To Appear: AMS Contemporary Mathematics Series, Volume **218**, 1998

A Parallel Non-Overlapping Domain-Decomposition Algorithm for Compressible Fluid Flow Problems on Triangulated Domains

Timothy J. Barth, Tony F. Chan, and Wei-Pai Tang

1. Introduction

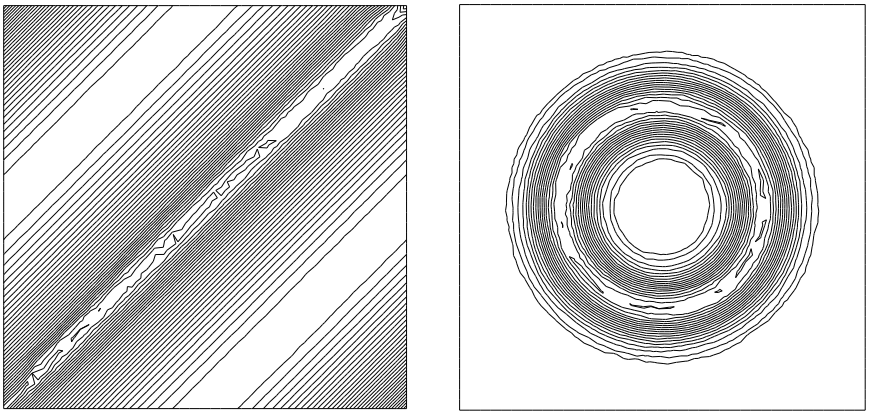
This paper considers an algebraic preconditioning algorithm for hyperbolic-elliptic fluid flow problems. The algorithm is based on a parallel non-overlapping Schur complement domain-decomposition technique for triangulated domains. In the Schur complement technique, the triangulation is first partitioned into a number of non-overlapping subdomains and interfaces. This suggests a reordering of triangulation vertices which separates subdomain and interface solution unknowns. The reordering induces a natural 2×2 block partitioning of the discretization matrix. Exact LU factorization of this block system yields a Schur complement matrix which couples subdomains and the interface together. The remaining sections of this paper present a family of approximate techniques for both constructing and applying the Schur complement as a domain-decomposition preconditioner. The approximate Schur complement serves as an algebraic coarse space operator, thus avoiding the known difficulties associated with the direct formation of a coarse space discretization. In developing Schur complement approximations, particular attention has been given to improving sequential and parallel efficiency of implementations without significantly degrading the quality of the preconditioner. A computer code based on these developments has been tested on the IBM SP2 using MPI message passing protocol. A number of 2-D calculations are presented for both scalar advection-diffusion equations as well as the Euler equations governing compressible fluid flow to demonstrate performance of the preconditioning algorithm.

The efficient numerical simulation of compressible fluid flow about complex geometries continues to be a challenging problem in large scale computing. Many

1991 *Mathematics Subject Classification.* Primary 65Y05, 65Y10; Secondary 76R99, 76N10.

The second author was partially supported by the National Science Foundation grant ASC-9720257, by NASA under contract NAS 2-96027 between NASA and the Universities Space Research Association (USRA).

The third author was partially supported by NASA under contract NAS 2-96027 between NASA and the Universities Space Research Association (USRA), by the Natural Sciences and Engineering Research Council of Canada and by the Information Technology Research Centre which is funded by the Province of Ontario.



(a) $u_x + u_y = 0$, Entrance/exit flow.

(b) $yu_x - xu_y = \epsilon \Delta u$, Recirculation flow in advection limit.

FIGURE 1. Two model advection flows.

computational problems of interest in combustion, turbulence, aerodynamic performance analysis and optimization will require orders of magnitude increases in mesh resolution and solution unknowns to adequately resolve relevant fluid flow features. In solving these large problems, algorithmic scalability [1] becomes fundamentally important. To understand algorithmic scalability, we think of the partial differential equation discretization process as producing linear or linearized systems of equations of the form

$$Ax - b = 0 \tag{1.1}$$

where A is some large (usually sparse) matrix, b is a given right-hand-side vector, and x is the desired solution. For many practical problems, the amount of arithmetic computation required to solve (1.1) by iterative methods can be estimated in terms of the condition number of the system $\kappa(A)$. If A is symmetric positive definite (SPD) the well-known conjugate gradient method converges at a constant rate which depends on κ. After n iterations of the conjugate gradient method, the error ϵ satisfies

$$\frac{\|\epsilon^n\|_2}{\|\epsilon^0\|_2} \leq \left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1} \right)^n . \tag{1.2}$$

The situation changes considerably for advection dominated problems. The matrix A ceases to be SPD so that the performance of iterative methods is not always linked to the condition number behavior of A. Moreover, the convergence properties associated with A can depend on nonlocal properties of the PDE. To see this, consider the advection and advection-diffusion problems shown in Fig. 1. The entrance/exit flow shown in Fig. 1(a) transports the solution and any error components along 45° characteristics which eventually exit the domain. This is contrasted with the recirculation flow shown in Fig. 1(b) which has circular characteristics in the advection dominated limit. In this (singular) limit, any radially symmetric error components persist for all time. The behavior of iterative methods for these two problems is

[1] the arithmetic complexity of algorithms with increasing number of solution unknowns

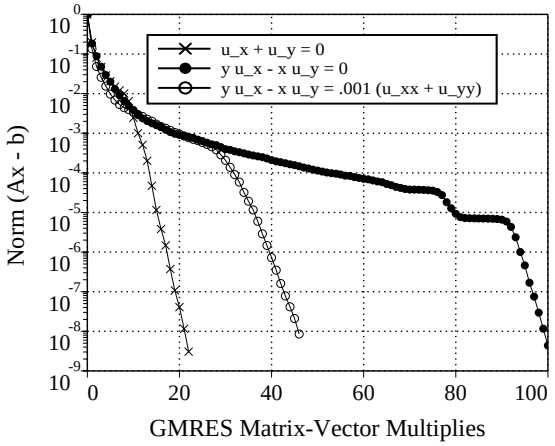


FIGURE 2. Convergence behavior of ILU preconditioned GMRES for entrance/exit and recirculation flow problems using GLS discretization in a triangulated square (1600 dofs).

notably different. Figure 2 graphs the convergence history of ILU-preconditioned GMRES in solving Cuthill-McKee ordered matrix problems for entrance/exit flow and recirculation flow discretized using the Galerkin least-squares (GLS) procedure described in Sec. 2. The entrance/exit flow matrix problem is solved to a 10^{-8} accuracy tolerance in approximately 20 ILU-GMRES iterations. The recirculation flow problem with $\epsilon = 10^{-3}$ requires 45 ILU-GMRES iterations to reach the 10^{-8} tolerance and approximately 100 ILU-GMRES iterations with $\epsilon = 0$. This difference in the number of iterations required for each problem increases dramatically as the mesh is refined. Any theory which addresses scalability and performance of iterative methods for hyperbolic-elliptic problems must address these effects.

2. Stabilized Numerical Discretization of Hyperbolic Systems

Non-overlapping domain-decomposition procedures such as those developed in Sec. 5 strongly motivate the use of compact-stencil spatial discretizations since larger discretization stencils produce larger interface sizes. For this reason, the Petrov-Galerkin approximation due to Hughes, Franca and Mallet [**13**] has been used in the present study. Consider the prototype conservation law system in m coupled independent variables in the spatial domain $\Omega \subset \mathbf{R}^d$ with boundary surface Γ and exterior normal $\mathbf{n}(x)$

$$\mathbf{u}_{,t} + \mathbf{f}^i_{,x_i} = 0, \quad (x,t) \in \Omega \times [0, \mathbf{R}^+] \tag{2.1}$$

$$(n_i \, \mathbf{f}^i_{,\mathbf{u}})^- \, (\mathbf{u} - \mathbf{g}) = 0, \quad (x,t) \in \Gamma \times [0, \mathbf{R}^+] \tag{2.2}$$

with implied summation over repeated indices. In this equation, $\mathbf{u} \in \mathbf{R}^m$ denotes the vector of conserved variables and $\mathbf{f}^i \in \mathbf{R}^m$ the inviscid flux vectors. The vector $\mathbf{g}$ can be suitably chosen to impose characteristic data or surface flow tangency using reflection principles. The conservation law system (2.1) is assumed to possess a generalized entropy pair so that the change of variables $\mathbf{u}(\mathbf{v}) : \mathbf{R}^m \mapsto \mathbf{R}^m$

symmetrizes the system in quasi-linear form

$$\mathbf{u}_{,\mathbf{v}}\mathbf{v}_{,t} + \mathbf{f}^i_{,\mathbf{v}}\mathbf{v}_{,x_i} = 0 \tag{2.3}$$

with $\mathbf{u}_{,\mathbf{v}}$ symmetric positive definite and $\mathbf{f}^i_{,\mathbf{v}}$ symmetric. The computational domain Ω is composed of non-overlapping simplicial elements T_i, $\Omega = \cup T_i$, $T_i \cap T_j = \emptyset$, $i \neq j$. For purposes of the present study, our attention is restricted to steady-state calculations. Time derivatives are retained in the Galerkin integral so that a pseudo-time marching strategy can be used for obtaining steady-state solutions. The Galerkin least-squares method due to Hughes, Franca and Mallet [**13**] can be defined via the following variational problem with time derivatives omitted from the least-squares bilinear form: Let $\mathcal{V}^h$ denote the finite element space

$$\mathcal{V}^h = \left\{ \mathbf{w}^h | \mathbf{w}^h \in \left(C^0(\Omega)\right)^m, \mathbf{w}^h|_T \in \left(\mathcal{P}_k(T)\right)^m \right\}. \tag{2.4}$$

Find $\mathbf{v}^h \in \mathcal{V}^h$ such that for all $\mathbf{w}^h \in \mathcal{V}^h$

$$B(\mathbf{v}^h, \mathbf{w}^h)_{gal} + B(\mathbf{v}^h, \mathbf{w}^h)_{ls} + B(\mathbf{v}^h, \mathbf{w}^h)_{bc} = 0 \tag{2.5}$$

with

$$\begin{aligned} B(\mathbf{v}, \mathbf{w})_{gal} &= \int_\Omega (\mathbf{w}^T \mathbf{u}(\mathbf{v})_{,t} - \mathbf{w}^T_{,x_i} \mathbf{f}^i(\mathbf{v}))\, d\Omega \\ B(\mathbf{v}, \mathbf{w})_{ls} &= \sum_{T \in \Omega} \int_T \left(\mathbf{f}^i_{,\mathbf{v}} \mathbf{w}_{,x_i}\right)^T \boldsymbol{\tau} \left(\mathbf{f}^i_{,\mathbf{v}} \mathbf{v}_{,x_i}\right)\, d\Omega \\ B(\mathbf{v}, \mathbf{w})_{bc} &= \int_\Gamma \mathbf{w}^T\ \mathbf{h}(\mathbf{v}, \mathbf{g}; \mathbf{n})\ d\Gamma \end{aligned}$$

where

$$\mathbf{h}(\mathbf{v}_-, \mathbf{v}_+, \mathbf{n}) = \frac{1}{2}\left(\mathbf{f}(\mathbf{u}(\mathbf{v}_-); \mathbf{n}) + \mathbf{f}(\mathbf{u}(\mathbf{v}_+); \mathbf{n})\right) - \frac{1}{2}|A(\mathbf{u}(\overline{\mathbf{v}}); \mathbf{n})|(\mathbf{u}(\mathbf{v}_+) - \mathbf{u}(\mathbf{v}_-)). \tag{2.6}$$

Inserting standard C^0 polynomial spatial approximations and mass-lumping of the remaining time derivative terms, yields coupled ordinary differential equations of the form:

$$D\,\mathbf{u}_t = \mathcal{R}(\mathbf{u}), \quad \mathcal{R}(\mathbf{u}) : \mathbf{R}^n \to \mathbf{R}^n \tag{2.7}$$

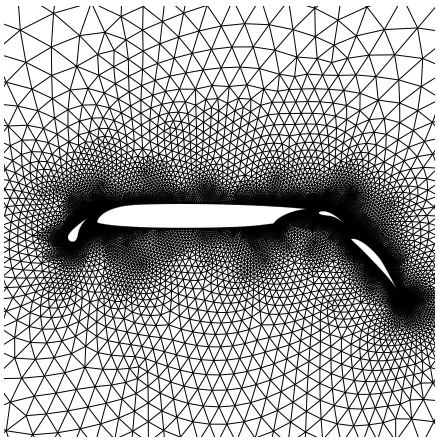
or in symmetric variables

$$D\,\mathbf{u}_{,\mathbf{v}}\mathbf{v}_t = \mathcal{R}(\mathbf{u}(\mathbf{v})), \tag{2.8}$$

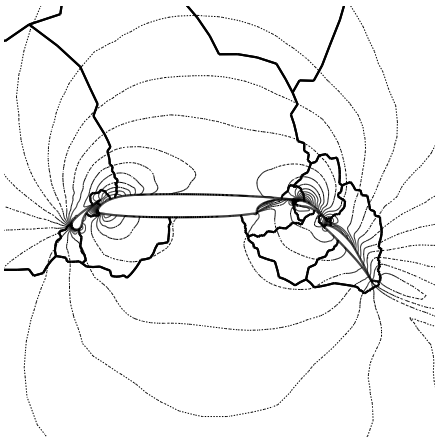
where D represents the (diagonal) lumped mass matrix. In the present study, backward Euler time integration with local time linearization is applied to Eqn. (2.7) yielding:

$$\left[\frac{1}{\Delta t} D - \left(\frac{\partial \mathcal{R}}{\partial \mathbf{u}}\right)^n\right](\mathbf{u}^{n+1} - \mathbf{u}^n) = \mathcal{R}(\mathbf{u}^n). \tag{2.9}$$

The above equation can also be viewed as a modified Newton method for solving the steady state equation $\mathcal{R}(\mathbf{u}) = 0$. For each modified Newton step, a large Jacobian matrix must be solved. In practice Δt is varied as an exponential function $\|\mathcal{R}(\mathbf{u})\|$ so that Newton's method is approached as $\|\mathcal{R}(\mathbf{u})\| \to 0$. Since each Newton iterate in (2.9) produces a linear system of the form (1.1), our attention focuses on this prototype linear form.



(a) Mesh triangulation (80,000 elements).



(b) Mach number solution contours (broken lines) and mesh partition boundaries (bold lines).

FIGURE 3. Multiple component airfoil geometry with 16 subdomain partitioning and sample solution contours ($M_\infty = .20, \alpha = 10°$).

3. Domain Partitioning

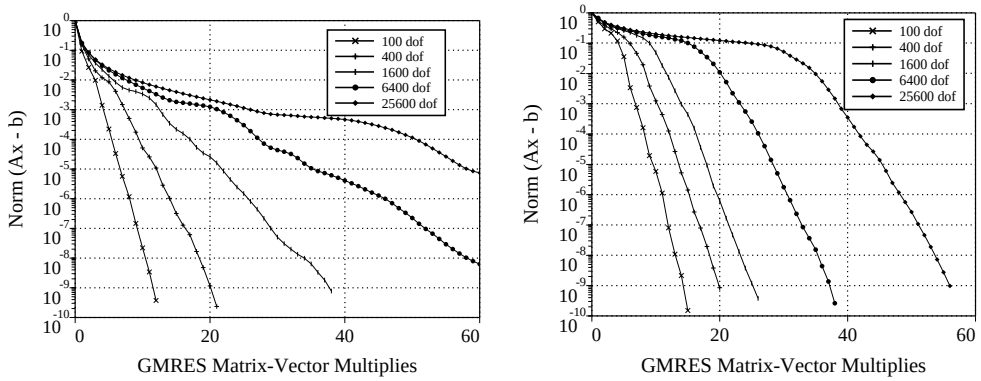
In the present study, meshes are partitioned using the multilevel k-way partitioning algorithm METIS developed by Karypis and Kumar [**14**]. Figure 3(a) shows a typical airfoil geometry and triangulated domain. To construct a non-overlapping partitioning, a dual triangulation graph has been provided to the METIS partitioning software. Figure 3(b) shows partition boundaries and sample solution contours using the spatial discretization technique described in the previous section. By partitioning the dual graph of the triangulation, the number of elements in each subdomain is automatically balanced by the METIS software. Unfortunately, a large percentage of computation in our domain-decomposition algorithm is proportional to the interface size associated with each subdomain. On general meshes containing non-uniform element densities, balancing subdomain sizes does not imply a balance of interface sizes. In fact, results shown in Sec. 6 show increased imbalance of interface sizes as meshes are partitioned into larger numbers of subdomains. This ultimately leads to poor load balancing of the parallel computation. This topic will be revisited in Sec. 6.

4. Matrix Preconditioning

Since the matrix A originating from (2.9) is assumed to be ill-conditioned, a first step is to consider the prototype linear system in right (or left) preconditioned form

$$(AP^{-1})Px - b = 0. \tag{4.1}$$

The solution is unchanged but the convergence rate of iterative methods now depends on properties of AP^{-1}. Ideally, one seeks preconditioning matrices P which



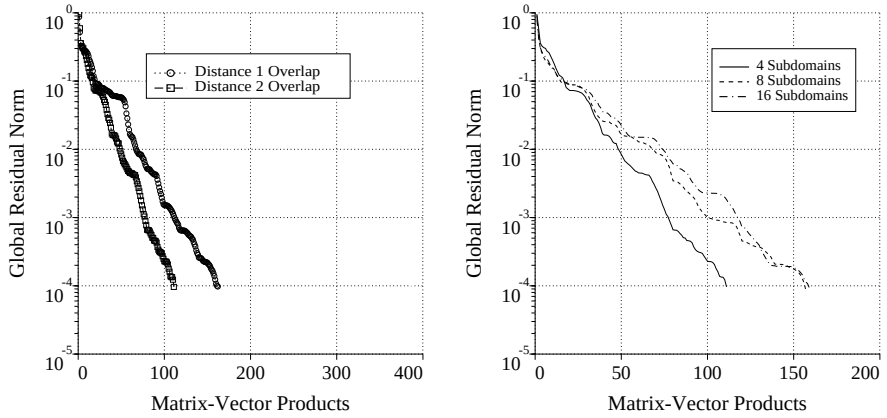
(a) Diffusion dominated problem. (b) Advection dominated problem.

FIGURE 4. Convergence dependence of ILU on the number of mesh points for diffusion and advection dominated problems using SUPG discretization and Cuthill-McKee ordering.

are easily solved and in some sense nearby A, e.g. $\kappa(AP^{-1}) = O(1)$ when A is SPD. Several candidate preconditioning matrices have been considered in this study:

4.1. ILU Factorization. A common preconditioning choice is incomplete lower-upper factorization with arbitrary fill level k, ILU[k]. Early application and analysis of ILU preconditioning is given in Evans [**11**], Stone [**22**], and Meijerink and van der Vorst [**16**]. Although the technique is algebraic and well-suited to sparse matrices, ILU-preconditioned systems are not generally scalable. For example, Dupont *et al.* [**10**] have shown that ILU[0] preconditioning does not asymptotically change the $O(h^{-2})$ condition number of the 5-point difference approximation to Laplace's equation. Figure 4 shows the convergence of ILU-preconditioned GMRES for Cuthill-McKee ordered matrix problems obtained from diffusion and advection dominated problems discretized using Galerkin and Galerkin least-squares techniques respectively with linear elements. Both problems show pronounced convergence deterioration as the number of solution unknowns (degrees of freedom) increases. Note that matrix orderings exist for discretized scalar advection equations that are vastly superior to Cuthill-McKee ordering. Unfortunately, these orderings do not generalize naturally to coupled systems of equations which do not have a single characteristic direction. Some ILU matrix ordering experiments are given in [**6**]. Keep in mind that ILU *does* recluster eigenvalues of the preconditioned matrix so that for small enough problems a noticeable improvement can often be observed when ILU preconditioning is combined with a Krylov projection sequence.

4.2. Multi-level Methods. In the past decade, multi-level approaches such as multigrid has proven to be one of the most effective techniques for solving discretizations of elliptic PDEs [**23**]. For certain classes of elliptic problems, multigrid attains optimal scalability. For hyperbolic-elliptic problems such as the steady-state Navier-Stokes equations, the success of multigrid is less convincing. For example,



(a) Effect of increasing mesh overlap.

(b) Effect of increasing number of subdomains.

FIGURE 5. Performance of GMRES with additive Schwarz preconditioning.

Ref. [**15**] presents numerical results using multigrid to solve compressible Navier-Stokes flow about a multiple-component wing geometry with asymptotic convergence rates approaching .98 (Fig. 12 in Ref. [**15**]). This is quite far from the usual convergence rates quoted for multigrid on elliptic model problems. This is not too surprising since multigrid for hyperbolic-elliptic problems is not well-understood. In addition, some multigrid algorithms require operations such as mesh coarsening which are poorly defined for general meshes (especially in 3-D) or place unattainable shape-regularity demands on mesh generation. Other techniques add new meshing constraints to existing software packages which limit the overall applicability of the software.

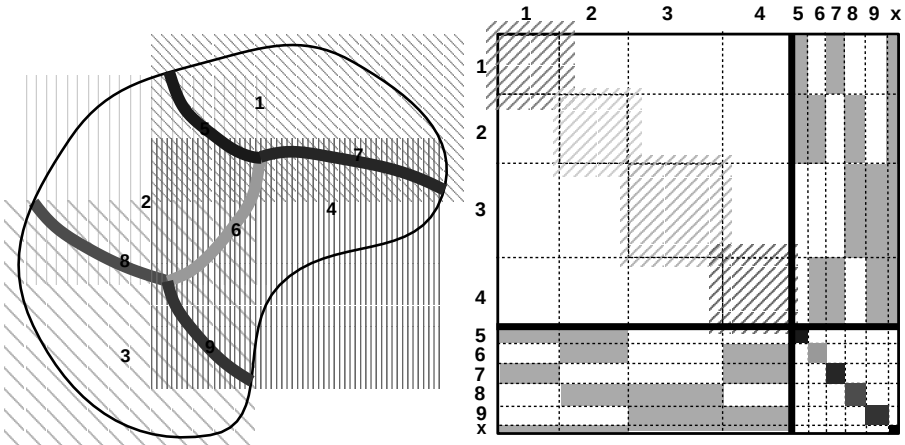
4.3. Additive Schwarz Methods. The additive Schwarz algorithm [**19**] is appealing since each subdomain solve can be performed in parallel. Unfortunately the performance of the algorithm deteriorates as the number of subdomains increases. Let H denote the characteristic size of each subdomain, δ the overlap distance, and h the mesh spacing. Dryja and Widlund [**8, 9**] give the following condition number bound for the method when used as a preconditioner for elliptic discretizations

$$\kappa(AP^{-1}) \leq CH^{-2}\left(1 + (H/\delta)^2\right) \tag{4.2}$$

where C is a constant independent of H and h. This result describes the deterioration as the number of subdomains increases (and H decreases). With some additional work this deterioration can be removed by the introduction of a global coarse subspace. Under the assumption of "generous overlap" the condition number bound [**8, 9, 4**] can be improved to

$$\kappa(AP^{-1}) \leq C\left(1 + (H/\delta)\right). \tag{4.3}$$

The addition of a coarse space approximation introduces implementation problems similar to those found in multigrid methods described above. Once again, the theory associated with additive Schwarz methods for hyperbolic PDE systems is



(a) Partitioned domain.

(b) Induced 2×2 block discretization matrix.

FIGURE 6. Domain decomposition and the corresponding block matrix.

not well-developed. Practical applications of the additive Schwarz method for the steady state calculation of hyperbolic PDE systems show similar deterioration of the method when the coarse space is omitted. Figure 5 shows the performance of the additive Schwarz algorithm used as a preconditioner for GMRES. The test matrix was taken from one step of Newton's method applied to an upwind finite volume discretization of the Euler equations at low Mach number ($M_\infty = .2$), see Barth [**1**] for further details. These calculations were performed without coarse mesh correction. As expected, the graphs show a degradation in quality with decreasing overlap and increasing number of mesh partitions.

4.4. Schur complement Algorithms. Schur complement preconditioning algorithms are a general family of algebraic techniques in non-overlapping domain-decomposition. These techniques can be interpreted as variants of the well-known substructuring method introduced by Przemieniecki [**17**] in structural analysis. When recursively applied, the method is related to the nested dissection algorithm. In the present development, we consider an arbitrary domain as illustrated in Fig. 6 that has been further decomposed into subdomains labeled $1-4$, interfaces labeled $5-9$, and cross points x. A natural 2×2 partitioning of the system is induced by permuting rows and columns of the discretization matrix so that subdomain unknowns are ordered first, interface unknowns second, and cross points ordered last

$$Ax = \begin{bmatrix} A_{\mathcal{D}\mathcal{D}} & A_{\mathcal{D}\mathcal{I}} \\ A_{\mathcal{I}\mathcal{D}} & A_{\mathcal{I}\mathcal{I}} \end{bmatrix} \begin{pmatrix} x_{\mathcal{D}} \\ x_{\mathcal{I}} \end{pmatrix} = \begin{pmatrix} f_{\mathcal{D}} \\ f_{\mathcal{I}} \end{pmatrix} \tag{4.4}$$

where $x_{\mathcal{D}}, x_{\mathcal{I}}$ denote the subdomain and interface variables respectively. The block LU factorization of A is then given by

$$A = LU = \begin{bmatrix} A_{\mathcal{D}\mathcal{D}} & 0 \\ A_{\mathcal{I}\mathcal{D}} & I \end{bmatrix} \begin{bmatrix} I & A_{\mathcal{D}\mathcal{D}}^{-1} A_{\mathcal{D}\mathcal{I}} \\ 0 & S \end{bmatrix}, \tag{4.5}$$

where

$$S = A_{\mathcal{I}\mathcal{I}} - A_{\mathcal{I}\mathcal{D}} A_{\mathcal{D}\mathcal{D}}^{-1} A_{\mathcal{D}\mathcal{I}} \tag{4.6}$$

is the Schur complement for the system. Note that $A_{\mathcal{D}\mathcal{D}}$ is block diagonal with each block associated with a subdomain matrix. Subdomains are decoupled from each other and only coupled to the interface. The subdomain decoupling property is exploited heavily in parallel implementations.

In the next section, we outline a naive parallel implementation of the "exact" factorization. This will serve as the basis for a number of simplifying approximations that will be discussed in later sections.

4.5. "Exact" Factorization. Given the domain partitioning illustrated in Fig. 6, a straightforward (but naive) parallel implementation would assign a processor to each subdomain and a single processor to the Schur complement. Let $\overline{\mathcal{I}}_i$ denote the union of interfaces surrounding $\mathcal{D}_i$. The entire solution process would then consist of the following steps:

Parallel Preprocessing:

1. Parallel computation of subdomain $A_{\mathcal{D}_i\mathcal{D}_i}$ matrix LU factors.
2. Parallel computation of Schur complement block entries associated with each subdomain $\mathcal{D}_i$

$$\Delta S_{\overline{\mathcal{I}}_i} = A_{\overline{\mathcal{I}}_i\mathcal{D}_i} A_{\mathcal{D}_i\mathcal{D}_i}^{-1} A_{\mathcal{D}_i\overline{\mathcal{I}}_i}. \tag{4.7}$$

3. Accumulation of the global Schur complement S matrix

$$S = A_{\mathcal{I}\mathcal{I}} - \sum_{i=1}^{\#subdomains} \Delta S_{\overline{\mathcal{I}}_i}. \tag{4.8}$$

Solution:

Step (1)	$u_{\mathcal{D}_i} = A_{\mathcal{D}_i\mathcal{D}_i}^{-1} b_{\mathcal{D}_i}$	(parallel)
Step (2)	$v_{\overline{\mathcal{I}}_i} = A_{\overline{\mathcal{I}}_i\mathcal{D}_i} u_{\mathcal{D}_i}$	(parallel)
Step (3)	$w_{\mathcal{I}} = b_{\mathcal{I}} - \sum_{i=1}^{\#subdomains} v_{\overline{\mathcal{I}}_i}$	(communication)
Step (4)	$x_{\mathcal{I}} = S^{-1} w_{\mathcal{I}}$	(sequential, communication)
Step (5)	$y_{\mathcal{D}_i} = A_{\mathcal{D}_i\overline{\mathcal{I}}_i} x_{\overline{\mathcal{I}}_i}$	(parallel)
Step (6)	$x_{\mathcal{D}_i} = u_{\mathcal{D}_i} - A_{\mathcal{D}_i\mathcal{D}_i}^{-1} y_{\mathcal{D}_i}$	(parallel)

This algorithm has several deficiencies. Steps 3 and 4 of the solution process are sequential and require communication between the Schur complement and subdomains. More generally, the algorithm is not scalable since the growth in size of the Schur complement with increasing number of subdomains eventually overwhelms the calculation in terms of memory, computation, and communication.

5. Iterative Schur complement Algorithms

A number of approximations have been investigated which simplify the exact factorization algorithm and address the growth in size of the Schur complement. During this investigation, our goal has been to develop algebraic techniques which can be applied to both elliptic and hyperbolic partial differential equations. These approximations include iterative (Krylov projection) subdomain and Schur complement solves, element dropping and other sparsity control strategies, localized subdomain solves in the formation of the Schur complement, and partitioning of

the interface and parallel distribution of the Schur complement matrix. Before describing each approximation and technique, we can make several observations:

Observation 1. (Ill-conditioning of Subproblems) For model elliptic problem discretizations, it is known in the two subdomain case that $\kappa(A_{\mathcal{D}_i\mathcal{D}_i}) = O((L/h)^2)$ and $\kappa(S) = O(L/h)$ where L denotes the domain size. From this perspective, both subproblems are ill-conditioned since the condition number depends on the mesh spacing parameter h. If one considers the scalability experiment, the situation changes in a subtle way. In the scalability experiment, the number of mesh points and the number of subdomains is increased such that the ratio of subdomain size to mesh spacing size H/h is held constant. The subdomain matrices for elliptic problem discretizations now exhibit a $O((H/h)^2)$ condition number so the cost associated with iteratively solving them (with or without preconditioning) is approximately constant as the problem size is increased. Therefore, this portion of the algorithm is scalable. Even so, it may be desirable to precondition the subdomain problems to reduce the overall cost. The Schur complement matrix retains (at best) the $O(L/h)$ condition number and becomes increasingly ill-conditioned as the mesh size is increased. Thus in the scalability experiment, it is ill-conditioning of the Schur complement matrix that must be controlled by adequate preconditioning, see for example Dryja, Smith and Widlund [**7**].

Observation 2. (Non-stationary Preconditioning) The use of Krylov projection methods to solve the local subdomain and Schur complement subproblems renders the global preconditioner non-stationary. Consequently, Krylov projection methods designed for non-stationary preconditioners should be used for the global problem. For this reason, FGMRES [**18**], a variant of GMRES designed for non-stationary preconditioning, has been used in the present work.

Observation 3. (Algebraic Coarse Space) The Schur complement serves as an algebraic coarse space operator since the system

$$Sx_{\mathcal{I}} = b_{\mathcal{I}} - A_{\mathcal{I}\mathcal{D}}A_{\mathcal{D}\mathcal{D}}^{-1}b_{\mathcal{D}} \tag{5.1}$$

globally couples solution unknowns on the entire interface. The rapid propagation of information to large distances is a crucial component of optimal algorithms.

5.1. ILU-GMRES Subdomain and Schur complement Solves. The first natural approximation is to replace exact inverses of the subdomain and Schur complement subproblems with an iterative Krylov projection method such as GMRES (or stabilized biconjugate gradient).

5.1.1. *Iterative Subdomain Solves.* Recall from the exact factorization algorithm that a subdomain solve is required once in the preprocessing step and twice in the solution step. This suggests replacing these three inverses with m_1, m_2, and m_3 steps of GMRES respectively. As mentioned in Observation 1, although the condition number of subdomain problems remains roughly constant in the scalability experiment, it still is beneficial to precondition subdomain problems to improve the overall efficiency of the global preconditioner. By preconditioning subdomain problems, the parameters m_1, m_2, m_3 can be kept small. This will be exploited in later approximations. Since the subdomain matrices are assumed given, it is straightforward to precondition subdomains using ILU[k]. For the GLS spatial discretization, satisfactory performance is achieved using ILU[2].

5.1.2. *Iterative Schur complement Solves.* It is possible to avoid explicitly computing the Schur complement matrix for use in Krylov projection methods by alternatively computing the action of S on a given vector p, i.e.

$$S\,p = A_{\mathcal{I}\mathcal{I}}\,p - A_{\mathcal{I}\mathcal{D}}A_{\mathcal{D}\mathcal{D}}^{-1}A_{\mathcal{D}\mathcal{I}}\,p. \tag{5.2}$$

Unfortunately S is ill-conditioned, thus some form of interface preconditioning is needed. For elliptic problems, the rapid decay of elements away from the diagonal in the Schur complement matrix [**12**] permits simple preconditioning techniques. Bramble, Pasciak, and Schatz [**3**] have shown that even the simple block Jacobi preconditioner yields a substantial improvement in condition number

$$\kappa(SP_S^{-1}) \leq CH^{-2}\left(1+\log^2(H/h)\right) \tag{5.3}$$

for C independent of h and H. For a small number of subdomains, this technique is very effective. To avoid the explicit formation of the diagonal blocks, a number of simplified approximations have been introduced over the last several years, see for examples Bjorstad [**2**] or Smith [**21**]. By introducing a further coarse space coupling of cross points to the interface, the condition number is further improved

$$\kappa(SP_S^{-1}) \leq C\left(1+\log^2(H/h)\right). \tag{5.4}$$

Unfortunately, the Schur complement associated with advection dominated discretizations may not exhibit the rapid element decay found in the elliptic case. This can occur when characteristic trajectories of the advection equation traverse a subdomain from one interface edge to another. Consequently, the Schur complement is not well-preconditioned by elliptic-like preconditioners that use the action of local problems. A more basic strategy has been developed in the present work whereby elements of the Schur complement are *explicitly computed.* Once the elements have been computed, ILU factorization is used to precondition the Schur complement iterative solution. In principle, ILU factorization with a suitable reordering of unknowns can compute the long distance interactions associated with simple advection fields corresponding to entrance/exit-like flows. For general advection fields, it remains a topic of current research to find reordering algorithms suitable for ILU factorization. The situation is further complicated for coupled systems of hyperbolic equations (even in two independent variables) where multiple characteristic directions and/or Cauchy-Riemann systems can be produced. At the present time, Cuthill-McKee ordering has been used on all matrices although improved reordering algorithms are currently under development.

In the present implementation, each subdomain processor computes (in parallel) and stores portions of the Schur complement matrix

$$\Delta S_{\overline{\mathcal{I}}_i} = A_{\overline{\mathcal{I}}_i\mathcal{D}_i}A_{\mathcal{D}_i\mathcal{D}_i}^{-1}A_{\mathcal{D}_i\overline{\mathcal{I}}_i}. \tag{5.5}$$

To gain improved parallel scalability, the interface edges and cross points are partitioned into a smaller number of generic "subinterfaces". This subinterface partitioning is accomplished by assigning a supernode to each interface edge separating two subdomains, forming the graph of the Schur complement matrix in terms of these supernodes, and applying the METIS partitioning software to this graph. Let $\overline{\overline{\mathcal{I}}}_j$ denote the j-th subinterface such that $\mathcal{I} = \cup_j \overline{\overline{\mathcal{I}}}_j$. Computation of the action of the Schur complement matrix on a vector p needed in Schur complement solves

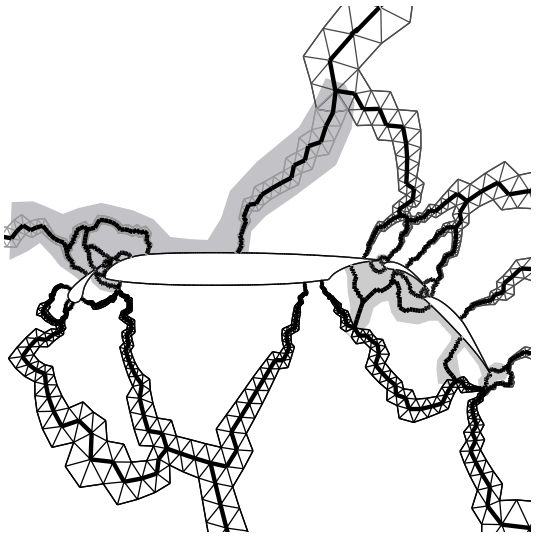


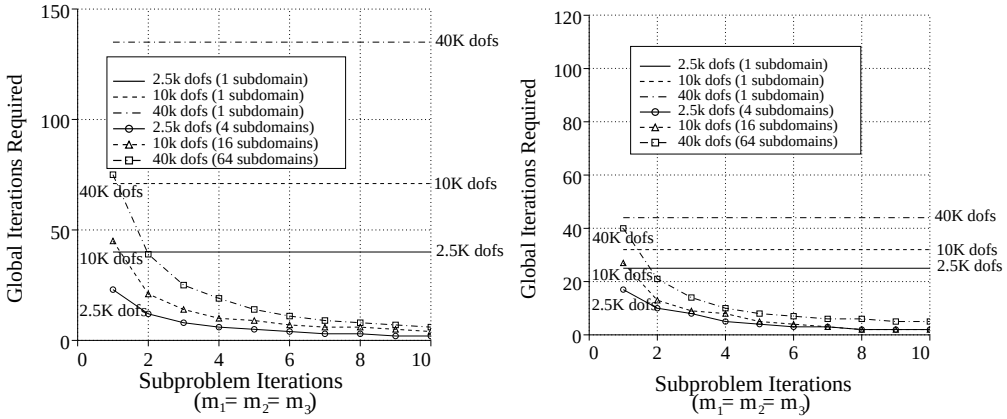
FIGURE 7. Interface (bold lines) decomposed into 4 subinterfaces indicated by alternating shaded regions.

now takes the (highly parallel) form

$$Sp = \sum_{j=1}^{\#subinterfaces} A_{\overline{\overline{\mathcal{I}}}_j \overline{\overline{\mathcal{I}}}_j}\, p(\overline{\overline{\mathcal{I}}}_j) - \sum_{i=1}^{\#subdomains} \Delta S_{\overline{\mathcal{I}}_i}\, p(\overline{\mathcal{I}}_i). \tag{5.6}$$

Using this formula it is straightforward to compute the action of S on a vector p to any required accuracy by choosing the subdomain iteration parameter m_i large enough. Figure 7 shows an interface and the immediate neighboring mesh that has been decomposed into 4 smaller subinterface partitions for a 32 subdomain partitioning. By choosing the number of subinterface partitions proportional to the square root of the number of 2-D subdomains and assigning a processor to each, the number of solution unknowns associated with each subinterface is held approximately constant in the scalability experiment. Note that the use of iterative subdomain solves renders both Eqns. (5.2) and (5.6) approximate.

In our investigation, the Schur complement is preconditioned using ILU factorization. This is not a straightforward task for two reasons: (1) portions of the Schur complement are distributed among subdomain processors, (2) the interface itself has been distributed among several subinterface processors. In the next section, a block element dropping strategy is proposed for gathering portions of the Schur complement together on subinterface processors for use in ILU preconditioning the Schur complement solve. Thus, a block Jacobi preconditioner is constructed for the Schur complement which is more powerful than the Bramble, Pasciak, and Schatz (BPS) form (without coarse space correction) since the blocks now correspond to larger subinterfaces rather than the smaller interface edges. Formally, BPS preconditioning without coarse space correction can be obtained for 2D elliptic discretizations by dropping additional terms in our Schur complement matrix approximation and ordering unknowns along interface edges so that the ILU factorization of the tridiagonal-like system for each interface edge becomes exact.



(a) Diffusion dominated problem. $u_{xx} + u_{yy} = 0$.

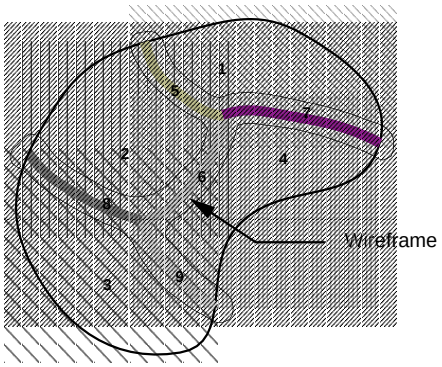
(b) Advection dominated problem. $u_x + u_y = 0$.

FIGURE 8. Effect of the subproblem iteration parameters m_i on the global FGMRES convergence, $m_1 = m_2 = m_3$ for meshes containing 2500, 10000, and 40000 solution unknowns.

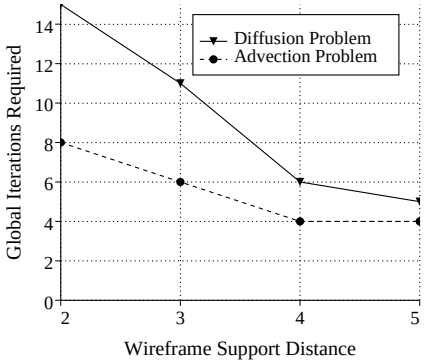
5.1.3. *Block Element Dropping.* In our implementation, portions of the Schur complement residing on subdomain processors are gathered together on subinterface processors for use in ILU preconditioning of the Schur complement solve. In assembling a Schur complement matrix approximation on each subinterface processor, certain matrix elements are neglected:

1. All elements that couple subinterfaces are ignored. This yields a block Jacobi approximation for subinterfaces.
2. All elements with matrix entry location that exceeds a user specified graph distance from the diagonal as measured on the triangulation graph are ignored. Recall that the Schur complement matrix can be very dense. The graph distance criteria is motivated by the rapid decay of elements away from the matrix diagonal for elliptic problems. In all subsequent calculations, a graph distance threshold of 2 has been chosen for block element dropping.

Figures 8(a) and 8(b) show calculations performed with the present non-overlapping domain-decomposition preconditioner for diffusion and advection problems. These figures graph the number of global FGMRES iterations needed to solve the discretization matrix problem to 10^{-6} accuracy tolerance as a function of the number of subproblem iterations. In this example, all the subproblem iteration parameters have been set equal to each other ($m_1 = m_2 = m_3$). The horizontal lines show poor scalability of single domain ILU-FGMRES on meshes containing 2500, 10000, and 40000 solution unknowns. The remaining curves show the behavior of the Schur complement preconditioned FGMRES on 4, 16, and 64 subdomain meshes. Satisfactory scalability for very small values (5 or 6) of the subproblem iteration parameter m_i is clearly observed.



(a) Wireframe region surrounding interface.



(b) Effect of wireframe support on preconditioner performance for diffusion $u_{xx} + u_{yy} = 0$ and advection $u_x + u_y = 0$ problems.

FIGURE 9. Wireframe region surrounding interface and preconditioner performance results for a fixed mesh size (1600 vertices) and 16 subdomain partitioning.

5.1.4. *Wireframe Approximation.* A major cost in the explicit construction of the Schur complement is the matrix-matrix product

$$A^{-1}_{\mathcal{D}_i\mathcal{D}_i} A_{\mathcal{D}_i\overline{\mathcal{I}}_i}. \tag{5.7}$$

Since the subdomain inverse is computed iteratively using ILU-GMRES iteration, forming (5.7) is equivalent to solving a multiple right-hand sides system with each right-hand side vector corresponding to a column of $A_{\mathcal{D}_i\overline{\mathcal{I}}_i}$. The number of columns of $A_{\mathcal{D}_i\overline{\mathcal{I}}_i}$ is precisely the number of solution unknowns located on the interface surrounding a subdomain. This computational cost can be quite large. Numerical experiments with Krylov projection methods designed for multiple right-hand side systems [**20**] showed only marginal improvement owing to the fact that the columns are essentially independent. In the following paragraphs, "wireframe" and "supersparse" approximations are introduced to reduce the cost in forming the Schur complement matrix.

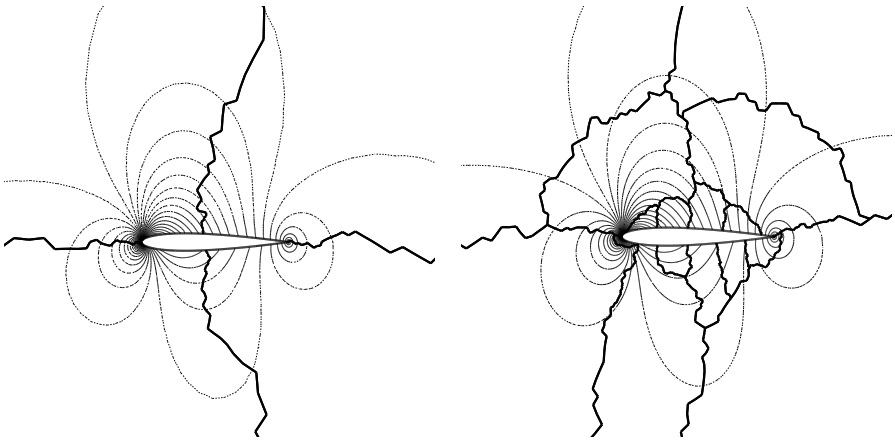
The wireframe approximation idea [**5**] is motivated from standard elliptic domain-decomposition theory by the rapid decay of elements in S with graph distance from the diagonal. Consider constructing a relatively thin *wireframe* region surrounding the interface as shown in Fig. 9(a). In forming the Eqn. (5.7) expression, subdomain solves are performed using the much smaller wireframe subdomains. In matrix terms, a principal submatrix of A, corresponding to the variables within the wireframe, is used to compute the (approximate) Schur complement of the interface variables. It is known from domain-decomposition theory that the exact Schur complement of the wireframe region is spectrally equivalent to the Schur complement of the whole domain. This wireframe approximation leads to a substantial savings in the computation of the Schur complement matrix. Note that the full subdomain matrices are used everywhere else in the Schur complement algorithm. The wireframe technique introduces a new adjustable parameter into the preconditioner which represents the width of the wireframe. For simplicity, this width is

specified in terms of graph distance on the mesh triangulation. Figure 9(b) demonstrates the performance of this approximation by graphing the total number of preconditioned FGMRES iterations required to solve the global matrix problem to a 10^{-6} accuracy tolerance while varying the width of the wireframe. As expected, the quality of the preconditioner improves rapidly with increasing wireframe width with full subdomain-like results obtained using modest wireframe widths. As a consequence of the wireframe construction, the time taken form the Schur complement has dropped by approximately 50%.

5.1.5. *Supersparse Matrix-Vector Operations* . It is possible to introduce further approximations which improve upon the overall efficiency in forming the Schur complement matrix. One simple idea is to exploit the extreme sparsity in columns of $A_{\mathcal{D}_i\overline{\mathcal{I}_i}}$ or equivalently the sparsity in the right-hand sides produced from $A^{-1}_{\mathcal{D}_i\mathcal{D}_i}A_{\mathcal{D}_i\overline{\mathcal{I}_i}}$ needed in the formation of the Schur complement. Observe that m steps of GMRES generates a small sequence of Krylov subspace vectors $[p, A\,p, A^2\,p, \ldots, A^m\,p]$ where p is a right-hand side vector. Consequently for small m, if both A and p are sparse then the sequence of matrix-vector products will be relatively sparse. Standard sparse matrix-vector product subroutines utilize the matrix in sparse storage format and the vector in dense storage format. In the present application, the vectors contain only a few non-zero entries so that standard sparse matrix-vector products waste many arithmetic operations. For this reason, a "supersparse" software library have been developed to take advantage of the sparsity in matrices as well as in vectors by storing both in compressed form. Unfortunately, when GMRES is preconditioned using ILU factorization, the Krylov sequence becomes $[p, A\,P^{-1}\,p, (A\,P^{-1})^2\,p, \ldots, (A\,P^{-1})^m\,p]$. Since the inverse of the ILU approximate factors $\widetilde{L}$ and $\widetilde{U}$ can be dense, the first application of ILU preconditioning produces a dense Krylov vector result. All subsequent Krylov vectors can become dense as well. To prevent this densification of vectors using ILU preconditioning, a fill-level-like strategy has been incorporated into the ILU *backsolve* step. Consider the ILU preconditioning problem, $\widetilde{L}\,\widetilde{U}\,r = b$. This system is conventionally solved by a lower triangular backsolve, $w = \widetilde{L}^{-1}b$, followed by a upper triangular backsolve $r = \widetilde{U}^{-1}w$. In our supersparse strategy, sparsity is controlled by imposing a non-zero fill pattern for the vectors w and r during lower and upper backsolves. The backsolve fill patterns are most easily specified in terms fill-level distance, i.e. graph distance from existing nonzeros of the right-hand side vector in which new fill in the resultant vector is allowed to occur. This idea is motivated from the element decay phenomena observed for elliptic problems. Table 1 shows the performance benefits of using supersparse computations together with backsolve fill-level specification for a 2-D test problem consisting of Euler flow past a multi-element airfoil geometry partitioned into 4 subdomains with 1600 mesh vertices in each subdomain. Computations were performed on the IBM SP2 parallel computer using MPI message passing protocol. Various values of backsolve fill-level distance were chosen while monitoring the number of global GMRES iterations needed to solve the matrix problem and the time taken to form the Schur complement preconditioner. Results for this problem indicate preconditioning performance comparable to exact ILU backsolves using backsolve fill-level distances of only 2 or 3 with a 60-70% reduction in cost.

Backsolve Fill-Level Distance k	Global GMRES Iterations	Time(k)/Time(∞)
0	26	0.325
1	22	0.313
2	21	0.337
3	20	0.362
4	20	0.392
∞	20	1.000

TABLE 1. Performance of the Schur complement preconditioner with supersparse arithmetic for a 2-D test problem consisting of Euler flow past a multi-element airfoil geometry partitioned into 4 subdomains with 1600 mesh vertices in each subdomain.



(a) Mach contours (4 subdomains, 20K elements).

(b) Mach contours (16 subdomains, 80K elements).

FIGURE 10. Mach number contours and mesh partition boundaries for NACA0012 airfoil geometry.

6. Numerical Results on the IBM SP2

In the remaining paragraphs, we assess the performance of the Schur complement preconditioned FGMRES in solving linear matrix problems associated an approximate Newton method for the nonlinear discretized compressible Euler equations. All calculations were performed on an IBM SP2 parallel computer using MPI message passing protocol. A scalability experiment was performed on meshes containing 4/1, 16/2, and 64/4 subdomains/subinterfaces with each subdomain containing 5000 mesh elements. Figures 10(a) and 10(b) show mesh partitionings and sample Mach number solution contours for subsonic ($M_\infty = .20, \alpha = 2.0°$) flow over the airfoil geometry. The flow field was computed using the stabilized GLS discretization and approximate Newton method described in Sec. 2. Figure 11 graphs the convergence of the approximate Newton method for the 16 subdomain test problem. Each approximate Newton iterate shown in Fig. 11 requires the

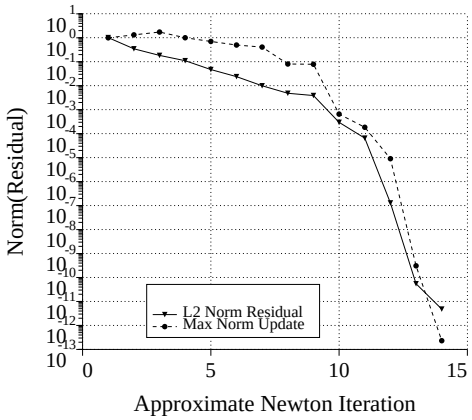
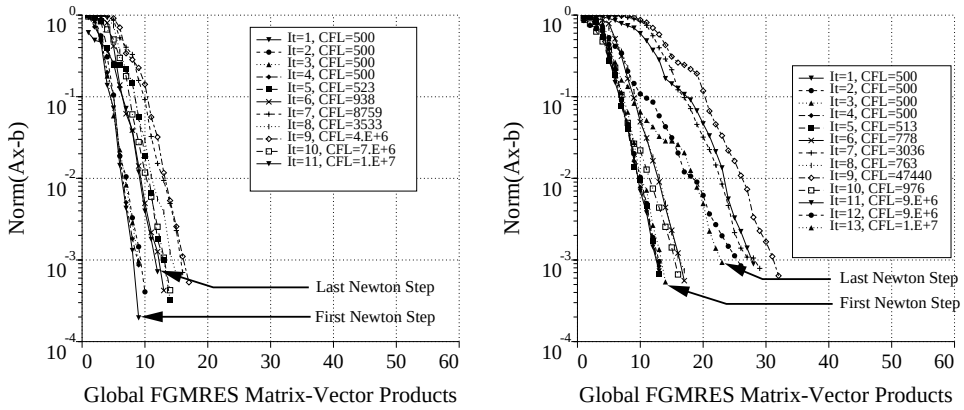


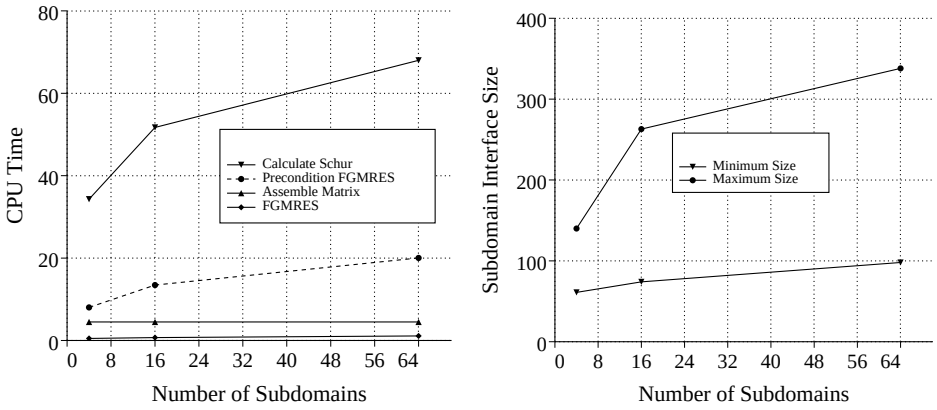
FIGURE 11. Nonlinear convergence behavior of the approximate Newton method for subsonic airfoil flow.



(a) 4 subdomains (20K elements). (b) 16 subdomains (80K elements).

FIGURE 12. FGMRES convergence history for each Newton step.

solution of a linear matrix system which has been solved using the Schur complement preconditioned FGMRES algorithm. Figure 12 graphs the convergence of the FGMRES algorithm for each matrix from the 4 and 16 subdomain test problems. These calculations were performed using ILU[2] and $m_1 = m_2 = m_3 = 5$ iterations on subproblems with supersparse distance equal to 5. The 4 subdomain mesh with 20000 total elements produces matrices that are easily solved in 9-17 global FGMRES iterations. Calculations corresponding to the largest CFL numbers are close approximations to exact Newton iterates. As is typically observed by these methods, the final few Newton iterates are solved more easily than matrices produced during earlier iterates. The most difficult matrix problem required 17 FGMRES iterations and the final Newton iterate required only 12 FGMRES iterations. The 16 subdomain mesh containing 80000 total elements produces matrices that are solved in 12-32 global FGMRES. Due to the nonlinearity in the spatial discretization, several approximate Newton iterates were relatively difficult to solve, requiring over



(a) Raw timing breakdown. (b) Interface Imbalance and growth.

FIGURE 13. Raw IBM SP2 timing breakdown and the effect of increased number of subdomains on smallest and largest interface sizes.

30 FGMRES iterations. As nonlinear convergence is obtained the matrix problems become less demanding. In this case, the final Newton iterate matrix required 22 FGMRES iterations. This iteration degradation from the 4 subdomain case can be reduced by increasing the subproblem iteration parameters m_1, m_2, m_3 but the overall computation time is increased. In the remaining timing graphs, we have sampled timings from 15 FGMRES iterations taken from the final Newton iterate on each mesh. For example, Fig. 13(a) gives a raw timing breakdown for several of the major calculations in the overall solver: calculation of the Schur complement matrix, preconditioning FGMRES with the Schur complement algorithm, matrix element computation and assembly, and FGMRES solve. Results are plotted on each of the meshes containing 4, 16, and 64 subdomains with 5000 elements per subdomain. Since the number of elements in each subdomain is held constant, the time taken to assemble the matrix is also constant. Observe that in our implementation the time to form and apply the Schur complement preconditioner currently dominates the calculation. Although the growth observed in these timings with increasing numbers of subdomains comes from several sources, the dominate effect comes from a very simple source: *the maximum interface size growth associated with subdomains.* This has a devastating impact on the parallel performance since at the Schur complement synchronization point all processors must wait for subdomains working on the largest interfaces to finish. Figure 13(b) plots this growth in maximum interface size as a function of number of subdomains in our scalability experiment. Although the number of elements in each subdomain has been held constant in this experiment, the largest interface associated with any subdomain has more than doubled. This essentially translates into a doubling in time to form the Schur complement matrix. This doubling in time is clearly observed in the raw timing breakdown in Fig. 13(a). At this point in time, we known of no partitioning method that actively addresses controlling the maximum interface size associated with subdomains. We suspect that other non-overlapping methods are sensitive to this effect as well.

7. Concluding Remarks

Experience with our non-overlapping domain-decomposition method with an algebraically generated coarse problem shows that we can successfully trade off some of the robustness of the exact Schur complement method for increased efficiency by making appropriately designed approximations. In particular, the localized wireframe approximation and the supersparse matrix-vector operations together result in reduced cost without significantly degrading the overall convergence rate.

It remains an outstanding problem to partition domains such that the maximum interface size does grow with increased number of subdomains and mesh size. In addition, it may be cost effective to combine this technique with multigrid or multiple-grid techniques to improve the robustness of Newton's method.

References

[1] T. J. Barth, *Parallel CFD algorithms on unstructured meshes*, Tech. Report AGARD R-807, Advisory Group for Aeronautical Research and Development, 1995, Special course on parallel computing in CFD.

[2] P. Bjorstad and O. B. Widlund, *Solving elliptic problems on regions partitioned into substructures*, SIAM J. Numer. Anal. **23** (1986), no. 6, 1093–1120.

[3] J. H. Bramble, J. E. Pasciak, and A. H. Schatz, *The construction of preconditioners for elliptic problems by substructuring, I*, Math. Comp. **47** (1986), no. 6, 103–134.

[4] T. Chan and J. Zou, *Additive schwarz domain decomposition methods for elliptic problems on unstructured meshes*, Tech. Report CAM 93-40, UCLA Department of Mathematics, December 1993.

[5] T. F. Chan and T. Mathew, *Domain decomposition algorithms*, Acta Numerica (1994), 61–143.

[6] D. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Toward a cost effective ilu preconditioner with high level fill*, BIT **32** (1992), 442–463.

[7] M. Dryja, B. F. Smith, and O. B. Widlund, *Schwarz analysis of iterative substructuring algorithms for elliptic problems in three dimensions*, SIAM J. Numer. Anal. **31** (1994), 1662–1694.

[8] M. Dryja and O.B. Widlund, *Some domain decomposition algorithms for elliptic problems*, Iterative Methods for Large Linear Systems (L. Hayes and D. Kincaid, eds.), 1989, pp. 273–291.

[9] M. Dryja and O.B. Widlund, *Additive schwarz methods for elliptic finite element problems in three dimensions*, Fifth Conference on Domain Decomposition Methods for Partial Differential Equations (T. F. Chan, D.E. Keyes, G.A. Meurant, J.S. Scroggs, and R.G. Voit, eds.), 1992.

[10] T. Dupont, R. Kendall, and H. Rachford, *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal. **5** (1968), 558–573.

[11] D. J. Evans, *The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices*, J. Inst. Maths. Applics. **4** (1968), 295–314.

[12] G. Golub and D. Mayers, *The use of preconditioning over irregular regions*, Comput. Meth. Appl. Mech. Eng. **6** (1984), 223–234.

[13] T. J. R. Hughes, L. P. Franca, and M. Mallet, *A new finite element formulation for CFD: I. symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics*, Comput. Meth. Appl. Mech. Eng. **54** (1986), 223–234.

[14] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, Tech. Report Report 95-064, U. of Minn. Computer Science Department, 1995.

[15] D. J. Mavriplis, *A three-dimensional multigrid Reynolds-averaged Navier-Stokes solver for unstructured meshes*, Tech. Report Report 94-29, ICASE, 1994.

[16] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp. **34** (1977), 148–162.

[17] J. S. Przemieniecki, *Matrix structural analysis of substructures*, Am. Inst. Aero. Astro. J. **1** (1963), 138–147.

[18] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Stat. Comp. **14** (1993), no. 2, 461–469.
[19] H. A. Schwarz, *Uber einige abbildungensaufgaben*, J. Reine Angew. Math. **70** (1869), 105–120.
[20] V. Simoncini and E. Gallopoulos, *An iterative method for nonsymmetric systems with multiple right-hand sides*, SIAM J. Sci. Comput. **16** (1995), no. 4, 917–933.
[21] B. Smith, P. Bjorstad, and W. Gropp, *Domain decomposition: parallel multilevel methods for elliptic partial differential equations*, Cambridge University Press, 1996.
[22] H. Stone, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal. **5** (1968), 530–558.
[23] J. Xu, *An introduction to multilevel methods*, Lecture notes: VIIth EPSRC numerical analysis summer school, 1997.

NASA Ames Research Center, Information Sciences Directorate, Mail Stop T27A-1, Moffett Field, CA 94035
E-mail address: barth@nas.nasa.gov

Department of Mathematics, University of California at Los Angeles, Los Angeles, CA 90095-1555
E-mail address: chan@math.ucla.edu

Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
E-mail address: wptang@bz.uwaterloo.ca