

INEXACT NEWTON METHODS WITH RESTRICTED ADDITIVE SCHWARZ BASED NONLINEAR ELIMINATION FOR PROBLEMS WITH HIGH LOCAL NONLINEARITY*

XIAO-CHUAN CAI[†] AND XUEFENG LI[‡]

Abstract. The classical inexact Newton algorithm is an efficient and popular technique for solving large sparse nonlinear system of equations. When the nonlinearities in the system are well-balanced, a near quadratic convergence is often observed, however, if some of the equations are much more nonlinear than the others in the system, the convergence is much slower. The slow convergence (or sometimes divergence) is often determined by the small number of equations in the system with the highest nonlinearities. The idea of nonlinear preconditioning has been proven to be very useful. Through subspace nonlinear solves, the local high nonlinearities are removed, and the fast convergence can then be restored when the inexact Newton algorithm is called after the preconditioning. Recently a left preconditioned inexact Newton’s method was proposed in which the nonlinear function is replaced by a preconditioned function with more balanced nonlinearities. In this paper, we combine an inexact Newton with a restricted additive Schwarz based nonlinear elimination. The new approach is easier to implement than the left preconditioned method since the nonlinear function doesn’t have to be replaced, and further more, the nonlinear elimination step doesn’t have to be called at every outer Newton iteration. We show numerically that it performs well for, as an example, solving the incompressible Navier-Stokes equations with high Reynolds numbers and on machines with large number of processors.

Key words. Nonlinear elimination, inexact Newton method, nonlinear restricted additive Schwarz, domain decomposition, nonlinear equations, parallel computing, incompressible flows

1. Introduction. In this paper we consider scalable and robust parallel algorithms for solving large sparse nonlinear systems of equations arising from the discretization of partial differential equations. In particular, we focus on the type of systems in which a small number of equations are more nonlinear than the rest of the equations. This type of problems appears quite often in many areas of computational science and engineering. For example, if a Newton method is used in an implicit CFD calculation, it happens very often that when there is a local high nonlinearity such as a boundary layer or a corner singularity, the convergence stagnates by the small number of residual functions associated with the local high nonlinearities [5, 15, 20, 25, 28, 29]. Recent progress in developing nonlinear algorithms that are not too sensitive to local high nonlinearities include the ideas of nonlinear elimination and nonlinear preconditioning. In nonlinear elimination [19, 24], the local high nonlinearities are removed implicitly so that they don’t participate in the global Newton iterations which act only on a subset of the equations in the system. The idea of nonlinear preconditioning [6] is to reduce the global impact of these few highly nonlinear components using a local nonlinear preconditioner. The global Newton is used for the entire set of equations in the system. In this paper, we propose and study numerically a different nonlinear preconditioning technique in which the nonlinear function doesn’t need to be changed, as in ([6]), and we show that this approach is considerably easier to implement. Consider the nonlinear system of equations $F(u) = 0$, in this new approach, we introduce a nonlinear elimination technique a ‘right’ preconditioner which attempts

*The research was supported in part by DOE under DE-FC02-04ER25595, and in part by NSF under grants EAR-0934647, CNS-0722023, DMS-0913089.

[†]Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (cai@cs.colorado.edu).

[‡]Department of Mathematics and Computer Science, Loyola University New Orleans, New Orleans, LA 70118 (li@loyno.edu).

to make the variable u more acceptable as compared to the approach in ([6]) which uses a ‘left’ preconditioner on the nonlinear function $F(\cdot)$. Since $F(\cdot)$ doesn’t have to be changed at all, the new approach can be incorporated into existing software packages for solving nonlinear systems without requiring too much additional effort. Physically speaking, left preconditioning and the new nonlinear elimination carry different meanings. For example, if $F(u)$ is derived from fluid dynamics, u is often the basic flow field such as the velocity and F often represents the conservation laws. In this situation, left preconditioning means the change of the conservative quantities, such as the total mass, and nonlinear elimination means the change of the velocity field.

To construct the nonlinear elimination operator, we introduce a nonlinear version of the linear restricted additive Schwarz preconditioner [8], which has an interesting property that is not shared with its sibling algorithms such as the additive or the multiplicative Schwarz algorithms. Let us briefly describe this property which we refer to as the *reduced boundary effect property*, which is something we discovered while running the numerical experiments. Suppose the nonlinear function, denoted as $F(u)$, is the discretization of a second order partial differential equation defined on the domain Ω . Let ω be a subdomain in Ω . When defining the overlapping domain decomposition preconditioner, a subdomain problem has to be solved in ω with an artificial boundary condition imposed on $\partial\omega$. Let u_c be the current approximate solution on Ω and u_ω the solution on ω . After the subdomain solve, a new approximate solution can be defined as u_{new} which equals to u_ω in ω and u_c in $\Omega \setminus \omega$. For the new approximate solution the corresponding residual function $F(u_{new})$ has a very distinctive feature, that is, it has a large jump along the interface $\partial\omega$; see the right figure in Fig.2.1 for an illustrative one dimensional example. To reduce the interface jump on the boundary of ω , we introduce a larger domain ω' ($\omega \subset \omega' \subset \Omega$) and solve the subdomain problem defined on ω' and then only keep the part of the solution inside ω ; i.e., throw away the bad part of the solution in $\omega' \setminus \omega$. Of course, to have a good approximation for the whole domain, we need many such subdomain solves covering the entire Ω .

Note that by throwing away the bad part of the solution we obtain not only a better solution but also lower the communication cost when the algorithm is implemented on a distributed memory computer. The theoretical understanding of the nonlinear restricted additive Schwarz preconditioner is out of the scope of this paper, and we believe it may not be a trivial matter because a satisfactory theory for the linear restricted additive Schwarz preconditioner is yet to be established.

Consider a given nonlinear function $F : R^n \rightarrow R^n$, we are interested in finding a vector $u^* \in R^n$, such that

$$(1.1) \quad F(u^*) = 0,$$

starting from an initial guess $u^{(0)} \in R^n$. Here $F = (F_1, \dots, F_n)^T$, $F_i = F_i(u_1, \dots, u_n)$, and $u = (u_1, \dots, u_n)^T$. Inexact Newton algorithms (IN) [10, 13, 21] are commonly used for solving such systems and can briefly be described here. Suppose $u^{(k)}$ is the current approximate solution; a new approximate solution $u^{(k+1)}$ can be computed through

$$(1.2) \quad u^{(k+1)} = u^{(k)} + \lambda^{(k)} p^{(k)},$$

where the inexact Newton direction $p^{(k)}$ satisfies

$$(1.3) \quad \|F(u^{(k)}) + F'(u^{(k)})p^{(k)}\| \leq \eta_k \|F(u^{(k)})\|.$$

Here $\eta_k \in [0, 1)$ is a scalar that determines how accurately the Jacobian system needs to be solved using, for example, Krylov subspace methods [2, 3, 13, 14], and $\lambda^{(k)}$ is another scalar that determines how far one should go in the selected inexact Newton direction [13]. In an ideal case, $\lambda^{(k)} = 1$, which means that a full inexact Newton step is taken, however, in many practical situations, $\lambda^{(k)}$ is much smaller than 1, and this is often due to the fact that some dominate terms of the residual vector $F(u^{(k)})$ can't be sufficiently reduced by following the Newton search direction. Small $\lambda^{(k)}$ implies slow convergence, or stagnation, in some cases. We will show in the paper that once nonlinear elimination is used, full inexact Newton steps become acceptable even for some difficult problems, such as incompressible Navier-Stokes with high Reynolds numbers.

The rest of the paper is organized as follows. In section 2, we introduce the nonlinear additive Schwarz preconditioned system and show how it reduces in certain special cases to known methods. In section 3, we discuss the details of the algorithm. Numerical examples are given in section 4. Some concluding remarks are given in Section 5.

2. A nonlinear restricted additive Schwarz algorithm. Restricted additive Schwarz (RAS) was first introduced in [8] as a linear preconditioner for solving sparse linear system of equations. It is a modification of the classical additive Schwarz method (AS, [27]). Comparing with AS, RAS needs less communication, and often less number of iterations, when implemented on distributed memory computers. In this section, we extend RAS to nonlinear problems and the motivation is completely different from what we had for linear problems.

To motivate the work, we first discuss a nonlinear elimination algorithm [24] and some of its features. Let $F(\cdot)$ be the discretization of a one-dimensional nonlinear boundary value problem (Poisson, for example) defined on a domain $\Omega = (0, 1)$ and u_c be an approximate solution of the nonlinear system $F(u) = 0$ defined on Ω with the corresponding residual $F(u_c)$. To illustrate the situation, we provide a picture of u_c and $F(u_c)$ in Fig.2.1 (the blue curves or the curves marked with “o”). It is clear that u_c is not close to the desired solution, since $F(u_c)$ is not close to zero. The idea of nonlinear elimination is to make some components of $F(u_c)$ zero, or close to zero. Let us denote by $\omega \subset \Omega$ as the interval in which $F(u_c)$ is to be reduced, then we can describe the algorithm as follows.

Step 1: Find u_ω , which approximately solves

$$(2.1) \quad F_\omega(u_\omega) = 0,$$

where F_ω is the restriction of $F(\cdot)$ in the subdomain ω . We assume u_ω satisfies the boundary condition

$$(2.2) \quad u_\omega|_{\partial\omega} = u_c|_{\partial\omega}.$$

Step 2: Obtain a new global approximate solution

$$u_{new} = \begin{cases} u_\omega & \text{in } \omega \\ u_c & \text{in } \Omega \setminus \omega. \end{cases}$$

The new solution u_{new} is shown in Fig.2.1 (the red curve on the left marked with “x”). The interesting thing is that the corresponding residual function $F(u_{new})$ (the red curve of the right figure in Fig.2.1 marked with “x”), which is indeed very small inside the subdomain ω and is not changed outside of ω . However, on the boundary

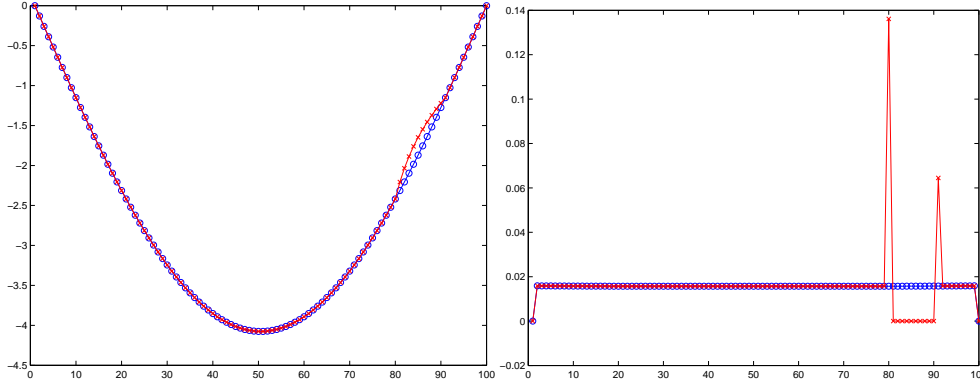


FIG. 2.1. The left figures are for u and the right figures are the residuals $F(u)$.

of ω , the function values increase quite a bit. This is not unexpected. the transition from u_ω to u_c may be hardly noticeable, but the first and second derivatives change a lot. The calculation of $F(u_{new})$ involves second derivatives at the boundary points of ω , and therefore $F(u_{new})$ has sharp jumps. The jumps need to be removed or reduced for the algorithm to be useful.

Below we introduce a nonlinear restricted additive Schwarz technique which is based on the nonlinear elimination idea just outlined, but with two key differences: (1) We do not assume we know which components to eliminate before hand; (2) After the local nonlinear elimination we keep the solution in the “true” interior of ω and throw away values near the boundary of ω .

For the purpose of parallel computing, we decompose the domain into subdomains and the eliminations are carried out on every subdomains in parallel. More precisely, if any components in a subdomain need to be eliminated, then all components in the subdomain are eliminated. Suppose the system to be solved has n unknowns and n equations. Let

$$S = (1, \dots, n)$$

be an index set; i.e., one integer for each unknown u_i and F_i . We assume that S_1, \dots, S_N is a non-overlapping partition of S in the sense that

$$\bigcup_{i=1}^N S_i = S, \quad S_i \cap S_j = \emptyset \text{ if } i \neq j, \quad \text{and } S_i \subset S.$$

Let n_i be the dimension of S_i ; then,

$$\sum_{i=1}^N n_i = n.$$

Using the partition of S , we introduce subspaces of R^n and the corresponding restriction and extension matrices. For each S_i we define $V_i \subset R^n$ as

$$V_i = \{v | v = (v_1, \dots, v_n)^T \in R^n, v_k = 0, \text{ if } k \notin S_i\}$$

and R_i^0 as an $n \times n$ restriction (also extension) matrix whose k th column is either the k th column of the $n \times n$ identity matrix $I_{n \times n}$ if $k \in S_i$ or zero if $k \notin S_i$. We next

introduce an overlapping partition of S . Let S_i^δ be an extension of S_i ; i.e.,

$$S_i \subset S_i^\delta, \text{ for } i = 1, \dots, N.$$

where δ is an positive integer indicating the size of the overlap. In the case that S is from a finite element mesh we usually assign some meaning to the integer δ as follows. If $\delta = 0$, then $S_i^0 = S_i$. If $\delta = 1$, we say the overlap is 1, which means only nodes in S that have distance 1 to S_i are included in S_i^1 . In general, S_i^δ contains all nodes in S whose distances to S_i are no more than δ . Because of the recursive definition, we have

$$S_i^{\delta_1} \subset S_i^{\delta_2}, \text{ if } \delta_2 > \delta_1, \text{ for } i = 1, \dots, N.$$

For each S_i^δ , we define R_i^δ as a restriction matrix whose k th column is either the k th column of the $n \times n$ identity matrix $I_{n \times n}$ if $k \in S_i^\delta$ or zero if $k \notin S_i^\delta$. We denote by n_i^δ as the dimension of S_i^δ . Similarly, we can define the subspace V_i^δ for S_i^δ .

Using the restriction operator, we define the subdomain nonlinear function as

$$F_i^\delta(u) = R_i^\delta F(u).$$

We next define the major component of the algorithm, namely the nonlinearly preconditioned function. For any given $v \in R^n$, in each overlapping subspace we find a vector $v_i^\delta \in V_i^\delta$ that solves the subdomain nonlinear system

$$(2.3) \quad F_i^\delta(v + v_i^\delta) = 0.$$

Note that the dimension of the problem is n_i^δ and v is considered as a given vector. Then, we define

$$(2.4) \quad w_i = R_i^0(v + v_i^\delta)$$

and the nonlinear preconditioning operator G as

$$(2.5) \quad w = G(v) = w_1 + w_2 + \dots + w_N.$$

We will refer to G as the nonlinear restricted additive Schwarz operator, or simply restricted additive Schwarz (RAS). Note that the evaluation of the function $G(v)$, for a given v , involves the solution of nonlinear systems on all subdomains S_i^δ . Very often, these nonlinear systems don't need to be solved very accurately. If the overlap is zero, then this is simply a nonlinear block Jacobi algorithm.

In (2.4), if R_i^0 is replaced by R_i^δ , the resulting nonlinear elimination operator would be the regular nonlinear additive Schwarz method. We tested the regular method for several nonlinear equations but didn't obtain satisfactory results.

In the linear case, this algorithm is the same as the restricted additive Schwarz preconditioned Richardson's method. Using the usual notation, if

$$F(v) = Av - b,$$

then, with a simple calculation, we have

$$G(v) = v - M_{RAS}^{-1}F(v),$$

where

$$M_{RAS}^{-1} = \sum_{i=1}^N R_i^0 A_i^{-1} R_i^\delta$$

and A_i^{-1} is the subspace inverse of $A_i = R_i^\delta A R_i^\delta$. We mention that in the linear case, there is another version of RAS, if we switch the operators R_i^0 with R_i^δ . We haven't studied this version for nonlinear problems.

3. The NKS-RAS algorithm. In this section we first give a high level description of the general algorithm and then discuss the details of each major component.

3.1. The basic algorithm. The goal is to solve equation (1.1) with a given initial guess $u^{(0)}$. Suppose we are at iteration k and $u^{(k)}$ is the current approximate solution.

ALGORITHM 3.1 (NKS-RAS).

Step 1 (The Nonlinearity Checking Step): Check the local and global stopping conditions.

- *If the global condition is satisfied, stop.*
- *If the local conditions indicate that nonlinearities are not balanced, go to Step 2.*
- *If the local conditions indicate that nonlinearities are balanced, set $\tilde{u}^{(k)} = u^{(k)}$, go to Step 3.*

Step 2 (The RAS Step): Solve local nonlinear problems on the overlapping subdomains to obtain the subdomain correction v_i^δ

$$R_i^\delta F(u^{(k)} + v_i^\delta) = 0$$

for $i = 1, \dots, N$.

Drop the solution in the overlapping part of the subdomain and compute the global function $G(u^{(k)})$

$$G(u^{(k)}) = \sum_{i=1}^N R_i^0(u^{(k)} + v_i^\delta), \quad \text{and set } \tilde{u}^{(k)} = G(u^{(k)}).$$

Go to Step 3.

Step 3 (The NKS Step): Compute the next approximate solution $u^{(k+1)}$ by approximately solving the following equation

$$F(u) = 0$$

with one step of NKS iteration using $\tilde{u}^{(k)}$ as the initial guess.

Go to Step 1.

3.2. The nonlinearity checking step. To check the balance of the nonlinearity is difficult, if not impossible. At the current approximate solution $u^{(k)}$, if we take the Taylor's expansion of the function as

$$F(u^{(k)}) + F'(u^{(k)})(u - u^{(k)}) + \frac{1}{2}F''(u^{(k)})(u - u^{(k)}, u - u^{(k)}) + O(\|u - u^{(k)}\|^3),$$

then the values of the bilinear form, $F''(u^{(k)})(u - u^{(k)}, u - u^{(k)})$, tell us how nonlinear the function is in the neighborhood of $u^{(k)}$, however, the desired values are often too

expensive to compute. We use the following algorithm to check the balance of the nonlinearity.

ALGORITHM 3.2 (Nonlinearity-checking). *At the k -th iteration,*

1. Compute $\|F_1^\delta(u^{(k)})\|, \dots, \|F_N^\delta(u^{(k)})\|$.
2. Determine m such that $\|F_m^\delta(u^{(k)})\| = \max\{\|F_i^\delta(u^{(k)})\| \mid 1 \leq i \leq N\}$.
3. Nonlinearity is not balanced if

$$\|F_m^\delta(u^{(k)})\| > \rho \|F(u^{(k)})\|$$

where $0 < \rho < 1$ is a pre-chosen constant; or else it is balanced.

The above mentioned algorithm is based purely on the partitioned domain. For the problems that we are working on (to be reported in the Numerical experiments section), $F(\cdot)$ arises from the discretization of the Navier-Stokes equations. The point-wise values of the residual $F(\cdot)$ represents how well the conservation laws are satisfied. $F_i^\delta(\cdot)$ represents how well the conservation laws are satisfied in the i^{th} subdomain. Other techniques may be used. For example, if there are multiple physical fields, such as velocity, vorticity, etc., then we can also check the residual values for each of the field separately and determine which field equations are more nonlinear than the others. A combination of the domain-based and field-based approaches may provide another choice in certain applications.

3.3. The RAS step. This is the nonlinear elimination step, and is called between two outer Newton iterations when the nonlinearity is not balanced. Its purpose is to provide a better “initial” guess for the next outer Newton iteration. When the nonlinearity is well-balanced, the global Newton method should work well by itself and this RAS step can be skipped. There are several issues: (1) how to define the subdomains? (2) how big an overlap to use? (3) how accurately the subdomain nonlinear systems should be solved? In this paper, the subdomains are all obtained via domain decomposition without considering the physical meanings of the equations. As to the size of the overlap, we will show some computational experiments in the next section using different sizes.

On each subdomain, the nonlinear subsystem to be solved takes the form

$$(3.1) \quad G_i(v_i^\delta) \equiv R_i^\delta F(u^{(k)} + v_i^\delta) = 0,$$

which can be solved using Newton method with zero initial guess, $(v_i^\delta)^{(0)} = 0$. Let $(v_i^\delta)^k$ be the current solution, then a new solution is computed by first finding a search direction $p_i^{(k)}$ satisfying

$$(3.2) \quad G'_i\left((v_i^\delta)^{(k)}\right) p_i^{(k)} = -G_i\left((v_i^\delta)^{(k)}\right)$$

and then compute

$$(3.3) \quad (v_i^\delta)^{(k+1)} = (v_i^\delta)^{(k)} + \lambda_i^{(k)} p_i^{(k)}.$$

Here $\lambda_i^{(k)}$ is a linesearch parameter. Since the solution v_i^δ is a correction to the outer Newton solution, it should be close to zero when $u^{(k)}$ is approaching the solution of the global system. If the subsystem is not too large, the Jacobian system can be solved simply with Gaussian elimination on a single processor. In some situations, if the subsystem is too large for a single processor, NKS can be used, with possibly further partitioning and another level of elimination for the purpose of parallel processing.

A subsystem is set up for each subdomain, but not all of them need to actually be “solved”. Very often, after checking the initial residual $\|G_i((v_i^\delta)^0)\|$, the subsystem can immediately be declared as solved since the stopping condition is satisfied

$$\left\|G_i((v_i^\delta)^{(k)})\right\| \leq \max \left\{ \varepsilon_r \|G_i((v_i^\delta)^{(0)})\|, \varepsilon_a \right\},$$

where ε_r and ε_a are the relative and absolute tolerances for the subdomain nonlinear problem. Once all the subsystems are solved, the solutions are gathered to form a global correction

$$(3.4) \quad \sum_{i=1}^N R_i^0 v_i^\delta$$

to be added to the current solution $u^{(k)}$. The operation (3.4) is mathematically very important as it removes the values in the overlapping regions. Computationally, it involves nothing: neither computation nor communication.

We note that the RAS based nonlinear elimination operator G has an interesting property that is

$$G^2 = G,$$

which simply means that the elimination step doesn't need to be performed more than once per Newton iteration. The proof is straightforward.

Because S_1, \dots, S_N is a non-overlapping partition of S , and R_j^0 is an $n \times n$ matrix whose k -th column is either the k -th column of the $n \times n$ identity matrix $I_{n \times n}$ if $k \in S_j$, or zero if $k \notin S_j$,

$$(3.5) \quad \sum_{j=1}^N R_j^0 v = I_{n \times n} v = v$$

Therefore, from equations (2.5) and (2.4),

$$(3.6) \quad w = G(v) = \sum_{j=1}^N R_j^0 (v + v_j^\delta) = v + \sum_{j=1}^N R_j^0 v_j^\delta$$

When the nonlinear elimination procedure (2.5) is applied twice in a row,

$$(3.7) \quad w' = G(G(v)) = G(w) = \sum_{i=1}^N R_i^0 (w + w_i^\delta) = \sum_{i=1}^N R_i^0 \left(v + \sum_{j=1}^N R_j^0 v_j^\delta + w_i^\delta \right)$$

where w_i^δ is the solution to

$$(3.8) \quad F_i^\delta(w + w_i^\delta) = 0, \text{ or, } F_i^\delta\left(v + \sum_{j=1}^N R_j^0 v_j^\delta + w_i^\delta\right) = 0.$$

Assume that solution to equation (2.3) is unique, we then conclude that

$$(3.9) \quad \sum_{j=1}^N R_j^0 v_j^\delta + w_i^\delta = v_i^\delta.$$

Substitute the result from equation (3.9) back into equation (3.7),

$$(3.10) \quad w' = G(G(v)) = G(w) = \sum_{i=1}^N R_i^0 \left(v + \sum_{j=1}^N R_j^0 v_j^\delta + w_i^\delta \right)$$

$$(3.11) \quad = \sum_{i=1}^N R_i^0 (v + v_i^\delta) = w = G(v)$$

3.4. The NKS step. This is the step to solve the global nonlinear system (1.1). The only connection of this step and the RAS step is the initial guess; otherwise they are fairly independent, except in practice, they often use the same data structure. Let $u^{(k)}$ be the current approximate solution, and J the Jacobian of $F(\cdot)$ at $u^{(k)}$,

$$J = F' = \left(\frac{\partial F_i}{\partial u_j} \right)_{n \times n}$$

and J_i , the restriction of the global Jacobian to the subdomain Ω_i^δ , i.e.,

$$J_i = (R_i^\delta J R_i^\delta)_{n_i^\delta \times n_i^\delta}$$

for $i = 1, \dots, N$. To advance the solution from the current solution $u^{(k)}$, we first find $\tilde{p}^{(k)}$ satisfying

$$(3.12) \quad \|F(u^{(k)}) + J(u^{(k)})M^{-1}\tilde{p}^{(k)}\| \leq \eta \|F(u^{(k)})\|$$

and then compute

$$(3.13) \quad u^{(k+1)} = u^{(k)} + \lambda^{(k)} p^{(k)},$$

where $p^{(k)} = M^{-1}\tilde{p}^{(k)}$ and $\lambda^{(k)}$ is a linesearch parameter. M^{-1} can be any preconditioner for J . Since the Schwarz framework is used at the nonlinear elimination step, it is natural to use the same framework for M . To define the additive Schwarz preconditioner, let M_i^{-1} be the inverse of J_i or some approximation of the inverse, then the additive Schwarz preconditioner can be written as

$$(3.14) \quad M_{AS}^{-1} = \sum_{i=1}^N R_i^\delta M_i^{-1} R_i^\delta.$$

We remark that this overlapping factor δ doesn't have to be the same as the one in the nonlinear RAS step. In principle, the partition of the subdomains for building the preconditioner (3.14) doesn't have to be the same partition as in the nonlinear RAS step, but for the convenience of the implementation, we simply use the same partition. Many other linear preconditioners, such as the linear RAS, can also be used in (3.12), instead of (3.14).

4. Numerical experiments. We report some results of our numerical experiments in this section using the new algorithm, NKS-RAS, and we also compare the new results with those obtained using a standard inexact Newton method. We consider a two-dimensional driven cavity flow problem [16], in the velocity-vorticity

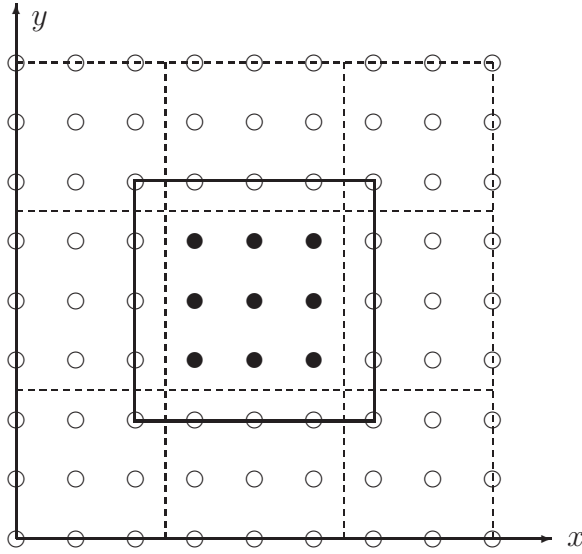


FIG. 4.1. This is a 9-subdomain partition of the unit square. The middle subdomain contains all 25 mesh points marked with either \circ or \bullet . The subdomain nonlinear problem is solved using all 25 mesh points, but the solution is kept only at the 9 interior points marked with \bullet .

formulation, in terms of the velocity u , v , and the vorticity ω , defined on the unit square $\Omega = (0, 1) \times (0, 1)$,

$$(4.1) \quad \begin{cases} -\Delta u - \frac{\partial \omega}{\partial y} & = 0 \\ -\Delta v + \frac{\partial \omega}{\partial x} & = 0 \\ -\frac{1}{Re} \Delta \omega + u \frac{\partial \omega}{\partial x} + v \frac{\partial \omega}{\partial y} & = 0. \end{cases}$$

The boundary conditions are: $u = v = 0$ for the bottom, left and right part of the boundary; and $u = 1$, $v = 0$ for the top boundary. The boundary condition on ω is given by its definition:

$$(4.2) \quad \omega(x, y) = -\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}.$$

We vary the Reynolds number Re in the experiments because, in general, as Re increases the problem becomes harder to solve.

The usual uniform mesh finite difference approximation with the 5-point stencil is used to discretize the boundary value problem. Upwinding is used for the divergence (convective) terms and central differencing for the gradient (source) terms. To obtain a nonlinear algebraic system of equations F , we use natural ordering for the mesh points, and at each mesh point, we arrange the unknowns in the order of u , v , and ω . The partitioning of F is through the partitioning of the mesh points. Fig. 4.1 shows a typical mesh, together with an overlapping partition with interior points marked as

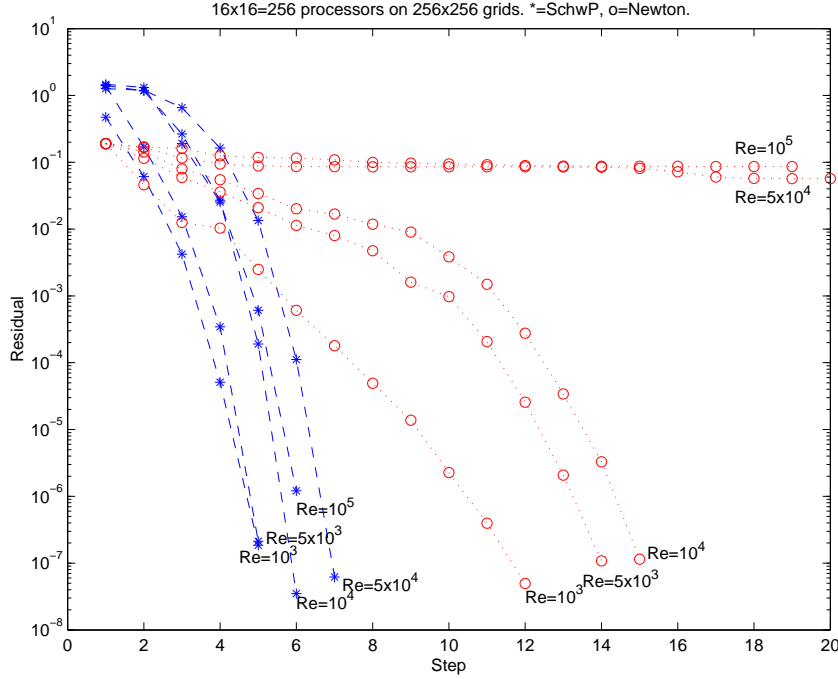


FIG. 4.2. Nonlinear residual history for the flow problem with different Reynolds numbers.

“•”. The size of the overlap is as indicated in Fig.4.1. The implementation is done using PETSc [1], and the results are obtained on an IBM BG/L.

In all tests, the initial guess for the global Newton iteration is zero for u , v and ω . We stop the global Newton iterations if

$$\|F(u^{(k)})\| \leq \max\{10^{-6}\|F(u^{(0)})\|, 10^{-10}\}.$$

In the nonlinear RAS step of the algorithm, we stop the subdomain Newton iterations if

$$\|F_i^\delta(u^{(k)} + v_i^\delta)\| \leq \max\{10^{-6}\|F_i^\delta(u^{(k)})\|, 10^{-10}\}.$$

The global Jacobian system is solved with GMRES(30). The global linear iteration is stopped if the relative tolerance

$$\|F(u^{(k)}) + F'(u^{(k)})p^{(k)}\| \leq \max\{10^{-4}\|F(u^{(k)})\|, 10^{-10}\}$$

is satisfied. In the nonlinearity checking step, we use

$$\rho = 0.75.$$

In practice, there is usually a range of ρ that one can choose from.

We first present a comparison of NKS-RAS with the regular NKS. In NKS the preconditioner is the regular additive Schwarz (3.14). In Fig.4.2, we show the history of the norm of the residual of several test runs with different Reynolds numbers on a 256×256 mesh using the regular NKS (marked with \circ) and NKS-RAS (marked with $*$). The mesh is partitioned into 256 subdomains (16 in each direction) and

each subdomain is assigned to one processor. As the Reynolds number increases, the nonlinear system becomes more difficult to solve. NKS with the standard line search fails to converge once the Reynolds number passes a certain value. We did not try to employ other techniques, such as pseudo-time stepping [22] or parameter/mesh continuations [28, 29], to improve the convergence of NKS. On the other hand, NKS-RAS converges for a much larger range of Reynolds numbers as shown in Fig.4.2 without employing any special techniques. The number of global Newton iterations does not change much as we increase the Reynolds number, and in all tests, the nonlinear RAS is called only once at the very first iteration. During the RAS iteration, the number of local Newton iterations ranges from 0 to 10.

An interesting question is why the use of the nonlinear RAS helps so much the convergence of the outer Newton iteration. To answer the question theoretically is beyond the scope of the paper. A similar question about why the linear RAS works so well as a linear preconditioner is still an unanswered question, as far as we know. Some recent papers show that in some special cases, a version of the linear RAS is in fact closely related to the parallel Schwarz algorithm of Lions [12, 23]. Sometimes the norm of the residual $\|F(u^{(k)})\|$, as shown in Fig.4.2, does not tell us much about the pointwise value of the residual function $F(u^{(k)})$ itself. As a matter of fact, the work in this paper is mostly motivated by inspecting the surface plots (not just the norm) of the residual function for each of the physical fields. In Fig 4.3, we show the surface plots for each of the three components of $F(u^{(k)})$ corresponding to the two velocity components u and v and the vorticity component ω . The top three figures are for the residual functions from the regular NKS iterations, and the bottom three figures are for the residual functions from the new algorithm.

For this particular mesh and Reynolds number, both algorithms converge and NKS takes a few more iterations than NKS-RAS. For the NKS run, we observe that the dominant part of the residual is the vorticity component near the top corner of the computational domain. For the NKS-RAS run, the dominant component of the vorticity function near the corner is removed by the RAS iteration (see the bottom figure of Fig. 4.3). Once the dominant component is removed, the global Newton algorithm converges in a small number of iterations. Six iterations are needed to reach convergence in this case. RAS is used in the first iteration to remove the dominant peak in the nonlinear residual. In the other five iterations, there are no dominant nonlinear components according to the nonlinearity-checking condition and therefore RAS is not used.

The side effect of the RAS iteration includes the spreading of the dominant residual to other components, and to other areas of the computational domain, near the boundary of the subdomains. The interesting thing is that the newly created peaks in the residual functions do not seem to decrease the overall convergence rate.

We next address various scalability issues of the algorithm. In Tables 4.1 and 4.2, we present the numbers of global Newton iterations, the average numbers of global GMRES iterations, the ranges of local Newton iterations in RAS, the overlap size, and the total compute times for various Reynolds numbers, the number of processors, and mesh sizes.

As the Reynolds number increases, the nonlinearity is more concentrated near the two top corners of the computational domain. That makes the distribution of nonlinearity more unbalanced. In fact, the classical NKS method starts to show divergent behaviors once the Reynolds number exceeds 3×10^4 on all of our test runs. On the other hand, NKS-RAS converges on all of our test runs for all the Reynolds

TABLE 4.1

The numbers of iterations and total compute times. Mesh size 128×128 on 64, 128 and 256 processors. The overlapping size is 3 for all cases.

# of processors	$Re = 10^3$	$Re = 5 \cdot 10^3$	$Re = 10^4$	$Re = 5 \cdot 10^4$	$Re = 10^5$
Global Newton iterations					
8×8	4	6	6	7	8
8×16	4	5	5	7	7
16×16	4	5	6	7	9
Average GMRES iterations					
8×8	48	39	38	39	38
8×16	56	50	52	42	40
16×16	77	67	68	61	51
Ranges of subdomain Newton iterations in RAS					
8×8	$0 \sim 4$	$0 \sim 6$	$0 \sim 7$	$0 \sim 8$	$0 \sim 9$
8×16	$0 \sim 4$	$0 \sim 5$	$0 \sim 5$	$0 \sim 5$	$0 \sim 7$
16×16	$0 \sim 4$	$0 \sim 5$	$0 \sim 6$	$0 \sim 7$	$0 \sim 8$
Total compute times (sec)					
8×8	0.9761	1.2520	1.2810	1.5050	1.6620
8×16	0.5846	0.6716	0.7078	0.8208	0.8292
16×16	0.4659	0.5268	0.6226	0.6661	0.7472

numbers ranging from $Re = 10^3$ to $Re = 10^5$. This shows that the RAS step has indeed provided some balancing of the nonlinearities to the nonlinear problem under consideration. Note that for large Re , bifurcation may happen, but we do not intend to deal with that in this paper.

There are two overlap parameters in the algorithms; one for the nonlinear RAS step and one for the linear Schwarz preconditioner in the NKS step. They don't have to be the same, and in some situations, much better results can be obtained by using different values of overlap. To find the best combination is often a trial and error step.

Here we consider a case where the RAS overlap is fixed to be 4 and the effect of the NKS overlap is given in Table 4.3. We can see that in general, the algorithm converges better as the overlap increases. On the other hand, a larger overlap also increases inter-process communications. Consequently, choices for overlap width are usually between 2 and 4.

TABLE 4.2

The numbers of iterations and total compute times. Mesh size 256×256 on 128, 256 and 512 processors. The overlapping size is 3 for all cases.

# of processors	$Re = 10^3$	$Re = 5 \cdot 10^3$	$Re = 10^4$	$Re = 5 \cdot 10^4$	$Re = 10^5$
Global Newton iterations					
8×16	5	5	6	9	9
16×16	5	5	6	7	9
16×32	5^\dagger	4	5	7	7
Average GMRES iterations					
8×16	85	65	62	49	45
16×16	107	90	98	96	97
16×32	121^\dagger	115	103	94	97
Ranges of subdomain Newton iterations in RAS					
8×16	$0 \sim 3$	$0 \sim 4$	$0 \sim 4$	$0 \sim 5$	$0 \sim 5$
16×16	$0 \sim 3$	$0 \sim 5$	$0 \sim 5$	$0 \sim 6$	$0 \sim 6$
16×32	$0 \sim 2^\dagger$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$
Total compute times (sec)					
8×16	3.082	2.720	3.108	3.914	3.749
16×16	1.888	1.752	2.131	2.440	3.030
16×32	1.868	1.088	1.214	1.515	1.550

† Overlapping size of 6 used in this case.

TABLE 4.3

Effect of the overlapping size δ . Mesh size 256×256 , $Re = 10^4$, on 128, 256 and 512 processors.

# of processors	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$
Global Newton iterations					
8×16	6	6	6	6	6
16×16	6	6	6	6	6
16×32	5	5	5	5	5
Average GMRES iterations					
8×16	96	77	62	54	51
16×16	169	111	98	86	78
16×32	165	134	103	87	80
Ranges of subdomain Newton iterations in RAS					
8×16	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$
16×16	$0 \sim 5$	$0 \sim 5$	$0 \sim 5$	$0 \sim 5$	$0 \sim 5$
16×32	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$	$0 \sim 4$
Total compute times (sec)					
8×16	2.992	2.999	3.108	3.211	3.567
16×16	2.246	2.013	2.131	2.209	2.369
16×32	1.169	1.230	1.214	1.222	1.315

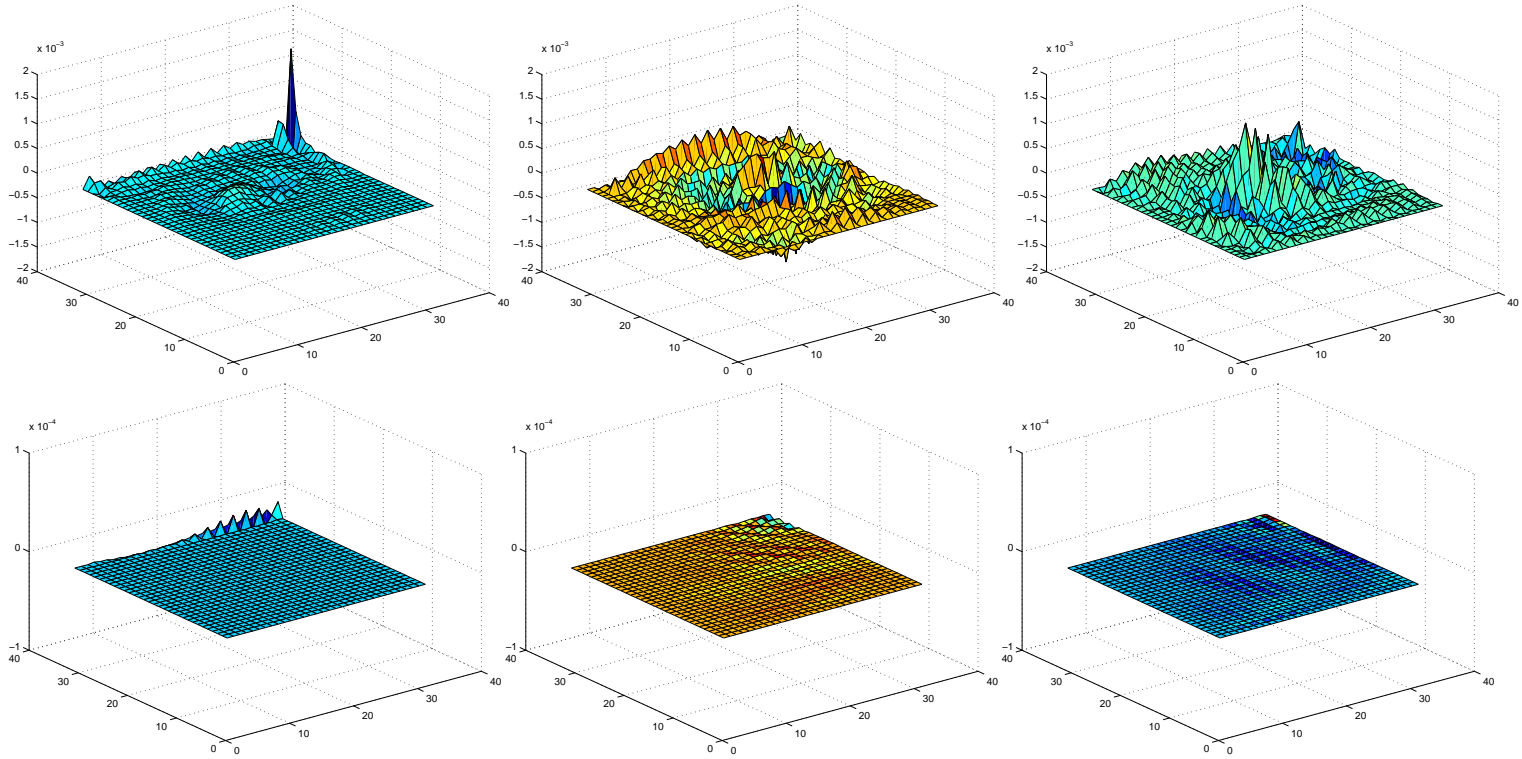


FIG. 4.3. A comparison of the residual surfaces obtained from Newton's method (top three figures) and the NKS-RAS method (bottom three figures). The left figures correspond to the ω component of the residual function, the middle figures correspond to the u component of the residual function, and the right figures correspond to the v component of the residual function. The results are for the 5th iteration of both methods. The calculation was carried out for $Re = 5.0 \times 10^3$ on a 256×256 mesh, partitioned into 16×16 subdomains. Note that the scale for the top figures is 10^{-3} and the scale for the bottom figures is 10^{-4} .

Finally we present a comparison, in Table 4.4, of the algorithm when RAS is used only once, with the case when RAS is used at every iteration. In most situations, it is enough to use RAS once. This suggests that once the local highly unbalanced components are removed, Newton method converges well without requiring any further intervention.

TABLE 4.4

A comparison of the numbers of Newton iterations when RAS is used only once and used at every iteration. The mesh size is 256×256 , with $16 \times 16 = 256$ processors. The overlapping size is 3 for all cases.

Re	RAS once	RAS every step
$1 \cdot 10^2$	4	3
$5 \cdot 10^2$	5	6
$1 \cdot 10^3$	5	5
$5 \cdot 10^3$	5	6
$1 \cdot 10^4$	6	9
$5 \cdot 10^4$	7	8
$1 \cdot 10^5$	9	11

5. Final remarks. A robust and fully parallel inexact Newton method was introduced using a restricted additive Schwarz based nonlinear elimination method and the inexact Newton-Krylov-Schwarz method. We demonstrated in this paper that this new RAS-NKS algorithm can be used to solve difficult nonlinear problems by balancing the distribution of nonlinearities in the system. As an example, we studied the performance of the new algorithm for solving the high Reynolds number incompressible Navier-Stokes equations. We observed that once the local high nonlinearity caused by the high Reynolds number is eliminated, the overall convergence becomes completely independent of the Reynolds number. Comparing with the previously introduced left-preconditioning method (additive Schwarz preconditioned inexact Newton method [6, 17, 18]), the new method has similar robustness and scalability properties, but is considerably easier to implement and is more flexible since it can be turned on and off during some of the outer Newton iterations.

As future work, we will consider two improvements of the algorithm. First, when the number of processors is very small, the subdomain problems become too large and the subdomain nonlinear iterations used in nonlinear RAS may fail to converge. Some adaptivity may be necessary to recursively use the NKS-RAS idea to solve the nonlinear subproblems on large subdomains. Second, when the number of processors is very large, multilevel versions for the algorithm is needed to obtain good parallel scalability.

Acknowledgements. We would like to thank Professor Frederic Nataf, and the referees for their insightful and constructive suggestions.

REFERENCES

- [1] S. BALAY, K. BUSCHELMAN, W. GROPP, D. KAUSHIK, M. KNEPLEY, L. MCINNES, B. SMITH, AND H. ZHANG, *PETSc Users Manual*, Argonne National Laboratory, 2008.
- [2] P. N. BROWN AND Y. SAAD, *Hybrid Krylov methods for nonlinear systems of equations*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 59–71.

- [3] P. N. BROWN AND Y. SAAD, *Convergence theory of nonlinear Newton-Krylov algorithms*, SIAM J. Optimization, 4 (1994), pp. 297–330.
- [4] X.-C. CAI AND M. DRYJA, *Domain decomposition methods for monotone nonlinear elliptic problems*, Contemporary Math., 180 (1994), pp. 21–27.
- [5] X.-C. CAI, W. D. GROPP, D. E. KEYES, R. G. MELVIN, AND D. P. YOUNG, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput., 19 (1998), pp. 246–265.
- [6] X.-C. CAI AND D. E. KEYES, *Nonlinearly preconditioned inexact Newton algorithm*, SIAM J. Sci. Comput., 24 (2002), pp. 183–200.
- [7] X.-C. CAI, D. E. KEYES, AND D. P. YOUNG, *A nonlinear additive Schwarz preconditioned inexact Newton method for shocked duct flow*, Proceedings of the 13th International Conference on Domain Decomposition Methods, 2001.
- [8] X.-C. CAI AND M. SARKIS, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.
- [9] T. CHAN AND K. JACKSON, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 533–542.
- [10] R. S. DEMBO, S. C. EISENSTAT, AND T. STEIHAUG, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
- [11] M. DRYJA AND W. HACKBUSCH, *On the nonlinear domain decomposition method*, BIT, (1997), pp. 296–311.
- [12] E. EFSTATHIOU AND M. GANDER, *Why restricted additive Schwarz converges faster than additive Schwarz*, BIT Numer. Math., 43 (2003), pp. 945–959.
- [13] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton methods*, SIAM J. Optimization, 4 (1994), pp. 393–422.
- [14] S. C. EISENSTAT AND H. F. WALKER, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16–32.
- [15] W. D. GROPP, D. E. KEYES, L. C. MCINNES AND M. D. TIDRIRI, *Globalized Newton-Krylov-Schwarz algorithms and software for parallel implicit CFD*, Int. J. High Performance Computing Applications, 14 (2000), pp. 102–136.
- [16] C. HIRSCH, *Numerical Computation of Internal and External Flows*, John Wiley & Sons, New York, 1990.
- [17] F.-N. HWANG AND X.-C. CAI, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., 204 (2005), pp. 666–691.
- [18] F.-N. HWANG AND X.-C. CAI, *A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms*, Comp. Methods Appl. Mech. Engin., 196 (2007), pp. 1603–1611.
- [19] F.-N. HWANG, H.-L. LIN, AND X.-C. CAI, *Two-level nonlinear elimination based preconditioners for inexact Newton methods with application in shocked duct flow calculation*, ETNA, 37 (2010), pp. 239–251.
- [20] H. JIANG AND P. A. FORSYTH, *Robust linear and nonlinear strategies for solution of the transonic Euler equations*, Computer and Fluids, 24 (1995), pp. 753–770.
- [21] C. T. KELLEY, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
- [22] C. T. KELLEY AND D. E. KEYES, *Convergence analysis of pseudo-transient continuation*, SIAM J. Num. Anal., 35 (1998), pp. 508–523.
- [23] F. KWOK, *Is additive Schwarz with harmonic extension just Lions method in disguise?* Lecture Notes in Comput. Sci. Engin., 78 (2010), pp. 439–446.
- [24] P. J. LANZKRON, D. J. ROSE, AND J. T. WILKES, *An analysis of approximate nonlinear elimination*, SIAM J. Sci. Comput., 17 (1996), pp. 538–559.
- [25] M. PARASCHIOIU, X.-C. CAI, M. SARKIS, D. P. YOUNG, AND D. KEYES, *Multi-domain multimodel formulation for compressible flows: Conservative interface coupling and parallel implicit solvers for 3D unstructured meshes*, AIAA Paper 99-0784, 1999.
- [26] M. PERNICE AND H. WALKER, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.
- [27] B. F. SMITH, P. E. BJØRSTAD, AND W. D. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
- [28] D. P. YOUNG, R. G. MELVIN, M. B. BIETERMAN, F. T. JOHNSON, AND S. S. SAMANT, *Global convergence of inexact Newton methods for transonic flow*, Int. J. Numer. Meths. Fluids, 11 (1990), pp. 1075–1095.
- [29] D. P. YOUNG, R. G. MERVIN, 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, J. Comput. Phys., 92 (1991), pp. 1-66.