NONLINEARLY PRECONDITIONED INEXACT NEWTON ALGORITHMS

XIAO-CHUAN CAI* AND DAVID E. KEYES†

Abstract. Inexact Newton algorithms are commonly used for solving large sparse nonlinear system of equations $F(u^*) = 0$ arising, for example, from the discretization of partial differential equations. Even with global strategies such as line search or trust region the methods often stagnate at local minima of $\|F\|$, especially for problems with unbalanced nonlinearities, because the methods do not have built-in machinery to deal with the unbalanced nonlinearities. To find the same solution u^*, one may want to solve, instead, an equivalent nonlinearly preconditioned system $\mathcal{F}(u^*) = 0$ whose nonlinearities are more balanced. In this paper, we propose and study a nonlinear additive Schwarz based parallel nonlinear preconditioner and show numerically that the new method converges well even for some difficult problems, such as high Reynolds number flows, where a traditional inexact Newton method fails.

Key words. Nonlinear preconditioning, inexact Newton methods, Krylov subspace methods, nonlinear additive Schwarz, domain decomposition, nonlinear equations, parallel computing, incompressible flows

1. Introduction. Many computational engineering problems require the numerical solution of large sparse nonlinear system of equations, i.e., for a given nonlinear function $F : R^n \to R^n$, compute a vector $u^* \in R^n$, such that

$$F(u^*) = 0, \tag{1.1}$$

starting from an initial guess $u^{(0)} \in R^n$. Here $F = (F_1, \ldots, F_n)^T$, $F_i = F_i(u_1, \ldots, u_n)$, and $u = (u_1, \ldots, u_n)^T$. Inexact Newton algorithms (IN) [8, 9, 12, 18] 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 the following steps:

ALGORITHM 1.1 (IN).

Step 1: Find the inexact Newton direction $p^{(k)}$ such that

$$\|F(u^{(k)}) - F'(u^{(k)})p^{(k)}\| \le \eta_k \|F(u^{(k)})\| \tag{1.2}$$

Step 2: Compute the new approximate solution

$$u^{(k+1)} = u^{(k)} - \lambda^{(k)} p^{(k)}. \tag{1.3}$$

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, 12, 13]. $\lambda^{(k)}$ is another scalar that determines how far one should go in the selected inexact Newton direction [8]. We comment on our strategy for selecting $\lambda^{(k)}$ in section 4.

IN has two well-known features, namely, (a) if the initial guess is close enough to the desired solution then the convergence is very fast provided that the η_k's are

*Department of Computer Science, University of Colorado, Boulder, CO 80309-0430 (cai@cs.colorado.edu). The work was supported in part by the NSF grants ASC-9457534, ECS-9725504, and ACI-0072089, and by Lawrence Livermore National Laboratory under subcontract B509471.

†Department of Mathematics & Statistics, Old Dominion University, Norfolk, VA 23529-0077; ISCR, Lawrence Livermore National Laboratory, Livermore, CA 94551-9989; and ICASE, NASA Langley Research Center, Hampton, VA 23681-2199 (keyes@icase.edu). This work was supported in part by NASA under contract NAS1-19480 and by Lawrence Livermore National Laboratory under subcontract B347882.

sufficiently small, and (b) such a good initial guess is generally very difficult to obtain, especially for nonlinear equations that have unbalanced nonlinearities [20]. The step length $\lambda^{(k)}$ is often determined by the components with the strongest nonlinearities, and this may lead to an extended period of stagnation in the nonlinear residual curve; see Fig 6.2 for a typical picture and others in the references [5, 15, 17, 24, 28, 29]. We say that the nonlinearities are "unbalanced" when $\lambda^{(k)}$ is, in effect, determined by a subset of the overall degrees of freedom.

In this paper, we develop some nonlinearly preconditioned inexact Newton algorithms (PIN): Find the solution $u^* \in R^n$ of (1.1) by solving an equivalent nonlinear system

$$\mathcal{F}(u^*) = 0. \tag{1.4}$$

Systems (1.4) and (1.1) are said to be equivalent if they have the same solution. We solve (1.4) using the inexact Newton method in the following form:

ALGORITHM 1.2 (PIN).

Step 1: Find the inexact Newton direction $p^{(k)}$ such that

$$\|\mathcal{F}(u^{(k)}) - \mathcal{F}'(u^{(k)})p^{(k)}\| \leq \eta_k \|\mathcal{F}(u^{(k)})\|. \tag{1.5}$$

Step 2: Compute the new approximate solution

$$u^{(k+1)} = u^{(k)} - \lambda^{(k)} p^{(k)}. \tag{1.6}$$

Here the step length parameter $\lambda^{(k)}$ is determined with a linesearch procedure based on the new merit function $1/2\|\mathcal{F}\|^2$.

$\mathcal{F}$ and F may have completely different forms, but they must have the same solution. In general, the linear case being an exception, $\mathcal{F}$ is a function of both F and u, and we do not expect to know explicitly how $\mathcal{F}$ depends on F or u. As an example, $\mathcal{F}$ may take the form of a composite function

$$\mathcal{F}(u^*) \equiv G(F(u^*)) = 0,$$

which makes G look like a preconditioner if we use the language invented for linear systems of equations. In this case, some desirable properties of the preconditioner $G : R^n \to R^n$ include:

1. If $G(x) = 0$, then $x = 0$.
2. $G \approx F^{-1}$ in some sense.
3. $G(F(w))$ is easily computable for $w \in R^n$.
4. If a Newton-Krylov type method is used for solving (1.4), then the matrix-vector product $(G(F(w)))'v$ should also be easily computable for $w, v \in R^n$.

As in the linear equation case [14], the definition of a preconditioner can not be given precisely, nor is it necessary. Also as in the linear equation case, preconditioning can greatly improve the robustness of the iterative methods, since the preconditioner is designed so that the new system (1.4) has more uniform nonlinearities. G is ordinarily not operationally useful, since it is defined only implicitly. This paper could be written without reference to G by defining $\mathcal{F}$ directly. However, the concept of a nonlinear preconditioner seems expositorily useful. Note that the Jacobian of the preconditioned function can be computed, at least in theory, using the chain rule, i.e.,

$$\mathcal{F}'(u) = \frac{\partial G}{\partial v}\frac{\partial F}{\partial u}. \tag{1.7}$$

If G is close to F^{-1} in the sense that $G(F(u)) \approx u$, then $\frac{\partial G}{\partial v}\frac{\partial F}{\partial u} \approx I$, i.e., $\mathcal{F}'(u) \approx I$. In this case, Algorithm 1.2 converges in one iteration, or few iterations, depending on how close is G to F^{-1}. In fact, the same thing happens as long as $G(F(u)) \approx Au$, where A is a constant matrix independent of u. On the other hand, if G is a linear function, then $\frac{\partial G}{\partial v}$ would be a constant matrix independent of u. In this case the Newton equation of the preconditioned system

$$\mathcal{F}'(u^{(k)})p^{(k)} = \mathcal{F}(u^{(k)})$$

reduces to the Newton equation of the original system

$$F'(u^{(k)})p^{(k)} = F(u^{(k)})$$

and G does not affect the nonlinear convergence of the method, except for the stopping conditions. However, G does change the conditioning of the linear Jacobian system, and this forms the basis for the matrix-free Newton-Krylov methods.

Most of the current research has been on the case of linear G; see, for example, [5, 25]. In this paper, we shall focus on the case when $\mathcal{F}$ is obtained using the single-level nonlinear additive Schwarz method. As an example, we show the nonlinear iteration history, in Figure 6.2, for solving a two-dimensional flow problem with various Reynolds numbers using the standard IN (top) and the newly introduced PIN (bottom). It can be seen clearly that PIN is much less sensitive to the increase in Reynolds number than IN. Details of the numerical experiment will be given later in the paper. Nonlinear Schwarz algorithms have been studied extensively as *iterative methods* [4, 10, 21, 22, 23, 26, 27], and are known, at least experimentally, to be not very robust, in general, unless the problem is monotone. However, we show in the paper that nonlinear Schwarz can be an excellent *nonlinear preconditioner*.

We remark that nonlinear methods can also be used as linear preconditioners as described in [7], but we will not look into this issue in this paper.

Nested linear and nonlinear solvers are often needed in the implementation of PIN, and as a consequence, the software is much harder to develop than for the regular IN. Our target applications are these problems that are difficult to solve using traditional Newton type methods. Those include (1) problems whose solutions have local singularities such as shocks or nonsmooth fronts; and (2) multi-physics problems with drastically different stiffness that require different nonlinear solvers based on a single physics submodel, such as coupled fluid-structure interaction problems.

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 derive a formula for the Jacobian of the nonlinear preconditioned system and a readily computable approximation thereto. The details of the algorithm, with some practical comments about each step, are given in section 4. Section 5 contains a proof of the equivalence of the original nonlinear system and the preconditioned system, under some reasonable local assumptions. Some numerical experiments motivating the value of the ASPIN algorithm appear in Section 6. In section 7, we make further comments and discuss some future research topics in nonlinear preconditioning.

2. A nonlinear additive Schwarz preconditioner. In this section, we describe a nonlinear preconditioner based on the additive Schwarz method [4, 10]. Let

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

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

$$\bigcup_{i=1}^{N} S_i = S, \text{ and } S_i \subset S.$$

Here we allow the subsets to have overlap. Let n_i be the dimension of S_i; then, in general,

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

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

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

and a $n \times n$ restriction (also extension) matrix I_{S_i} whose kth column is either the kth column of the $n \times n$ identity matrix $I_{n\times n}$ if $k \in S_i$ or zero if $k \notin S_i$. Similarly, let s be a subset of S; we denote by I_s the restriction on s. Note that the matrix I_s is always symmetric and the same matrix can be used as both restriction and extension operator. Many other forms of restriction/extension are available in the literature; however, we only consider the simplest form in this paper.

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

$$F_{S_i} = I_{S_i} F.$$

We next define the major component of the algorithm, namely the nonlinearly preconditioned function. For any given $v \in R^n$, define $T_i(v) \in V_i$ as the solution of the following subspace nonlinear system

$$F_{S_i}(v - T_i(v)) = 0, \tag{2.1}$$

for $i = 1, \ldots, N$. We introduce a new function

$$\mathcal{F}(u) = \sum_{i=1}^{N} T_i(u), \tag{2.2}$$

which we will refer to as the nonlinearly preconditioned $F(u)$. The main contribution of this paper is the following algorithm.

ALGORITHM 2.1. *Find the solution u^* of (1.1) by solving the nonlinearly preconditioned system*

$$\mathcal{F}(u) = 0 \tag{2.3}$$

with $u^{(0)}$ as the initial guess.

REMARK 2.1. *In the linear case, this algorithm is the same as the additive Schwarz algorithm. Using the usual notation, if*

$$F(u) = Au - b,$$

then

$$\mathcal{F}(u) = \left(\sum_{i=1}^{N} A_i^{-1} \right) (Au - b),$$

where A_i^{-1} *is the subspace inverse of* $A_i = I_{S_i} A I_{S_i}$ *in* V_i.

REMARK 2.2. *The evaluation of the function* $\mathcal{F}(v)$*, for a given* v*, involves the calculation of the* T_i*, which in turn involves the solution of nonlinear systems on* S_i.

REMARK 2.3. *If the overlap is zero, then this is simply a nonlinear block Jacobi preconditioner.*

REMARK 2.4. *If (2.3) is solved with Picard iteration, or Richardson's method, then the algorithm is simply the nonlinear additive Schwarz method, which is not a robust algorithm, as is known from experience with linear and nonlinear problems.*

3. Basic properties of the Jacobian. If (2.3) is solved using a Newton type algorithm, then the Jacobian is needed in one form or another. We here provide a computable form of it, and discuss some of its basic properties. Let J be the Jacobian of the original nonlinear system, i.e.,

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

and J_{S_i}, the Jacobian of the subdomain nonlinear system, i.e.,

$$J_{S_i} = (I_{S_i} J I_{S_i})_{n \times n}$$

for $i = 1, \ldots, N$. Note that if $F(\cdot)$ is sparse nonlinear function, then J is a sparse matrix and so are the J_{S_i}. Unfortunately, the same thing cannot be said about the preconditioned function $\mathcal{F}(\cdot)$. Its Jacobian, generally speaking, is a dense matrix, and is very expensive to compute and store, as one may imagine. In the following discussion, we denote by

$$\mathcal{J} = \mathcal{F}' = \left(\sum_{i=1}^{N} \mathcal{J}_{S_i} \right)_{n \times n} \tag{3.1}$$

and

$$\mathcal{J}_{S_i} = \left(\frac{\partial T_i}{\partial u_j} \right)_{n \times n} \tag{3.2}$$

the Jacobian of the preconditioned whole system, and the subsystems, respectively. Because of the definition of T_i, $\mathcal{J}_{S_i}$ is a $n \times n$ matrix. $T_i(u)$ has n_i non-trivial function components in S_i, n independent variables $u_1, \ldots, u_n$, and its other $n - n_i$ components are zeros.

Suppose we want to compute the Jacobian $\mathcal{J}$ at a given point $u \in R^n$. Consider one subdomain S_i. Let $S_i^c = S \setminus S_i$ be the complement of S_i in S, we can write $u = (u_{S_i}, u_{S_i^c})$, which is correct up to a re-ordering of the independent variables $u_{S_i} = I_{S_i} u$ and $u_{S_i^c} = I_{S_i^c} u$. Using the definition of $T_i(u)$, we have that

$$F_{S_i}(u_{S_i} - T_i(u_{S_i}, u_{S_i^c}), u_{S_i^c}) = 0. \tag{3.3}$$

Taking the derivative of the above function with respect to u_{S_i}, we obtain

$$\left(\frac{\partial F_{S_i}}{\partial v_{S_i}}\right)\left(I_{S_i} - \frac{\partial T_i(u)}{\partial u_{S_i}}\right) = 0. \tag{3.4}$$

Here $v_{S_i} \equiv u_{S_i} - T_i(u_{S_i}, u_{S_i^c})$. (3.4) implies that

$$\frac{\partial T_i(u)}{\partial u_{S_i}} = I_{S_i}, \tag{3.5}$$

assuming the subsystem Jacobian matrix $\frac{\partial F_{S_i}}{\partial v_{S_i}}$ is nonsingular in the subspace V_i. Next, we take the derivative of (3.3) with respect to $u_{S_i^c}$,

$$-\frac{\partial F_{S_i}}{\partial v_{S_i}}\frac{\partial T_i(u)}{\partial u_{S_i^c}} + \frac{\partial F_{S_i}}{\partial u_{S_i^c}} = 0, \tag{3.6}$$

which is equivalent to

$$\frac{\partial T_i(u)}{\partial u_{S_i^c}} = \left(\frac{\partial F_{S_i}}{\partial v_{S_i}}\right)^{-1}\frac{\partial F_{S_i}}{\partial u_{S_i^c}}. \tag{3.7}$$

Note that

$$\frac{\partial T_i(u)}{\partial u} = \frac{\partial T_i(u)}{\partial u_{S_i}} + \frac{\partial T_i(u)}{\partial u_{S_i^c}}$$

since the sets S_i and S_i^c do not overlap each other. Combining (3.4) and (3.5), we obtain

$$\frac{\partial T_i(u)}{\partial u} = \left(\frac{\partial F_{S_i}}{\partial v_{S_i}}\right)^{-1}\frac{\partial F}{\partial u}. \tag{3.8}$$

Although (3.8) is computable, it is more convenient and cheaper in practice to use the following approximation

$$\frac{\partial T_i(u)}{\partial u} \approx J_{S_i}^{-1} J. \tag{3.9}$$

Summing up (3.9) for all subdomains, we have a formula for the Jacobian of the preconditioned nonlinear system in the form of

$$\tilde{\mathcal{J}} \equiv \sum_{i=1}^{N} J_{S_i}^{-1} J. \tag{3.10}$$

We shall use (3.10) in numerical experiments presented later in this paper. (3.10) is an interesting formula since it corresponds to the additive Schwarz preconditioned linear Jacobian system of the original unpreconditioned equation. This fact implies that, first of all, we know how to solve the Jacobian system of the preconditioned nonlinear system, and second, the Jacobian itself is already partly well-conditioned. In other words, nonlinear preconditioning automatically offers a linear preconditioning for the corresponding Jacobian system.

4. Additive Schwarz preconditioned inexact Newton algorithm. We describe a nonlinear additive Schwarz preconditioned inexact Newton algorithm (ASPIN). Suppose $u^{(0)}$ is a given initial guess, and $u^{(k)}$ is the current approximate solution; a new approximate solution $u^{(k+1)}$ can be computed through the following steps:

ALGORITHM 4.1 (ASPIN).

Step 1: Compute the nonlinear residual $g^{(k)} = \mathcal{F}(u^{(k)})$ through the following steps:

a) Find $g_i^{(k)} = T_{S_i} u^{(k)}$, $i = 1, \ldots, N$, by solving the local subdomain nonlinear systems

$$F_{S_i}(u^{(k)} - g_i^{(k)}) = 0$$

with a starting point $g_i^{(k)} = 0$.

b) Form the global residual

$$g^{(k)} = \sum_{i=1}^{N} g_i^{(k)}.$$

c) Check stopping conditions on $g^{(k)}$.

Step 2: Find the inexact Newton direction $p^{(k)}$ by solving the Jacobian system approximately

$$\sum_{i=1}^{N} J_{S_i}^{-1} J p^{(k)} = g^{(k)}$$

in the sense that

$$\left\| g^{(k)} - \sum_{i=1}^{N} J_{S_i}^{-1} J p^{(k)} \right\| \leq \eta_k \|g^{(k)}\|$$

for some $\eta_k \in [0, 1)$. Various approximations to the matrix elements of the Jacobian may be used in forming the preconditioned Jacobian vector products required in the inexact solution process.

Step 3: Compute the new approximate solution

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

where $\lambda^{(k)}$ is a damping parameter.

ASPIN may look a bit complicated, but as a matter of fact, the required user input is the same as that for the regular IN Algorithm 1.1, i.e., the user needs to supply only two routines for each subdomain:

(1) the evaluation of the original function $F_{S_i}(w)$. This is needed in both Step 1 a) and Step 2 if the Jacobian is to be computed using finite-difference methods. It is also needed in Step 3 in the line search steps.

(2) the Jacobian of the original function J_{S_i} in terms of a matrix-vector multiplication. This is needed in both Step 1 a) and Step 2.

We now briefly discuss the basic steps of the algorithm. In Step 1 a) of Algorithm 4.1, N subdomain nonlinear systems have to be solved in order to evaluate the preconditioned function $\mathcal{F}$ at a given point. More explicitly, we solve

$$G_{S_i}\left(g_i^{(k)}\right) \equiv F_{S_i}\left(u_{S_i}^{(k)} - g_i^{(k)}, u_{S_i^c}^{(k)}\right) = 0, \tag{4.1}$$

which has n_i equations and n_i unknowns, using Algorithm 1.1 with a starting value 0, for $i = 1, \ldots, N$. Note that the vector $u_{S_i^c}^{(k)} \notin V_i$ is needed to evaluate $G_{S_i}(\xi)$, for $\xi \in V_i$, and this requires the ghost points in a mesh-based software implementation. In a parallel implementation, the ghost values often belong to several neighboring processors and communication is required to obtain their current values. We note, however, that the ghost values do not change during the solution of the subdomain nonlinear system.

In Step 2, pieces of the Jacobian matrix are computed. The full Jacobian matrix $\mathcal{J}$ never needs to be formed. In our distributed memory parallel implementation, the submatrices J_{S_i} are formed, and saved, and the multiplication of J with a given vector is carried out using the submatrices J_{S_i}. Therefore the global J matrix is never needed. Several techniques are available for computing the J_{S_i}, for example, using an analytic formula, multi-colored finite differencing, or automatic differentiation. A triangular factorization of J_{S_i} is also performed at this step, in our implementation, and the resulting matrices are stored.

Note that the matrix

$$\sum_{i=1}^{N} J_{S_i}^{-1}$$

should not be considered as a linear preconditioner since it does not appear on the right-hand side of the linear system. However, using the additive Schwarz preconditioning theory, we know that for many applications the matrix $\sum_{i=1}^{N} J_{S_i}^{-1} J$ is well-conditioned, under certain conditions. We also note that if an inexact solver is used to compute $\sum_{i=1}^{N} J_{S_i}^{-1} w$ in Step 2, the Newton search direction would be changed and, as a result, the algorithm becomes more inexact than a regular inexact Newton algorithm, in which the matrix is assumed to be exact but the linear system is not solved exactly.

As noted above the Jacobian system in Step 2 does not have the standard form of a preconditioned sparse linear system

$$M^{-1}Ax = M^{-1}b.$$

However, standard linear solver software packages can still be used with some slight modification, such as removing the line that performs

$$b := M^{-1}b.$$

Since the explicit sparse format of $\sum_{i=1}^{N} J_{S_i}^{-1} J$ is often not available, further preconditioning of the Jacobian system using sparse matrix based techniques, such as ILU, is difficult.

A particular interesting case is when the overlap is zero; then the diagonal blocks of $\sum_{i=1}^{N} J_{S_i}^{-1} J$ are all identities and therefore do not involve any computations when multiplied with vectors. Let us take a two-subdomain case for example,

$$J = \begin{pmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{pmatrix}$$

and $J_{S_i} = J_{ii}$, then

$$\sum_{i=1}^{2} J_{S_i}^{-1} J = \begin{pmatrix} I & J_{11}^{-1} J_{12} \\ J_{22}^{-1} J_{21} & I \end{pmatrix}.$$

The same thing can also be done for the overlapping case. This is a small savings when there many small subspaces. However, the savings can be big if there are relatively few subspaces, but the sizes are large. For example, in the case of a coupled fluid-structure interaction simulation, there could be only two subdomains; one for the fluid flow and one for the structure.

In Step 3, the step length $\lambda^{(k)}$ is determined using a standard line search technique [8] based on the function

$$f(u) = \frac{1}{2} \|\mathcal{F}(u)\|^2.$$

More precisely, we first compute the initial reduction

$$s^{(k)} = (g^{(k)})^T \sum_{i=1}^{N} J_{S_i}^{-1} J p^{(k)}. \tag{4.2}$$

Then, $\lambda^{(k)}$ is picked such that

$$f(u^{(k)} - \lambda^{(k)} p^{(k)}) \leq f(u^{(k)}) - \alpha \lambda^{(k)} s^{(k)}.$$

Here α is a pre-selected parameter (use $\alpha = 10^{-4}$). The standard cubic backtracking algorithm [8] is used in our computations.

5. Some analysis of ASPIN. In this section we provide some preliminary analysis of ASPIN. We show that, under certain conditions, the original nonlinear system (1.1) and the nonlinearly preconditioned system (2.3) have the same solution. Such kind of analysis is trivial for the case of linear preconditioning, but not so for the case of nonlinear preconditioning. Our analysis will be based mostly on the results of [4, 10].

We consider the nonlinear problem (1.1). Let D be a neighborhood of the exact solution u^*. We assume that the function $F(u)$ is well-defined in D, and in addition, we assume that

ASSUMPTION 5.1. *$F'(u)$ is continuous in D and the matrix $F'(u^*)$ is nonsingular.*

These assumptions are satisfied, for example, for the class of monotone elliptic partial differential equations [4, 10]. Following the nonlinear additive Schwarz theory of [10], we have

LEMMA 5.1 (Dryja and Hackbusch). *The subproblems (2.1) are all uniquely solvable in a neighborhood of u^* in D.*

LEMMA 5.2 (Dryja and Hackbusch). *The matrix $\tilde{\mathcal{J}} = \sum_{i=1}^{N} J_{S_i}^{-1} J$ is nonsingular for any u in a neighborhood of u^* in D.*

The difference between $\tilde{\mathcal{J}}$ and $\mathcal{J}$ is that, in the latter case, the subdomain derivatives are taken at the point $v_{S_i} = u_{S_i} - T_i(u_{S_i}, u_{S_i^c})$ instead of u_{S_i}. Due to the continuity assumption on F, we know that $T_i(u) \to 0$ as $u \to u^*$. Therefore using Assumption 5.1, Lemma 5.1, and Lemma 5.2, it is clear that there exists a subdomain D' satisfying $u^* \in D' \subset D$, such that

LEMMA 5.3. *The matrix $\mathcal{J}$, defined by (3.1), (3.2) and (3.8), is nonsingular for any $u \in D'$.*

THEOREM 5.4. *Under Assumption 5.1, the nonlinear systems (1.1) and (2.3) are equivalent in the sense that they have the same solution.*

Proof. Let us first assume that u^* is the solution of (1.1), i.e., $F(u^*) = 0$, which immediately implies that

$$F_{S_i}(u^*) = 0. \tag{5.1}$$

By definition, T_i satisfies

$$F_{S_i}(u^* - T_i(u^*)) = 0. \tag{5.2}$$

Comparing (5.1) and (5.2), and using the local unique solvability Lemma 5.1, we must have

$$T_i(u^*) = 0$$

for $i = 1, \ldots, N$. Therefore, u^* is a solution of (2.3); i.e.,

$$\sum_{i=1}^{N} T_i(u^*) = 0. \tag{5.3}$$

This proves not only that the solution of (1.1) is a solution of (2.3), but also that (2.3) has at least one solution. Next, we show that (2.3) has *only* one solution in the common neighborhood of Lemmas 5.1 and 5.2, which will then imply that the solution must be u^*.

Using Lemma 5.3, we have that $\mathcal{F}'$ is nonsingular in a neighborhood of u^* in D, therefore the solution is unique, according to the inverse function theorem of calculus. This concludes the proof. □

We remark that our analysis doesn't require that the nonlinear problem (1.1) has a global unique solution. As long as the solution is locally isolated, our analysis should apply.

6. Numerical experiments. We show a few numerical experiments in this section using ASPIN, and compare with the results obtained using a standard inexact Newton's algorithm. We are mostly interested in the kind of problems on which the regular inexact Newton type algorithm does not work well. The application of ASPIN for a full potential equation can be found in the report [6], and in this paper, we shall focus our discussion on the following two-dimensional driven cavity flow problem [16], using the velocity-vorticity formulation, in terms of the velocity u, v, and the vorticity ω, defined on the unit square $\Omega = (0,1) \times (0,1)$,

$$\begin{cases} -\Delta u - \dfrac{\partial \omega}{\partial y} &= 0 \\ -\Delta v + \dfrac{\partial \omega}{\partial x} &= 0 \\ -\frac{1}{Re}\Delta \omega + u\dfrac{\partial \omega}{\partial x} + v\dfrac{\partial \omega}{\partial y} &= 0. \end{cases} \tag{6.1}$$

Here Re is Reynolds number. The boundary conditions are:

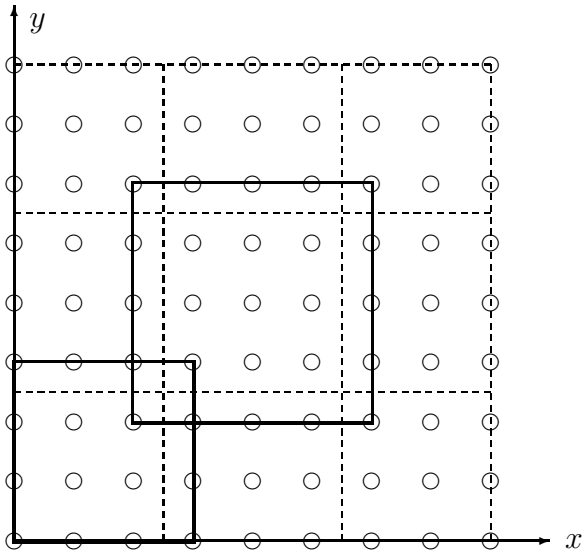

FIG. 6.1. *A* 9×9 *fine mesh with* 3×3 *subdomain partition. The 'o' are the mesh points. The dashed lines indicate a* $3 \times 3 = 9$ *subdomain nonoverlapping partitioning. The solid lines indicate the "overlapping = 1" subdomains.*

- bottom, left and right: $u = v = 0$
- top: $u = 1$, $v = 0$

We vary the Reynolds number in the experiments. The boundary condition on ω is given by its definition:

$$\omega(x, y) = -\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}. \tag{6.2}$$

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 knowns in the order of u, v, and ω. The partitioning of F is through the partitioning of the mesh points. In other words, the partition is neither physics-based nor element-based. Figure 6.1 shows a typical mesh, together with an overlapping partition. The subdomains may have different sizes depending on whether they touch the boundary of Ω. The size of the overlap is as indicated in Figure 6.1. Note that since this is mesh-point based partition, the zero overlap case in fact corresponds to the half-cell overlap case of the element-based partition, which is used more often in the literature on domain decomposition methods for finite element problems [11].

The subdomain Jacobian matrices J_{S_i} are formed using a multi-colored finite difference scheme.

The implementation is done using PETSc [1], and the results are obtained on a cluster of DEC workstations. Double precision is used throughout the computations. The initial iterate is zero for u, v and ω. Either 2×2 or 4×4 processor decompositions are used in all experiments reported here. We report here only the machine

independent properties of the algorithms.

6.1. Parameter definitions. We stop the global PIN iterations if

$$\|\mathcal{F}(u^{(k)})\| \leq \epsilon_{global-nonlinear}\|\mathcal{F}(u^{(0)})\|.$$

$\epsilon_{global-nonlinear} = 10^{-10}$ is used for all the tests. The global linear iteration for solving the global Jacobian system is stopped if the relative tolerance

$$\|\mathcal{F}(u^{(k)}) - \sum_{i=1}^{N} J_{S_i}^{-1} J(u^{(k)})p^{(k)}\| \leq \epsilon_{global-linear-rtol}\|\mathcal{F}(u^{(k)})\|$$

is satisfied. In fact we pick $\eta_k = \epsilon_{global-linear-rtol}$ independent of k, throughout the nonlinear iterations. Several different values of $\epsilon_{global-linear-rtol}$ are used as given in the tables below.

At the kth global nonlinear iteration, nonlinear subsystems

$$F_{S_i}\left(u^{(k)} - g_i^{(k)}\right) = 0,$$

defined in Step 1 a) of Algorithm 4.1, have to be solved. We use the standard IN with a cubic line search for such systems with initial guess $g_{i,0}^{(k)} = 0$. The local nonlinear iteration in subdomain S_i is stopped if the following condition is satisfied:

$$\left\|F_{S_i}\left(g_{i,l}^{(k)}\right)\right\| \leq \epsilon_{local-nonlinear-rtol}\left\|F_{S_i}\left(g_{i,0}^{(k)}\right)\right\|.$$

The overall cost of the algorithm depends heavily on the choice of $\epsilon_{local-nonlinear-rtol}$. We report computational results using a few different values for it.

6.2. Comparison with a Newton-Krylov-Schwarz algorithm. We compare the newly developed algorithm ASPIN with a well-understood inexact Newton algorithm using a cubic backtracking line search as the global strategy, as described in [8]. At each IN iteration, the Newton direction $p^{(k)}$ satisfies

$$\|F(u^{(k)}) - F'(u^{(k)})p^{(k)}\| \leq \eta_k\|F(u^{(k)})\|. \tag{6.3}$$

Several values of η_k were tested and the results are similar. The stagnation appears in all cases. Here we show the result with $\eta_k = 10^{-10}$. GMRES with an one-level additive Schwarz preconditioner is used as the linear solver with the same partition and overlap as in the corresponding ASPIN algorithm. The history of nonlinear residuals is shown in Figure 6.2 (top) with several different Reynolds numbers on a fixed fine mesh of size 128×128.

6.3. Test results and observations. As the Reynolds number increases, the nonlinear system becomes more and more difficult to solve. In this paper, the Reynolds number should be understood only as a parameter that controls the nonlinearity. We do not claim that the physics of the viscous boundary layer is resolved independently of Reynolds number for our fixed discretization. The Newton-Krylov-Schwarz algorithm fails to converge once the Reynolds number passes the value $Re = 770.0$ on this 128×128 mesh. Standard techniques for going further would employ pseudo-time stepping [19], nonlinear continuation in h or Re [29], or mesh sequencing [29]. ASPIN converges for a much larger range of Reynolds numbers as shown in Figure 6.2 without employing any of the standard tricks. Furthermore, the number of PIN iterations does

not change much as we increase the Reynolds number. A key to the success of the method is that the subdomain nonlinear problems are well solved, which helps balance the nonlinearities of the outer Newton iterations.

In Table 6.1, we present the numbers of global nonlinear PIN iterations and the numbers of global GMRES iterations per PIN iteration for various Reynolds numbers and overlapping sizes. Two key stopping parameters are $\epsilon_{global-linear-rtol}$ for the global linear Jacobian systems and $\epsilon_{local-nonlinear-rtol}$ for the local nonlinear systems. We test several combinations of two values 10^{-6} and 10^{-3}. As shown in the table, the total number of PIN iteration does not change much as we change $\epsilon_{global-linear-rtol}$ and $\epsilon_{local-nonlinear-rtol}$; however, it does increase from 2 or 3 to 6 or 9 when the Reynolds number increases from 1 to 10^4. The bottom part of Table 6.1 shows the corresponding numbers of GMRES iterations per PIN iteration. These linear iteration numbers change drastically as we switch to different stopping parameters. Solving the global Jacobian too accurately costs many GMRES iterations and does not result in corresponding savings in the total number of PIN iterations.

Table 6.1 also compares the results with two sizes of overlap. A small number of PIN iterations can be saved as one increases the overlapping size from 0 to 1, or more, as shown also in Table 6.3. The corresponding number of global linear iterations decreases considerably. We should mention that the size of subdomain nonlinear systems increases as one increases the overlap, especially for three dimensional problems. The communication cost in a distributed parallel implementation also increases as we increase the overlap. Recent experiments seem to indicate that small overlap, such as overlap=1, is preferred balancing the saving of the computational cost and the increase of the communication cost, see for example [11, 15]. Of course, the observation is highly machine and network dependent.

In Table 6.2, we look at the number of Newton iterations for solving the subdomain nonlinear systems. In this test case, we partition the domain into 16 subdomains, 4 in each direction, and number them naturally from the bottom to top, and left to right. Four subdomains Ω_{13}, Ω_{14}, Ω_{15}, and Ω_{16} touch the moving lid. The solution of the problem is less smooth near the lid, especially when the Reynolds number is large. As expected, the subdomains near the lid need more iterations; two to three times more than what is needed in the smooth subdomains for the large Reynolds number cases.

We next show how the iteration numbers change as we change the number of subdomains with a fixed 128×128 fine mesh. The results are displayed in Table 6.4. As we increase the number of subdomains from 4 to 16 the number of global PIN iterations does not change much; up or down by 1 is most likely due to the last bits of the stopping conditions rather than the change of the algorithm. Note that when we change the number of subdomains, the inexact Newton direction changes, and as a result, the algorithm changes. As a matter of fact, we are comparing two mathematically different algorithms. The bottom part of Table 6.4 shows that the number of GMRES iterations per PIN increases quite a bit as we increase the number of subdomains. In a one-level Schwarz preconditioned linear solver, we would expect the condition number to deteriorate linearly in the number of partitions in each coordinate direction and the iteration count to go up as the square root of this.

7. Some further comments. We comment on a few important issues concerning the newly proposed algorithm including parallel scalabilities and load balancing in parallel implementations.

TABLE 6.1

Global PIN iterations. Fine mesh 128×128*,* 4×4 *subdomain partition on 16 processors. Subdomain linear systems are solved exactly.* $\epsilon_{global-linear-rtol}$ *is the stopping condition for the global GMRES iterations.* $\epsilon_{local-nonlinear-rtol}$ *is the stopping condition for the local nonlinear iterations. The finite difference step size is* 10^{-8}*.*

	number of PIN iterations				
overlap	$Re = 10^0$	$Re = 10^1$	$Re = 10^2$	$Re = 10^3$	$Re = 10^4$
	$\epsilon_{global-linear-rtol} = 10^{-6}$, $\epsilon_{local-nonlinear-rtol} = 10^{-6}$				
0	2	3	4	8	7
1	2	3	4	7	7
	$\epsilon_{global-linear-rtol} = 10^{-6}$, $\epsilon_{local-nonlinear-rtol} = 10^{-3}$				
0	2	3	4	8	7
1	2	3	4	8	7
	$\epsilon_{global-linear-rtol} = 10^{-3}$, $\epsilon_{local-nonlinear-rtol} = 10^{-6}$				
0	4	4	5	7	8
1	3	3	4	7	7
	$\epsilon_{global-linear-rtol} = 10^{-3}$, $\epsilon_{local-nonlinear-rtol} = 10^{-3}$				
0	4	4	5	7	9
1	3	2	4	7	6
	number of GMRES iterations per PIN				
	$\epsilon_{global-linear-rtol} = 10^{-6}$, $\epsilon_{local-nonlinear-rtol} = 10^{-6}$				
0	118	100	152	146	122
1	71	61	87	90	69
	$\epsilon_{global-linear-rtol} = 10^{-6}$, $\epsilon_{local-nonlinear-rtol} = 10^{-3}$				
0	118	100	153	141	130
1	71	62	87	80	69
	$\epsilon_{global-linear-rtol} = 10^{-3}$, $\epsilon_{local-nonlinear-rtol} = 10^{-6}$				
0	60	61	56	46	34
1	42	37	40	32	27
	$\epsilon_{global-linear-rtol} = 10^{-3}$, $\epsilon_{local-nonlinear-rtol} = 10^{-3}$				
0	60	61	56	45	37
1	42	37	40	31	26

Parallel scalability is a very important issue when using linear or nonlinear iterative methods for solving problems with a large number of unknowns on machines with a large number of processors. It usually involves two separate questions, namely how the iteration numbers change with the number of processors and with the number of unknowns. From our limited experience, the number of ASPIN iterations is not sensitive at all to either the number of processors or the number of unknowns. In other words, the number of nonlinear PIN iterations is completely scalable as is often seen with Newton's method on PDE problems. However, this can not be carried over to the linear solver. To make the linear solver scalable, a coarse grid space is definitely needed. Our current software implementation is not capable of dealing with the coarse space, therefore no further discussion of this issue can be offered at this point.

Load balancing is another important issue for parallel performance that we do not

TABLE 6.2

Total number of subdomain nonlinear iterations. Fine mesh 128×128*,* 4×4 *subdomain partition on 16 processors. Subdomains are naturally ordered. Subdomain linear systems are solved exactly.* $\epsilon_{global-linear-rtol} = 10^{-3}$ *is the stopping condition for the global GMRES iterations.* $\epsilon_{local-nonlinear-rtol} = 10^{-3}$ *is the stopping condition for the local nonlinear iterations.* $overlap = 1$*. The finite difference step size is* 10^{-8}*.*

subdomain #	$Re = 10^0$	$Re = 10^1$	$Re = 10^2$	$Re = 10^3$	$Re = 10^4$
Ω_1	2	2	3	7	12
Ω_2	2	2	3	6	8
Ω_3	2	2	3	6	8
Ω_4	2	2	3	7	9
Ω_5	2	2	3	8	10
Ω_6	2	2	3	7	10
Ω_7	2	2	3	6	9
Ω_8	2	2	3	9	10
Ω_9	2	2	3	10	12
Ω_{10}	2	2	3	9	10
Ω_{11}	2	2	3	8	10
Ω_{12}	2	2	3	10	10
Ω_{13}	4	4	6	12	18
Ω_{14}	4	4	6	13	19
Ω_{15}	4	4	6	13	18
Ω_{16}	4	4	7	15	19

TABLE 6.3

Varying the overlapping size. Fine mesh 128×128*,* 4×4 *subdomain partition on 16 processors. Subdomain linear systems are solved exactly.* $\epsilon_{global-linear-rtol} = 10^{-3}$ *is the stopping condition for the global GMRES iterations.* $\epsilon_{local-nonlinear-rtol} = 10^{-3}$ *is the stopping condition for the local nonlinear iterations. The finite difference step size is* 10^{-8}*.*

overlap	0	1	2	3	4	5
PIN	7	7	6	6	4	6
GMRES/PIN	45	31	23	21	18	17

TABLE 6.4

Different subdomain partitions with the same fine mesh 128×128*. Subdomain linear systems are solved exactly.* $\epsilon_{global-linear-rtol} = 10^{-3}$ *is the stopping condition for the global GMRES iterations.* $\epsilon_{local-nonlinear-rtol} = 10^{-3}$ *is the stopping condition for the local nonlinear iterations.* $overlap = 1$*. The finite difference step size is* 10^{-8}*.*

	number of PIN iterations				
subdomain partition	$Re = 10^0$	$Re = 10^1$	$Re = 10^2$	$Re = 10^3$	$Re = 10^4$
$N = 2 \times 2 = 4$	3	3	4	6	7
$N = 4 \times 4 = 16$	3	2	4	7	6
	number of GMRES iterations per PIN				
$N = 2 \times 2 = 4$	22	21	23	20	15
$N = 4 \times 4 = 16$	42	37	40	31	26

TABLE 6.5

Different fine meshes on 16 processors. Subdomain linear systems are solved exactly. $\epsilon_{global-linear-rtol} = 10^{-3}$ *is the stopping condition for the global GMRES iterations.* $\epsilon_{local-nonlinear-rtol} = 10^{-3}$ *is the stopping condition for the local nonlinear iterations. overlap* $= 1$. *The finite difference step size is* 10^{-8}.

	number of PIN iterations				
fine mesh	$Re = 10^0$	$Re = 10^1$	$Re = 10^2$	$Re = 10^3$	$Re = 10^4$
32×32	3	3	4	6	6
64×64	3	3	4	6	7
128×128	3	2	4	7	6
	number of GMRES iterations per PIN				
32×32	16	16	18	15	16
64×64	25	25	24	22	20
128×128	42	37	40	31	26

address in the paper. As shown in Table 6.2, the computational cost is much higher in the subdomains near the lid than the other subdomains, in particular for the large Reynolds number cases. To balance the computational load, ideally, one should partition the domain such that these subdomains that require more linear/nonlinear iterations contain less mesh points. However, the solution dependent cost information is not available until after a few iterations, and therefore the ideal partition has to obtained dynamically as the computation is being carried out.

We discussed only one partitioning strategy based on the geometry of the mesh and the number of processors available in our computing system. Many other partitioning strategies need to be investigated. For example, physics-based partitions: all the velocity unknowns as Ω_1 and the vorticity unknowns as Ω_2. In this case, the number of subdomains may have nothing to do with the number of processors. Further partitions may be needed on both Ω_1 and Ω_2 for the purpose of parallel processing. One possible advantage of this physics-based partition is that the nonlinearities between different physical quantities can be balanced.

An extreme case of a mesh-based partition would be that each subdomain contains only one grid point. Then, the dimension of the subdomain nonlinear system is the same as the number of variables associated with a grid point, 3 for our test case. In this situation, ASPIN becomes a pointwise nonlinear scaling algorithm. As noted in [8], linear scaling does not change the nonlinear convergence of Newton's method, but nonlinear scaling does. Further investigation should be of great interest.

Acknowledgement: We thank the anonymous referees and L. Marcinkowski for many valuable suggestions. We also thank the PETSc team, S. Balay, W. Gropp, L. McInnes, and B. Smith, for their help in the software implementation of the algorithms.

REFERENCES

[1] S. BALAY, W. GROPP, L. MCINNES, AND B. SMITH, *The Portable, Extensible Toolkit for Scientific Computing, version 2.0.28*, `http://www.mcs.anl.gov/petsc`, code and documentation, 2000.

[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, 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.
[7] T. Chan and K. Jackson, *Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 533–542.
[8] J. E. Dennis and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, NJ, 1983.
[9] R. S. Dembo, S. C. Eisenstat, and T. Steihaug, *Inexact Newton methods*, SIAM J. Numer. Anal., 19 (1982), pp. 400–408.
[10] M. Dryja and W. Hackbusch, *On the nonlinear domain decomposition method*, BIT, (1997), pp. 296-311.
[11] M. Dryja and O. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.
[12] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optimization, 4 (1994), pp. 393–422.
[13] 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.
[14] G. H. Golub and C. Van Loan, *Matrix Computations*, Academic Press, New York, 1981.
[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] 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.
[18] C. T. Kelley, *Iterative Methods for Linear and Nonlinear Equations*, SIAM, Philadelphia, 1995.
[19] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM J. Num. Anal., 35 (1998), pp. 508-523.
[20] 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.
[21] P. L. Lions, *On the Schwarz alternating method. I*, First International Symposium on Domain Decomposition Methods for Partial Differential Equations, R. Glowinski, G. H. Golub, G. A. Meurant, and J. Périaux, eds., SIAM, Philadelphia, pp. 1-42, 1988.
[22] P. L. Lions, *On the Schwarz alternating method. II*, Domain Decomposition Methods, T. Chan, R. Glowinski, J. Périaux, and O. Widlund, eds., SIAM, Philadelphia, pp. 47–70, 1989.
[23] S. H. Lui, *On Schwarz alternating methods for incompressible Navier-Stokes equations in n dimensions*, Tech. Report, Department of Mathematics, Hong Kong Univ. of Sci. and Tech., 1998.
[24] M. Paraschivoiu, 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.
[25] M. Pernice and H. Walker, *NITSOL: A Newton iterative solver for nonlinear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 302–318.
[26] B. F. Smith, P. E. Bjørstad, and W. D. Gropp, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, 1996.
[27] X.-C. Tai and M. Espedal, *Rate of convergence of some space decomposition methods for linear and nonlinear problems*, SIAM J. Numer. Anal., 35 (1998), pp. 1558–1570.
[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.

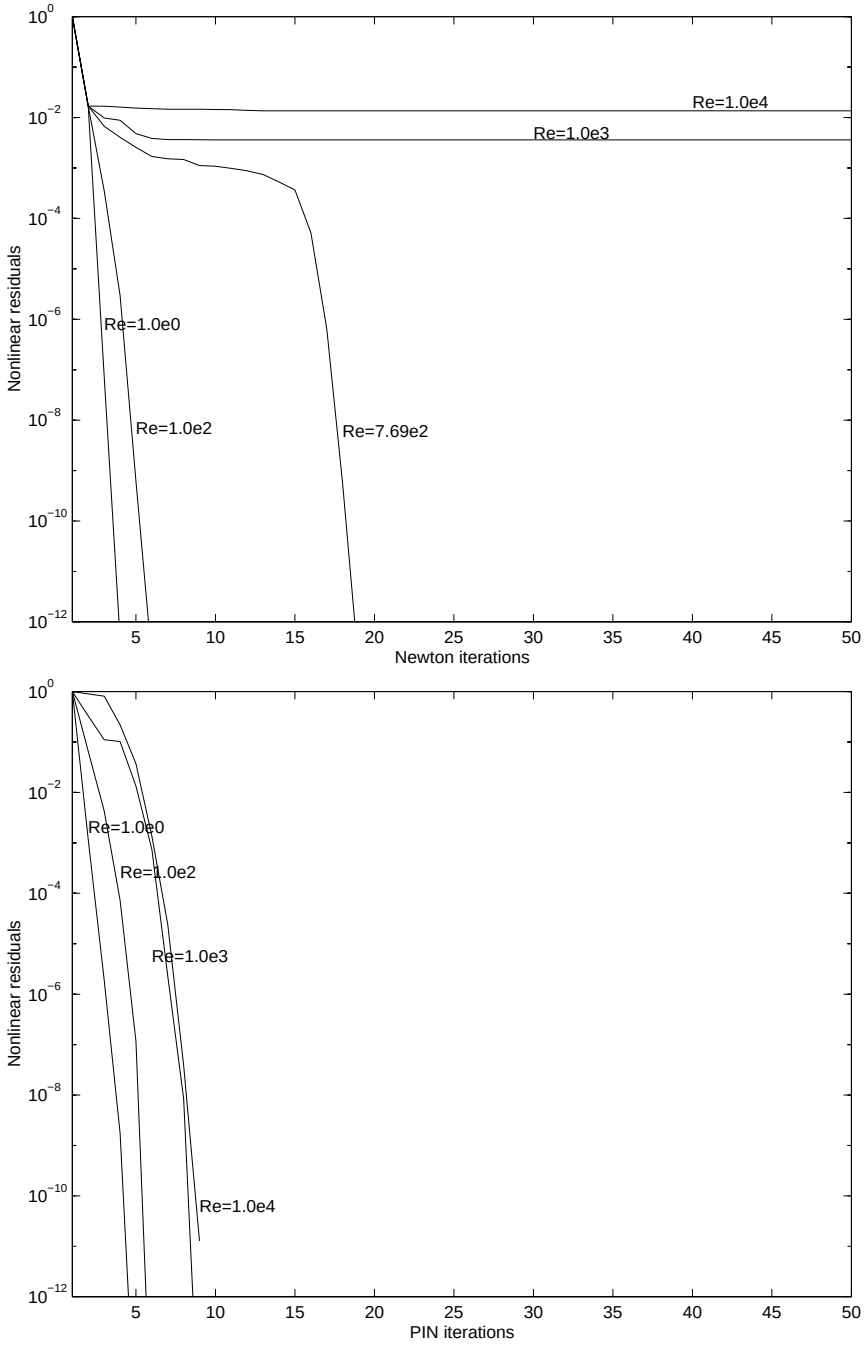

FIG. 6.2. *Nonlinear residual history for the flow problem with different Reynolds numbers.*