Newton-Krylov-Schwarz: An Implicit Solver for CFD*

Xiao-Chuan Cai
Department of Computer Science
University of Colorado - Boulder
Boulder, CO 80309-0430
cai@cs.colorado.edu

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

V. Venkatakrishnan
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681-0001
venkat@icase.edu

Abstract

Newton-Krylov methods and Krylov-Schwarz (domain decomposition) methods have begun to become established in computational fluid dynamics (CFD) over the past decade. The former employ a Krylov method inside of Newton's method in a Jacobian-free manner, through directional differencing. The latter employ an overlapping Schwarz domain decomposition to derive a preconditioner for the Krylov accelerator that relies primarily on local information, for data-parallel concurrency. They may be composed as Newton-Krylov-Schwarz (NKS) methods, which seem particularly well suited for solving nonlinear elliptic systems in high-latency, distributed-memory environments. We give a brief description of this family of algorithms, with an emphasis on domain decomposition iterative aspects. We then describe numerical simulations with Newton-Krylov-Schwarz methods on aerodynamics applications emphasizing comparisons with a standard defect-correction approach, subdomain preconditioner consistency, subdomain preconditioner quality, and the effect of a coarse grid.

*This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the authors were in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681-0001.

1 INTRODUCTION

Several trends contribute to the importance of parallel implicit algorithms in CFD. Multidisciplinary analysis and optimization put a premium on the ability of algorithms to achieve low residual solutions rapidly, since analysis codes for individual components are typically solved iteratively and their results are often differenced for sensitivities. Problems possessing multiple scales provide the classical motivation for implicit algorithms and arise frequently in locally adaptive contexts or in dynamical contexts with multiple time scales, such as aeroelasticity. Meanwhile, the never slackening demand for resolution and prompt turnaround forces consideration of parallelism, and, for cost effectiveness, particularly parallelism of the high-latency, low-bandwidth variety represented by workstation clusters.

A Newton-Krylov-Schwarz (NKS) method combines a Newton-Krylov (NK) method such as nonlinear GMRES [2], with a Krylov-Schwarz (KS) method, such as additive Schwarz [8]. The key linkage is provided by the Krylov method, of which the restarted form of GMRES [21] is perhaps the best-known example for nonselfadjoint problems. From a computational point of view, the most important characteristic of a Krylov method for the linear system $Au = f$ is that information about the matrix A needs to be accessed only in the form of matrix-vector products in a small number (relative to the dimension of the matrix) of carefully chosen directions. NK methods are suited for nonlinear problems in which it is unreasonable to compute or store a true Jacobian. However, if the Jacobian A is ill-conditioned, the Krylov method will require an unacceptably large number of iterations. The system can be transformed into the equivalent form $B_1^{-1}AB_2^{-1}v = B_1^{-1}f$, where $v = B_2u$, through the action of left and right preconditioners, B_1 and B_2. It is in the choice of preconditioning where the battle for low computational cost and scalable parallelism is usually won or lost.

In KS methods, the preconditioning is introduced on a subdomain-by-subdomain basis, which provides good data locality for parallel implementations over a range of granularities, and allows significant architectural adaptivity. The emphasis today is on operation count complexity and parallel efficiency, which means that Schwarz is usually employed with very modest subdomain overlap and in a two-level form, in which a small global problem is solved together with the local subdomain problems at each iteration. Mathematically, if $Au = f$ arises as the linearized correction step of a discretized PDE computation, Schwarz operates by:

1. Decomposing the space of the solution u: $\mathcal{U} = \sum_k \mathcal{U}_k$;
2. Finding the restriction of A to each $\mathcal{U}_k$: $A_k = R_k A R_k^T$, for some restriction operators $R_k : \mathcal{U} \rightarrow \mathcal{U}_k$ and extension operators $R_k^T : \mathcal{U}_k \rightarrow \mathcal{U}$;

3. Forming B^{-1} from the A_k^{-1}, where the inverse of A_k is well defined within the k^{th} subspace.

In Schwarz-style domain decomposition, the subspace $\mathcal{U}_k$ corresponding to subdomain k is the span of nodal basis or other expansion functions with support over the subdomain. A practical Schwarz preconditioner is

$$B^{-1} \equiv \sum_k \; R_k^T (\tilde{A}_k)^{-1} R_k, \tag{1}$$

where $\tilde{A}_k$ is a convenient approximation to $A_k \equiv R_k A R_k^T$. In this paper, $\tilde{A}_k$ is usually an incomplete LU (ILU) factorization of A_k, with modest fill permitted. For $k = 1, 2, \ldots$, the R_k and R_k^T are simply gather and scatter operators, respectively, one for each subdomain with small overlap between the subdomains. For an optional $k = 0$ term corresponding to the coarse space, R_0 represents a full-weighting restriction operator in the sense of multigrid, and R_0^T is the corresponding prolongation. We never actually assemble either A or B^{-1} globally. Rather, when their action on a vector is needed, a processor governing each subdomain executes local operations, after receiving a thin buffer of data required from its neighbors to complete stencil operations on the boundary of the subdomain. For the assembly and solution of the coarse-grid component of the preconditioner, data exchanges further than nearest neighbor must generally occur.

The two-level form of additive Schwarz can be proved to possess mesh-independent and granularity-independent condition number in elliptically dominated problems, including non-symmetric and indefinite problems, when the coarse global and fine local operators are solved with sufficient precision. Ref. [4] contains several examples demonstrating this optimality when exact subdomain solvers are used, and thus shows their superiority to global incomplete LU factorizations. Architecturally adaptive strategies for dealing with the coarse-grid component of the preconditioner are outlined in [9]. The collection [16] is representative of the state of the art of algorithms, applications, and parallel implementations.

The NKS technique is compared in this paper against a defect correction algorithm common to many implicit codes. The objective of either algorithm is to solve the steady-state conservation equations $f(u) = 0$ through the pseudo-transient form $\frac{\partial u}{\partial t} + f(u) = 0$, where the time derivative is approximated by backwards differencing, with a time step that ultimately approaches infinity. A standard defect correction approach employs an accurate right-hand side residual discretization, $f_{high}(u)$, and a convenient left-hand side Jacobian approximation, $J_{low}(u)$, based on a low-accuracy residual $f_{low}(u)$, to compute a sequence of corrections, $\delta u \equiv u^{n+1} - u^n$. Computational short-cuts are employed in the creation of the left-hand side matrix, which may, for instance, be stabilized by a degree of first-order upwinding that would not be acceptable in the discretization of the residual itself.

The so-called "defect" is $f_{high}(u) - f_{low}(u)$, and the nonlinear defect correction scheme to drive $f_{high}(u)$ to zero is to solve approximately for u^{n+1} in

$$f_{low}(u^{n+1}) = f_{low}(u^n) - f_{high}(u^n), \tag{2}$$

which may be linearized as

$$J_{low}(u^n)\delta u = -f_{high}(u^n). \tag{3}$$

In the case of pseudo-transient computations, the approximate Jacobian J_{low} is based on a low-accuracy residual:

$$J_{low} = \frac{D}{\delta t} + \frac{\partial f_{low}}{\partial u}, \tag{4}$$

where D is a scaling matrix. It is required either to solve with J_{low}, itself, or with some further algebraic or parallel approximation, $\tilde{J}_{low}$. Inconsistency between the left- and right-hand sides prevents the use of large time steps, δt, and prevents (3) from being a true Newton method.

A Newton-Krylov approach employs a (nearly) consistent left-hand side obtained by directionally differencing the actual residual, f_{high}:

$$J_{high}(u^n)\ \delta u = -f_{high}(u^n), \tag{5}$$

in which the action of J_{high} on a vector is obtained through directional differencing, for instance,

$$J_{high}(u^n)v \approx \frac{1}{h}\left[f_{high}(u^n + hv) - f_{high}(u^n)\right], \tag{6}$$

where h is a small parameter. The operators on both sides of (5) are based on consistent high-order discretizations; hence time steps can be advanced to values as large as linear conditioning permits, recovering a true Newton method in the limit.

In practice, the convergence of the method is sensitive to the choice of h in (6), which is not entirely trivial. When u and v are comparably scaled, it should ideally sit near the square-root of the machine unit roundoff, $\sqrt{\varepsilon_{mach}}$, or around 10^{-7}–10^{-8} in 64-bit precision. Smaller values improve the Taylor approximation upon which (6) is based. Larger values preserve more significant digits when the perturbed residuals on the right-hand side of (6) are differenced in finite precision. In a nondimensionalized formulation, the elements of u^n in (6) will have an RMS of approximately unity, but the elements of v will have an RMS smaller than unity by a factor of $\sqrt{n}$, where n is the dimension of the discrete unknown vector, since GMRES calls the matrix-vector evaluation routine with $||v||_2 = 1$. We therefore set h to be $\sqrt{n \cdot \varepsilon_{mach}}$. When less is known about the scaling of u^n and v, a reasonable choice is $\sqrt{\varepsilon_{mach}} \cdot (u^n, v)/||v||^2$, with guard code to set $h = \sqrt{\varepsilon_{mach}}$ if $||v||$ is too small. For a fuller

discussion, see [2]. For numerical experiments demonstrating the importance of the relative scaling of h in the CFD context, see [17, 20].

Preconditioning (5) by $\tilde{J}_{low}$, for instance on the left, as in

$$(\tilde{J}_{low})^{-1} J_{high}(u^n)\ \delta u = -(\tilde{J}_{low})^{-1} f_{high}(u^n), \tag{7}$$

shifts the inconsistency from the nonlinear to the linear aspects of the problem. This should be contrasted with the customary preconditioned form of (3),

$$(\tilde{J}_{low})^{-1} J_{low}(u^n)\ \delta u = -(\tilde{J}_{low})^{-1} f_{high}(u^n). \tag{8}$$

At this level of abstraction, it is not clear which is better — many nonlinear steps with cheap subiterations (8), or a few nonlinear steps with expensive subiterations (7). Execution time comparisons are more practical arbiters than are rates of convergence for the steady-state residual norm, but running times are sensitive to parametric tuning as well as to architectural parameters. We present a comparison of (7) and (8) in Section 4.

A more comprehensive set of comparisons of this type, comparing (7), (8), and

$$(\tilde{J}_{high})^{-1} J_{high}(u^n)\ \delta u = -(\tilde{J}_{high})^{-1} f_{high}(u^n) \tag{9}$$

may be found in [13]. Of course, (9) relies on possessing the full high-order Jacobian, and is not a matrix-free method.

We might mislead if we closed this section while failing to emphasize the importance of a globalization strategy when using any of the methods (7–9). Pseudo-transient continuation is usually recommended when a Newton-like method is used on flow problems in primitive variables. In such cases, the steady-state nonlinear residual norm should not be expected to decrease monotonically, and step-selection strategies should not be geared to monotonic decrease. Potential- or streamfunction-based formulations can more confidently be posed directly as steady-state problems, but in this case damping strategies for λ^n in $u^{n+1} = u^n + \lambda^n \delta u$ may be critical to convergence. Strategies for λ^n that provide as a minimum that $||f(u^{n+1})|| \leq ||f(u^n)||$ may need to be supplemented by feasibility checks on the components of u^{n+1} and/or a strategy that prevents spuriously large (though feasible) fluctuations in the components. In addition, artificial continuation parameters, such as upwinding strength, may enhance the convergence of globally divergent or slowly convergent iterations with near-discontinuities in the solution. For transonic potential computations, we have found invaluable the practical advice on "viscosity damping" in Section 8 of [25].

2 PARALLEL SCALABILITY OF KRYLOV-SCHWARZ

Practical scalable parallelism is one of the major motivations for research on and implementation of domain decomposition methods in CFD, the operative buzzwords being "faster, bigger, and cheaper." Though it would be premature to attempt to draw conclusions about the optimal algorithm/architecture combination for a given CFD analysis, we offer some experimental evidence for the excellent scalability of Krylov-Schwarz methods on a simple problem for which it is relatively easy to isolate the factors that trade off against each other in any such study. Without digressing into the refined nomenclature of scability analysis [12], we loosely call an algorithm/architecture combination "scalable" if its parallel efficiency is constant asymptotically, in any of several coordinated limits of discrete problem size n and parallel granularity p.

For an iterative numerical method, in which the total execution time $T(n,p)$ is the product of an iteration count $I(n,p)$ with an average cost-per-iteration $C(n,p)$, it is useful to separate the parallel efficiency into two factors: numerical efficiency and implementation efficiency. Numerical efficiency η_n measures the degradation of the convergence rate as the problem is scaled, and implementation efficiency η_i measures the degradation in the cost per iteration as the problem is scaled. For instance, we may take $\eta_n \equiv I(n,1)/I(n,p)$ and $\eta_i \equiv C(n,1)/[p \cdot C(n,p)]$. The numerical efficiency is usually very difficult to predict for a nonlinear problem, particularly when refining the grid (increasing n) resolves new physics. However, for certain domain decomposition methods applied to model linear problems with smooth solutions, the relative numerical efficiency $I(n_1,p_1)/I(n_2,p_2)$, with $p_2 > p_1$ and $n_1/p_1 = n_2/p_2$, can be proved to be 100% asymptotically.

The proof relies on the link between the rate of convergence of Krylov methods and the condition number of the (preconditioned) operator $B^{-1}A$, and on the link between the condition number and the extremal eigenvalues in the symmetric case, which can be estimated by Rayleigh quotients. Upper and lower bounds on the condition number of $B^{-1}A$ may be constructed that are independent of the mesh cell diameter h and the subdomain diameter H, or that depend only upon their ratio. In turn, h and H can be inversely related to n and p in simple problems. The theory, which has evolved over a decade to cover nonsmooth, nonsymmetric and indefinite problems, as well as nonnested spaces, is digested among other places in [7] and [22]. Two such numerically optimal methods for the scalar self-adjoint elliptic problem are the two-level additive Schwarz method [8] and BPS-I [1], which is a wire-basket (Schur complement-based) method. Before presenting parallel CFD results, we illustrate the performance achievable by such methods on contemporary parallel systems.

Table 1: Fixed-size scalability results – Poisson problem

# grid cells	# proc.	# iter.	seconds	sec./iter.
262,144	1	1*	183.5	183.5
262,144	4	8	325.7	40.7
262,144	16	12	96.7	8.1
262,144	64	12	17.6	1.5

Table 2: Fixed-memory-per-node scalability results – Poisson problem

# grid cells	# proc.	# iter.	seconds	sec./iter.
16,384	1	1*	8.75	8.75
65,536	4	7	58.4	8.34
262,144	16	12	96.7	8.06
1,048,576	64	11	91.2	8.29

Implementation efficiencies have improved substantially since we conducted our first such study in 1985.

A message-passing code for the Poisson problem on a unit square described in [15] was converted to MPI [18] and ported to several machines. With convergence defined as five orders of magnitude reduction in (unpreconditioned) residual, as in [15], we tested both fixed-size scalability and fixed-memory-per-node scalability on an Intel Paragon with 1, 4, 16, or 64 subdomains, with one subdomain per processor. The results for a fixed-size 512×512 grid are shown in Table 1. Table 2 is based on a problem size that grows from 16K to 1M unknowns, with a 128×128 subdomain problem on every processor. (The third row is common to both tables.)

On each subdomain, a direct FFT-based method is employed, so that only one iteration is required in the uni-processor case. Asymptotically, approximately 12 iterations are required, independent of the granularity. As seen in the column "sec./iter." in Table 2, the implementation efficiency is near perfect in this granularity range. Problems can be solved in constant time as resolution and processing power are increased in proportion. (We expect implementation efficiency to degrade eventually at higher granularity, due to a sequential bottleneck in the coarse-grid part of the preconditioner.) Consulting the "sec./iter." column of Table 1, we note a super-unitary implementation efficiency – as p increases by a factor of 4, runtime decreases by more than a factor of 4. This is attributable to the cacheing or paging advantages of domain-based array blocking, which are clearly more important than communication effects in this range of n and p.

Table 3: Inter-architecture comparison – Poisson problem

Machine	# proc.	seconds
IBM SP2	16	65.2
Intel Paragon	64	91.2
SPARC Cluster	16	124.5

In general CFD applications, finding a cost-effective coarse-grid operator is not straightforward, and one is often resigned to a Schwarz-preconditioned operator that deteriorates in numerical efficiency as the granularity of the decomposition increases. In such cases, optimizing execution time as a function of granularity is difficult, apart from numerical experimentation.

The largest problem solved on the Paragon (a grid of 1024×1024) was also run on 16 nodes of an IBM SP2 and on 16 SPARCstations connected by Ethernet (during a relatively quiescent period of the CPUs and the network). The overall runtime results are shown in Table 3. It is apparent that large memory-per-node workstations on an Ethernet are competitive with more expensive machines built around proprietary dedicated-link interconnects (a mesh for the Paragon, a multi-stage bi-directional switch for the SP2). This is due to the relatively small communication-to-computation ratio for Krylov-Schwarz methods, and is encouraging for cost-effective large-scale CFD computations, at least for dedicated clusters. Parallel workstation cluster implementations [6] of structured-grid Euler problems reveal the vulnerability of highly synchronous algorithms, including Krylov methods with their frequent inner product calls, to a non-dedicated environment. The other three main sources of communication inefficiency in parallel algorithms, namely load imbalance, latency, and finite bandwidth, are believed to impose much less serious limits on the number of workstations that can be clustered together to solve PDEs than frequent synchronization of non-dedicated resources.

3 AERODYNAMICS APPLICATIONS

In this section, we present parallel numerical results for two different formulations of inviscid, subsonic compressible external flow over two-dimensional airfoils using Newton-Krylov-Schwarz. The formulation with greater fidelity to the flow physics is the set of Euler equations:

$$\nabla \cdot (\rho \mathbf{v}) = 0 \tag{10}$$

$$\nabla \cdot (\rho \mathbf{v}\mathbf{v} + pI) = 0 \tag{11}$$

$$\nabla \cdot ((\rho e + p)\mathbf{v}) \quad = \quad 0 \tag{12}$$

where ρ is the fluid density, $\mathbf{v}$ the velocity, p the pressure, and e the specific total energy, together with the ideal gas law, $p = \rho(\gamma - 1)(e - |\mathbf{v}|^2/2)$, where γ is the ratio of specific heats.

Under additional restrictive assumptions of irrotationality ($\mathbf{v} = \nabla\Phi$) and isentropy ($\nabla(p/\rho^\gamma) = 0$), one can derive [11] a scalar equation for the velocity potential Φ:

$$\nabla \cdot (\rho \nabla \Phi) = 0. \tag{13}$$

We report on a model computation based on (13), which has the advantage of being simple in structure, and hence relatively easy to test algorithmic varieties upon. We then report on a computation based on a state-of-the-art unstructured grid solver for (10–12).

3.1 Full Potential Flow

A nonlinear full potential code has been built as a "laboratory" for parallel algorithms for nonlinear elliptic problems. A two-level Schwarz preconditioner was implemented on top of an early version of the PETSc [10] library, offering variable-fill incomplete factorization, a variety of subdomain preconditionings, variable subdomain overlap, and a coarse grid of variable density. The coarse grid is not necessarily nested in the underlying decomposition into subdomains, which would be an impractical restriction in real world problems with adaptively refined meshes. Full details may be found in [5]; we summarize some representative trends.

Consider the scalar nonlinear BVP

$$\nabla \cdot (\rho(||\nabla\Phi||)\nabla\Phi) = 0,$$

where

$$\rho = \rho_\infty \left(1 + \frac{\gamma - 1}{2} M_\infty^2 (1 - \frac{q^2}{q_\infty^2})\right)^{\frac{1}{(\gamma-1)}},$$

and where $q \equiv ||\nabla\Phi||$. $M_\infty \equiv q_\infty/a_\infty$ is the free-stream Mach number. Boundary conditions of Neumann or Dirichlet type are derived from inviscid boundary conditions for the velocity. The simplest possible problem of aerodynamic interest is a thin nonlifting symmetric airfoil at zero angle of attack. The airfoil lies along the x-axis, where its shape is parameterized by $f = y(x)$. So-called transpiration boundary conditions permit a uniform grid to be employed on a rectilinear domain. A complete set of boundary conditions, adequate for testing the nonlinear algebraic solver, if not for extracting accurate results about the underlying continuous problem, is:

- Upstream and far field: $\Phi = q_\infty \cdot x$,
- Downstream: $\Phi_{,n} = q_\infty$,
- Symmetry: $\Phi_{,n} = 0$,
- On parameterized airfoil ($y = f(x)$): $\Phi_{,n} = -q_\infty f'(x)$.

See [25], which motivates our present study, for a more refined treatment of boundary conditions.

We study convergence rate and parallel efficiency as functions of the accuracy of the subdomain solvers, the overlap of the subdomains, the density of the coarse grid component of the preconditioner, and the granularity of the decomposition. All tests presented below are on a uniform grid of 250K unknowns (512×512) for flow over a NACA0012 airfoil at a free-stream Mach number of 0.5. In this problem, it is never necessary to use pseudo-time-stepping to reach the domain of convergence of Newton's method. The initial iterate is the simple uniform flow $\Phi(x, y) = \int_{x_0}^{x} q_\infty \mathrm{d}x$.

Each of Tables 4 through 6 examines the sensitivity of the convergence to a single preconditioner parameter with all others controlled. The decomposition of the domain into eight rectangular subdomains is also controlled. The total number of outer Newton steps, the accumulated number of inner GMRES steps, and the overall running time of the parallel computation are tabulated.

Table 4 shows the effect of varying the level of fill k in ILU(k). As k varies from 0 to the full discrete dimension of the local Jacobian matrix, the subdomain solves gain increasing exactness. However, there is a law of diminishing returns in convergence rate, and overall execution time actually increases after a minimum, as the cost per iteration begins to rise more rapidly than the number of iterations falls. It has been observed in other contexts for the same full potential equation [24] that Schwarz methods are forgiving of inexactness in the individual subdomain solves. Furthermore, a factorization with a fixed level of fill is increasingly accurate as the subdomain over which it is defined gets smaller. Thus, for fine-grained computations, it is not cost-effective to work too hard on the individual subdomains. In this table, the overlap is fixed at $3h$ and the coarse grid is 4×5 — fewer than one coarse-grid vertex for every 10,000 fine-grid vertices.

Table 5 shows the effect of varying the overlap between subdomains. The *ovlp* listed is the distance that each subdomain is extended into its neighbors; hence the overall zone of overlap is twice as wide. As overlap varies from one mesh cell diameter, h, to roughly half the subdomain diameter, convergence rate improves. (For additive methods, too much overlap can lead to deteriorating condition number, however.) Even in the regime of monotonically

Table 4: Effect of preconditioner level of fill – Full Potential problem

k	0	1	2	3	4	5
Newton	11	7	6	*5*	5	5
GMRES	494	277	222	*169*	167	166
Time	891.44	512.16	417.37	*325.01*	327.30	332.86

Table 5: Effect of subdomain overlap – Full Potential problem

ovlp	1	2	3	4	5
Newton	5	5	*5*	5	5
GMRES	117	92	*77*	66	64
Time	278.23	246.10	*222.75*	211.65	213.34

improving convergence with increase in overlap there is a law of diminishing returns, and overall execution time increases after a minimum, as the cost per iteration rises. In this table, an exact solver is used on all subdomains, and the coarse grid is 8×9.

Table 6 shows the effect of varying the coarse grid density, with exact subdomain solves and subdomain overlap of $3h$. Exceedingly modest coarse grids provide a major improvement over no coarse grid at all; however, this is a relatively smooth problem. As with the other parameters, the marginal benefit of effort spent on the coarse grid decreases rapidly and ultimately reverses as the cost per iteration overtakes improved convergence rate.

Table 7 explores the implementation scalability of the NKS method, as seen over the range of two successive doublings, from 8 to 16 and from 16 to 32 processors, for three fixed-size preconditioner combinations (corresponding to the parameter choices in the italicized columns of the first three tables). Dividing elapsed computation times by the number of GMRES inner iterations, to obtain an average cost per iteration on this fixed-size problem, yields the scalability results shown. There is a superunitary relative efficiency for one of the preconditioners, attributable to more favorable cache blocking. The fixed-size parallel

Table 6: Effect of coarse-grid density – Full Potential problem

Coarse Grid	0×0	2×3	4×5	6×7	8×9
Newton	5	4	5	*5*	5
GMRES	164	95	98	*89*	77
Time	373.32	240.82	259.60	*245.02*	243.24

Table 7: Fixed-size scalability results – Full Potential code on an IBM SP2 (three different preconditioners)

	$k = 3$ ILU fill level		$3h$ subdomain overlap		6×7 coarse grid density	
# proc.	sec./iter.	rel. eff.	sec./iter.	rel. eff.	sec./iter.	rel. eff.
8	1.92	(1.00)	2.89	(1.00)	2.75	(1.00)
16	1.01	0.95	1.39	1.04	1.59	0.86
32	0.55	0.87	0.73	0.99	0.89	0.77

efficiencies remain above 75% throughout the preconditioner parameter and granularity range considered.

3.2 Euler Flow

The problem of inviscid incompressible flow around a two-dimensional four-element airfoil in landing configuration was studied in terms of convergence rate and parallel performance in [23], and the same code was converted to NKS form for the present study. The details of the discretization are left to the original reference. From [23] we consider the vertex-based discretization with a first-order Roe scheme on the left (out of which we form $\tilde{J}_{low}^{-1}$), and a second-order Roe scheme on the right (which defines f_{high}). The flow is subsonic (Ma = 0.2), with an angle of attack of 5°. Adaptively placed unstructured grids of approximately 6,000 and 16,000 vertices were decomposed into from 1 to 128 load-balanced subdomains, including all power-of-two granularities in between. We report below on the problem of 6,019 vertices, with four degrees of freedom per vertex (giving 24,076 as the algebraic dimension of the discrete problem). This is certainly small by parallel computational standards, though it is probably reasonably adequate in two dimensions from a physical modeling point of view, since the unstructured grid is not restricted to quasi-uniformity, and mesh cells are concentrated into small regions between the airfoils requiring the greatest refinement. The clustering can be seen in Fig. 1, which shows just a near field subset of the grid. (The grid recedes into the far field with smoothly increasing cell sizes. If the entire grid is is scaled to the page size, the flaps are too small to be visible.)

Figure 2 compares the convergence histories of the defect correction and NKS solvers, over a range of time sufficient to that permit the reduction of the residual of the NKS method to drop to within an order of magnitude of ε_{mach}. Both solvers utilize a residual-adaptive setting of the CFL number (related to the size of the time step δt in the pseudo-transient code), known as "switched evolution/relaxation" (SER) [19]. Starting from some small initial

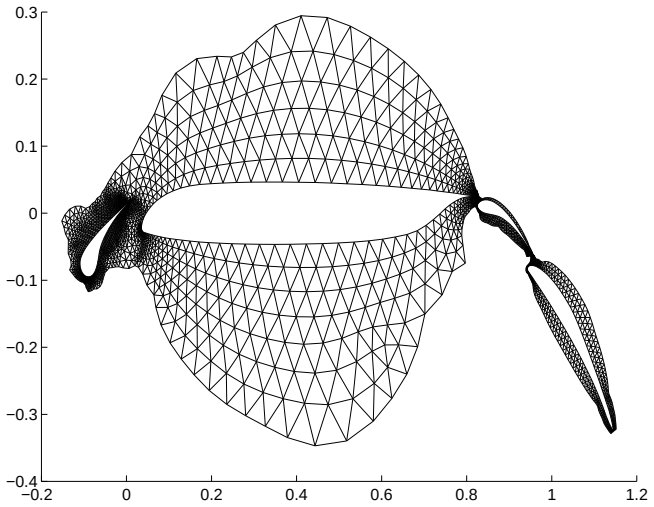

Figure 1: Zoom of the unstructured grid cells in the near field.

CFL number, CFL is adaptively advanced according to:

$$\mathrm{CFL}^{l+1} = \mathrm{CFL}^{l} \cdot \frac{||f(u)^{l-1}||}{||f(u)^{l}||}.$$

As $||f(u^l)|| \to 0$, $\delta t \to \infty$. In practice, it is wise to bound the relative growth of CFL in any one step by some factor and/or to bound it asymptotically in the range of 10^3 – 10^6 to preserve a modest diagonal dominance for the linear subiterations. Since convergence is not generally monotonic in $||f(u)||$, CFL may also adaptively decrease, and it should be ratcheted away from too large a relative decrease, as well.

Both solvers use the same Schwarz preconditioner, namely one-cell overlap and point-block ILU(0) in each subdomain. NKS is clearly superior to defect correction in convergence rate, though the cost per iteration is sufficiently high that defect correction is faster in execution time up to a modest residual reduction. (The cross-over point in the right plot is at about a reduction of 10^4 of the initial residual. A polyalgorithm, initially defect correction then switched to NKS when defect correction prohibits fast growth in CFL, may ultimately be much faster than either pure algorithm exclusively, as demonstrated for a related problem in [20], and as found in preliminary experiments for the present problem.) The asymptotic convergence rate is shown to be linear, since we truncated the Newton iterations well above the tolerances necessary to guarantee superlinear or quadratic convergence. Improving the constant in linear convergence is reason enough to use the matrix-free split discretization method (5). Table 8 compares the performance of the NKS version of the solver across three doublings of the processor force of the Intel Paragon for this fixed-size problem. The second-order evaluation of fluxes in $f_{high}(u)$ requires that first conserved variables, and later their fluxes, be communicated across subdomain boundaries each time the routine to evaluate

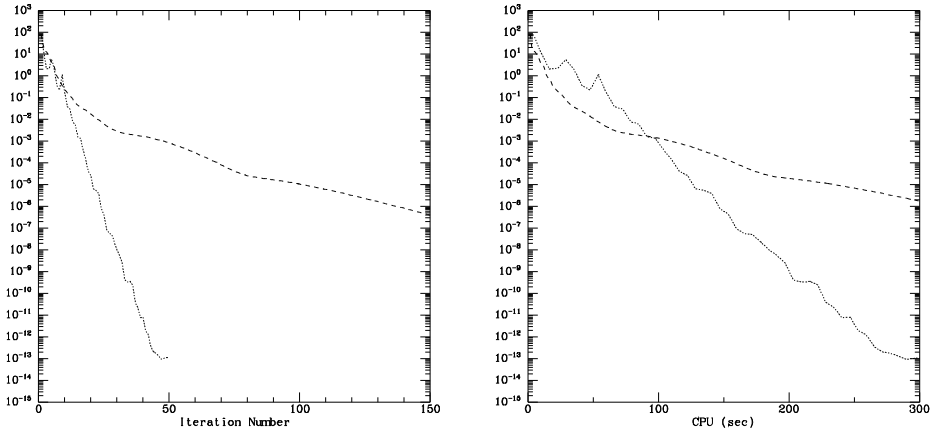

Figure 2: Norm of steady-state residual vs. iterations (left) and vs. execution time on 32 nodes of the Intel Paragon (right) for the defect correction scheme (dashed), and the NKS method (dotted).

the nonlinear residual is called. This imposes an extra communication burden per iteration on the matrix-free NKS solver, relative to a method that explicitly stores the elements of the Jacobian. Nevertheless, for residual norm reductions of more than a few orders of magnitude, the parallelized NKS solver is faster than the parallelized defect correction solver. The number of subdomains matches the number of processors, so convergence rate of the preconditioned system degrades slowly with increasing granularity, as coupling is lost in the preconditioner. However, the number of Krylov vectors per Newton iteration is bounded (at 2 restart cycles of 25 each), so the data translates directly to parallelization efficiency of the truncated Newton method.

In this example, no coarse grid is used, but [23] compares the defect correction form of the algorithm with and without a coarse grid. The coarse grid appears multiplicatively rather than additively, as in (1). The restriction operator consists of summing subdomain boundary fluxes and the prolongation operator is essentially piecewise constant subdomain extension followed by a boundary relaxation process. On the original platform of the Intel iPSC/2, the convergence rate advantage of the coarse grid is nearly completely cancelled by the sequential bottleneck. The coarse grid aspect of the preconditioner demands further attention.

Table 8: Wall-clock performance and relative parallel efficiency for unstructured Euler code on an Intel Paragon.

# proc.	sec./iter.	rel. eff.
4	36.09	(1.00)
8	19.21	0.94
16	10.65	0.85
32	6.25	0.72

4 CONCLUSIONS AND RELATED EXTENSIONS

We have shown that steady aerodynamics problems in two different formulations (full potential and Euler) can be effectively solved, and cost-effectively solved in parallel, by NKS methods.

The NK technique has been compared with V-cycle multigrid on Euler and Navier-Stokes problems without parallelizing the preconditioning in [14, 20]. For a subsonic unstructured grid example, NK trails multigrid in execution time by a factor of only about 1.5. This penalty can be accepted when it is realized that the NK method has the advantage of doing all of its computation without generation of a family of coarse unstructured grids (which is difficult for three-dimensional unstructured grids). This work has been extended to three-dimensional problems in [20].

Large-scale time-dependent problems suffering from multiple scales often require parallel implicit algorithms. The KS technique has been shown effective in the unsteady Navier-Stokes context in [3]. In [3], two of the same parameters explored herein (level of fill in the local ILU factorizations and subdomain overlap) are varied to produce a Schwarz preconditioner whose strength can be adjusted to adapt to the varying time-evolving ill-conditioning of the linear system arising at each implicit time step.

A variety of CFD applications are (or have inner) nonlinear elliptically-dominated problems amenable to solution by NKS algorithms, which are characterized by low storage requirements (for an implicit method) and locally concentrated data dependencies with small overlaps between the preconditioner blocks. The addition of a global coarse grid in the Schwarz preconditioner is often effective, where architecturally convenient. A deterrent to the widespread adoption of NKS algorithms is the large number of parameters that require tuning. Each component (Newton, Krylov, and Schwarz) has its own set of parameters, the most important of which, in our experience, is the convergence criterion for the inner Krylov subiterations. In large-scale, poorly preconditioned problems, including the test problems of this paper, tunings that guarantee quadratic convergence lead to unacceptable inner it-

eration counts and/or memory consumption. However, the plethora of parameters can be exploited, in principle, to produce optimal tradeoffs in space and time for a given problem class. Though parametric tuning is important to performance, conservative robust choices are not difficult.

References

[1] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring, I. *Mathematics of Computation*, 47:103–134, 1986.

[2] P. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal of Scientific and Statistical Computing*, 11:450–481, 1990.

[3] X.-C. Cai, C. Farhat, and M. Sarkis. Schwarz methods for the unsteady compressible Navier-Stokes equations on unstructured meshes. In R. Glowinski, J. Periaux, Z. Shi, and O. Widlund, editors, *Domain Decomposition Methods in Sciences and Engineering*, Chichester, 1996. Wiley.

[4] X.-C. Cai, W. D. Gropp, and D. E. Keyes. A comparison of some domain decomposition and ILU preconditioned iterative methods for nonsymmetric elliptic problems. *Journal of Numerical Linear Algebra and Applications*, 1:477–504, 1994.

[5] X.-C. Cai, W. D. Gropp, and D. E. Keyes. Parallel implementation of Newton-Krylov-Schwarz algorithms for the transonic full potential equation. ICASE Technical Report, in preparation, 1995.

[6] X.-C. Cai, W. D. Gropp, D. E. Keyes, and M .D. Tidriri. Newton-Krylov-Schwarz methods in CFD. In F. Hebeker and R. Rannacher, editors, *Proceedings of an International Workshop on Numerical Methods for the Navier-Stokes Equations*, pages 17–30, Braunschweig, 1994. Vieweg Verlag.

[7] T. F. Chan and T. Mathew. Domain decomposition algorithms. *Acta Numerica*, pages 61–143, 1994.

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

[9] W. D. Gropp and D. E. Keyes. Domain decomposition on parallel computers. *Impact of Computing in Science and Engineering*, 1:421–439, 1989.

[10] W. D. Gropp, L. C. McInnes, and B. F. Smith. PETSc 2.0 users guide. Technical Report ANL 95/11, Argonne National Laboratory, 1995.

[11] C. Hirsch, editor. *Numerical Computation of Internal and External Flows, Vol. 2*. Wiley, Chichester, 1990.

[12] K. Hwang. *Advanced Computer Architectures: Parallelism, Scalability, and Programmability (Chap. 3)*. McGraw-Hill, New York, 1993.

[13] H. Jiang and P. A. Forsyth. Robust linear and nonlinear strategies for solution of the transonic Euler equations. *Computers & Fluids*, 24:753–770, 1995.

[14] D. E. Keyes. Aerodynamic applications of Newton-Krylov-Schwarz solvers. In R. Narasimha, editor, *Proceedings of the 14th International Conference on Numerical Methods in Fluid Dynamics*, New York, 1995. Springer Verlag.

[15] D. E. Keyes and W. D. Gropp. A comparison of domain decomposition techniques for elliptic partial differential equations and their parallel implementation. *SIAM Journal of Scientific and Statistical Computing*, 8:s166–s202, 1987.

[16] D. E. Keyes and J. Xu, editors. *Proceedings of the Seventh International Conference on Domain Decomposition Methods*. Number 180 in Contemporary Mathematics. AMS, Providence, 1995.

[17] D. A. Knoll and P. R. McHugh. Inexact Newton's method solutions to the incompressible Navier-Stokes and energy equations using standard and matrix-free implementations. Technical Report 93-3332, AIAA, 1993.

[18] MPI Forum. MPI: A message-passing interface standard. *International Journal of Supercomputer Applications*, 8(3/4), 1994.

[19] W. Mulder and B. Van Leer. Experiments with implicit upwind methods for the Euler equations. *Journal of Computational Physics*, 59:232–246, 1985.

[20] E. J. Nielsen, R. W. Walters, W. K. Anderson, and D. E. Keyes. Application of Newton-Krylov methodology to a three-dimensional unstructured Euler code. Technical Report 95-1733, AIAA, 1995.

[21] Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing*, 7:856–869, 1986.

[22] B. F. Smith, P. E. Bjorstad, and W. D. Gropp. *Domain Decomposition: Parallel Multilevel Algorithms for Elliptic Partial Differential Equations.* Cambridge Univ. Press, Cambridge, 1995.

[23] V. Venkatakrishnan. Parallel implicit unstructured grid Euler solvers. *AIAA Journal*, 32:1985–1991, 1994.

[24] D. P. Young, C. C. Ashcraft, R. G. Melvin, M. B. Bieterman, W. P. Huffman, F. T. Johnson, C. L. Hilmes, and J. E. Bussoletti. Ordering and incomplete factorization issues for matrices arising from the TRANAIR CFD code. Technical Report BCSTECH-93-025, Boeing Computer Services, 1993.

[25] D. P. Young, R. G. Melvin, M. B. Bieterman, F. T. Johnson, S. S. Samant, and J. E. Bussoletti. A locally refined rectangular grid finite element method: Application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92:1–66, 1991.