work is an axisymmetric isobaric laminar flame, such as the gaseous co-flowing jet flame sketched in Fig. 1.

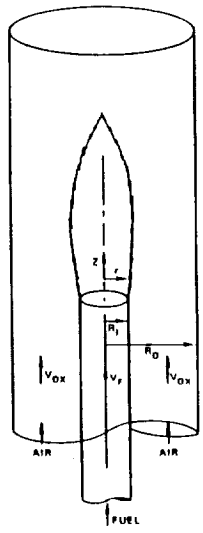

Figure 1: Schematic of a confined non-premixed laminar jet flame.

Let r and z denote the radial and axial directions, v_r and v_z the respective velocity components, and ρ the density. We introduce the variable density Stokes streamfunction ψ such that

$$\rho r v_r = -\frac{\partial \psi}{\partial z} \quad \text{and} \quad \rho r v_z = \frac{\partial \psi}{\partial r},$$

and the vorticity

$$\omega = \frac{\partial v_r}{\partial z} - \frac{\partial v_z}{\partial r}.$$

The species concentrations by mass are Y_k, for $k = 1, 2, \ldots, K$, where K is the number of distinct chemical species in the reaction mechanism. The temperature is T. These principal fields satisfy the following equations.

Streamfunction (overall mass conservation):

$$\frac{\partial}{\partial z}\left(\frac{1}{r\rho}\frac{\partial \psi}{\partial z}\right) + \frac{\partial}{\partial r}\left(\frac{1}{r\rho}\frac{\partial \psi}{\partial r}\right) + \omega = 0. \tag{2.1}$$

Vorticity (momentum conservation):

$$r^2\left[\frac{\partial}{\partial z}\left(\frac{\omega}{r}\frac{\partial \psi}{\partial r}\right) - \frac{\partial}{\partial r}\left(\frac{\omega}{r}\frac{\partial \psi}{\partial z}\right)\right] - \frac{\partial}{\partial r}\left(r^3\frac{\partial}{\partial r}\left(\frac{\mu}{r}\omega\right)\right) - \frac{\partial}{\partial z}\left(r^3\frac{\partial}{\partial z}\left(\frac{\mu}{r}\omega\right)\right)$$
$$+ r^2 g\frac{\partial \rho}{\partial r} + r^2\nabla\left(\frac{v_r^2 + v_z^2}{2}\right)\cdot \text{iso } \rho = 0. \tag{2.2}$$

Species (detailed mass conservation):

$$\frac{\partial}{\partial z}\left(Y_k\frac{\partial \psi}{\partial r}\right) - \frac{\partial}{\partial r}\left(Y_k\frac{\partial \psi}{\partial z}\right) + \frac{\partial}{\partial r}\left(r\rho Y_k V_{k_r}\right) + \frac{\partial}{\partial z}\left(r\rho Y_k V_{k_z}\right)$$
$$- rW_k\dot{w}_k = 0, \quad k = 1, 2, \ldots, K. \tag{2.3}$$

Temperature (internal energy conservation):

$$c_p\left[\frac{\partial}{\partial z}\left(T\frac{\partial \psi}{\partial r}\right)-\frac{\partial}{\partial r}\left(T\frac{\partial \psi}{\partial z}\right)\right]-\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right)-\frac{\partial}{\partial z}\left(r\lambda\frac{\partial T}{\partial z}\right)$$
$$+r\sum_{k=1}^{K}\left\{\rho c_{pk}Y_k\left(V_{k_r}\frac{\partial T}{\partial r}+V_{k_z}\frac{\partial T}{\partial z}\right)\right\}+r\sum_{k=1}^{K}h_kW_k\dot{w}_k=0. \tag{2.4}$$

Other parameters appearing in this system are: the viscosity of the mixture μ, the acceleration of gravity g, the diffusion velocities of the k^{th} species in the mixture (V_{k_r}, V_{k_z}), the specific heat of the k^{th} species c_{pk}, the specific heat of the mixture c_p, the thermal conductivity of the mixture λ, the molecular mass of the k^{th} species W_k, and the generation/consumption rate of the k^{th} species $\dot{w}_k$.

The system is closed with the multicomponent ideal gas law,

$$\rho=\frac{pW}{RT}, \tag{2.5}$$

where W is the mean molecular mass of the local mixture, p the pressure, and R the universal gas constant. The Arrhenius reaction rate law for a system of J reactions gives

$$\dot{w}_k=\sum_{j=1}^{J}(\nu''_{jk}-\nu'_{jk})\left[k_j\prod_{n=1}^{K}\left(\frac{\rho Y_n}{W_n}\right)^{\nu'_{jn}}-\frac{k_j}{K_j^c}\prod_{n=1}^{K}\left(\frac{\rho Y_n}{W_n}\right)^{\nu''_{jn}}\right], \tag{2.6}$$

where the forward rate coefficient of reaction j is given by

$$k_j=A_jT^{\beta_j}\exp(-E_j/RT), \tag{2.7}$$

and where, in turn, A_j, the pre-exponential rate constants, β_j, the temperature exponents, E_j, the activation energies, and K_j^c, the equilibrium constants, are assumed known. The ν_{jk} are the stoichiometric coefficients of the k^{th} species in the j^{th} reaction. Superscripts $'$ and $''$ on the ν_{jk} denote the reactant and product sides of the stoichiometric equation, respectively; thus $\dot{w}_k$ is positive where species k is produced. Thermodynamic and constitutive equations provide the temperature- and composition-dependent specific heats, viscosities, thermal conductivities, and diffusion velocities. The matrix of stoichiometric coefficients is typically very sparse, with each of the J elementary reactions involving only a few of the K species. For hydrogen or lower hydrocarbon flames in air, J and K are $O(10^1-10^2)$.

Though formally elliptic, the presence of strong convection renders all of the governing equations except that of the streamfunction approximately parabolic in z.

In most practical combustors, the reactants are supplied in a non-premixed state in streams from different reservoirs, and the rate of combustion is limited by the diffusive mixing of the reactants rather than chemical reaction rates. The flame sheet model exploits this large *Damköhler number* (ratio of diffusion to reaction time scales) by replacing a reaction zone of finite thickness with an interface (of unknown location) across which gradients of the temperature and species are discontinuous.* The interface subdivides the physical domain Ω into an oxidizer-free zone Ω_F and a fuel-free zone Ω_O, in either of which the full composition and thermodynamic state of the gas mixture can be recovered from a single conserved scalar. The flame sheet is an economical means of computing initial iterates for detailed kinetics calculations [16], since it requires just

*Our version of the flame sheet includes some additional simplifying assumptions; see [16] for details.

three components per gridpoint (the variable density Stokes streamfunction ψ, the vorticity ω, and a conserved scalar S) instead of the dozens that would be required with full chemistry. The equations which replace (2.3)-(2.7) are:
Conserved Scalar (Energy and Mass Conservation):

$$\frac{\partial}{\partial z}\left(S\frac{\partial \psi}{\partial r}\right) - \frac{\partial}{\partial r}\left(S\frac{\partial \psi}{\partial z}\right) - \frac{\partial}{\partial r}\left(r\rho D\frac{\partial S}{\partial r}\right) - \frac{\partial}{\partial z}\left(r\rho D\frac{\partial S}{\partial z}\right) = 0, \tag{2.8}$$

Algebraic State Relations:

$$\rho(\vec{x}) = \left\{ \begin{array}{ll} \rho_F(S(\vec{x})), & \vec{x} \in \Omega_F \\ \rho_O(S(\vec{x})), & \vec{x} \in \Omega_O \end{array} \right\}, \tag{2.9}$$

$$\mu(\vec{x}) = \left\{ \begin{array}{ll} \mu_F(S(\vec{x})), & \vec{x} \in \Omega_F \\ \mu_O(S(\vec{x})), & \vec{x} \in \Omega_O \end{array} \right\}, \tag{2.10}$$

$$D(\vec{x}) = \left\{ \begin{array}{ll} D_F(S(\vec{x})), & \vec{x} \in \Omega_F \\ D_O(S(\vec{x})), & \vec{x} \in \Omega_O \end{array} \right\}, \tag{2.11}$$

The fields of T and the Y_k may also be obtained from S and are convenient intermediates in the evaluation of the right-hand sides of (2.9)-(2.11). Though the flame sheet is a more compact problem in terms of the number of unknowns, the spatial density distribution, with sharp gradients leading up to a gradient discontinuity at the flame front, carries the difficult nonlinearities of the original system. The ratio of maximum to minimum density in the flow field is typically $O(10^1)$, and the ratios of the extreme viscosity and mass diffusion coefficients, μ in (2.2) and ρD in (2.8), are on the order of the square root of the extreme density ratio. Fig. 2 displays the converged flame sheet density profile for the problem studied in sections 5 and 6, and the grid on which it was obtained. Note the severe gradients near the fuel/air inlet boundary at $r = 0.2$ cm. For plotting clarity, the true axial-to-radial aspect ratio of 8:1 has been distorted.

As an illustration of the quality of the flame sheet results for the temperature distribution, we display Fig. 3 from [16] which compares a flame sheet solution on a one-dimensional 38-point grid to a detailed kinetics solution on an adaptive 65-point grid. However, the flame sheet results have little information to offer concerning the chemical structure of the reaction zone, because of the restrictive assumptions employed, especially that the product of the fuel and oxidizer concentrations is everywhere zero.

3. The Discrete Governing System and Its Solution

The governing equations, along with appropriate boundary conditions, are differenced on a two-dimensional tensor product grid (but see also [30] for *local* adaptive gridding results) which is generated adaptively from an initial coarse grid by subequidistribution of gradients and curvatures of the solution components. This concentrates grid points in the regions of high-activity (fronts and peaks) in the domain. Second-order differences are used throughout except for gradient boundary conditions, and for the convective terms of the conservation equations in which first-order upwind differences are employed. This discretization can be accommodated within the standard nine-point stencil (the corner points being necessitated by the mixed derivatives in the convective terms) and, apart from the source term, insures the diagonal dominance of the Jacobian. The evaluation of the governing equation residuals at a given solution iterate constitutes a significant expense, so

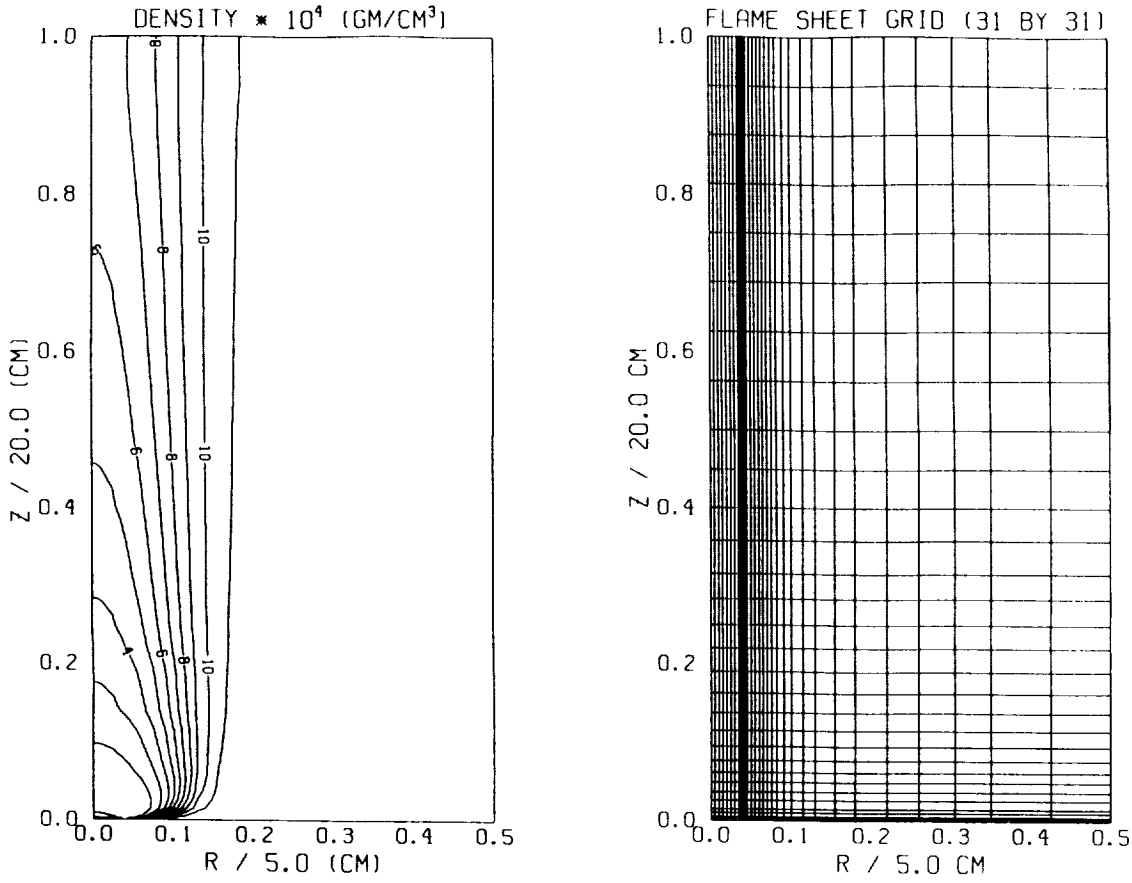

Figure 2: **Density contours in a co-flowing non-premixed methane-air flame, computed in the flame sheet limit.** Except for the spatial variability of mixture molecular weight W, the temperature field is inverse to this through (2.5), and thus has similar structure. The hottest region (and the density trough) lies near the fuel inlet. The grid is shown on the right.

methods which make efficient use of the function evaluations are required. For such problems, robust variations of Newton's method are often preferable to less fully coupled iterative methods or associated explicit time-marching methods (see, *e.g.*, [28]).

We write the overall system in the form

$$F(\phi) = 0, \tag{3.1}$$

where ϕ represents a column vector of all of the unknowns. Equation (3.1) may be solved efficiently by a damped modified Newton method provided that an initial iterate $\phi^{(0)}$ sufficiently close to the solution $\phi^\star$ is supplied. The damped modified Newton iteration is given by

$$\phi^{(k+1)} = \phi^{(k)} + \lambda^{(k)}\delta\phi^{(k)}, \tag{3.2}$$

where

$$\delta\phi^{(k)} = -(\tilde{J}^{(k)})^{-1}F(\phi^{(k)}), \tag{3.3}$$

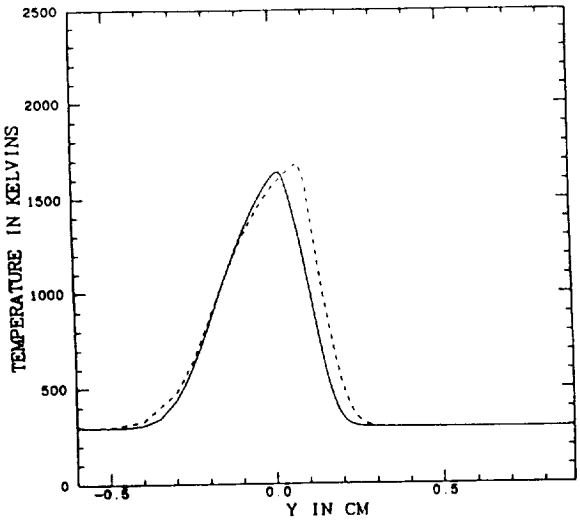

Figure 3: Comparison between one-dimensional temperature profiles computed from flame sheet and a detailed kinetics formulations in a non-premixed methane-air flame.

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 some (possibly 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 $F(\phi^{(k)})$.

A sufficiently good initial condition for Newton's method is usually obtainable by an elementary continuation procedure, namely driving a pseudo-transient form of (3.1),

$$D\frac{\partial \phi}{\partial t} + F(\phi) = 0, \tag{3.4}$$

where D is a scaling matrix, part of the way towards its steady state. If (3.4) is implicitly time-differenced with the backward Euler method using time step Δt from a given initial state $\bar{\phi}$, a new system,

$$\bar{F}(\phi) \equiv D\frac{(\phi - \bar{\phi})}{\Delta t} + F(\phi) = 0, \tag{3.5}$$

results, which possesses the Jacobian

$$\bar{J} \equiv \frac{D}{\Delta t} + \frac{\partial F}{\partial \phi}.$$

With an appropriate D and for a succession of a sufficiently small Δt, which will, however, in pursuit of efficiency, be larger than what would be required for accurate resolution of the physical transient, the iteration of (3.5) is a numerically robust procedure for generating an iterate from which Newton's method will converge. Moreover, due to the close relationship of (3.5) to (3.1), only minor modifications of the steady-state code for (3.1) are required to make it a transient/steady-state hybrid. Adaptive control of the size of the time steps used in (3.5) can enhance the convergence rate of the hybrid algorithm. One useful strategy is to choose Δt based on a loose temporal truncation error bound on the most rapidly varying component of the solution. As ϕ approaches ϕ^* and Δt becomes large, this effects a smooth transition to the (infinite time step) steady-state formulation.

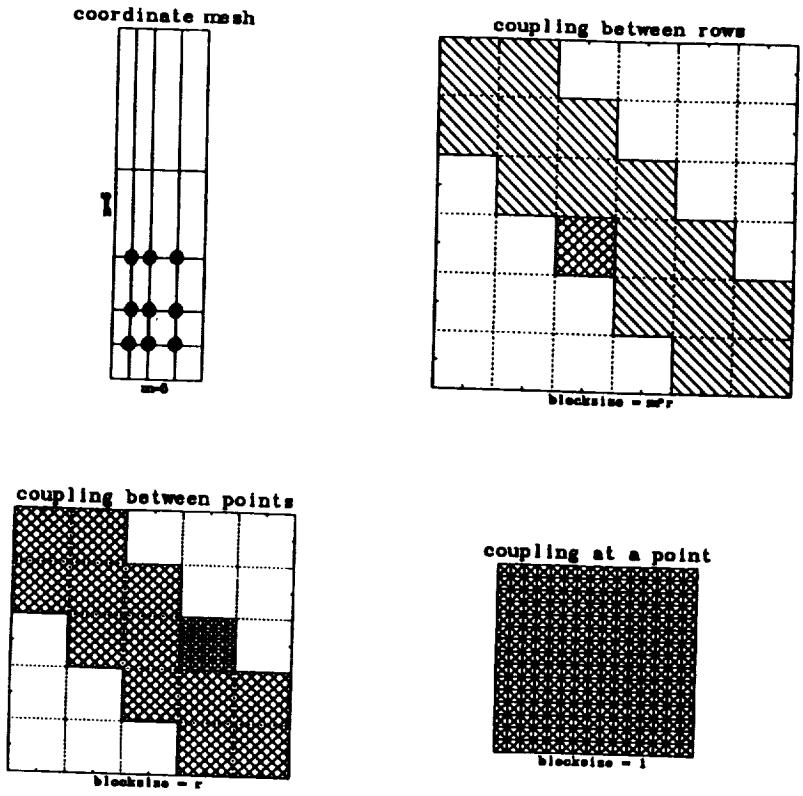

Figure 4: **A schematic of the Jacobian matrix of the 9-point finite-difference operator showing its sparsity structure.** (a) upper left: a two-dimensional 5×6 grid with the stencil corresponding to $(i, j) = (3, 3)$ highlighted. (b) upper right: gross sparsity structure of the Jacobian, n^2 blocks of size $mr \times mr$, representing the coupling between rows. (c) lower left: an enlargement of the cross-hatched block, m^2 blocks of size $r \times r$, representing the coupling between points within a row. (d) lower right: an enlargement of the double-cross-hatched block, r^2 blocks of size 1×1, representing the coupling between degrees of freedom at a point.

Ordering the solution components at each gridpoint within a lexicographical ordering of the gridpoints themselves results in a Jacobian which has a standard block nine-diagonal structure. To quantify the overall sparsity, let there be m and n gridpoints in the radial and axial coordinate directions, respectively, and r unknowns per gridpoint ($r = 3$ for the flame sheet). The $r \times r$ blocks must be assumed fully dense to accommodate the most general kinetic mechanism and composition-dependence of the transport properties. The fraction of Jacobian elements which are nonzero is then approximately $9/(nm)$. For m and n on the order of 30, which is certainly smaller than what is required for well-resolved reaction zones, this number is already approximately only 1%. For ease of visualization, an $m = 5$, $n = 6$ example is given in Fig. 4.

The complexity of the governing equations in either formulation above precludes analytic expression of the elements of the Jacobian; therefore a finite-difference approximation to the Jacobian must be used. Extending the sparse Jacobian evaluation ideas of Curtis, Powell and Reid [9], as described in more detail elsewhere [29], we can form all of the nonzero elements from $(9r + 1)$ independent vector residual evaluations. This is still a large number of function evaluations, with the result that coefficient generation becomes at least a co-dominant term in running time of the code.

The method described in this section has been successfully applied to a laminar methane-air jet in serial by means of a three-pass solution method, and results have appeared in [30]. In the first pass, which was carried out on a relatively coarse grid, the flame sheet model was used to generate hydrodynamic and temperature profiles. In the second pass, the temperature profile was maintained at the flame sheet solution, and the full system was solved with the energy equation (2.4) omitted. Finally, the energy equation was readmitted, the temperature profile allowed to adjust, and the full system solved on several successively finer grids, selected adaptively to resolve the high activity regions of the flame. This procedure, which is similar to that found invaluable in the one-dimensional context [16], promotes convergence of the full nonlinear iterations by gradually phasing in the details of the physical model, so that a good starting estimate exists for each pass, and promotes efficiency by keeping the size of the discrete system relatively small until the endgame. Even so, the equivalent of approximately 10 Cray-XMP hours is required for the complete effort.

Each of the three phases employs Newton's method as described above. From (3.2)-(3.3) and the paragraph following, we identify for further analysis the five basic tasks which together account for almost all of the execution time required by the code: (1) DAXPY vector arithmetic, (2) the evaluation of norms, (3) the evaluation of residual vectors, (4) the evaluation of Jacobians, and (5) the solution of linear equations involving the Jacobian matrix.

4. Complexity Analysis for a Domain-Decomposed Solver

A complexity analysis of a fairly involved computational procedure is most clearly approached in a modular fashion. Accordingly, we begin by constructing operation counts for the five major computational subtasks listed at the end of the previous section, then combine them in proportion. We restrict our analysis here to strip-wise decompositions and to leading-order effects. A strip-wise decomposition with radially oriented strips is natural for a domain consisting of a symmetry plane of the domain sketched in Fig. 1 because the high aspect ratio will lead to tight radial coupling in the diffusive terms of the governing equations, whereas the axial coupling is dominated by convection, and hence predominantly unidirectional.

We begin with a model for the computer itself. A network of p homogeneous processor elements is assumed, each of which has sufficient local memory or a devoted share of global memory to represent the data of the associated subdomain. We define γ as the unit of scalar floating point operation time. Any vector-processing capability of these elements is ignored, although we acknowledge that the potential savings of vectorization are considerable in this context [11].

The processors communicate by passing messages or accessing shared memory through a global bus, depending upon the interconnection network. We define α as the time overhead to route a message of any length between a pair of neighboring processors or enforce a "lock" on a shared memory location, and β as the additional transfer time per floating point word, which is simply the reciprocal of the bus bandwidth for access to shared memory. For simplicity, we assume that only *sending* messages (or *depositing* data in the shared memory) takes time. *Receiving* (or *reading* data) is assumed to be free. Since all sent messages are presumed received at some point, any additional reception cost which ought to be included to model a particular machine can be incorporated by simple readjustment of α and β (*e.g.*, doubling them if the two costs are comparable).

We define two architecture-dependent functions of p: C_E, the *local exchange coefficient* between processors sharing neighboring *sides*, and C_R, the *global reduction coefficient.* The global reduction operation produces a single scalar from scalars defined on each processor, whether by addition in the case of a global inner product whose partial sums are locally accumulated, or by the logical "and" operation in the case of a convergence test. These coefficients which multiply terms in α and β take into account the possibly degraded ability of the network to exchange data as the number of processors grows. In the case of the ring of processors or a hypercube, C_E is unity regardless of the number of processors because there are channels solely devoted to these nearest-neighbor message routes in the network. In the case of the shared-memory machine, we make for simplicity the extreme assumption that the bandwidth to memory for each processor must be divided by the number of processors; hence C_E is p itself. For a ring or a hypercube, C_R is twice the diameter of the network, since a partial sum must be collected, then redistributed. Hence C_R is p or $2\log p$, respectively. C_R is $2p$ for a shared-memory machine.

The complexity model depends in addition on basic discrete problem dimensions: the resolution of the grid n (assumed the same in each direction for simplicity), and the number of components per gridpoint r. We also require an arithmetic operation count coefficient C_A, which depends on the details of the physics and chemistry incorporated into the governing equations and the number of points in the discrete stencil. The product of C_A with $n^2 r$ gives the number of arithmetic operations per residual vector evaluation. In terms of these parameters the major algorithmic subtasks have the following complexities.

DAXPYs:

$$\left[\frac{n^2}{p}r\right]\gamma.$$

The DAXPY requires no communication and parallelizes perfectly over any partitioning.

Global Norms:

$$\left[\frac{n^2}{p}r\right]\gamma + [C_R]\,\alpha + [C_R]\,\beta.$$

The norm requires local inner products followed by a global reduction of the scalar partial sums.

Residual Vector Evaluation:

$$\left[\frac{n^2}{p}C_A r\right]\gamma + [2C_E]\,\alpha + [2nrC_E]\,\beta.$$

To satisfy the data requirements of the nine-point finite difference stencils on the artificial boundaries of the subdomains, each strip subdomain must exchange its two artificial boundary rows with its neighbors.

Jacobian Evaluation:

$$\left[\frac{n^2}{p}\left((9r+1)C_A r + 18r^2\right)\right]\gamma +$$

$$[(2(9r+1)+6)C_E]\,\alpha + \left[(2(9r+1)nr + 6nr^2)C_E\right]\,\beta.$$

The Jacobian evaluation cost includes $(9r+1)$ independent function evaluations, along with some finite difference arithmetic to generate the matrix elements. The communication terms include both the sharing of subdomain boundary data needed for the function evaluation and the exchange

of $6n$ $r \times r$ Jacobian blocks generated on adjacent processors using the grid-coloring described in [29].

For a base-line linear algebraic method, we employ concurrent block-line relaxation, sweeping in the downstream axial direction within each subdomain to follow the convection. In the limit of one processor, this is the block-line Gauss-Seidel method; in the limit of n processors, so that the each subdomain consists of only one row, this is the block-line Jacobi method. Treating only the three central diagonal blocks implicitly, each row of the sweep requires the formation of a modified right-hand side which includes a term in r^2 from each of the other six Jacobian blocks in the same row, followed by a block tridiagonal solve. The factorization of the block tridiagonal system need be performed only once per Jacobian evaluation. The assemblage of the right-hand sides (which includes the exchange of the rows lining the artificial boundaries) and the back-substitution is required on each sweep, as is a convergence check based on a global norm. Thus, we have:

Block Relaxation Factorization Phase:

$$\left[\frac{n^2}{p}\frac{7}{3}r^3\right]\gamma.$$

Block Relaxation Sweep Phase:

$$\left[\frac{n^2}{p}(9r^2)\right]\gamma + [2C_E + C_R]\,\alpha + [2nrC_E + C_R]\,\beta.$$

If the linear systems involving the Jacobian are solved to a level of convergence which is independent of the granularity of the decomposition, then the number of Newton steps is independent of the type of decomposition and the number of processors.* We may therefore use the Newton step as the largest aggregate in the complexity model, and work out the complexity on a per-Newton step basis. Neglecting the overhead associated with possible problem-dependent damping and selection of time steps, each Newton step consists of one DAXPY, one norm, one function evaluation, $1/N$ Jacobian evaluations and L relaxation sweeps, This count introduces the final two parameters of the model: the average number of Newton steps between Jacobian evaluations, N, and the average number of relaxation sweeps per Newton step, L. For systems as complicated as (2.1) through (2.7), no theory exists from which either N or L can be specified as a function of n and p. We must therefore obtain N and L from problem-specific experiments.

For present purposes, we will focus on an individual point in physical and numerical parameter space, leaving only the machine and network parameters free. (A more general approach is taken in [17].) We adopt the 16-species, 46-reaction mechanism of [22] for the oxidation of methane. Here $r = 19$ and the operation count coefficient C_A is determined from counting and relative timing measurements to be approximately 2.6×10^3. $L \approx 200$ and $N \approx 5$ are also available from averaging over various runs of the serial code on a grid of approximately 1200 nonuniformly-spaced points.

With the preceding formulae, we can estimate the execution time T_p per Newton step of the p-processor parallelized solver through a relation of the form:

$$T_p = \gamma f_\gamma + \alpha f_\alpha + \beta f_\beta. \tag{4.1}$$

From the assigned values, we can extract the leading terms of each component of sum as:

*This condition may not be satisfied for relaxation methods in which convergence is based solely upon the size of the update, since the deteriorating condition number of the iteration matrix with finer granularity may allow a larger residual at convergence.

$$f_\gamma = (\frac{C_A}{N} + L)\frac{n^2}{p}r^2, \quad f_\alpha = (\frac{18r}{N} + 2L)C_E + LC_R, \quad f_\beta = (\frac{24r}{N} + 2L)nrC_E + LC_R.$$

The linear appearance of L, the iteration count for the relaxation method, in the leading terms of the communication overhead puts a premium on keeping this parameter as small as possible for good parallel efficiency. $\beta/\gamma = O(1)$ and $\alpha/\gamma = O(10^2)$ are typical. As p, which appears to a negative power in f_γ and to non-negative powers in C_E and C_R increases, the communication terms potentially dominate, rendering the parallel arithmetic capability worthless. For p greater than some $p_{\text{crit}}(\alpha/\gamma, \beta/\gamma)$, the phenomenon known as speed-*down* can occur, in which adding processors increases wall-clock execution time.

5. Alternative Linear Algebraic Methods

In this section we compare a variety of linear algebraic methods on a series of four Jacobian-residual systems of the form

$$J(\phi^{(k)})\ \delta\phi^{(k)} = -F(\phi^{(k)}).$$

The results for the iteration count and global inner product count are shown in Table 1. The first three Jacobians come from flame sheet problems on a 31×31 grid, the last on a more refined 63×63 grid, with discrete dimensions of 2,883 and 11,907, respectively. The first is drawn from the first pseudo-transient Newton time step on the 31×31 grid, following interpolation from the Newton-converged solution on a previous (coarser) grid. A Δt of 1.0×10^{-5} was employed as a continuation parameter for this particular Newton step, which had an linear residual of $O(10^4)$. The next derives from the 84^{th} and last pseudo-transient Newton step, after Δt had adaptively grown to 1.0×10^{-1}. The third comes from a pure Newton step ($\Delta t = \infty$). Finally, after interpolation of the converged solution to the 63×63 grid, a Jacobian was written based on an initial Δt of 1.0×10^{-9} on a problem with a linear residual of $O(10^8)$. Though this seems like an excessively conservative initial time step, the strong nonlinearities and steep gradients prevent the pseudo-transient procedure from converging if a much larger step is attempted. Three of the four linear algebraic methods presented below are adequate for obtaining convergence of the linearized problem at larger time steps (affording less diagonal dominance), but the updated nonlinear residual norms will diverge during the pseudo-transient Newton stepping.

Any linear system solution algorithm intended for use in fluid dynamical applications should be robust with respect both to asymmetry, to allow for the presence of convective terms, and indefiniteness, to allow for the presence of linearized source terms whose coefficients oppose the algebraic sign of the diagonal term of the discrete convective-diffusive operator. The latter can be particularly important in the modeling of chemically reacting flows in which at any point in the flow field some species may be created while others are consumed. In the flame sheet context, however, indefiniteness does not occur, since (2.8) contains no source terms.

The first technique is block-line relaxation, the standard SOR method [14] (except for the misnomer when ω, the over-relaxation parameter, is less than one). The Jacobian (denoted by A in this section), partitioned at the level shown in Fig. 4(b) is decomposed additively in the form $A = D - C_L - C_U$. In the limit of strong convection with upwind differencing, the lower triangular part C_L is more influential than the upper triangular part C_U, and the splitting matrix $(\frac{1}{\omega}D - C_L)$ is chosen, with no symmetrizing step. With $L = D^{-1}C_L$ and $U = D^{-1}C_U$, the overall iteration matrix may be written as $(I - \omega L)^{-1}(\omega U + (1 - \omega)I)$, which form reveals how ω may be adjusted downward to gain control over a divergent problem. Of course, the application of D^{-1} involves a block tridiagonal solve at the level of Fig. 4(c), in which a dense LU factorization is performed at the level of Fig. 4(d).

		$n = 31, \Delta t_i$		$n = 31, \Delta t_f$		$n = 31, \Delta t_\infty$		$n = 63, \Delta t_i$	
Method	Param.	I	IP	I	IP	I	IP	I	IP
Block-Line	$\omega = 0.50$	84	*84*	43	*43*	58	*58*	-	-
	$\omega = 0.75$	51	*51*	23	*23*	31	*31*	-	-
	$\omega = 1.00$	31	*31*	69	*69*	-	-	-	-
	$\omega = 1.20$	17	*17*	-	-	-	-	-	-
GMR/ILU	$K = 10$	7	*28*	22	*113*	28	*146*	54	*281*
	$K = 50$	"	"	21	*231*	24	*300*	25	*325*
GMR/BILU	$K = 10$	6	*15*	16	*76*	17	*83*	26	*131*
	$K = 50$	"	"	14	*105*	14	*105*	18	*171*
Cheby/BILU	$C = 10$	9	*9*	18	*32*	20	*34*	46	*74*
	$C = 20$	"	"	22	*36*	22	*36*	42	*56*

Table 1: Iteration count I, and inner product count IP for Jacobians from different stages of the flame sheet problem, using the four linear algebraic techniques described in the text. Convergence for the block-line method was based on an absolute tolerance of 1.0×10^{-5} on the normalized 2-norm of the update vector. Convergence for the generalized minimum residual and Chebyshev iterations was based on a relative tolerance of 1.0×10^{-5} on the 2-norm of the residual vector. Comparing the two convergence criteria, the block-line criterion was slightly lenient, in that its terminal relative residual reductions were less than five orders of magnitude in some cases. (A hypen indicates divergence. A double quote indicates a trivial repeated result.)

We note that the block-line method is very effective in the first system, which permits over-relaxation, but becomes increasingly less so as the calculation evolves, and eventually fails to converge even at $\omega = 0.5$ in the largest problem. Since each iteration requires an inner product to evaluate the convergence criterion, I and IP are equal for this case.

The second technique is the Generalized Minimum Residual (GMR) method [25] with an point incomplete LU factorization preconditioner, permitting no fill-in. The code PCGPAK [27] was employed for these runs. Given a system of equations, $Mx = f$, M nonsingular, and an initial iterate, x_0, with initial residual, $r_0 = f - Mx_0$, GMR computes the solution x from finding $z \in K_k$ such that

$$(r_0 - Mz, v) = 0,$$

for all $v \in L_k$, and setting $x = x_0 + z$, where K_k and L_k are Krylov spaces based on r_0:

$$K_k \equiv \text{span}\{r_0, Mr_0, \ldots, M^{k-1}r_0\}, \quad L_k \equiv \text{span}\{Mr_0, M^2r_0, \ldots, M^kr_0\}.$$

The solution x computed after k steps of GMR minimizes $||r||_2$ in the affine space $x_0 + K_k$. In a practical algorithm, an orthogonal basis for K_k is built up by means of a Gram-Schmidt or Householder process, which obviates the necessity of working with the normal equations. Suitable computer implementations of GMR have been given in [25] and [31], of which the former is employed in these tests. Among the desirable properties of GMR are: (1) the only reference to M is in form of matrix-vector products, (2) it cannot break down (in exact arithmetic) short of delivering the solution even for nonsymmetric systems with indefinite symmetric part, (3) it requires less storage and

fewer operations per step than the mathematically equivalent GCR and ORTHODIR algorithms, and (4) the 2-norm of the residual is non-increasing and can be monitored without constructing intermediate solution iterates. The main disadvantage of GMR is the lack of a bounded recurrence relation, which causes the operation count and storage requirements to grow quadratically and linearly, respectively, in the iteration index. We note in particular that $k(k+1)/2$ inner products are required in a k-step procedure. In many applications, restarting GMR after a predetermined number of steps is amelioratory, but restarted GMR can also fail through stagnation.

GMR is often uneconomical when left to act by itself on a general reaction-convection-diffusion operator. In an effort to control the work and storage required by GMR when A has a widely spread spectrum, we precondition the iterations by taking M in the formulae above to be AB^{-1}, where B is an approximation to A whose inverse is relatively inexpensive to apply. This is "right" preconditioning, which first solves $M\tilde{x} = f$ for $\tilde{x}$, then $Bx = \tilde{x}$ for x. We adopt right over left preconditioning because in the latter the matrix B enters into the GMR residual convergence criterion in a direct way, making convergence comparisons between different preconditioning techniques difficult. Approximate factorizations of the original matrix into triangular matrices, such as incomplete LU-decomposition (ILU) [21], are useful general purpose preconditioners for GMR and other Krylov methods. "Incomplete" refers to the fact that certain elements which would fill in and destroy the sparsity of A in a complete factorization are never computed or stored, saving operations and memory. The present examples employ the crudest of techniques appropriate to nonsymmetric systems, a Crout-Doolittle ILU(0), where the zero indicates that no fill-in outside the original sparsity pattern of A is allowed (see [32]).

From the table, we note that GMR/ILU is more robust than the block-line method, but requires substantial memory and inner products in "difficult" problems. Two versions of GMR/ILU are presented. The $K = 10$ version is a restarted GMR which discards the Krylov subspace every 10 iterations, constructs the new residual vector, and iterates again. Using a restart cycle reduces inner product count by keeping k small in the quadratic term. However, as in the fourth system presented in the table, overall iteration count can increase and take back some of this gain in the IP category. The $K = 50$ version is essentially $K = \infty$ in the cases considered, since I never exceeded 50. The preconditioner is essential. As a baseline reference for the first and best conditioned case in the table, only two decades of residual reduction were obtained after 50 steps of unpreconditioned GMR, or after 200 steps of restarted ($K = 10$) unpreconditioned GMR.

The third technique is again GMR, but with a block preconditioning at the level of Fig. 4(c). This algorithm was hand-coded from scratch in anticipation of a parallel port (see next section). Block preconditioning exploits the fact that the tightest coupling in our systems is between unknowns at the same gridpoint, which it handles exactly. The superiority of BILU over ILU is apparent in all table entries. Of course, the preconditioning matrix is itself more expensive to form and apply.

The fourth technique is BILU-preconditioned adaptive Chebyshev iteration [19], suitable for cases in which the indefiniteness can be controlled and advantageous in parallel contexts because of its short recurrence relation. The code CHEBYCODE [1] was employed for these runs, supplemented by the same block-ILU code used in the previous technique.

From the initial iterate, x_0, the Chebyshev algorithm generates a sequence of iterates such that

$$r_k = P_k(M)r_0,$$

where

$$P_k(z) = T_k\left(\frac{d-z}{c}\right) \Big/ T_k\left(\frac{d}{c}\right),$$

and T_k is the k^{th} Chebyshev polynomial of the first kind. The parameter d defines the center and

c the foci (at $d \pm c$) of a family of confocal ellipses in the complex plane on which the Chebyshev polynomials are rapidly decreasing in norm as k increases. There is a smallest such ellipse which contains the spectrum of M. If it does not contain the origin, the sequence of residuals above is rapidly convergent. Like GMR, Chebyshev iteration deals directly with a nonsymmetric matrix M, without reference to the normal equations. Unlike GMR, Chebyshev iteration requires the estimation of parameters describing the convex hull of eigenvalues of M. After an initial guess, these estimates are obtained adaptively using the modified power method labeled "Method 3" in [20] as a by-product of successive residuals. For a sub-sequence of four residuals, the cost of these estimates is 14 inner products per adaptive step. The present examples employ $d = 1$, $c = 0$ as the initial guess, which describes circles centered at the point $z = 1$. This is a reasonable estimate for a good preconditioner like block-ILU, since the spectrum of AB^{-1} lies in a small region near 1. (In all the cases we have inspected, the real parts are bounded between 0.2 and 1.5 and the imaginary parts between ± 0.4.)

The table shows that Chebyshev/BILU is always less powerful than GMR/ILU in terms of iteration count (and hence matrix-vector products using M), but always superior in terms of inner product count. Two versions of Chebyshev/BILU are presented. The $C = 10$ version forces an adaptive estimation of d and c every ten steps unless the iterations are converging sufficiently rapidly. The $C = 20$ version allows twice as many slowly converging steps between adaptations. (Divergence also invokes the the adaptive procedure in CHEBYCODE, but this did not occur in our examples.) Iteration count is not monotonic in C (unlike K in GMR), so problem-specific investigation is worthwhile.

In summarizing the discussion of Table 1, block-preconditioned GMR and Chebyshev methods have complementary strengths. The GMR version always requires fewer references to M and (roughly proportionally) less CPU time on a serial computer, even when restarted at intervals commensurate with Chebyshev adaptive cycles. However, the decision on which one to prefer in parallel depends ultimately on the communication-to-computation rates of the hardware in question, since the Chebyshev version is always superior in inner products.

6. Parallel Experiments for Flame Sheet Jacobians

We have seen in section 4 that domain decomposition of the Newton solver is trivial except for the solution of the linear systems, and in section 5 that GMR/BILU is the method of choice for our flame sheet Jacobians in the serial context. We also have from the complexity analysis that individual matrix-vector multiplies are no worse from a communication point of view than the residual vector evaluations required by other stages of Newton's method. Therefore, provided that iteration counts can be kept low, say $O(10r/N)$, it is only the preconditioner phase of the GMR algorithm that poses a parallel challenge. A global ILU can be a bottleneck in parallel implementations because of the sequential nature of triangular factorizations and solves. Though wavefront-based or red-black reorderings of the standard sequential operations can alleviate this problem [26] in the context of sparse banded matrices, domain-decomposition approaches side-step it altogether by applying ILU within subdomains only. We now describe two domain-decomposed implementations of GMR/BILU. Both consist of creating separate BILU preconditioners for each subdomain and building them into a global preconditioner which is basically block-diagonal at the subdomain level. The two methods differ in that one of the preconditioners is allowed nonzero border blocks to provide for the flow of information between subdomains in the preconditioner phase.

The GMR iterations are already "inner iterations" from the point of view of the overall nonlinear process. To avoid any deeper nesting of iterations in applications, we choose *a priori* to iterate simultaneously on all of the unknowns in the linear system, in the sense that the subdomain

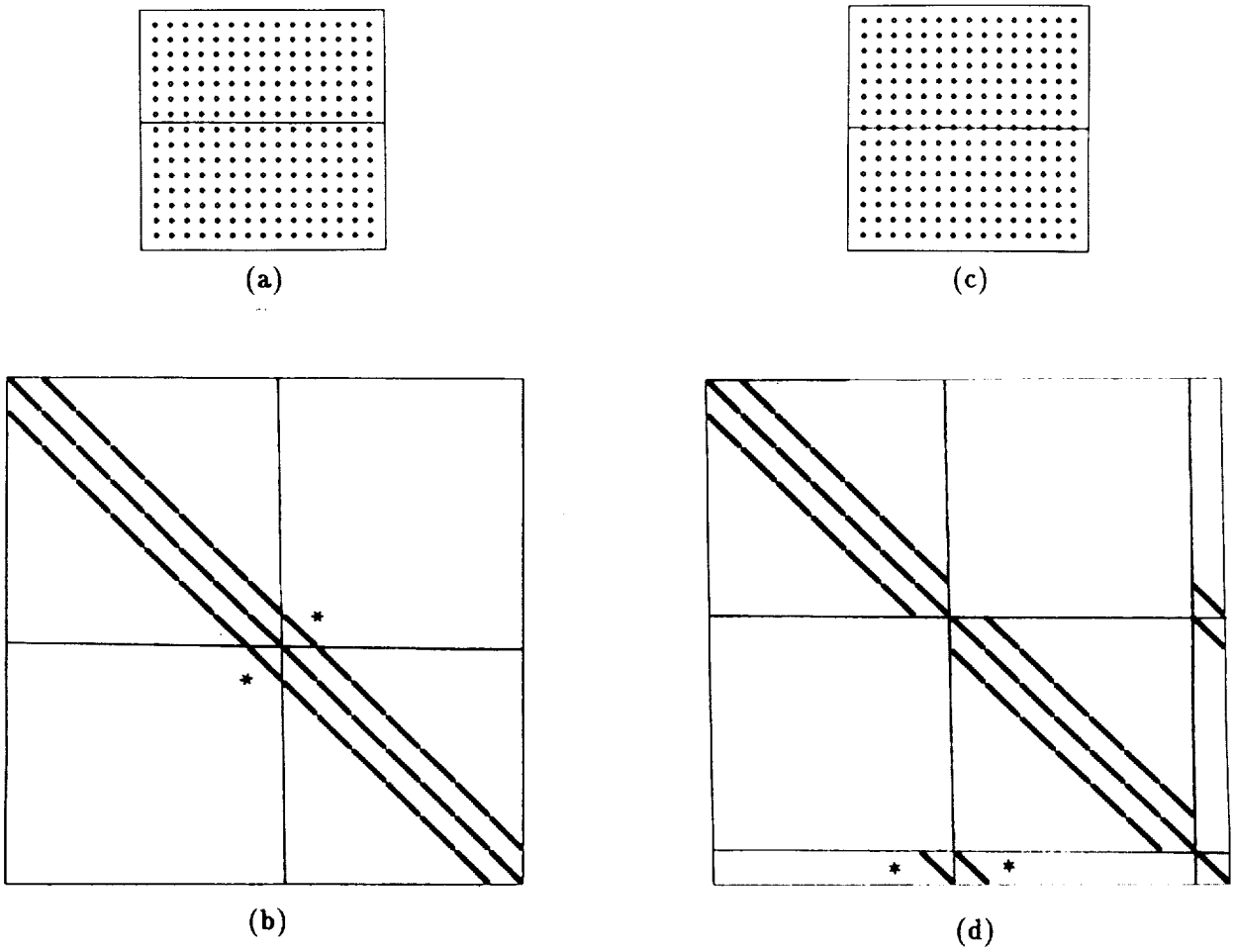

Figure 5: Sparsity patterns for two different decompositions of a rectangular region into two strips using a 9-point finite-difference template: (a) without edge; (b) sparsity pattern of A for (a); (c) with edge; (d) sparsity pattern of A for (c). The B operators have the asterisked blocks removed for parallelism. Within each subdomain the gridpoints are ordered lexicographically.

problems are not individually iterated to convergence before their values are used to update the right-hand sides of the equations for the interfacial unknowns. This form of full matrix domain decomposition was advocated in [5] for problems in which no fast solver is known for the subdomain interior problems, and has been demonstrated therein to lead to an optimally convergent scheme for a class of self-adjoint strongly elliptic operators, provided (among other things) that spectrally equivalent subdomain preconditioners are employed.

The best paradigm for domain decomposition is the decomposition into two strips of a rectangular region overlaid by a tensor-product grid. In one case the cut passes "between" grid points; in the other, it follows a line of gridpoints, which are ordered separately. For a 9-point operator on a grid with 16 interior subintervals in each direction (with Dirichlet boundary conditions eliminated) the resulting sparsity pattern for the operators A and B indicated graphically in Fig. 5.

The matrix of Fig. 5(b) can be written

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad \text{with a preconditioner} B = \begin{pmatrix} \tilde{A}_{11} & 0 \\ 0 & \tilde{A}_{22} \end{pmatrix} \tag{6.1}$$

where the $\tilde{A}_{ii}$ are ILU approximations. In this method, the creation of independent threads of computation in the preconditioner comes at the expense of severing the interdomain coupling altogether. It therefore parallelizes with virtually no overhead, but potentially suffers in iteration count as the granularity of the decomposition is refined and more and more of the coupling is discarded. Its limit is block-point diagonal preconditioning. Some experiments with this technique have been reported in [2] and [3], and we have also tested it on all of the Jacobians of the previous section. It is referred to as the "uncoupled" preconditioning in the tables to follow.

The matrix of Fig. 5(d) is

$$A = \begin{pmatrix} A_{11} & 0 & A_{13} \\ 0 & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}. \tag{6.2}$$

Here, A_{33} renders the coupling between the points on the interface itself. The conformally partitioned preconditioning matrix we propose for A is

$$B = \begin{pmatrix} \tilde{A}_{11} & 0 & \tilde{A}_{13} \\ 0 & \tilde{A}_{22} & \tilde{A}_{23} \\ 0 & 0 & \tilde{C} \end{pmatrix}, \tag{6.3}$$

where $\tilde{C}$ approximates the Schur complement of A_{11} and A_{22} in A. The exact Schur complement, C, may be obtained from block-Gaussian elimination on A as:

$$C \equiv A_{33} - A_{31}A_{11}^{-1}A_{13} - A_{32}A_{22}^{-1}A_{23}. \tag{6.4}$$

The tilde-notation in the definition of B accommodates the replacement, if convenient, of the exact A_{ij} with approximations thereto. We assume throughout that the A_{ii} are invertible. (This is certainly a reasonable requirement for a discrete convective-diffusive operator.) Under this assumption, C is also invertible [8]. This technique is referred to as the "Modified Schur Complement (MSC)-coupled" preconditioning. For further details on the motivation and construction of MSC-interface preconditioning, see [7, 15, 18]. For present purposes it is important only to note that the MSC-coupled version is hierarchical; that is, it is built on top of the subdomain preconditioning and requires a small number of ILU solves on each subdomain to construct. If the subdomain preconditioning is poor, there is no reason to expect that the MSC preconditioning of the interfaces is good. Note that the inverse of either B can be applied with one solve in each subdomain.

Tables 2 through 5 in this section report parallel results on the four Jacobian systems considered above.

The first case, the initial pseudo-transient step on the 31 x 31 grid, is solved with each possible number of processors between 1 and 16. One notes that the iteration count grows monotonically with the granularity of the decomposition for the uncoupled version, a degradation of convergence penalty for throwing away information. Somewhat surprisingly, the iteration count has non-monotonic behavior in the coupled case. The fact that the 8-subdomain case requires fewer iterations than the global algorithm causes super-linear speed-up; that is the 8-processor code executes more than 8 times as fast as the 1-processor code. This effect is due to an interaction between the MSC and ILU preconditioners, and eventually is swamped by the loss of the "chunky" subdomain condition at high processor numbers. The actual parallel efficiencies e are plotted in

Fig. 6. In order to separate iteration count effects from synchronization and communication effects, a third curve appears in this figure. For the third curve, which is for the MSC-coupled case, the iteration count has been fixed at the $p = 1$ total of 6, independent of residual convergence. The peaks of this curve occur at $p = 2^d$, at which all of the subdomain interiors have the same discrete thickness. (For $p = 2^d$, there are $d + 1$ subdomains of exactly $2^{5-d} - 1$ rows each.) At all other p, there are load imbalances between the subdomains which allow idling.

	Uncoupled			MSC-Coupled		
p	I	T	e	I	T	e
1	6	24.9	1.00	6	25.1	1.00
2	6	13.4	.93	6	13.4	.94
3	6	9.1	.91	6	9.3	.90
4	6	6.8	.91	6	6.8	.92
5	6	5.6	.89	5	5.1	.98
6	6	4.9	.85	6	5.1	.82
7	9	6.1	.58	5	3.7	.97
8	9	4.9	.64	5	3.0	1.05
9	8	4.4	.63	6	3.5	.80
10	9	4.8	.52	6	3.4	.74
11	10	4.1	.55	7	3.1	.74
12	10	4.2	.49	7	3.1	.67
13	12	4.9	.39	7	3.1	.62
14	13	5.4	.33	8	3.5	.51
15	13	5.4	.31	8	3.5	.48
16	13	3.9	.40	8	2.6	.60

Table 2: Iteration count I, CPU time T, and efficiency e for Jacobians from the initial pseudo-transient Newton step of the flame sheet problem with horizontal strips on a 31×31 grid as a function of number of processors p.

For the next case, the final pseudo-transient step on the 31×31 grid, and subsequently, attention is focused on iteration behavior, by considering $p = 2^d$ only. One notes that the iteration count grows monotonically and rather badly for both preconditioners; however, wall-clock speed-ups are obtained all the way to the limit of 16 processors. Neither method is preferable at all values of p; however, the uncoupled one is surprisingly good. This competitive behavior of the uncoupled algorithm is *not* generally found with scalar convective-diffusive systems (see [18]), since spatial coupling is dominant in such problems. In reacting flows, the grid-point blocking is enough for a decent preconditioning. In fact, the attempt of the MSC preconditioning to account for spatially-connected neighbors interacts negatively with the BILU in several cases.

For the third case, the full Newton step on the 31×31 grid, all of the conclusions of the previous case are repeated. The efficiencies are the lowest of any of the four cases due to degradation of convergence, but speed-ups do occur all the way up to $p = 16$.

The final case is a reprise of the initial one. The MSC-coupled version displays a retrograde iteration count, and very high terminal efficiencies are obtained, in spite of the loss of "chunkiness".

All of the decompositions presented in this section were for strips normal to the axial direction. Cases for strips normal to the radial direction were also run, but were always inferior for a corresponding number of subdomains. This is a natural reflection of the tight radial coupling in the physics of the problem, which is preserved in strips which span from the axis to the confining

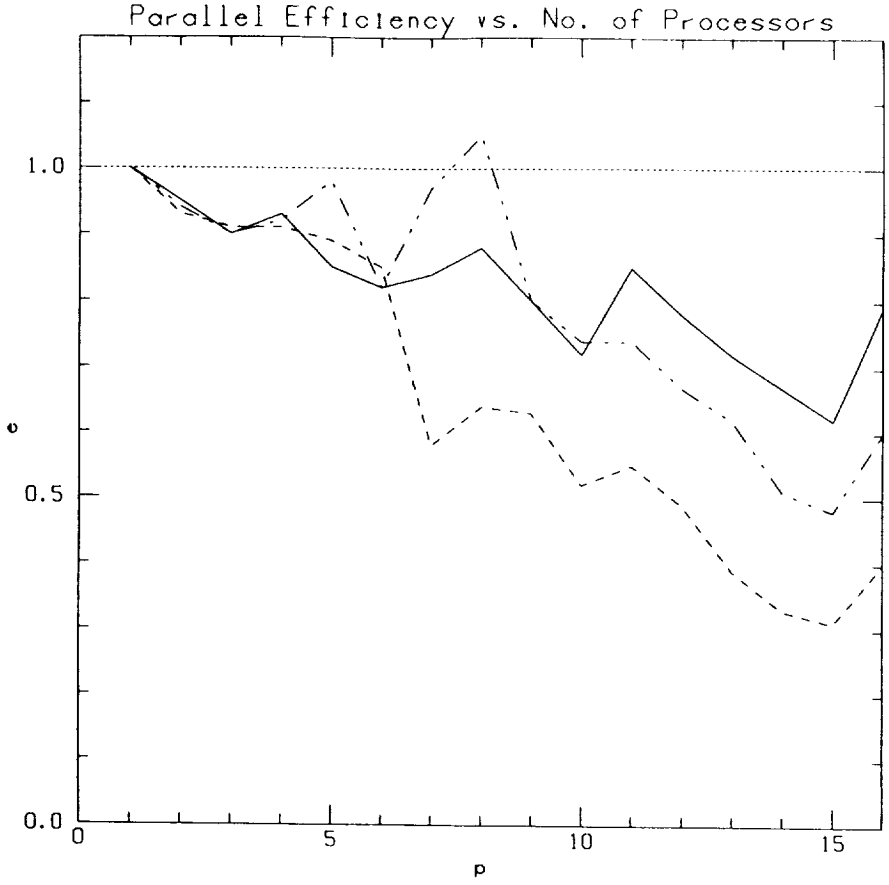

Figure 6: Graphs of efficiency versus number of processors for the solving the first linear system at the first pseudo-transient Newton step on the 31 x 31 flame sheet problem. The uniform dashed curve is for the uncoupled algorithm and the patterned dashed curve for the MSC-coupled version. The solid curve is for a reference case in which the number of iterations has been fixed at a constant value of 6.

	Uncoupled			MSC-Coupled		
p	I	T	e	I	T	e
1	14	56.0	1.00	14	56.2	1.00
2	15	31.7	.88	17	36.5	.77
4	16	17.0	.82	19	21.0	.67
8	20	11.2	.63	21	11.9	.59
16	25	8.1	.43	21	6.7	.52

Table 3: Iteration count I, CPU time T, and efficiency e for Jacobians from the final pseudo-transient Newton step of the flame sheet problem with horizontal strips on a 31 x 31 grid as a function of number of processors p.

	Uncoupled			MSC-Coupled		
p	I	T	e	I	T	e
1	14	56.0	1.00	14	56.2	1.00
2	16	34.2	.82	18	38.9	.72
4	18	19.3	.73	24	27.5	.51
8	22	12.6	.55	30	18.2	.39
16	28	9.5	.38	34	12.1	.29

Table 4: Iteration count I, CPU time T, and efficiency e for Jacobians from a full Newton step of the flame sheet problem with horizontal strips on a 31×31 grid as a function of number of processors p.

	Uncoupled			MSC-Coupled		
p	I	T	e	I	T	e
1	18	307.	1.00	18	309.	1.00
2	18	163.	.94	18	155.	1.00
4	19	85.2	.90	18	81.2	.95
8	19	45.9	.83	17	39.0	.99
16	21	27.6	.70	17	23.3	.83

Table 5: Iteration count I, CPU time T, and efficiency e for Jacobians from the initial pseudo-transient Newton step of the flame sheet problem with horizontal strips on a 63×63 grid as a function of number of processors p.

wall. It is also an argument against box-wise decompositions, which are desirable in more isotropic problems from a surface-to-volume ratio point of view.

Summarizing the results of this section, GMR/BILU has acceptable behavior in all representative stages of a flame sheet calculation, but guidelines on when the interface coupling has sufficient incremental value to warrant its use are still lacking.

7. Conclusions and Further Directions

We conclude that domain decomposition is a practical means of getting appreciable speed-ups ($\approx$ 10) on a 16-processor array for representative two-dimensional finite-differenced systems of PDEs. The generalized minimum residual method with point-blocked ILU preconditioning is powerful in this context, but places more demands on the global communication properties of the interprocessor network than a Chebyshev method with the same preconditioning.

Further research is suggested in areas of general algorithmic interest and those specific to reacting flow systems.

A hybrid GMR-Chebyshev algorithm has been suggested in [10], which would seem very well suited to parallel implementation, since it combines low inner product count of Chebyshev iteration with an eigenvalue estimation scheme using a GMR-Arnoldi method, in which even the inner products themselves are by-products of an iteration-advancing step.

It would be of interest to compare our subdomain-based parallel ILU preconditioners to those in which the factorization (block-point or otherwise) is carried out over an entire domain. Since no interfacial blocks requiring approximation are introduced, it is expected that this technique, appropriately blocked, will have a better iteration count than subdomain-based techniques on most

problems; however, its parallel efficiency can be so low in some problems (due to synchronization delays) that it is inferior even to unpreconditioned GMR [4] as a parallel algorithm. A wide range of comparisons are needed here.

There is no consideration of complex domain geometry in the present work although ease of generalization to this case is an equally relevant motivation. The case of problems requiring too much memory to be managed by just one processor is another one in which the proposed technique is attractive, despite possible iteration count degradation relative to a global technique. Comparisons to non-domain decomposition-based techniques for non-tensor product grids and out-of-core solvers would be relevant on large scientific and engineering problems.

Finally, from the applications point of view, Jacobians from detailed kinetics reacting flow models should be studied. The regime of many components per gridpoint ($r \approx 20$ rather than $r = 3$) is expected to further bring out of the advantages of block-point preconditioning, reduce the importance of accurate interfacial coupling in the overall iteration count, and reduce the communication-to-computation ratio by adding zero-space-dimensional work. Verification of these statements will mean that future turn-around times for multidimensional reacting flow calculations on currently available supercomputers will be measured in hours rather than days.

8. Acknowledgements

The support of Dr. R. G. Voigt of the Institute for Computer Applications in Science and Engineering, and Prof. M. H. Schultz of the Yale Research Center for Scientific Computation is gratefully acknowledged. This synthetic work, bridging architectures and applications with novel algorithms, would not have been possible without collaborations during the past few years with Profs. W. D. Gropp (Dept. of Computer Science) and M. D. Smooke (Dept. of Mechanical Engineering) of Yale.

9. References

[1] S. F. Ashby, *CHEBYCODE: A FORTRAN Implementation of Manteuffel's Adaptive Chebyshev Algorithm,* Technical Report UIUCDCS-R-85-1203, Dept. of Computer Science, University of Illinois, May 1985.

[2] C. Ashcraft and R. Grimes, *On Vectorizing Incomplete Factorizations and SSOR Preconditioners,* Technical Report ETA-TR-41, Boeing Computer Services, December 1986.

[3] C. Ashcraft, *Domain Decoupled Incomplete Factorizations,* Technical Report ETA-TR-49, Boeing Computer Services, April 1987.

[4] D. J. Baxter, *personal communication,* 1987.

[5] J. H. Bramble, J. E. Pasciak and A. H. Schatz, *The Construction of Preconditioners for Elliptic Problems by Substructuring, I,* Math. Comp., 47 (1986), pp. 103–134.

[6] T. F. Chan, R. Glowinski, J. Periaux and O. Widlund,, *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989 (to appear).

[7] T. F. Chan and D. Resasco, *A Survey of Preconditioners for Domain Decomposition,* Technical Report 414, Computer Science Dept., Yale University, September 1985. In Proceedings of the IV Coloquio de Matemáticas del CINVESTAV, Workshop in Numerical Analysis and its applications, Taxco, Mexico, Aug. 18-24, 1985.

[8] R. W. Cottle, *Manifestations of the Schur Complement,* Lin. Alg. Appl., 8 (1974), pp. 189–211.

[9] A. R. Curtis, M. J. Powell and J. K. Reid, *On the Estimation of Sparse Jacobian Matrices,* J. Inst. Math. Appl., 13 (1974), pp. 117–119.

[10] H. C. Elman, Y. Saad & P. E. Saylor, *A Hybrid Chebyshev Krylov Subspace Algorithm for Solving Nonsymmetric Systems of Linear Equations,* Technical Report 301, Dept. of Computer Science, Yale University, February 1984.

[11] V. Giovangigli & N. Darabiha, Vector Computers and Complex Chemistry Combustion, C. M. Brauner & Cl. Schmidt-Lainé ed., *Mathematical Modeling in Combustion and Related Topics - Proceedings of a NATO Advanced Research Workshop, April 27-30, 1987, Lyon, France,* Martinus Nijhoff, Dordrecht, the Netherlands, 1988, pp. 491–503.

[12] R. Glowinski, G. H. Golub, G. A. Meurant and J. Periaux,, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations,* SIAM, Philadelphia, 1988.

[13] A. D. Gosman, W. M. Pun, A. K. Runchal, D. B. Spalding & M. Wolfshtein, *Heat and Mass Transfer in Recirculating Flows,* Academic Press, New York, 1969.

[14] L. A. Hageman & D. M. Young, *Applied Iterative Methods,* Academic Press, New York, 1981.

[15] D. E. Keyes and W. D. Gropp, *A Comparison of Domain Decomposition Techniques for Elliptic Partial Differential Equations and their Parallel Implementation,* SIAM J. Sci. Stat. Comp., 8 (1987), pp. s166-202.

[16] D. E. Keyes and M. D. Smooke, *Flame Sheet Starting Estimates for Counterflow Diffusion Flame Problems,* J. Comp. Phys., 73 (1987), pp. 267–288.

[17] ————, A Parallelized Elliptic Solver for Reacting Flows, A. K. Noor ed., *Parallel Computations and Their Impact on Mechanics,* ASME, 1987, pp. 375–402.

[18] D. E. Keyes & W. D. Gropp, Domain Decomposition Techniques for Nonsymmetric Systems of Elliptic Boundary Value Problems: Examples from CFD, T. F. Chan, R. Glowinski, J. Periaux & O. Widlund ed., *Second International Symposium on Domain Decomposition Methods,* SIAM, Philadelphia, 1989 (to appear).

[19] T. A. Manteuffel, *The Tchebychev Iteration for Nonsymmetric Linear Systems,* Numer. Math., 28 (1977), pp. 307–327.

[20] ————, *Adaptive Procedure for Estimating Parameters for the Nonsymmetric Tchebychev Iteration,* Numer. Math., 31 (1978), pp. 183–208.

[21] J. A. Meierink and H. A. Van der Vorst, *Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they Occur in Practical Problems,* J. Comp. Phys., 44 (1981), pp. 134–155.

[22] J. A. Miller, R. J. Kee, M. D. Smooke & J. F. Grcar, *The Computation of the Structure and Extinction of a Methane-Air Stagnation Point Diffusion Flame,* Technical Report WSS/CI 84-20, The Combustion Institute, 1984. (presented at the 1984 Spring Meeting of the Western States Section of the Combustion Institute, University of Colorado, Boulder).

[23] A. K. Noor, ed., *Parallel Computations and Their Impact on Mechanics,* ASME, New York, 1987.

[24] W. Proskurowski, ed., *Applied Numerical Mathematics (special issue on domain decomposition),* 1989.

[25] Y. Saad and M. Schultz, *GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems,* SIAM J. Sci. Stat. Comp., 7 (1986), pp. 856–869.

[26] Y. Saad & M. Schultz, *Parallel Implementations of Preconditioned Conjugate Gradient Methods,* Technical Report YALEU/DCS/RR-425, Computer Science Dept., Yale University, October 1985.

[27] Scientific Computing Associates, *PCGPAK User's Guide,* 1984.

[28] M. D. Smooke, *Solution of Burner-Stabilized Pre-Mixed Laminar Flames by Boundary Value Methods,* J. Comp. Phys., 48 (1982), pp. 72–105.

[29] M. D. Smooke, R. E. Mitchell & J. F. Grcar, Numerical Solution of a Confined Laminar Diffusion Flame, G. Birkhoff & A. Schoenstadt ed., *Elliptic Problem Solvers II,* Academic Press, New York, 1984, pp. 557–568.

[30] M. D. Smooke, A. A. Turnbull, R. E. Mitchell & D. E. Keyes, Solution of Two-Dimensional Axisymmetric Laminar Diffusion Flames by Adaptive Boundary Value Methods, C. M. Brauner & Cl. Schmidt-Lainé ed., *Mathematical Modeling in Combustion and Related Topics - Proceedings of a NATO Advanced Research Workshop, April 27-30, 1987, Lyon, France,* Martinus Nijhoff, Dordrecht, the Netherlands, 1987, pp. 261–300.

[31] H. F. Walker, *Implementation of the GMRES Method Using Householder Transformations,* SIAM J. Sci. Stat. Comp., 9 (1988), pp. 152–163.

[32] J. W. Watts, III, *A Conjugate Gradient-Truncated Direct Method for the Iterative Solution of the Reservoir Simulation Pressure Equation,* Soc. Petrol. Engin. J., 21 (1981), pp. 345–353.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-181719 ICASE Report No. 88-52		

4. Title and Subtitle	5. Report Date
DOMAIN DECOMPOSITION METHODS FOR THE PARALLEL COMPUTATION OF REACTING FLOWS	September 1988
	6. Performing Organization Code
7. Author(s) David E. Keyes	8. Performing Organization Report No. 88-52
	10. Work Unit No. 505-90-21-01
9. Performing Organization Name and Address Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23665-5225	11. Contract or Grant No. NAS1-18107, AFOSR 88-0117
	13. Type of Report and Period Covered Contractor Report
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225	14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Richard W. Barnwell

Final Report

Submitted to Minnesota Supercomputer Institute's Workshop on Practical Iterative Methods for Large-Scale Problems

16. Abstract

Domain decomposition is a natural route to parallel computing for partial differential equation solvers. In this procedure, subdomains of which the original domain of definition is comprised are assigned to independent processors at the price of periodic coordination between processors to compute global parameters and maintain the requisite degree of continuity of the solution at the subdomain interfaces. In the domain-decomposed solution of steady multidimensional systems of PDEs by finite difference methods using a pseudo-transient version of Newton iteration, the only portion of the computation which generally stands in the way of efficient parallelization is the solution of the large, sparse linear systems arising at each Newton step. For some Jacobian matrices drawn from an actual two-dimensional reacting flow problem, we make comparisons between relaxation-based linear solvers and also preconditioned iterative methods of Conjugate Gradient and Chebyshev type, focusing attention on both iteration count and global inner product count. The generalized minimum residual method with block-ILU preconditioning is judged the best serial method among these considered, and parallel numerical experiments on the Encore Multimax demonstrate for it approximately 10-fold speed-up on 16 processors. The three special features of reacting flow models in relation to these linear systems are: the possibly large number of degrees of freedom per gridpoint, the dominance of dense intra-point source-term coupling over interpoint convective-diffusive coupling throughout significant portions of the flow-field, and strong nonlinearities which restrict the time-step to small values (independent of linear algebraic considerations) throughout significant portions of the iteration history. Though these features are exploited to advantage herein, many aspects of the paper are applicable to the modeling of general convective-diffusive systems.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
domain decomposition, iterative methods, Newton's method, parallel algorithms, computational fluid dynamics, combustion, nonlinear elliptic problems	64 - Numerical Analysis Unclassified - unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	25	A02