

Parallel Implicit PDE Computations: Algorithms and Software

W. D. Gropp^a, D. E. Keyes^b, L. C. McInnes^c, and M. D. Tidriri^d

^a Mathematics and Computer Science Div., Argonne National Laboratory, Argonne, IL 60639-4844 USA, www.mcs.anl.gov/~gropp

^b Computer Science Dept., Old Dominion Univ., Norfolk, VA 23529-0162 USA, & ICASE, NASA LaRC, Hampton, VA 23681-0001 USA, www.cs.odu.edu/~keyes

^c Mathematics and Computer Science Div., Argonne National Laboratory, Argonne, IL 60639-4844 USA, www.mcs.anl.gov/~mcinnes

^d Mathematics Dept., Iowa State University, Ames, IA 50011-2064 USA

Implicit solution methods are increasingly important in applications modeled by PDEs with disparate time and spatial scales. Meanwhile, the demand for high resolution with reasonable turnaround requires “routine” parallelization for such applications. The pseudo-transient matrix-free Newton-Krylov-Schwarz (Ψ NKS) algorithmic framework is presented as an answer to these demands. We illustrate for three-dimensional transonic Euler flow about an M6 wing, on up to 64 processors of an IBM SP2 and a Cray T3E, that Ψ NKS can simultaneously deliver: (1) globalized, asymptotically rapid convergence through adaptive pseudo-transient continuation and Newton’s method, (2) high per-processor performance through attention to distributed memory and cache locality at the algorithmic level, and (3) reasonable parallelizability for an implicit method through favorable communication-to-computation scaling and deferred synchronization. Two disadvantages of Ψ NKS methods are their sensitivity to the coding of the underlying PDE discretization and the large number of parameters that must be selected to govern convergence behavior. We therefore distill several recommendations from our experience with various algorithmic components of Ψ NKS, and we describe a freely available, MPI-based portable parallel software implementation of the solver employed in our results.

1. Introduction

Disparate time and spatial scales abound in CFD applications. When convergence to a low-residual steady state is sought, such as when CFD analyses are used in computational design and optimization, a Newton method is asymptotically cost effective, and the ability to solve implicit linear systems with a true Jacobian can also be exploited in deriving sensitivities. Newton methods for PDEs must usually be robustified through a continuation scheme, such as pseudo-transience [6], and require the solution of large, sparse nonsymmetric linear systems, to which we apply Krylov methods such as GMRES [8]. In order to control the number of Krylov iterations, while obtaining concurrency pro-

portional to the number of processors, we precondition with domain-decomposed additive Schwarz methods [3].

Effective use of Ψ NKS in CFD codes requires attention to several details. We describe the sensitivity of the methodology to: the implicitness of the boundary conditions, the convergence of the inner Krylov iterations, and the aggressiveness of the pseudo-transient continuation. Some of these considerations are generic to any system modeled by PDEs.

Various aspects of the Ψ NKS framework have been pioneered by others over the past two decades. While the present work contributes some architecturally-oriented software advances, our principal goal is to integrate collective algorithmic progress and to reconcile trade-offs between interrelated algorithmic components so as to promote robustness, rapid convergence, and parallel scalability in the context of an important family of applications. We believe that the most important capabilities of Ψ NKS algorithms have been brought together in freely available, widely portable parallel software for the first time in the PETSc package [1], through which the illustrative results of this paper have been obtained.

2. Algorithmic Framework

Ψ NKS methods are designed to solve steady-state systems of nonlinear boundary value problems discretized as $f(u) = 0$, where u is a vector of unknowns representing the state of the system (typically nodal values of multiple fields defined at the same set of grid locations) with solution u^* , and $f(u)$ is a vector-valued function of residuals of the governing equations. We are primarily interested in three-dimensional settings in which the number of gridpoints is 10^5 – 10^6 , and the number of components of u is correspondingly larger by a factor of five or more. We consider a problem on a mapped, structured grid. Work on tetrahedral grids is also complete and will appear elsewhere.

Pseudo-transient continuation solves the steady-state problem $f(u) = 0$ through a sequence of problems $g_\ell(u) \equiv \frac{1}{\tau_\ell}(u - u^{\ell-1}) + f(u) = 0$, $\ell = 1, 2, \dots$, derived from a method-of-lines model, $\frac{\partial u}{\partial t} = -f(u)$, each of which is solved (approximately) for u^ℓ . The physical transient is followed when the time step τ_ℓ is sufficiently small, which provides a feasible sequence of states. Furthermore, the Jacobians associated with $g_\ell(u) = 0$ are well-conditioned when τ_ℓ is small. τ_ℓ is advanced from $\tau_0 \ll 1$ to $\tau_\ell \rightarrow \infty$ as $\ell \rightarrow \infty$, so that u^ℓ approaches the root of $f(u) = 0$. It should be emphasized that pseudo-transient continuation does *not* require reduction in $\|f(u^\ell)\|$ at each step as do typical linesearch or trust region globalization strategies [5]; it can climb hills.

The timestepping scheme determines more than the nonlinear robustness. The linear conditioning, and hence the strength of preconditioner and number of inner iterations required, are inherited from the timestepping. Our choice is a modified Successive Evolution-Relaxation (SER) [7], which (after an initial frozen phase) lets the timestep grow in inverse proportion to residual norm progress: $\tau_\ell = \tau_{\ell-1} \cdot \|f(u^{\ell-2})\|/\|f(u^{\ell-1})\|$, within bounds relative to the current step. The globalization theory of [6] for SER employs a three-phase approach (small timestep approach to the Newton convergence domain, timestep buildup, and Newton-like asymptotic tail), the main result of which is that there is either convergence from u_0 to u^* , or an easily detectable contraction of τ_ℓ towards 0, allowing recovery actions.

We use an “inexact” Newton method, solving $f(u) = 0$ through a sequence $u^\ell =$

$u^{\ell-1} + \alpha \cdot \delta u^\ell$, where δu^ℓ approximately satisfies the true Newton correction equation $f'(u^{\ell-1}) \delta u^\ell = -f(u^{\ell-1})$, in the sense that the linear residual norm $\|f'(u^{\ell-1}) \delta u^\ell + f(u^{\ell-1})\|$ is sufficiently small. The right-hand side of the linear Newton correction equation, $f(u^{\ell-1})$, is evaluated to full discretization order, so the inexactness arises from incomplete convergence or from the employment of an inexact Jacobian. The Jacobian may be approximated at various levels: by evaluating it to a lower discretization order, by lagging it to a previous iteration, or by approximating the functional differentiation by vector finite differences. Inexact Newton methods require a strategy for terminating the inner linear iterations, in effect choosing η_ℓ in $\|f(u^{\ell-1}) + f'(u^{\ell-1})(u - u^{\ell-1})\| \leq \eta_\ell \|f(u^{\ell-1})\|$. Following extensive experimentation, we find that a fixed (modest) relative reduction in the linear residual is most efficient for our test cases.

A Newton-Krylov (NK) method uses a Krylov method to solve the Newton correction equation for δu^ℓ . A Newton-Krylov-Schwarz method combines an NK method with a Krylov-Schwarz (KS) method. 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 $B^{-1}Ax = B^{-1}b$ through the action of a preconditioner, B , whose inverse action approximates that of A , but at smaller cost. 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 through a conveniently computable approximation to a local Jacobian. Such Schwarz-type preconditioning provides good data locality for parallel implementations over a range of parallel granularities, allowing significant architectural adaptability.

The Restricted Additive Schwarz Method (RASM) [4] eliminates interprocess communication during the interpolation phase of conventional additive Schwarz. In particular, let a PDE domain be decomposed into a set of possibly overlapping subdomains Ω_i and express the extraction from the global discrete representation of the components in subdomain i by the Boolean matrix R_i , with corresponding extension operator R_i^T . Then RASM can be expressed as $B_{RASM}^{-1} = \sum_i R_i^T \tilde{A}_i^{-1} R_i$, where A_i is the subdomain stiffness matrix $R_i A R_i^T$, \tilde{A}_i^{-1} is an approximate inverse within the i^{th} subspace, and R_i' is a truncated form of R_i with no overlap. The three-phase process of applying B_{RAS} to a global vector distributed over the subdomains consists of collecting data from local and neighboring subdomains, performing a local linear solve on each subdomain, and retaining only locally owned updates. Each subdomain processes its portion of the output concurrently. This leads to parallelism on the scale of the number of subdomains, and it also leads to efficient execution on each subdomain, since the working set of data for \tilde{A}_i^{-1} is smaller than for a global preconditioner. The resulting enhanced performance of cache-based microprocessors is as compelling a reason as concurrency for Schwarz methods.

3. Compressible Inviscid Flow Model

To illustrate the Ψ NKS algorithm in the parallel context and the feasibility of harvesting valuable discretization modules of sequential legacy codes with this algorithm, we solve the three-dimensional compressible Euler equations on mapped, structured grids using a second-order, Roe-type, finite-volume discretization. The governing system of PDEs and its discretization is standard and has been authoritatively described in [13]. The basis for

our implementation is a sequential Fortran77 code from D. Whitfield. Out of the box, the JULIANNE code includes a discrete Newton-relaxation pseudo-transient continuation solver with explicit enforcement of boundary conditions. We retained the discretization as embodied in flux balance routines for steady-state residual construction and finite-difference Jacobian construction. The function evaluations are undertaken to second order in the upwinding scheme. The Jacobian matrix (used mainly as a preconditioner, but also tested as an explicit Jacobian) is evaluated to first order. We replaced the explicit characteristic boundary conditions with fully implicit characteristic variants [10] and added the differentiable Van Albada limiter option to three existing limiters. The structure of the C-H grid around our test ONERA M6 wing may be inferred from Fig. 1.

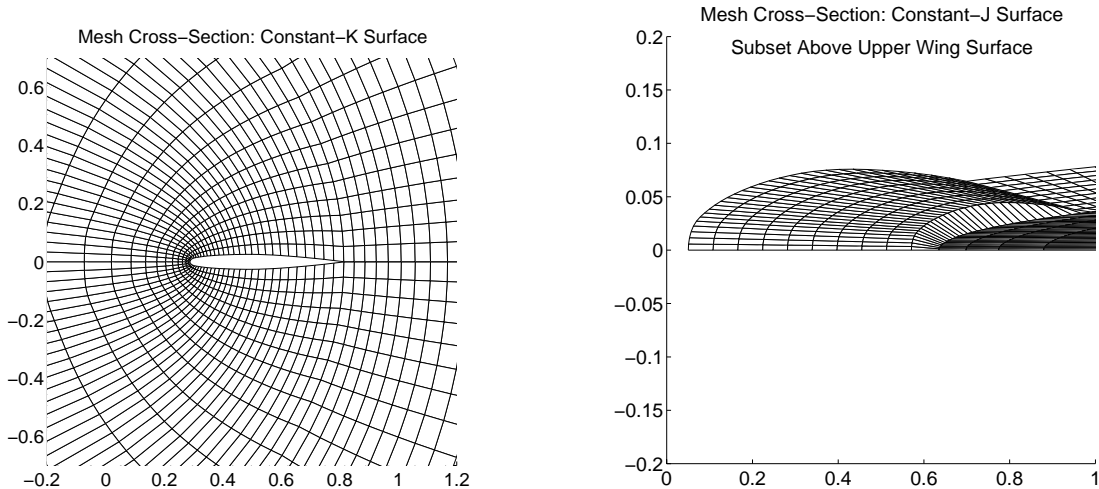


Figure 1. Constant-coordinate cuts of the “medium” grid: (a) a clipped cross-section of the constant- k surface five cells away from the wing root; (b) a clipped perspective of the constant- j surface on the upper side of the wing, looking in from the wing tip. (Index i wraps around the wing streamwise.)

For the freestream and impermeable wing surfaces, we use locally one-dimensional characteristic variable boundary conditions. See [12] for the derivation of the explicit forms of the BCs and [10,11,13] for the importance of using an implicit form of these boundary conditions to maintain stability as timesteps are adaptively increased. All of the boundary conditions for the ghost vertex unknowns are local (involving, at most, values at immediately interior vertices), with the exception of the C-cut ghost-to-interior identity mappings across the trailing edge streamline. For traditional lexicographic orderings of the gridpoints, the entries of the Jacobian matrix that tie together these spatially identical but logically remote degrees of freedom will lie outside of the normal band structure, and would generate extensive fill in a full factorization. Since we employ a special diagonal block data structure in our algebraic solvers, these nonzeros appear in the matrix-free

action of the Jacobian only, not in the Jacobian preconditioner.

4. Parallel Implementation Using PETSc

The parallelization paradigm we recommend in approaching a legacy code is a compromise between the “compiler does all” and the “hand-coded by expert” approaches. We employ the “Portable, Extensible Toolkit for Scientific Computing” (PETSc) [1,2], a library that attempts to handle in a highly efficient way, through a uniform interface, the low-level details of the distributed memory hierarchy. Examples of such details include striking the right balance between buffering messages and minimizing buffer copies, overlapping communication and computation, organizing node code for strong cache locality, preallocating memory in sizable chunks rather than incrementally, and separating tasks into one-time and every-time subtasks using the inspector/executor paradigm. The benefits to be gained from these and from other numerically neutral but architecturally sensitive techniques are so significant that it is efficient in both the programmer-time and execution-time senses to express them in general purpose code.

PETSc is a large and versatile package integrating distributed vectors, distributed matrices in several sparse storage formats, Krylov subspace methods, preconditioners, and Newton-like nonlinear methods with built-in trust region or linesearch strategies and continuation for robustness. It has been designed to provide the numerical infrastructure for application codes involving the implicit numerical solution of PDEs, and it sits atop MPI for portability to most parallel machines. The PETSc library is written in C, but may be accessed from user codes written in C, Fortran, and C++. PETSc version 2, first released in June 1995, has been downloaded thousands of times by users worldwide. PETSc has features relevant to computational fluid dynamicists, including matrix-free Krylov methods, blocked forms of parallel preconditioners, and various types of time-stepping.

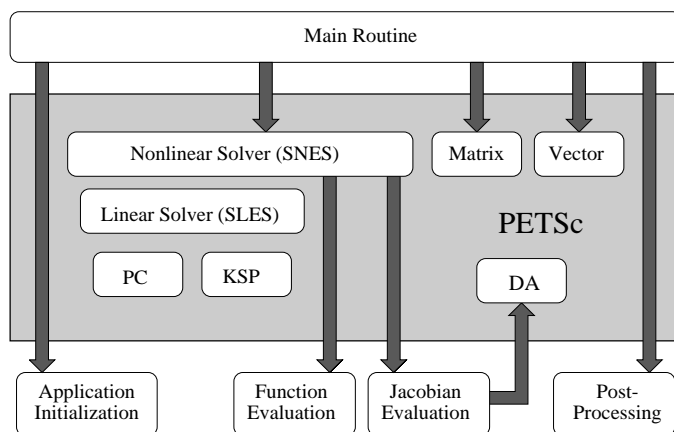


Figure 2. Coarsened calling tree of the JULIANNE-PETSc code, showing the user-supplied main program and callback routines for providing the initial nonlinear iterate, computing the nonlinear residual vector at a PETSc-requested state, and evaluating the Jacobian (preconditioner) matrix.

A diagram of the calling tree of a typical Ψ NKS application appears in Fig. 2. The arrows represent calls that cross the boundary between application-specific code and PETSc library code; all other details are suppressed. The top-level user routine performs I/O related to initialization, restart, and post-processing and calls PETSc subroutines to create data structures for vectors and matrices and to initiate the nonlinear solver. PETSc calls user routines for function evaluations $f(u)$ and (approximate) Jacobian evaluations $f'(u)$ at given state vectors. Auxiliary information required for the evaluation of f and $f'(u)$ that is not carried as part of u is communicated through PETSc via a user-defined “context” that encapsulates application-specific data. (Such information typically includes dimensioning data, grid data, physical parameters, and quantities that could be derived from the state u , but are most conveniently stored instead of recalculated, such as constitutive quantities.)

5. Numerical Experiments

We illustrate in this section the importance of some of the design decisions above in the context of a legacy CFD application that has been parallelized and accelerated through Ψ NKS. We briefly describe the test problems and then discuss issues of boundary conditions, preconditioner quality, convergence tuning, and scalability.

The ONERA M6 wing is a standard three-dimensional test case, for which extensive experimental data is given in [9]. A frequently studied parameter combination combines a freestream Mach number of 0.84 with an angle of attack of 3.06° . This transonic case gives rise to a characteristic λ -shock, as depicted with in Fig. 3. Tests on three different

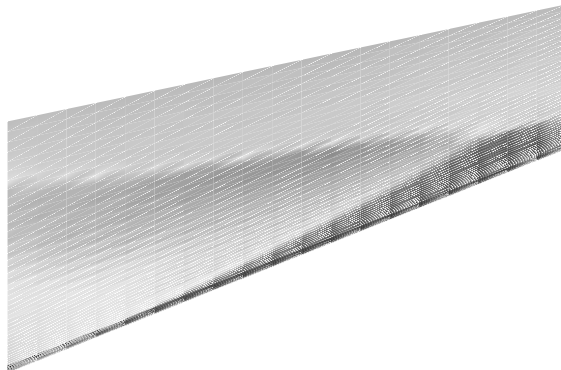


Figure 3. Mach number contours (the local tangential velocity magnitude divided by the local sound speed) of the converged flowfield on the upper wing surface for the finest grid.

successively doubled grid resolutions (“coarse” $50 \times 10 \times 10$, “medium” $98 \times 18 \times 18$, and “fine” $194 \times 34 \times 34$) demonstrate the grid-independence of the solution on the last two grids. Comparison of the aerodynamic coefficients with the data in [9] shows good agreement for an inviscid model.

We employ only the simplest decomposition strategy in the Schwarz preconditioner:

recursive bisection cuts along the i , j , and k coordinate directions. The order of the cuts is determined so as to produce logically rectilinear subdomains that are as close to unit aspect ratio as possible, a strategy that minimizes communication volume.

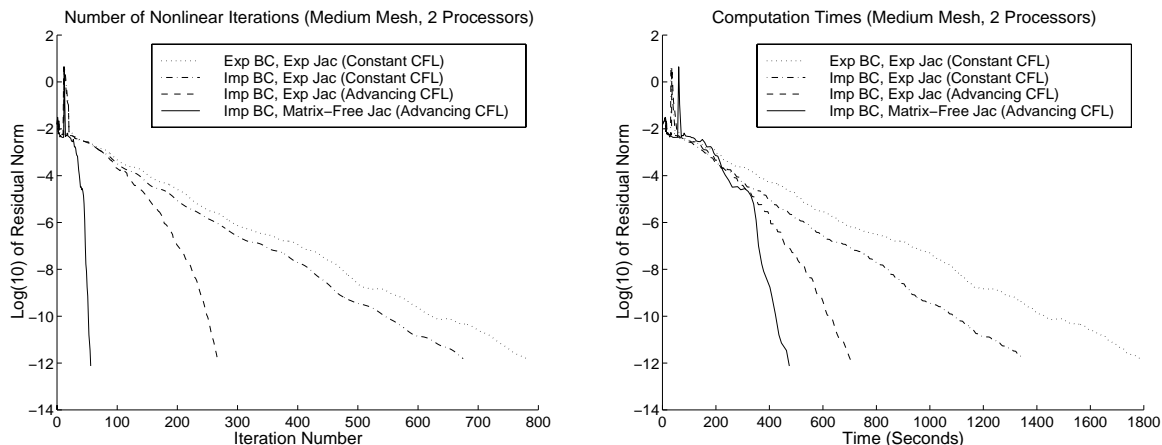


Figure 4. Comparison of four globalized solution algorithms, showing the cumulative advantages of implicit BCs, advancing CFL, and matrix-free representation of the Jacobian, in terms of both iteration count (number of pseudo-timesteps) and overall execution time.

Figure 4 compares the convergence in terms of both nonlinear iterations and total time (on two processors of an SP2) of the various approaches under consideration: explicit boundary conditions (limiting CFL to 7.5), implicit BCs with the same CFL, implicit BCs with advancing CFL, and implicit BCs with advancing CFL and matrix-free application of the Jacobian. The data is for the medium grid problem from a uniform flow initial condition; the relative behavior of the four algorithms is representative of all grid sizes. In all cases throughout this paper, the explicit Jacobian is computed to first order accuracy in space via finite differences, stored in a block sparse format, and refreshed once every ten pseudo-timesteps. This Jacobian serves as the left-hand-side matrix of the Newton systems, or as the preconditioner for the matrix-free method. This frequency of Jacobian recomputation provides a reasonable tradeoff in terms of convergence rate and the computational cost of matrix evaluation. In all cases, the flow is allowed to “settle in” before the boundary of the airfoil is rendered impermeable in the tenth pseudo-timestep, accounting for the sudden jumps in the residual norm convergence plots.

For the cases in Fig. 4, we solved the linearized Newton systems approximately with a relative linear convergence tolerance of 10^{-2} using preconditioned GMRES. For the matrix-free variant we employed the additive Schwarz preconditioner with overlap 1, and for the other cases we used the cheaper but sufficiently powerful subdomain-block Jacobi method; both versions used one subdomain per processor, solved with point-block ILU(0). When retaining a fixed CFL of 7.5, only 3–4 linear iterations were required for each step to reach the tolerance for both explicit and implicit boundary condition variants. Experiments indicate that due to the mismatch of first-order and second-order schemes

in the left- and right-hand sides of these defect correction methods, more accurate linear solves and more frequent Jacobian refreshment do not appreciably accelerate convergence. For the cases with increasing CFL, the linear systems become more challenging to solve as they transition toward true Newton systems, requiring up to 30 iterations to reach the specified tolerance.

The key observation from this data is that the combination of implicit boundary conditions coupled with the higher-order discretization enabled by the matrix-free technique delivers solution of the nonlinear problem to machine precision several times faster than using standard defect correction methods. Neither the use of implicit boundary conditions alone nor the use of increasing CFL with a low-order Jacobian allows quadratic convergence.

Preconditioner quality dramatically impacts the overall efficiency of the parallel Ψ NKS methodology, as demonstrated in the closely related Figs. 5 and 6 for three variants of additive Schwarz techniques. These figures compare convergence rate (in terms of residual norm) versus both nonlinear iteration number and time for the medium grid on 16 processors; analogous results are achieved for other problem sizes and processor configurations. We contrast in Fig. 5 a zero-overlap (subdomain-)block Jacobi preconditioner with a two-cell-overlap standard Additive Schwarz Method (ASM) and a two-cell-overlap Restricted Additive Schwarz Method (RASM). We observe that a modest degree of overlap is crucial in providing a quality preconditioner, which in turn allows the overall scheme to progress rapidly in the nonlinear sense. Further, we see that the RASM method, which eliminates communication during the interpolation phase, not only saves time in terms of communication overhead, but also provides more powerful preconditioning, as evidenced by faster convergence in terms of nonlinear iteration count!

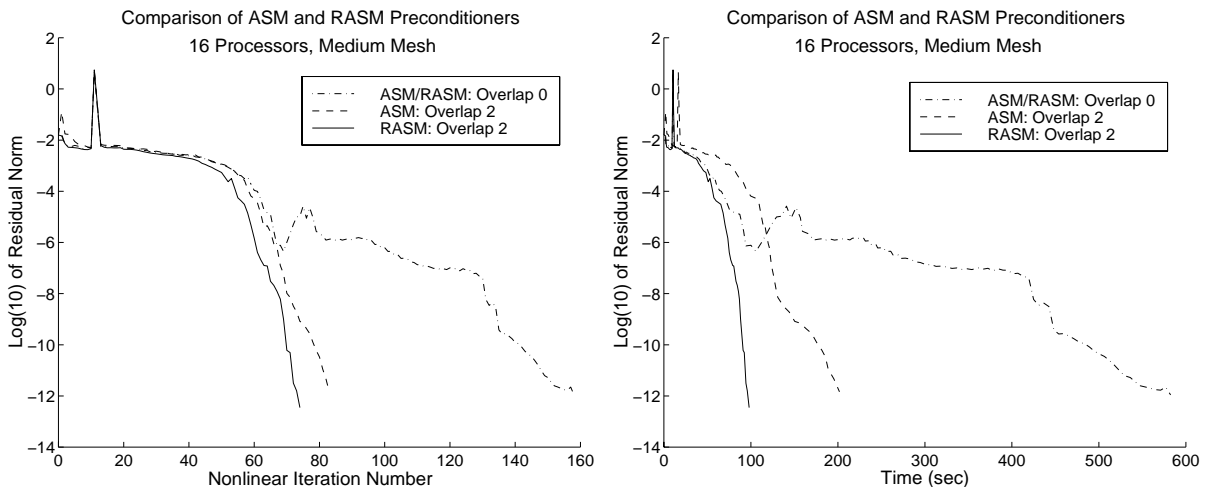


Figure 5. Comparison of three domain-decomposed preconditioners: subdomain-block Jacobi, standard Additive Schwarz with overlap of 2 cells, and Restricted Additive Schwarz with overlap of 2 cells. All methods use point-block ILU(0) on the 16 subdomains.

Figure 6 contrasts various degrees of overlap for RASM, with the solid curves in common with those of Fig. 5. In particular, we see that two-cell overlap provides a good balance in terms of power and cost. Less overlap trades off cheaper cost per iteration for a preconditioner that does not allow the nonlinear iterations to converge as rapidly, while more overlap is costly to apply and does not sufficiently contribute to faster nonlinear convergence.

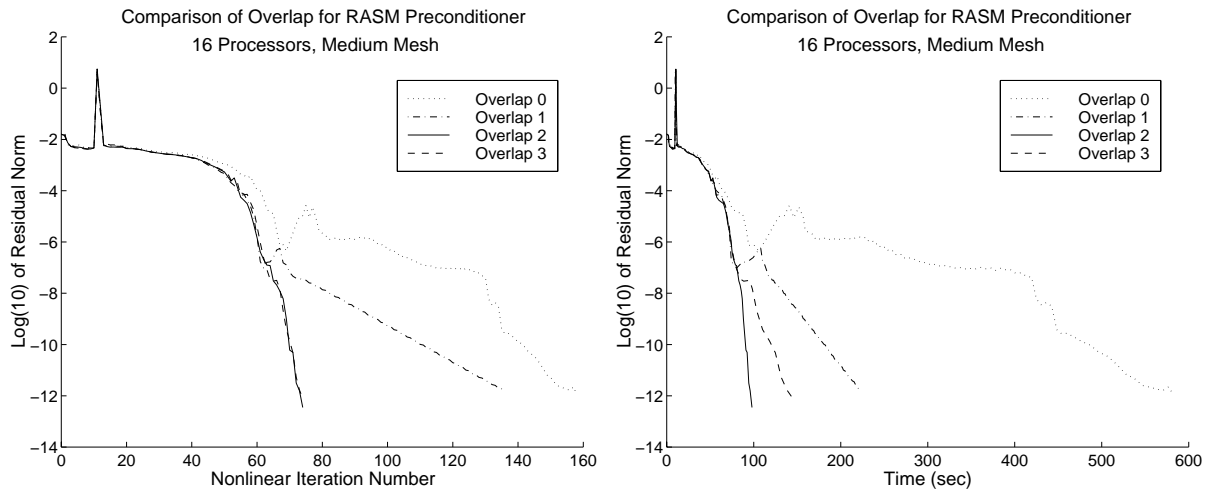


Figure 6. Comparison of 4 domain-decomposed preconditioners: subdomain-block Jacobi and restricted Additive Schwarz with overlap of 1, 2, and 3 cells. All methods solve point-block ILU(0) on the 16 subdomains.

Preconditioner quality and convergence tuning for terminating the inner Krylov iterations in the Newton correction are tightly coupled and must be considered in concert. Figure 7 contrasts various convergence criteria using the Restricted Additive Schwarz preconditioner with overlap of 2. This figure presents convergence data in terms of solution times and work per linear iteration for three methods: fixed Krylov subspace dimension, fixed relative tolerance for residual norm reduction, and a hybrid strategy that terminates upon the earlier of satisfying an adaptive residual norm strategy or exceeding a Krylov subspace dimension. The data is for the 16-processor fine-grid case, but is representative of experiments on various numbers of processors for other grids. The left-hand-side graph within Fig. 7, which plots the log of the residual norm versus solution time, indicates that a fixed loose fixed relative tolerance of 10^{-2} is preferable over the alternatives when a sufficiently robust preconditioner is employed.

The corresponding right-hand-side graph better illustrates where along the overall nonlinear simulation process the bulk of the linear computational work occurs. In particular, we see that the common practice of specifying a fixed amount of work per nonlinear iteration does not fare as well as alternatives that focus linear solver work where it is truly needed during the second and third phases [6] of the Ψ NKS process. The fixed-work case for this problem employed 30 iterations of GMRES; experiments with fewer iterations and

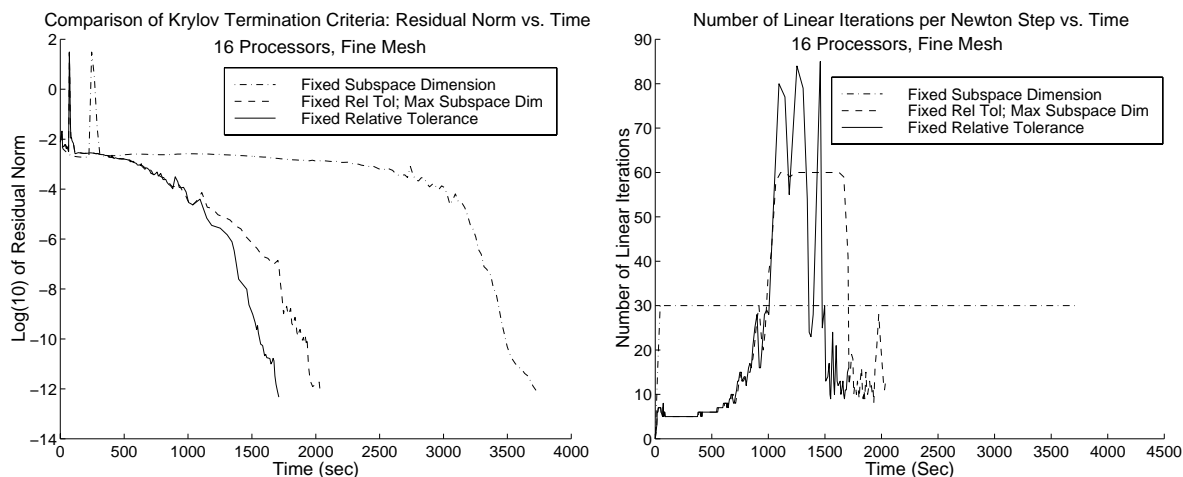


Figure 7. Comparison of three different tunings of the linear convergence tolerance: fixed work, fixed relative tolerance, and a loose hybrid.)

more frequent restarts encountered stagnation before reaching steady state. In contrast, the alternatives using a fixed relative tolerance or adaptive hybrid fared much better.

We report computation rates on the IBM SP2 for the matrix-vector product and an entire linear solve using an explicitly stored Jacobian, with implicit boundary conditions averaged over a fixed number of Newton corrections of a particular pseudo-timestep. The linear Newton systems are solved using GMRES and block Jacobi preconditioning, where each processor has one block that is solved with ILU(0). For the fine-grid problem with 1,121,320 unknowns, the computation rate on 16 processors for the matrix-vector product is 1.28 Gflop/s, while the complete linear solve achieves 1.01 Gflop/s. On 64 processors the matrix-vector product runs at 4.22 Gflop/s (a speedup of 3.3 out of 4), while the complete linear solve achieves 3.74 Gflop/s (a speedup of 3.7 out of 4). The corresponding *parallel sustained* computation rates of 60 to 80 Mflop/s per processor show that the SPMD node code for this sparse PDE problem resides well in cache, thanks to domain blocking. (Recall that the LINPACK-100 benchmark with dense BLAS3 arithmetic delivers 130 Mflop/s on a single sequentially programmed node of the SP2.)

The scalability of the matrix-free variant is presented in Fig. 8 and Table 1. Fig. 8 shows the scaling of the complete nonlinear simulation for the fine grid on 16, 32, and 64 processors of an IBM SP with 120 MHz P2SC nodes with two 128 MB memory cards each and a TB3 switch. The data indicate a modest decrease in nonlinear convergence rate as the number of processors grows; overall solution times scale reasonably well.

Finally, Table 1 compares scalability on two contemporary high-performance machines, the IBM SP2 and Cray T3E. Times are given for a single matrix-free NKS iteration on the fine grid, including evaluation of the right-hand side with the linear solve. We consider 16-, 32-, and 64-processor configurations, and present speedup over the 16-processor case.

We have shown that the pseudo-transient matrix-free Newton-Krylov-Schwarz (Ψ NKS) methodology can be effective for the parallel solution of large-scale CFD problems, as ex-

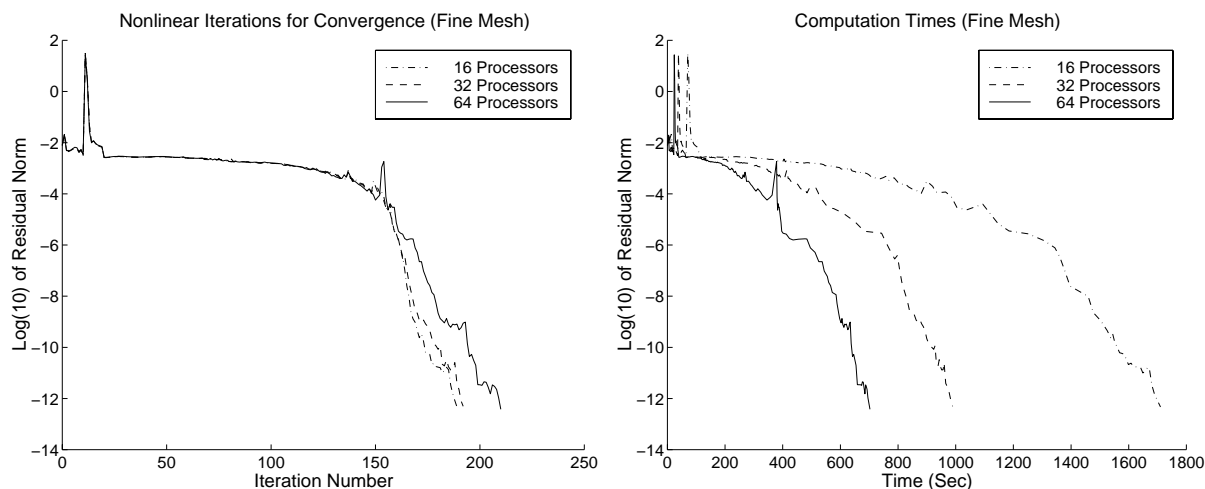


Figure 8. Scalability of the nonlinear solver (fine grid).

Table 1

Scalability on the SP2 and T3E: times for one nonlinear iteration on the fine grid.

| Machine | Number of Processors | Time (Sec) | Speedup over 16 Processors |
|----------|----------------------|------------|----------------------------|
| IBM SP2 | 16 | 4.39 | — |
| | 32 | 2.46 | 1.8 |
| | 64 | 1.60 | 2.7 |
| Cray T3E | 16 | 5.15 | — |
| | 32 | 2.71 | 1.9 |
| | 64 | 1.71 | 3.0 |

emphified by the three-dimensional Euler equations. For a given problem class, the various parameters that affect algorithmic performance can systematically be studied and tuned; and algorithmic comparisons herein show how any one of several mis-tuned functionalities can cause convergence to be far slower than the best attainable. In our experience, conversion of a legacy code to the matrix-free NKS framework is a “filtering” process for the code, just as conversion to a multigrid solver is often said to be. During conversion, unrecognized explicit updates may be discovered, and unrecognized nondifferentiability or destabilizing sensitivity may be detected in gradient computation.

We believe that an approach to developing software that is simultaneously capable of flexibly integrating innovative numerical methods and porting them to emerging architectures is crucial for cost-effective maintenance of long-term high-performance simulation capability. Parallel implicit Newton-like solvers open the door to fast, low-residual analyses, which should have dramatic implications for multiscale problems and for the practice of large-scale PDE-constrained optimization.

Acknowledgements

The authors owe a large debt of gratitude to David Whitfield of the Engineering Research Center at Mississippi State University for providing JULIANNE as the legacy code that drove several aspects of this work. Satish Balay and Barry Smith of Argonne National Laboratory co-developed the PETSc software employed in this paper, together with Gropp and McInnes, under the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38. Dr. George Lea of The National Science Foundation supported the work of Keyes and McInnes at Old Dominion University under ECS-9527169, and all four authors have collaborated while in residence at ICASE under NASA contract NAS1-19480, where early results of this paper were employed in a “Bring Your Own Code” Workshop on the Parallelization of PDE-based Codes in December, 1996. Time on the NASA SP2s was provided under the Computational Aero Sciences section of the HPCCP.

REFERENCES

1. Satish Balay, William Gropp, Lois Curfman McInnes, and Barry Smith. PETSc 2.0 users manual. Technical Report ANL-95/11 - Revision 2.0.17, Argonne National Laboratory, October 1996.
2. Satish Balay, William Gropp, Lois Curfman McInnes, and Barry Smith. PETSc home page. <http://www.mcs.anl.gov/petsc/petsc.html>, August 1997.
3. X.-C. Cai. Some domain decomposition algorithms for nonselfadjoint elliptic and parabolic partial differential equations. Technical Report 461, Courant Institute, September 1989.
4. X.-C. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. Technical report, Computer Science Department, University of Colorado-Boulder, 1997.
5. J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, 1983.
6. C. T. Kelley and D. E. Keyes. Convergence analysis of pseudo-transient continuation. Technical Report 96-46, ICASE, 1996. *SIAM J. Num. Anal.*, to appear.
7. W. Mulder and B. Van Leer. Experiments with implicit upwind methods for the Euler equations. *J. Comp. Phys.*, 59:232–246, 1985.
8. Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.
9. V. Schmitt and F. Charpin. Pressure distributions on the ONERA M6 wing at transonic mach numbers. Technical Report AR-138, AGARD, May 1979.
10. M. D. Tidriri. Krylov methods for compressible flows. Technical Report 95-48, ICASE, June 1995.
11. M. D. Tidriri. Schwarz-based algorithms for compressible flows. Technical Report 96-4, ICASE, January 1996.
12. D. Whitfield and J. M. Janus. Three-dimensional unsteady Euler equations using flux vector splitting. In *Proceedings of the AIAA 17th Fluid Dynamics, Plasma Dynamics, and Lasers Conference*, 1984.
13. D. Whitfield and L. Taylor. Discretized Newton-relaxation solution of high resolution flux-difference split schemes. In *Proceedings of the AIAA Tenth Computational Fluid Dynamics Conference*, pages 134–145, 1991.