ICASE

PARALLEL NEWTON-KRYLOV-SCHWARZ ALGORITHMS FOR THE TRANSONIC FULL POTENTIAL EQUATION

Xiao-Chuan Cai, William D. Gropp, David E. Keyes, Robin G. Melvin, and David P. Young

NASA Contract No. NAS1-19480
May 1996

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001

Operated by Universities Space Research Association

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23681-0001
Parallel Newton-Krylov-Schwarz Algorithms
for the Transonic Full Potential Equation

Xiao-Chuan Cai
Department of Computer Science
University of Colorado at Boulder, Boulder, CO 80309
cai@cs.colorado.edu

William D. Gropp
Mathematics and Computer Science Division
Argonne National Laboratory, Argonne, IL 60439
gropp@mcs.anl.gov

David E. Keyes
Department of Computer Science
Old Dominion University, Norfolk, VA 23529-0162
keyes@icase.edu

Robin G. Melvin and David P. Young
The Boeing Company
Seattle, WA 98124
rgm4152@cfdd53.cfd.ca.boeing.com
dpy6629@cfdd51.cfd.ca.boeing.com

ABSTRACT

We study parallel two-level overlapping Schwarz algorithms for solving nonlinear finite element
problems, in particular, for the full potential equation of aerodynamics discretized in two dimensions
with bilinear elements. The overall algorithm, Newton-Krylov-Schwarz (NKS), employs an inexact
finite-difference Newton method and a Krylov space iterative method, with a two-level overlapping
Schwarz method as a preconditioner. We demonstrate that NKS, combined with a density upwinding
continuation strategy for problems with weak shocks, is robust and economical for this class of
mixed elliptic-hyperbolic nonlinear partial differential equations, with proper specification of several
parameters. We study upwinding parameters, inner convergence tolerance, coarse grid density,
subdomain overlap, and the level of fill-in in the incomplete factorization, and report their effect on
numerical convergence rate, overall execution time, and parallel efficiency on a distributed-memory
parallel computer.
1 Introduction

In the past few years domain decomposition methods for linear partial differential equations, including overlapping Schwarz methods [9, 12, 13, 37], have graduated from theory into practice in many applications [17, 27, 28, 34]. In this paper, we study several aspects of the parallel implementation of a Krylov-Schwarz domain decomposition algorithm for the finite element solution of the nonlinear full potential equation of aerodynamics, extending our model studies of linear convection-diffusion problems in [5] and of linear aerodynamic design optimization problems in [33]. Newton-Krylov methods [2, 3, 14, 15, 39] are potentially well suited and increasingly popular for the implicit solution of nonlinear problems whenever it is expensive to compute or store a true Jacobian. We employ a combined algorithm, called Newton-Krylov-Schwarz, and focus on the interplay of the three nested components of the algorithm, since the amount of work done in each component affects and is affected by the work done in the others.

Newton-Krylov-Schwarz is a general purpose parallel solver for nonlinear partial differential equations and has been applied to complex multicomponent systems of compressible and reacting flows in, e.g., [6, 7, 29]. This paper is concerned with the simpler scalar problem of the full potential equation, which describes inviscid, irrotational, isentropic compressible flow. Though the full potential model is highly idealized, it remains the model of choice of external aerodynamic designers to date, because codes based thereupon offer reasonable turnaround times and in many cases high accuracy compared to state-of-the-art Navier-Stokes solvers. Though derived under the condition of isentropy, the full potential model remains useful in flows with weak shocks, with pre-shock Mach numbers of about 1.2 or less. It can also be extended by boundary layer patching to incorporate viscous effects, by a branch cut to accommodate lift, and by source terms to simulate powered engines. In engineering practice, accurately modeling such nonideal effects in complex geometries accounts for almost all of the lines of code, but the solution of the resulting discrete equations accounts for the majority of the execution time. The lower per-cell storage and computational requirements of the potential model allow the use of grids dense enough to achieve low truncation error levels for complex geometries. The full potential equation also avoids the spurious entropy generation near stagnation often associated with Euler and Navier-Stokes codes for industrial complex geometries of interest. We justify the simply coded examples in this paper by our focus on a solution algorithm that should not require any changes other than greater irregularity in its sparse data structures to be useful in more practical settings.

With Newton’s method as the outer iteration, a highly nonsymmetric and/or indefinite large, sparse Jacobian equation needs to be solved at every iteration to a certain accuracy, which is often progressively tightened in response to a falling nonlinear residual norm. The most popular family of preconditioners for large sparse Jacobians on structured or unstructured grids, incomplete factorization, is difficult to parallelize efficiently flop-for-flop in its global form. In our approach, the ILU-preconditioner for the Newton correction equations is replaced by a multi-level overlapping Schwarz preconditioner. The latter is not only scalably parallelizable (up to available granularities), but also possesses an asymptotically optimal mesh- and granularity-independent convergence rate for elliptically dominated problems. Our two-level overlapping additive Schwarz algorithm uses a non-nested coarse space. Subdomain granularity, quality of subdomain solves, coarse grid density, strategy for
coarse grid solution, and inner iteration termination criteria are important factors in overall performance. We report numerical experiments on an IBM SP2 with up to 32 processors.

The outline of this paper is as follows. In §2, we briefly derive the form of the full potential equation that serves as the point of departure for the numerics. The finite element discretization and the construction of an approximate Jacobian for the full potential equation are discussed in §3. §4 is devoted to the description of the basic components of the NKS algorithm. Several parallel implementation issues are explained in §5. Numerical results are summarized in §6. Finally, we offer some general remarks on the use of NKS algorithms in §7.

2 The full potential problem

For completeness, we summarize the derivation and assumptions of the full potential equation of aerodynamics. For a more thorough development, see [23].

The equation of mass conservation in a steady state fluid flow can be written in divergence form,

\[ \nabla \cdot (\rho v) = 0, \]

where \( v = (v_1, v_2)^T \) is the velocity and \( \rho \) is the local density, respectively. We assume that the flow is irrotational, which implies that there exists a velocity potential \( \Phi \) such that \( v = \nabla \Phi \). Furthermore, the relation \( \frac{p}{\rho^2} = \text{const.} \) holds for isentropic flow of a perfect gas. With the above assumptions, we can integrate the inviscid momentum equations and obtain Bernoulli’s equation

\[ \frac{q^2}{2} + \frac{a^2}{\gamma - 1} = \text{const.}, \]

where \( q = (v_1^2 + v_2^2)^{1/2} = \|\nabla \Phi\|_2 \) is the local flow speed. The sound speed \( a \) is defined by \( a^2 = dp/d\rho \), where \( p \) is the local static pressure. By means of the above relations, the five unknown fields \( v_1, v_2, p, a, \) and \( \rho \) can be eliminated in favor of a single unknown function \( \Phi \), which solves the full potential equation:

\[ \nabla \cdot (\rho (\Phi) \nabla \Phi) = 0. \]

Two forms of this equation are standard in the literature, depending upon whether the density is referenced to a uniform freestream (at \( c_\infty \)) or to a stagnation point condition. We derive the freestream version as follows. From Bernoulli’s equation (2),

\[ \frac{q^2}{2} + \frac{a^2}{\gamma - 1} = \frac{q_\infty^2}{2} + \frac{a_\infty^2}{\gamma - 1}, \]

we have that

\[ \frac{a^2}{a_\infty^2} = 1 + \frac{(\gamma - 1)(q_\infty^2 - q^2)}{2a_\infty^2} = 1 + \frac{\gamma - 1}{2} M_\infty^2 \left( 1 - \frac{q^2}{q_\infty^2} \right), \]

where \( M = q/a \) is the Mach number, and \( M_\infty \) is the freestream Mach number. From the definition of the sound speed and the pressure-density relation, we obtain

\[ a^2 = \frac{d(c\rho^\gamma)}{d\rho} = c\gamma \rho^{\gamma - 1}, \]
or equivalently, \((\frac{a}{a_\infty})^2 = (\frac{\rho}{\rho_\infty})^\gamma - 1\). Therefore,

\[
\rho(\Phi) = \rho_\infty \left(1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{\|\nabla \Phi\|^2}{q_\infty^2}) \right)^{1/(\gamma - 1)}.
\] (7)

Observe that while the density is positive in regions of validity, (3) may be locally hyperbolic.

Equation (3) requires boundary conditions. In this paper, we consider only subsonic farfield boundaries. Since our emphasis is on the performance of the Schwarz preconditioning, we study a symmetric nonlifting case, thus avoiding consideration of the Kutta-Joukowsky boundary condition. To keep the geometry of the domain trivial, we use a classical transpiration boundary condition on a slit to represent the airfoil. Transpiration refers to a continuously parameterized injection and removal of fluid along a portion of the boundary to create a recirculation pocket with a bounding streamline attached to the domain boundary at both ends, over which the flow of interest passes inviscidly. Transpiration is implemented as an inhomogeneous Neumann condition. A theoretical discussion of the use of transpiration boundary conditions to model displaced surfaces can be found in [25]. For the farfield boundary condition, we use Dirichlet values of the potential upstream. More sophisticated farfield conditions are possible, and are required in the case of a lifting airfoil, but these conditions are sufficient for excellent agreement of our numerical results with standard nonlifting solutions.

3 Finite element approximation

Following Boeing's TRANAIR code [42], we employ a finite element formulation of the two-dimensional full potential equation using bilinear elements. The existence, uniqueness, and regularity of the solution are not central to this paper, but have been discussed in the papers [31, 35] and references therein. Related finite element approaches for this class of full potential equations can also be found in [1, 16]. A finite volume scheme was given in [30] and a mortar element-based domain decomposition scheme was recently parallelized to high efficiency in [26].

3.1 Basic finite element scheme

The finite element problem is formulated in terms of the weak form

\[
a(\Phi, v) = \int_\Omega \rho(\Phi) \nabla \Phi \cdot \nabla v \, d\Omega.
\]

We use bilinear elements on a rectangular partition of \(\Omega\) denoted by \(\Omega_h = \{r_i, i = 1, \ldots, M_h\}\). Let \(\{\phi_i(x, y)\}\) be the usual nodal basis functions. The numerical solution we seek has the form

\[
\Phi(x, y) = \sum \Phi_i \phi_i(x, y),
\]

and satisfies the following nonlinear algebraic equations

\[
\sum_{i=1}^{M_h} \rho(\Phi(x_i^+, y_i^+)) \int_{\Gamma_i} \nabla \Phi \nabla v \, d\Omega = 0,
\] (8)
for all \( v \) in the test function space. Here \((x^*_i, y^*_i)\) is the center point of the rectangle \( \tau_i \).

To simplify and speed up numerical integration, we introduce certain approximations when dealing with some of the nonlinear forms. The way that we treat the local nonlinear numerical integration in (8) is like that in [42]. Let us define a system of nonlinear equations

\[
F(\Phi) = \begin{pmatrix}
F_1(\Phi_1, \ldots, \Phi_N, \rho(\Phi_1, \ldots, \Phi_N)) \\
\vdots \\
F_N(\Phi_1, \ldots, \Phi_N, \rho(\Phi_1, \ldots, \Phi_N))
\end{pmatrix} = 0,
\]

where

\[
F_i(\Phi, \rho(\Phi)) = \sum_{\tau \in \Omega_h} \rho(\Phi(x^*_\tau, y^*_\tau)) \int_{\tau} \nabla \Phi \nabla \phi_i d\Omega,
\]

and \((x^*_\tau, y^*_\tau)\) is the center point of the element \( \tau \). We construct the Jacobian matrix \( J = \{J_{ij}\} \) of the nonlinear system \( F = 0 \), approximately, as follows. For each pair of indices \( i, j \), we define

\[
\frac{\partial F_i(\Phi)}{\partial \Phi_j} = \sum_{\tau \in \Omega_h} \int_{\tau} \rho(\Phi)(\nabla \phi_j \cdot \nabla \phi_i) d\Omega +
\]

\[
\sum_{\tau \in \Omega_h} \int_{\tau} \frac{d\rho}{ds} \frac{\partial}{\partial \Phi_j} (\|\nabla \Phi\|^2_2)(\nabla \Phi \cdot \nabla \phi_i) d\Omega,
\]

where \( s = \|\nabla \Phi\|^2_2 \). To simplify the numerical integration, the exact Jacobian value above is replaced by

\[
J_{ij} = \sum_{\tau \in \Omega_h} \rho(\Phi(x^*_\tau, y^*_\tau)) \int_{\tau} (\nabla \phi_j \cdot \nabla \phi_i) d\Omega +
\]

\[
\sum_{\tau \in \Omega_h} \frac{d\rho}{ds} \int_{\tau} \frac{\partial}{\partial \Phi_j} (\|\nabla \Phi\|^2_2)(\nabla \Phi \cdot \nabla \phi_i) d\Omega,
\]

where the value of \( d\rho/ds \) is calculated at the element center point. We remark here that, because the density function \( \rho \) is not a constant, the Jacobian matrix is generally non-symmetric and possibly indefinite. The explicit construction of the Jacobian matrix is not necessary if we use an unpreconditioned Newton-Krylov method; however, to implement a Schwarz preconditioner, explicit approximation of the Jacobian matrix is needed in each subdomain.

### 3.2 Density upwinding schemes

For subsonic problems, the above mentioned finite element method is sufficient; however, for transonic cases upwinding has to be introduced in the density calculation in order to capture the weak shock in the solution. The proper use of an upwinding scheme is essential both to the success of the overall approach in finding the correct location and strength of the shock and to the convergence, or the fast convergence, of the inexact Newton's method.

As mentioned earlier, density \( \rho \) is assumed to be a constant in each element, and this constant is ordinarily determined by the four values of \( \Phi \) at the corners of the element,
Figure 1: The finite element stencil. $\Phi$ is stored at the cell vertices and $\rho$ at the cell centers.

through (7). Following [24, 42], if an element is determined to be supersonic, or nearly so, its density value is replaced by

$$\tilde{\rho} = \rho - \mu V \cdot \nabla \rho,$$

(12)

where $V$ is the normalized element velocity and $\nabla \rho$ is an upwind undivided difference. For example, with reference to Fig. 1, if $V = (V_x, V_y)$ and $V_x, V_y > 0$ in the element marked with $\rho_1$, then

$$\tilde{\rho}_1 = \rho_1 - \mu (V_x (\rho_1 - \rho_6) + V_y (\rho_1 - \rho_8)).$$

Here $\mu$ is the element switching function,

$$\mu = \nu_0 \max \{0, 1 - M^2/M_c^2\},$$

(13)

where $M$ is the element Mach number, $M_c$ is a pre-selected cutoff Mach number chosen to introduce dissipation just below Mach 1.0, and $\nu_0$ is a constant usually set to something between 1.0 and 3.0 to increase the amount dissipation in the supersonic elements. Parameters $M_c$ and $\nu_0$ may be varied to advantage between Newton steps in problems with shocks. Roughly speaking, $M_c$ controls the spatial extent of the upwinding; as it drops below 1.0 upwinding is triggered in a greater number of subsonic (but nearly sonic) cells. $M_c$, $\nu_0$, and $V$ together control the amount of the upwinding in a triggered cell. A low $M_c$ and a high $\nu_0$ stabilize convergence but diffuse the shock. As iterations progress, $M_c$ should approach 1.0 and $\nu_0$ should be decreased to steepen up a shock whose location and strength has converged. A well resolved shock will take many Newton steps to settle on its correct location, whereas a diffuse shock centers quickly on this location. A carefully chosen sequence of $M_c$ and $\nu_0$ can considerably accelerate the Newton's convergence; more details on this "viscosity damping" can be found in [41]. Another way to control the convergence of Newton's method is through the use of iterated maximization of the switching function, as described below and apparently first discussed in [22].

For each element, $\mu$ as defined by (13) is called the zeroth level switching function and is denoted more precisely as $\mu^{(0)}$. In Fig. 1, there is a nonzero $\mu^{(0)}$ for each element marked with $\rho_1$. The first level switching function for the element marked with $\rho_1$ is defined as

$$\mu_1^{(1)} = \max \{\mu_1^{(0)}, \ldots, \mu_9^{(0)}\},$$
namely, \( \mu^{(1)}_i \) is the maximum of the all the \( \mu \) values in its immediate neighborhood. A 
\((k + 1)\)-level switching function is defined recursively as the maximum of the neighboring
\( k \)-level \( \mu \) values.

Results for \( k = 2 \) are reported in §6. A rather tight Mach cutoff is used, namely
\( M_c^2 = 0.95 \), and we set \( \nu_0 \) to 1.0.

We remark that large \( k \) results in greater discrete data dependency, or larger effective
stencil size, in both the nonlinear function and the Jacobian. For example, if \( k = 0 \), the
stencil contains at most 9 points (e.g., the nine mesh points immediately surrounding \((i, j)\)
in Fig. 1). If \( k = 1 \), then some of the “×” points may join the stencil depending the
flow direction, and the stencil may contain as many as 16 points. The increase in the
stencil bandwidth does not cause much of a problem in the nonlinear function evaluation,
but would substantially increase the memory requirement of the Jacobian matrix, which
is constructed and stored for the Schwarz preconditioner at the beginning of each Newton
iteration. To keep the memory requirements small in practice, we do not calculate or store
the matrix elements introduced by using the iterated switching function. Our numerical
experiments show that this extra level of approximation of the Jacobian matrix does not, in
fact, appreciably reduce its power as a preconditioner. This is analogous to the practice in
[7] of using Jacobian blocks based on first-order upwinding to drive a second-order upwinded
residual to zero, in an inexact Newton iteration. Though not much discussed in the theory of
approximate Newton methods for systems arising from PDEs, such techniques are commonly
applied in stationary iterations in steady-state aerodynamics codes. Especially in three
space dimensions, using simplified upwinding in the Jacobian matrix dramatically reduces
cost at a small expense in convergence rate degradation.

4 Newton-Krylov-Schwarz algorithms

NKS is a family of general purpose algorithms for solving nonlinear boundary value prob-
lems of partial differential equations. In terms of software development, NKS has three
components that can be handled independently. However, to achieve reasonable overall
convergence, the three components have to be tuned simultaneously. We discuss these
components in turn.

4.1 The matrix-free Newton method

In this subsection, we briefly discuss the well known matrix-free inexact finite-difference
Newton algorithm, and the Eisenstat-Walker forcing functions [15]. Starting from an initial
guess \( \Phi_0 \), which is sufficiently close to the solution, a solution of the nonlinear system (9)
is sought by using an inexact Newton method: For some \( \eta_k \in [0, 1) \) find \( s_k \) that satisfies

\[
\| F(\Phi_k) + J(\Phi_k)s_k \| \leq \eta_k
\]

and set \( \Phi_{k+1} = \Phi_k + \lambda_k s_k \), where \( \lambda_k \in (0, 1) \) is determined by a line search procedure
[11]. In practice, the method is insensitive to the details of the method used to determine
\( \lambda_k \). Much more important is nonlinear continuation in grid density, dissipation, and other
parameters. The iteration is continued until convergence, typically defined in terms of a
sufficiently small \( ||F(\Phi_k)|| \). The vector \( s_k \) is obtained by approximately solving the linear Jacobian system

\[
J(\Phi_k)s_k = -F(\Phi_k)
\]

with a Krylov space iterative method. The action of Jacobian \( J \) on an arbitrary Krylov vector \( w \) can be approximated by

\[
J(\Phi_k)w \approx \frac{1}{\epsilon} \left( F(\Phi_k + \epsilon w) - F(\Phi_k) \right).
\]

Finite-differencing with \( \epsilon \) makes such matrix-free methods potentially more susceptible to finite word-length effects than ordinary Krylov methods. Left preconditioning of the Jacobian with an operator \( B^{-1} \) can be accommodated via

\[
B^{-1} J(\Phi_k) w \approx \frac{1}{\epsilon} \left( B^{-1} F((\Phi_k + \epsilon w)) - \tilde{F}(\Phi_k) \right),
\]

where \( \tilde{F}(\Phi_k) = B^{-1} F(\Phi_k) \) is stored once, and right preconditioning via

\[
J(\Phi_k) B^{-1} w \approx \frac{1}{\epsilon} \left( F((\Phi_k + \epsilon B^{-1} w)) - F(\Phi_k) \right).
\]

Right preconditioning is preferable when the focus is on comparing different preconditioners \textit{in vitro}, since the true linear residual norm that is measured as a by-product in Krylov method GMRES (see next subsection) and used in the termination test is independent of any right preconditioning. On the other hand, any left preconditioning changes this by-product residual norm. For this very reason, left preconditioning may be preferable when GMRES is applied \textit{in vivo} as the solver for an inexact Newton method. When the preconditioning \( B^{-1} \) is of high quality, the left-preconditioned residual serves as an estimate of the error in the Newton update vector. This estimate can be employed in a termination condition. In this paper, one of our emphases is assessing preconditioner quality, and we report only right-preconditioned results.

The most expensive component of the algorithm is the solution of the linear system with the Jacobian at each Newton iteration. As discussed in Eisenstat and Walker [15], when \( \Phi_k \) is far from the solution, the local linear model used in deriving the Newton method may disagree considerably with the nonlinear function itself, and it is unproductive to "over-solve" these linear systems. We tested several stopping conditions, including those discussed in [15], and found that the best choice for our problems, based on elapsed execution time for a fixed relative nonlinear residual norm reduction, is simply to set \( \eta_k = 10^{-2} ||F(\Phi_k)||_2 \). In fact, even the looser \( \eta_k = 10^{-1} ||F(\Phi_k)||_2 \) is sufficient for the first few Newton iterations, but not much time is saved by switching dynamically among these two already loose criteria, so we use the first throughout.

### 4.2 Krylov iterative methods

We use the GMRES method [36], to solve the linear system of algebraic equations

\[
Pz = b,
\]
where $P$ is the matrix appears in (15), and $b$ is the negative of the nonlinear Newton residual vector in (9). The method begins with an initial approximate solution $x_0 \in \mathbb{R}^n$ and an initial residual $r_0 = b - Px_0$. At the $m$th iteration, a correction vector $z_m$ is computed in the Krylov subspace

$$K_m(r_0) = \text{span}\{r_0, Pr_0, \ldots, P^{m-1}r_0\}$$

that minimizes the residual, $\min_{z \in K_m(r_0)} \|b - P(x_0 + z)\|_2$. The $m$th iterate is thus $x_m = x_0 + z_m$. To fit the available memory, one is sometimes forced to use the $k$-step restarted GMRES method [36]. However, in this case neither an optimal convergence property nor even convergence is guaranteed. In our experiments, we do not need to solve the linear systems very accurately; i.e., $\eta = 10^{-2}$ in

$$\|b - Px_m\|_a \leq \eta\|r_0\|_2$$

is sufficient to capture an accurate solution to the nonlinear problem, in both subsonic and transonic cases. We do observe that, for certain maximum Krylov subspace dimensions (for example 30, in a problem with approximately $10^4$ times as many discrete unknowns) and certain Mach numbers ($M_{\infty} = 0.8$), the restarted GMRES can never reduce the initial residual below $10^{-5}$. In other words, there is no linear convergence. It is further noticed in such cases that the residual norm measured as a by-product in GMRES is no longer the same as, or even close to, the true residual norm except at the restarting points, where it is freshly updated. A loose linear convergence tolerance avoids this problem by returning to the Newton method with a step that is far from exact. In the delicate balance between few nearly exact Newton steps with expensive inner linear solutions and many inexact Newton steps with bounded-cost inner linear solutions, we find the bottom line of overall execution time best served by bounding the inner linear work. This approach is also found most effective in the context of inviscid aerodynamics based on the primitive variable Euler equations in [7]. It deprives Newton’s method of its asymptotic quadratic convergence, but provides steep linear convergence.

### 4.3 Two-level overlapping Schwarz preconditioners with non-nested coarse spaces

In this subsection, we discuss a two-level overlapping Schwarz preconditioner with inexact subdomain solvers and non-nested coarse grid. Let $\Omega$ be the domain of the full potential equation. We first partition the domain into nonoverlapping substructures $\Omega_i, i = 1, \ldots, N$. To obtain an overlapping decomposition of the domain, we extend each subregion $\Omega_i$ to a larger region $\Omega'_i$, i.e., $\Omega_i \subset \Omega'_i$. Only simple box decomposition is considered in this paper: all the subdomains $\Omega_i$ and $\Omega'_i$ are rectangular and are made up of integral numbers of fine mesh cells. For simplicity, we also assume that all substructures are of the same size. More precisely, the size of $\Omega_i, i = 1, \ldots, N$, is $H_x \times H_y$ and the size of $\Omega'_i$ is $H'_x \times H'_y$, where the

---

4We believe, after Saad (personal communication), that this may be due to a lack of floating point commutativity in the product that expresses $z_m$ in GMRES, namely $z_m = PV_my$, where $V_m$ is a Gram-Schmidt basis for $K_m$ and $y$ is a coefficient vector of dimension $m$ that satisfies a related least squares problem (see [36]). The effect seems related to drastic variations in the magnitude of successive elements of $y$. 

---
$H'$ are chosen so as to ensure a discrete overlap, denoted by $ovlp$, which is uniform in the number of fine grid cells all around the perimeter, i.e.,

$$ovlp = (H'_x - H_x)/2 = (H'_y - H_y)/2,$$

for interior subdomains. For boundary subdomains, we simply cut off the part that is outside $\Omega$. Fig. 3, which appears later with the definition of numerical boundary conditions, illustrates a decomposition with an overlap of three fine mesh cells.

On each extended subdomain $\Omega'_i$, we construct a so-called subdomain preconditioner $B_i = \{J_{ij}\}$, where the node indexed by $(i, j)$ belongs to the interior of $\Omega'_i$. $J_{ij}$ is calculated by using the formula (11). The density upwinding discussed earlier is used in the transonic cases. Homogeneous Dirichlet BCs are used on the internal subdomain boundary $\partial\Omega'_i \cap \Omega$, and the appropriate external boundary condition is used on the physical boundary if present.

We next discuss the construction of the coarse grid and the coarse grid preconditioner. The coarse grid is built independently of the fine mesh. We cover $\Omega$ with another uniform rectangular mesh $\Omega_H = \{r_H^i, i = 1, \ldots, M_H\}$, and at each coarse node we introduce a bilinear finite element basis function $\Psi_i(x, y)$. The set of coarse nodes is not generally a subset of the fine mesh nodes. In other words, the discrete subspaces defined by the two meshes are generally non-nested [4]. Both coarse and fine grids cover the entire $\Omega$, and they share the same boundary, which they both resolve exactly because of its prescribed simplicity. (The case of a multi-level Schwarz preconditioner for geometrically complex grids, in which only the finest level exactly resolves the boundary geometry, is considered in [10].) The coarse grid preconditioning matrix $B_0$ is defined by using formula (11) with respect to the basis functions $\{\Psi_i\}$. The coarse grid matrix arises from an independent discretization, not an agglomeration of fine grid matrix. No upwinding is used on the coarse grid even in the transonic case. Empirically, the convergence may be slowed down if the density upwinding is used at the coarse grid, since a poorly located shock may be "resolved" and added to the fine grid solution. We do not fully understand the reason for this slowdown, and believe we are not alone in regarding the choice of a coarse grid operator for mixed elliptic-hyperbolic problems as one of the most important outstanding questions in multilevel preconditioning.

The interaction of the coarse and the subdomain preconditioners is through the interpolation and restriction operations. We define the coarse-to-fine interpolation matrix, $I_H$, as follows. Let $I_H = \{I_{ij}\}$ be an $M_h \times M_H$ matrix, and

$$I_{ij} = \Psi_j(x_i),$$

where $x_i$ is $i$th fine mesh node. The fine-to-coarse restriction matrix is defined as $(I_H^T)^T$, the transpose of $I_H^T$. The additive Schwarz preconditioner can be written as

$$B^{-1} = I_H^T B_0^{-1}(I_H^T)^T + I_1 B_1^{-1}(I_1)^T + \cdots + I_N B_N^{-1}(I_N)^T. \quad (17)$$

Let $n'_i$ be the total number of nodes in $\Omega'_i$, then $I_i$ is an $M_h \times n'_i$ extension matrix that extends each vector defined on $\Omega'_i$ to a vector defined on the entire fine mesh by padding an $n'_i \times n'_i$ identity matrix with zero rows.

Various inexact additive Schwarz preconditioners can be constructed by replacing the matrices $B_i$, $i > 0$, in (17) with convenient and inexpensive to compute matrices, such as
those obtained by using local incomplete factorizations. The coarse grid operator $B_0^{-1}$ is always applied exactly. Some detailed comparisons of (17) with global ILU preconditioners on rather general scalar problems can be found in [5]. Experience with transonic potential problems in the Boeing TRANAIR code can be found in [40].

5 Parallel implementation issues

We implemented the family of NKS algorithms on the IBM SP2. The top-level message-passing calls are implemented through the Chameleon Package of Gropp and Smith [19], which uses the IBM MPL library.

The code is written in a hostless manner. Each processor is assigned one subdomain, and the information pertaining to the interior of the subdomain is uniquely owned by that processor and is not available to any other processors except by message passing. Following the parallel complexity study in [18], the low-storage coarse mesh information is duplicated in each of the processors. On each processor, we store the subvectors and subblocks of the Jacobian matrix associated with an extended subdomain. For the coarse-grid preconditioner, the right-hand vector is built by a parallel fine-to-coarse restriction operation. Once the right-hand vector is obtained, the coarse linear system is solved simultaneously on all of the processors. The solution is then added to the local subdomain solutions by using a parallel coarse-to-fine interpolation operation. In all the experiments that we have done, the size of the coarse linear system is so small that the CPU time spent on it is negligible.

Figure 2: Illustration of two-way buffer copies required at each nearest-neighbor boundary. For each action of the Schwarz preconditioner on a vector the data needed in the extended regions are copied from the interior of neighboring subdomains. The amount of data moved for each processor is proportional to the area of overlap.

The multiplication of a vector with the Schwarz preconditioner is the most expensive operation in terms of memory consumption and execution time. At the beginning of each nonlinear iteration, the $\Phi$-dependent local and coarse grid preconditioning matrices are computed explicitly, and stored in Compressed Sparse Row (CSR) format. According to the desired type of local solver (see below), the matrices are factored, and the upper and lower triangular parts stored. The matrices for the interpolation and restriction between the coarse and fine meshes are independent of $\Phi$, and are calculated in a preprocessing
step. After the solution of each subproblem is obtained, those portions that lie within the overlapping regions (bounded by the dashed boxes in Fig. 2) are sent to neighboring subdomains to complete the summation defined in (17). The length of the message is proportional to the area of overlap.

Figure 3: Domain $\Omega$ with an exaggerated NACA 0012 curve at the bottom. The dashed lines indicate the partition of the domain into nonoverlapping substructures, and the dotted lines indicate the overlapping subdomains. The incomplete fine mesh of solid lines illustrates an overlap of 3 subintervals. $\Gamma_6$ is the inflow, $\Gamma_5$ the freestream, and $\Gamma_4$ the outflow boundary.

6 Numerical results

In this section, we report some numerical results obtained on the IBM SP2 with up to 32 processors for both subsonic and transonic flows. The SP2 offers subsets of dedicated nodes through a batch scheduler. Other jobs on different dedicated subsets share the communication network, but processor allocation tends to concentrate intercommunicating processors onto independent subnetworks. We report five performance metrics for each run: (1) the total number of Newton iterations; (2) the total number of GMRES iterations; (3) the total execution time (including the pre-processing step such as the decomposition of the mesh, the calculation of message lengths and the allocation of sparse matrices, all communication and synchronization overhead, etc.), which is an average over all processors; (4) the megaflop rate, which is a sum of the rates on each processor; and (5) the total communication time, which is an average over all processors (isolated out of (3)). Metrics (1) and (2) are of interest in understanding convergence rates, while (3), (4) and (5) are useful in assessing bottom-line performance and modeling scalability.
The computer code was first developed on a network of workstations, and then moved to the IBM SP2, changing only a UNIX makefile. To obtain the best performance of the code, in terms of either the elapsed time or the megaflop rate, is not the main purpose of this paper. We provide the execution time and megaflop information for all the calculations for completeness. Though compiler optimization was used, the listed megaflop rates are an order of magnitude below their peak values. Greater attention to caching is undoubtedly required to improve this situation, and will potentially be simplified when addressed in the future by the domain-oriented structure of the software.

6.1 Test problem and parameter selection

\( \Omega \) is a unit-aspect ratio square partitioned into a uniform rectangular meshes up to 512 \( \times \) 512 in size. Let \( q_{\infty} \), the farfield flow speed, be normalized to 1. In Fig. 3, let \( \phi_{\infty} = \int_{\Omega} q_{\infty} dx \). We assume the following boundary conditions.

- On the farfield boundaries \( \Gamma_4, \Gamma_5 \) and \( \Gamma_6 \), we assume \( \phi = \phi_{\infty} \).
- On \( \Gamma_2 \),
  \[
  \frac{\partial \phi}{\partial y} = -\nabla \phi_{\infty} \cdot (n_x, n_y),
  \]
  where \( n = (n_x, n_y) \) is the unit outward normal, and where \( y = f(x) \) describes the shape of airfoil for \( x \in \Gamma_2 \). Once the function \( f(x) \) is given, this condition becomes
  \[
  \frac{\partial \phi}{\partial y} = -q_{\infty} f'(x).
  \]
- On \( \Gamma_1 \) and \( \Gamma_3 \), we impose for symmetry the no penetration condition
  \[
  \frac{\partial \phi}{\partial n} = \frac{\partial \phi}{\partial y} = 0.
  \]

The functional form used for the NACA0012 geometry [38] is

\[
f(x) = 0.17814(\sqrt{x} - x) + 0.10128(x(1-x)) - 0.10968x^2(1-x) + 0.06090x^3(1-x),
\]
for \( x \in (0, 1) \). This unit interval is scaled to \((1/3, 2/3)\) in the overall domain. The blunt leading edge of the airfoil poses a technical problem for the transpiration boundary condition, since \( f'(x) \) is undefined there, so we slightly modify the function \( f(x) \). The curve in the interval \([0, 0.047059]\) is replaced by a parabola with a matching function value at \( x = 0 \), and matching function and first derivative values at \( x = 0.047059 \).

A number of parameters need to be specified in the NKS algorithms. The selection of some parameters, such as the number of subdomains, is related to the granularity of the architecture, not to the equation, itself. Altogether, we have

- Switching-function parameters, in the transonic case (§3.2). The level of maximization of the switching function is set to 2, \( \nu_0 \) is 1.0, and the cutoff Mach value is \( M_c^2 = 0.95 \).
• Finite differencing parameter, \( \epsilon \) (§4.1). We find that for the nondimensional scalar full potential equation, the numerics are not very sensitive to \( \epsilon \). We simply set it to \( 10^{-8} \), near the square root of the machine epsilon.

• Newton convergence parameters (§4.1). The initial guess is a simple interpolation of the farfield boundary condition. Nonlinear convergence is declared following a \( 10^{-10} \) relative reduction of the initial residual. The step size reduction ratio in the line search is 0.5 and the termination tolerance is \( 10^{-4} \).

• Krylov convergence parameters (§4.2). The convergence tolerance for the linear iterative solver at each Newton iteration \( \eta_k = 10^{-2} \| F(\Phi_k) \|_2 \). We restart GMRES at every 30th iteration.

• Number of subdomains, \( n_s \) (§4.3). Since only the additive version of Schwarz is under consideration, we always set that the number of subdomains is the same as the number of processors, \( np \), which varies from 8 (the minimum required to store the problem) to 32 (the maximum available within power-of-two configurations). (In a multiplicative algorithm [37], we would set \( n \) to \( np \) times the number of colors.)

• Overlapping size, \( ovlp \) (§4.3). In fact, there are two overlapping sizes, in \( x \) and \( y \) directions. In this paper, we assume the same number of fine mesh cells, \( ovlp = 1, \ldots, 5 \), are extended in both directions.

• Coarse grid size (§4.3). This varies from no coarse grid (\( 0 \times 0 \)) to a coarse grid with a modest number of points in each subdomain (\( 10 \times 11 \)). (The coarse grid cells are square, but asymmetry in the employment of Neumann boundary conditions in the \( x \) and \( y \) directions makes the total number of gridpoints off by one.)

• Level of fill, \( k \), in ILU (§4.3). According to our past experience with multilevel preconditioning [5] and similar experience on an industrial-grade transonic potential code [33], relatively modest fill-in is optimal for small subdomains. Intuitively, little is lost relative to the coupling already sacrificed at subdomain boundaries. However, as the local memory keeps increasing on powerful modern parallel computers, such as the IBM SP2, the size of the subdomain problems can be quite large. For large subdomain problems, low level of fill-in is no longer as effective. \( k \) varies from 0 to 5 in our experiments, then jumps discontinuously to the full band in the case of exact subdomain solves.

6.2 Observations — subsonic case

Our first test case corresponds to a subsonic problem with \( M_\infty = 0.1 \). The linear systems that arise fall within the elliptic theory for Schwarz [37]. It takes 6 Newton iterations to reduce the initial nonlinear residual by a factor of \( 10^{-10} \). Because of the Krylov dimension cut-off, the convergence is linear; see the left panel in Fig. 4. The top portion of Table 1 shows the convergence performance for a fixed-size problem of \( 512 \times 512 \) uniform cells with an increasing number of subdomains: 8, 16 and 32. The overlap size is fixed at \( 3h \). The density of the unnested uniform coarse grid varies from \( 0 \times 0 \) to \( 10 \times 11 \). Key observations from this
example are as follows: (1) Even a modest coarse grid makes a significant improvement in an additive Schwarz preconditioner, especially when the number of subdomains is large. As much as 40% of the execution time can be saved when adding a 2 × 3 coarse grid to a no coarse grid preconditioner, for the 32-subdomain case. (2) A law of diminishing returns sets in at roughly one point per subdomain. (3) When using 8 processors, the total communication time is always less than 5% of the total computational time, however, it becomes as much as 26% when using 32 processors. (This includes synchronization delays as well as the time actually delivering the message packets from application process to application process.)

Table 2 shows the effects of the overlap size. For simplicity, we fix the coarse grid to 6 × 7 for all test cases. The overlap size is given here in absolute terms, i.e., the distance between the boundary of the unextended subdomain and the extended subdomain, not relative to the diameter of the subdomain. All the subproblems are solved with the exact Gaussian elimination in sparse format. Since the fine mesh size is fixed, when using small number of processors, such as np = 8, the single processor memory requirement is substantial. In this case, increase the overlap size can indeed reduce the total number of GMRES iterations, but the reduction of the total execution time is rather limited.

In Table 3, we present the results when the subproblems are solved with ILU(k) for various levels of fill-in. The overlap size is 3h, and the coarse grid is 7 × 8. The conclusion from the tests shown is that the larger the k, the faster the method becomes; see the boxed numbers in Table 3. When using a small number of processors, like 8, the best execution time is obtained with ILU(5); compare the upper portions of Tables 1, 2 and 3. However, if the processor number is large, the optimal result can only be obtained by considering several parameters: ovlp, k, the coarse mesh size, and perhaps others. We have not simultaneously varied all relevant parameters to get the best results, but have presented controlled slices through parameter space for insight.

Load balancing should not be a significant issue in the dedicated-processor subsonic case. All processors have nearly the same computational load, except those which have to handle the Neumann boundary conditions. This is no longer true for the transonic calculation, when a shock resides in some of the subdomains. See §6.4.
Figure 5: For $M_\infty = 0.8$, the left figure shows the history of the Newton residual, and the right shows the (upper surface) $C_p$ curve at convergence.

Figure 6: The left figure shows the convergence history of the $C_p$ curves at $M_\infty = 0.8$. The right figure shows the Mach contours of the final solution at $M_\infty = 0.8$.

### 6.3 Observations — transonic case

Figure 5 shows the convergence history and converged $C_p$ for a transonic problem with $M_\infty = 0.8$. The first and most important observation is that without a proper upwinding discretization, all three components of NKS can fail. Fig. 6 shows the convergence history in terms of the $C_p$ curves. We note that it takes only 4 to 5 iterations for the Newton’s method to establish the neighborhood of the shock, but another 15 or so iterations to move it to the exact location. Mach contours at the final solution are given in Fig. 6. While the shock is setting up, the linear convergence of Newton’s method is interrupted; see the left panel of Fig. 5.

The results for coarse grids of varying size are summarized at the bottom part of Table 1. The columns marked $0 \times 0$ and $2 \times 3$ reveal an interesting result for a mixed elliptic-hyperbolic problem. The inclusion of a small coarse grid can reduce the total number of the linear
6.4 Parallel efficiency

The parallel efficiency of the present algorithm-software-hardware system is encouraging, but it is useful to sort out in detail where efficiency is lost in going from 8 to 32 processors. We display the parallel performance in Table 4, whose first three columns are excerpted from the first and last columns of Table 3. The execution time data in the last column of Table 3 is the best, or nearly the best, for each Mach number and parallel granularity out of all of the parameter combinations considered, and is therefore the most meaningful from which to draw parallel efficiency conclusions, though more flattering conclusions could be drawn from runs that were performing more computation per node per communication exchange.

After the number of processors, we list the number of linear GMRES iterations per 6 Newton steps in the upper (subsonic) half of the table, and per 19 Newton steps in the lower (transonic) half of the table. Then we list the execution time, per 6 or 19 Newton steps, respectively, these being the typical number of Newton steps required to fully solve
the nonlinear problem. Fig. 7 shows that, for the transonic case, the number of GMRES steps can vary significantly over the course of a complete set of Newton iterations, but the mean and the median are close.

Consider the following idealized model for the execution time of the fixed-size problem on \( p \) processors. Let \( T(p) \) be the overall execution time, \( I(p) \) the number of linear iterations, and \( C(p) \) the average cost per iteration. (We note that in GMRES, the average cost per iteration is not independent of the number of iterations, because of the orthogonalization overhead of later Krylov vectors is greater than earlier vectors, but we assume that the dominant cost per iteration is the parallel Schwarz preconditioning.)

The overall parallel efficiency is defined as 
\[
\eta(p) = \frac{T(1)}{p T(p)},
\]
where 
\[
T(p) = I(p) \cdot C(p).
\]
Since we lack results for \( p = 1 \) on this problem of industrial size (512 x 512), we replace all efficiencies by relative efficiencies with respect to the minimum configuration of \( p_1 = 8 \).

The overall relative parallel efficiency is therefore defined as
\[
\eta(p_1 \rightarrow p) = \frac{p_1 T(p)}{p T(p)}.
\]
The numerical efficiency, a measure of the robustness of the preconditioning with respect to increasing granularity, is
\[
\eta_{\text{numer}}(p_1 \rightarrow p) = \frac{I(p_1)}{I(p)}.
\]
The implementation efficiency is the remaining factor,
\[
\eta_{\text{impl}}(p_1 \rightarrow p) = \frac{p_1(C(p_1))}{p C(p)},
\]
so that
\[
\eta(p_1 \rightarrow p) = \eta_{\text{numer}}(p_1 \rightarrow p) \times \eta_{\text{impl}}(p).
\]

The numerical efficiency is nearly 90% or above for all cases — that is, the convergence rate of the preconditioned linear system hardly degrades with increasing parallel granularity. Approximately 13 GMRES steps are required for each subsonic Jacobian system and approximately 23 GMRES steps for each transonic Jacobian system. This insensitivity to granularity for a multilevel preconditioned operator is predicted by the Schwarz theory for the subsonic case, and seems to be a fortunate consequence of the relatively confined supersonic pocket of flow in the transonic case.

The implementation efficiency accounts for the most significant factor of overall efficiency decline. The difference between the subsonic and transonic implementation efficiencies at high granularity can be attributed to load imbalance, since the cells requiring upwinding are concentrated into a small number of processors. (A more sophisticated dynamic mapping algorithm could address this problem, but this is beyond present scope.) The subsonic degradation of 76% in going from 4 to 32 nodes is identified as the chief remaining loss. Redundant work and higher communication-to-computation ratio in the overlap regions, which account for a steadily increasing fraction of all points in a fixed-size problem explain the majority of this efficiency loss, which would disappear in a scaled problem with fixed-size subdomains on each processor.

### 6.5 Sequential comparison with global ILU(\( k \)) preconditioners

The results of this section establish Schwarz preconditioning as numerically attractive and reasonably parallel efficient, but it is natural to ask whether its utility is limited to distributed memory implementations of Newton-Krylov methods. To satisfy curiosity on this point, we conclude with tests of Schwarz preconditioning against the popular global ILU(\( k \)), \( k = 0, \ldots, 5 \), family of preconditioners on a non-dedicated single-processor SUN SPARCstation with 512MB of memory. The results are summarized in Table 5. Because of the overlap and the coarse solve, the Schwarz preconditioner needs more memory, even if all subdomain problems are solved inexactly with ILU(5), than the other global ILU(\( k \)) preconditioners. On the other hand, Schwarz outperforms all the global solvers in terms of total GMRES
iteration count and the total execution time. Part of the reason for the fine performance of the Schwarz method is the much higher uni-processor Megaflap rating, which is presumably related to much improved cache locality.

7 Conclusions

We have investigated computationally the effectiveness of Newton-Krylov-Schwarz methods applied to the full potential equation of aerodynamics in some simplified situations in two space dimensions. Best performance is obtained with modest overlap, a modest coarse grid (one or two points per processor), modest-to-generous fill in the subdomain ILU preconditioners, and uniformly loose convergence tolerances on the Krylov iterations within each Newton step. For subsonic problems, the theoretically expected performance of the method is essentially achieved. For the transonic case, the numerics are more encouraging than existing theory. Overall computation time is approximately six times greater for the transonic than for the subsonic case, with current upwinding strategies. This can be factored into a three-fold increase in the number of Newton steps in the transonic case, and a two-fold increase in the number of Krylov iterations per Newton step.

Two strategies that should be employed on more nonlinearly taxing problems that we have not considered here are mesh sequencing and pseudo-transient continuation. Their purpose is to deliver an initial iterate for the steady-state form of Newton's method employed in this paper that is already in the local domain of convergence on the finest grid. (Observe, for instance, that the number of Newton steps required on \( M_\infty = 0.8 \) problem the 256 \( \times \) 256 grid in Table 5, is roughly half that of the corresponding problem on the 512 \( \times \) 512 grid in Table 3. If the shock is correctly located on a (relatively) coarse grid, the plateau of Fig. 5 will be diminished on a finer grid that is initialized from the coarse grid solution.) Our rapid turnaround times for two-dimensional problems artificially deemphasize the importance of these strategies in large, complex nonlinear problems. In addition to globalizing the Newton convergence, continuation strategies tend to improve the linear conditioning of the intermediate problems, and are therefore potentially useful even in problems (such as ours) for which simple initial guesses on the finest grid lead to convergence.

The broadest motivation for Newton-Krylov-Schwarz methods is the need to solve large-scale problems with complex discretizations on distributed-memory systems with limited memory per node. The matrix-free aspect of Newton permits shortcuts in Jacobian formation storage while the domain decomposition aspect of Schwarz leads to load-balanced data-to-memory maps that render communication subdominant in the preconditioning. The amount of work done in the Krylov iteration can be adjusted to produce an overall method with the best balance between the nested components.
References


Table 1: **Varying the coarse grid size.** Fine mesh $512 \times 512$, $M_\infty = 0.1$ and 0.8, sparse LU for all subproblems, $ovlp = 3h$. “Newton” is the total number of Newton iterations. “GMRES” is the total number of GMRES iterations occurring in all of the Newton iterations. “EXEC” is the total execution time per processor in seconds for the entire calculation. “COMM” is the total communication time per processor in seconds.

<table>
<thead>
<tr>
<th>np</th>
<th>Coarse Grid</th>
<th>$0 \times 0$</th>
<th>$2 \times 3$</th>
<th>$4 \times 5$</th>
<th>$6 \times 7$</th>
<th>$8 \times 9$</th>
<th>$10 \times 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>$M_\infty = 0.1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>144</td>
<td>81</td>
<td>59</td>
<td>53</td>
<td>50</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>136.79</td>
<td>125.30</td>
<td>104.18</td>
<td>97.28</td>
<td>94.18</td>
<td>94.63</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>2.14</td>
<td>2.10</td>
<td>1.95</td>
<td>2.39</td>
<td>3.28</td>
<td>4.24</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>183.59</td>
<td>157.02</td>
<td>170.67</td>
<td>177.63</td>
<td>180.94</td>
<td>180.93</td>
</tr>
<tr>
<td>16</td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>167</td>
<td>92</td>
<td>66</td>
<td>54</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>72.18</td>
<td>50.34</td>
<td>42.10</td>
<td>39.18</td>
<td>40.95</td>
<td>42.32</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>2.25</td>
<td>2.38</td>
<td>2.37</td>
<td>2.88</td>
<td>3.97</td>
<td>5.48</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>295.19</td>
<td>266.83</td>
<td>277.39</td>
<td>280.01</td>
<td>272.65</td>
<td>265.44</td>
</tr>
<tr>
<td>32</td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>227</td>
<td>105</td>
<td>72</td>
<td>64</td>
<td>53</td>
<td>57</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>47.94</td>
<td>29.55</td>
<td>24.32</td>
<td>24.46</td>
<td>23.43</td>
<td>26.59</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>3.36</td>
<td>2.92</td>
<td>2.95</td>
<td>4.13</td>
<td>4.51</td>
<td>7.00</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>498.04</td>
<td>477.99</td>
<td>463.72</td>
<td>440.64</td>
<td>423.16</td>
<td>390.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$M_\infty = 0.8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Newton</td>
<td>20</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>814</td>
<td>435</td>
<td>359</td>
<td>350</td>
<td>307</td>
<td>311</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>757.45</td>
<td>548.52</td>
<td>490.64</td>
<td>498.08</td>
<td>458.75</td>
<td>466.20</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>10.97</td>
<td>11.63</td>
<td>12.16</td>
<td>15.68</td>
<td>19.07</td>
<td>24.52</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>158.70</td>
<td>158.47</td>
<td>162.20</td>
<td>163.44</td>
<td>163.00</td>
<td>161.15</td>
</tr>
<tr>
<td>16</td>
<td>Newton</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>20</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>849</td>
<td>483</td>
<td>398</td>
<td>349</td>
<td>320</td>
<td>330</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>432.78</td>
<td>327.77</td>
<td>306.25</td>
<td>291.14</td>
<td>277.06</td>
<td>289.89</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>11.69</td>
<td>12.90</td>
<td>14.14</td>
<td>18.43</td>
<td>23.26</td>
<td>32.79</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>233.50</td>
<td>224.48</td>
<td>219.17</td>
<td>215.38</td>
<td>210.78</td>
<td>204.68</td>
</tr>
<tr>
<td>32</td>
<td>Newton</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>1142</td>
<td>636</td>
<td>454</td>
<td>387</td>
<td>391</td>
<td>385</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>333.79</td>
<td>244.36</td>
<td>207.54</td>
<td>200.21</td>
<td>215.46</td>
<td>227.84</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>16.09</td>
<td>18.43</td>
<td>19.40</td>
<td>25.26</td>
<td>34.46</td>
<td>47.88</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>360.18</td>
<td>333.53</td>
<td>317.77</td>
<td>298.29</td>
<td>284.96</td>
<td>267.36</td>
</tr>
</tbody>
</table>
Table 2: Varying the overlapping size $ovlp$. Fine mesh $512 \times 512$, $M_\infty = 0.1$ and 0.8. Exact LU for all subproblems. Coarse grid size $6 \times 7$. "Newton" is the total number of Newton iterations. "GMRES" is the total number of GMRES iterations occur in all of the Newton iterations. "EXEC" is the execution time per processor in seconds for the entire calculation. "COMM" is the total communication time per processor in seconds.

<table>
<thead>
<tr>
<th>$np$</th>
<th>$ovlp = 1h$</th>
<th>$ovlp = 2h$</th>
<th>$ovlp = 3h$</th>
<th>$ovlp = 4h$</th>
<th>$ovlp = 5h$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M_\infty = 0.1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>77</td>
<td>60</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>EXEC</td>
<td>101.12</td>
<td>99.90</td>
<td>97.31</td>
<td>98.31</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>3.22</td>
<td>2.59</td>
<td>2.46</td>
<td>2.36</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>161.03</td>
<td>174.33</td>
<td>177.61</td>
<td>186.71</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>82</td>
<td>61</td>
<td>54</td>
<td>49</td>
</tr>
<tr>
<td>16</td>
<td>EXEC</td>
<td>51.77</td>
<td>45.46</td>
<td>44.05</td>
<td>44.42</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>3.87</td>
<td>3.07</td>
<td>2.95</td>
<td>2.77</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>267.85</td>
<td>276.36</td>
<td>279.31</td>
<td>277.96</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>93</td>
<td>72</td>
<td>64</td>
<td>57</td>
</tr>
<tr>
<td>32</td>
<td>EXEC</td>
<td>30.35</td>
<td>28.28</td>
<td>26.75</td>
<td>26.31</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>5.42</td>
<td>4.24</td>
<td>3.64</td>
<td>3.58</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>387.34</td>
<td>421.17</td>
<td>441.80</td>
<td>452.00</td>
</tr>
<tr>
<td></td>
<td>$M_\infty = 0.8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>435</td>
<td>365</td>
<td>350</td>
<td>319</td>
</tr>
<tr>
<td>8</td>
<td>EXEC</td>
<td>527.50</td>
<td>489.65</td>
<td>498.78</td>
<td>491.47</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>17.66</td>
<td>15.87</td>
<td>16.31</td>
<td>15.86</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>145.19</td>
<td>160.11</td>
<td>163.11</td>
<td>170.98</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>19</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>462</td>
<td>377</td>
<td>349</td>
<td>324</td>
</tr>
<tr>
<td>16</td>
<td>EXEC</td>
<td>315.39</td>
<td>294.34</td>
<td>292.02</td>
<td>280.61</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>22.74</td>
<td>19.22</td>
<td>18.70</td>
<td>17.67</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>216.43</td>
<td>211.85</td>
<td>196.48</td>
<td>221.34</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>19</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>627</td>
<td>445</td>
<td>387</td>
<td>354</td>
</tr>
<tr>
<td>32</td>
<td>EXEC</td>
<td>251.96</td>
<td>207.12</td>
<td>199.90</td>
<td>188.99</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>38.08</td>
<td>27.51</td>
<td>24.12</td>
<td>23.02</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>276.02</td>
<td>301.20</td>
<td>298.73</td>
<td>313.36</td>
</tr>
</tbody>
</table>
Table 3: **Varying the level of ILU($k$) fill-in.** Fine mesh $512 \times 512$, $M_{\infty} = 0.1$ and 0.8. Coarse grid is $7 \times 8$. $olv_{p} = 3h$. “Newton” is the total number of Newton iterations. “GMRES” is the total number of GMRES iterations occur in all of the Newton iterations. “EXEC” is the execution time per processor in seconds for the entire calculation. “COMM” is the total communication time per processor in seconds.

<table>
<thead>
<tr>
<th>$np$</th>
<th>$ILU(k)$</th>
<th>$k = 0$</th>
<th>$k = 1$</th>
<th>$k = 2$</th>
<th>$k = 3$</th>
<th>$k = 4$</th>
<th>$k = 5$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$M_{\infty} = 0.1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>307</td>
<td>175</td>
<td>127</td>
<td>98</td>
<td>84</td>
<td>75</td>
</tr>
<tr>
<td>8</td>
<td>EXEC</td>
<td>171.46</td>
<td>110.07</td>
<td>88.26</td>
<td>76.11</td>
<td>68.28</td>
<td>71.74</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>14.08</td>
<td>8.10</td>
<td>5.87</td>
<td>4.54</td>
<td>4.44</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>119.80</td>
<td>113.63</td>
<td>109.46</td>
<td>102.36</td>
<td>96.00</td>
<td>95.43</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>171.46</td>
<td>110.07</td>
<td>88.26</td>
<td>76.11</td>
<td>68.28</td>
<td>71.74</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>307</td>
<td>175</td>
<td>127</td>
<td>98</td>
<td>84</td>
<td>75</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>171.46</td>
<td>110.07</td>
<td>88.26</td>
<td>76.11</td>
<td>68.28</td>
<td>71.74</td>
</tr>
<tr>
<td>16</td>
<td>COMM</td>
<td>14.08</td>
<td>8.10</td>
<td>5.87</td>
<td>4.54</td>
<td>4.44</td>
<td>3.62</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>119.80</td>
<td>113.63</td>
<td>109.46</td>
<td>102.36</td>
<td>96.00</td>
<td>95.43</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>299</td>
<td>178</td>
<td>129</td>
<td>101</td>
<td>87</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>97.82</td>
<td>64.14</td>
<td>51.19</td>
<td>43.34</td>
<td>40.81</td>
<td>40.07</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>16.73</td>
<td>9.42</td>
<td>6.85</td>
<td>5.38</td>
<td>4.80</td>
<td>3.88</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>208.04</td>
<td>201.60</td>
<td>195.17</td>
<td>188.34</td>
<td>183.56</td>
<td>179.69</td>
</tr>
<tr>
<td></td>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>298</td>
<td>179</td>
<td>130</td>
<td>104</td>
<td>90</td>
<td>82</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>64.34</td>
<td>41.92</td>
<td>32.73</td>
<td>28.18</td>
<td>26.34</td>
<td>25.57</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>19.28</td>
<td>11.44</td>
<td>8.14</td>
<td>6.63</td>
<td>5.70</td>
<td>4.91</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>324.76</td>
<td>321.72</td>
<td>318.07</td>
<td>308.78</td>
<td>304.06</td>
<td>305.19</td>
</tr>
<tr>
<td></td>
<td>$M_{\infty} = 0.8$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Newton</td>
<td>19</td>
<td>20</td>
<td>18</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>1638</td>
<td>911</td>
<td>622</td>
<td>547</td>
<td>462</td>
<td>424</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>863.15</td>
<td>523.33</td>
<td>382.27</td>
<td>364.11</td>
<td>335.47</td>
<td>330.76</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>77.30</td>
<td>42.64</td>
<td>29.13</td>
<td>25.90</td>
<td>21.62</td>
<td>19.81</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>127.49</td>
<td>128.37</td>
<td>124.04</td>
<td>123.70</td>
<td>122.13</td>
<td>119.90</td>
</tr>
<tr>
<td>16</td>
<td>Newton</td>
<td>19</td>
<td>20</td>
<td>20</td>
<td>19</td>
<td>19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>1693</td>
<td>924</td>
<td>707</td>
<td>531</td>
<td>460</td>
<td>423</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>526.40</td>
<td>316.98</td>
<td>255.54</td>
<td>209.27</td>
<td>195.96</td>
<td>192.16</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>90.88</td>
<td>49.9</td>
<td>38.32</td>
<td>28.55</td>
<td>25.13</td>
<td>22.46</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>220.22</td>
<td>213.73</td>
<td>215.42</td>
<td>214.21</td>
<td>210.64</td>
<td>208.59</td>
</tr>
<tr>
<td>32</td>
<td>Newton</td>
<td>19</td>
<td>20</td>
<td>20</td>
<td>19</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>GMRES</td>
<td>1746</td>
<td>961</td>
<td>799</td>
<td>596</td>
<td>557</td>
<td>501</td>
</tr>
<tr>
<td></td>
<td>EXEC</td>
<td>361.81</td>
<td>217.29</td>
<td>187.37</td>
<td>150.59</td>
<td>150.29</td>
<td>145.12</td>
</tr>
<tr>
<td></td>
<td>COMM</td>
<td>110.89</td>
<td>61.46</td>
<td>51.32</td>
<td>38.02</td>
<td>35.60</td>
<td>32.07</td>
</tr>
<tr>
<td></td>
<td>Mflop/s</td>
<td>342.40</td>
<td>336.48</td>
<td>342.84</td>
<td>341.52</td>
<td>343.77</td>
<td>340.06</td>
</tr>
</tbody>
</table>

24
Table 4: **Parallel efficiency.** Fine mesh $512 \times 512$, $M_\infty = 0.1$ and 0.8. Coarse grid is $7 \times 8$. *ovlp* $= 3h$. "GMRES" and "EXEC" are the total number of GMRES iterations and seconds of execution time per 6 Newton steps in the upper half of the table, and per 19 Newton steps in the lower half.

<table>
<thead>
<tr>
<th>np</th>
<th>GMRES</th>
<th>EXEC</th>
<th>$\eta_{\text{numer}}$</th>
<th>$\eta_{\text{impl}}$</th>
<th>$\eta$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>M_\infty = 0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>75</td>
<td>71.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>78</td>
<td>40.1</td>
<td>0.961</td>
<td>0.931</td>
<td>0.895</td>
</tr>
<tr>
<td>32</td>
<td>82</td>
<td>25.6</td>
<td>0.915</td>
<td>0.766</td>
<td>0.701</td>
</tr>
<tr>
<td>M_\infty = 0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>424</td>
<td>330.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>423</td>
<td>192.2</td>
<td>1.002</td>
<td>0.860</td>
<td>0.861</td>
</tr>
<tr>
<td>32</td>
<td>475</td>
<td>137.9</td>
<td>0.891</td>
<td>0.674</td>
<td>0.601</td>
</tr>
</tbody>
</table>

Table 5: **Sequential comparison** of the additive Schwarz preconditioner(OSM) with the global ILU($k$), $k = 0, \ldots, 5$, preconditioners on a single processor Sun workstation. The fine mesh is $256 \times 256$. The specifications of OSM are: 8 subdomains, $3h$ overlap, $7 \times 8$ coarse grid, and ILU(5) as the subdomain solver. MEM is the total memory needed to store the preconditioning matrix in Megabytes.

<table>
<thead>
<tr>
<th></th>
<th>OSM</th>
<th>ILU(0)</th>
<th>ILU(1)</th>
<th>ILU(2)</th>
<th>ILU(3)</th>
<th>ILU(4)</th>
<th>ILU(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>M_\infty = 0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>GMRES</td>
<td>48.29</td>
<td>2754.54</td>
<td>1599.84</td>
<td>1178.55</td>
<td>970.72</td>
<td>817.42</td>
<td>771.04</td>
</tr>
<tr>
<td>EXEC</td>
<td>1240.18</td>
<td>7766.42</td>
<td>3826.05</td>
<td>2848.73</td>
<td>2278.81</td>
<td>1761.89</td>
<td>1526.82</td>
</tr>
<tr>
<td>Mflop/s</td>
<td>32.12</td>
<td>6.68</td>
<td>6.48</td>
<td>6.19</td>
<td>6.04</td>
<td>5.85</td>
<td>5.40</td>
</tr>
<tr>
<td>M_\infty = 0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Newton</td>
<td>11</td>
<td>12</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>GMRES</td>
<td>136</td>
<td>1391</td>
<td>666</td>
<td>464</td>
<td>328</td>
<td>268</td>
<td>217</td>
</tr>
<tr>
<td>EXEC</td>
<td>1240.18</td>
<td>7766.42</td>
<td>3826.05</td>
<td>2848.73</td>
<td>2278.81</td>
<td>1761.89</td>
<td>1526.82</td>
</tr>
<tr>
<td>MEM(MB)</td>
<td>24.40</td>
<td>10.64</td>
<td>10.63</td>
<td>13.75</td>
<td>16.95</td>
<td>20.26</td>
<td>23.06</td>
</tr>
<tr>
<td>Mflop/s</td>
<td>36.57</td>
<td>6.53</td>
<td>6.48</td>
<td>6.20</td>
<td>5.63</td>
<td>5.99</td>
<td>5.79</td>
</tr>
</tbody>
</table>

25
We study parallel two-level overlapping Schwarz algorithms for solving nonlinear finite element problems, in particular, for the full potential equation of aerodynamics discretized in two dimensions with bilinear elements. The overall algorithm, Newton-Krylov-Schwarz (NKS), employs an inexact finite-difference Newton method and a Krylov space iterative method, with a two-level overlapping Schwarz method as a preconditioner. We demonstrate that NKS, combined with a density upwinding continuation strategy for problems with weak shocks, is robust and economical for this class of mixed elliptic-hyperbolic nonlinear partial differential equations, with proper specification of several parameters. We study upwinding parameters, inner convergence tolerance, coarse grid density, subdomain overlap, and the level of fill-in in the incomplete factorization, and report their effect on numerical convergence rate, overall execution time, and parallel efficiency on a distributed-memory parallel computer.