Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 196 (2007) 1603–1611
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

A class of parallel two-level nonlinear Schwarz preconditioned inexact Newton algorithms ☆

Feng-Nan Hwang [a], Xiao-Chuan Cai [b,*]

[a] Department of Mathematics, National Central University, Jhongli City, Taoyuan County 32001, Taiwan
[b] Department of Computer Science, University of Colorado at Boulder, Campus Box 430, Boulder, CO 80309, USA

Received 2 July 2005; received in revised form 27 December 2005; accepted 5 March 2006

Abstract

We propose and test a new class of two-level nonlinear additive Schwarz preconditioned inexact Newton algorithms (ASPIN). The two-level ASPIN combines a local nonlinear additive Schwarz preconditioner and a global *linear* coarse preconditioner. This approach is more attractive than the two-level method introduced in [X.-C. Cai, D.E. Keyes, L. Marcinkowski, Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics, Int. J. Numer. Methods Fluids, 40 (2002), 1463–1470], which is nonlinear on both levels. Since the coarse part of the global function evaluation requires only the solution of a linear coarse system rather than a nonlinear coarse system derived from the discretization of original partial differential equations, the overall computational cost is reduced considerably. Our parallel numerical results based on an incompressible lid-driven flow problem show that the new two-level ASPIN is quite scalable with respect to the number of processors and the fine mesh size when the coarse mesh size is fine enough, and in addition the convergence is not sensitive to the Reynolds numbers.

Keywords: Incompressible Navier–Stokes equations; Multilevel nonlinear preconditioning; Inexact Newton; Nonlinear additive Schwarz; Domain decomposition; Parallel computing

1. Introduction

One of the challenging problems in computational fluid dynamics is to solve large, sparse, nonlinear systems of equations arising from the discretization of incompressible Navier–Stokes equations at high Reynolds numbers [16,21,24], particularly when the solution has boundary layers or singularities. For such problems most existing nonlinear iterative methods [3,8,18] do not work well. They are either not robust enough to deal with a wide range of Reynolds numbers, e.g. Newton method, or show an unacceptably slow convergence, e.g. continuation methods. Two properties are highly desirable in the design of an algorithm for solving such system of equations on parallel computers: *robustness* and *scalability*. An algorithm is called robust if the convergence is not too sensitive to changes of some system parameters, for example, the initial guess for iterative methods, the mesh size, and other physical parameters, such as the Reynolds number. An algorithm is called scalable if the iteration count (and the computing timing) is nearly constant as the number of processors and the number of unknowns increase proportionally. To remedy these issues, we develop a general multilevel nonlinear preconditioning technique that is fast, robust and scalable. Our method is based on the inexact Newton method with backtracking (INB) [8,18] and nonlinear additive Schwarz methods [4–6,12,13]. We briefly describe INB below. Let

☆ The research was supported in part by the Department of Energy, DE-FC02-01ER25479, and in part by the US National Science Foundation, CCR-0219190, ACI-0072089 and ACI-0305666. The first author was also supported in part by the National Science Council in Taiwan, NSC94-2115-M-008-017.
* Corresponding author. Fax: +1 303 4922844.
E-mail addresses: hwangf@math.ncu.edu.tw (F.-N. Hwang), cai@cs.colorado.edu (X.-C. Cai).

0045-7825/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2006.03.019

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

be a nonlinear system of equations and $x^{(0)}$ a given initial guess. Assume $x^{(k)}$ is the current approximate solution. Then a new approximate solution $x^{(k+1)}$ of (1.1) can be computed by first finding an inexact Newton direction $s^{(k)}$ satisfying $\|F(x^{(k)}) + F'(x^{(k)})s^{(k)}\|_2 \leqslant \eta_k \|F(x^{(k)})\|_2$, then by obtaining $x^{(k+1)}$ with $x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}$, where the parameter $\lambda^{(k)}$ is computed via backtracking. In INB, the scalar η_k is often called the "forcing term", which determines how accurately the Jacobian system needs to be solved by some iterative method, such as a Krylov subspace type method, GMRES [20]. If the chosen forcing terms are small enough, the algorithm reduces to the exact Newton algorithm. The scalar $\lambda^{(k)}$ is selected so that $f(x^{(k)} + \lambda^{(k)} s^{(k)}) \leqslant f(x^{(k)}) + \alpha \lambda^{(k)} \nabla f(x^{(k)})^{\mathrm{T}} s^{(k)}$, where the merit function f is defined as $\|F(x)\|_2^2/2$. Here, a line search technique [8] is employed to determine the step length $\lambda^{(k)}$, and the parameter α is used to assure that the reduction of f is sufficient. Although INB has the desirable property of local fast convergence, like other nonlinear iterative methods, it is very fragile. It converges rapidly for a well-selected set of parameters (certain initial guesses and ranges of Reynolds numbers), but diverges if we slightly change some of these parameters. On the other hand, it may converge well at the beginning of the iterations, then suddenly stall for no apparent reason [4,13].

Recently, in [4,12,13], some nonlinear preconditioning methods were developed, and the nonlinear convergence of these methods is much less sensitive to these unfriendly parameters if INB is applied instead to the so-called nonlinearly preconditioned system

$$\mathscr{F}(x^*) = 0. \tag{1.2}$$

Here the word "preconditioner" refers to the fact that the systems (1.1) and (1.2) have the same solution and the new system (1.2) is, in some sense, better conditioned, both linearly and nonlinearly. One preconditioner is constructed by using some one-level nonlinear additive Schwarz method. To a certain extent, the robustness problem is solved by the one-level method, which converges for a wide range of Reynolds numbers and mesh sizes. However, the parallel scalability remains an open issue due to the lack of communication between subdomains. To improve the inter-subdomain communication, a two-level method was then proposed in [5], which is nonlinear on both the coarse and fine levels. The method works well if the number of processors is small, but when the number of processors is large, the *nonlinear* coarse solver require too much CPU and communication times [17]. Hence, we propose a class of combined *linear and nonlinear* additive Schwarz preconditioner and show numerically that by using a linear coarse preconditioner we can maintain the nonlinear robustness and at the same time reduce considerably the nonlinear complexity. A preliminary version of the new method was discussed in a short proceedings paper [14].

The paper is organized as follows. In Section 2, we discuss two-level nonlinear Schwarz preconditioned inexact Newton algorithms and introduce a new linear coarse preconditioner, which plays the central role in the scalability of the algorithm. We also describe two-dimensional steady-state incompressible Navier–Stokes equations and their finite element discretization, which is taken as an example to illustrate the applicability of the method. Then, in Section 3, we present some numerical results obtained on a parallel computer for a lid-driven cavity flow problem. Particularly, we focus on the parallel linear and nonlinear scalability of the method. Finally, Section 4 presents some concluding remarks.

2. Two-level nonlinearly preconditioned inexact Newton algorithms

In this section, we describe a new class of two-level preconditioners based on a combination of local nonlinear additive Schwarz preconditioners and a global linear coarse preconditioner. The local nonlinear preconditioners make the method more robust in the sense that the method is able to converge for a wide range of Reynolds numbers and mesh sizes, while the linear coarse preconditioner makes the method more scalable in the sense that the number of linear iterations does not depend much on the number of parallel processors. This is very important for solving large-scale problems on massively parallel computers.

2.1. A model problem

For simplicity, we restrict our discussion to a two-component system (velocity and pressure) resulting from the finite element discretization of incompressible Navier–Stokes equations. The generalization to other multi-component problems is straightforward. Consider two-dimensional steady-state incompressible Navier–Stokes equations in the primitive variable form [11,19] defined on a bounded domain Ω with a polygonal boundary Γ:

$$\begin{cases} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - 2\nu \nabla \cdot \epsilon(\boldsymbol{u}) + \nabla p = 0 & \text{in } \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in } \Omega, \\ \boldsymbol{u} = \boldsymbol{g} & \text{on } \Gamma, \end{cases} \tag{2.1}$$

where $\boldsymbol{u}$ is the velocity, p is the pressure, ν is the dynamic viscosity, which is inversely proportional to the Reynolds number, and $\epsilon(\boldsymbol{u}) = 1/2(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^{\mathrm{T}})$ is the symmetric part of the velocity gradient. Since only Dirichlet boundary condition is specified, the pressure p is determined up to a constant. An additional condition, $\int_\Omega p \, \mathrm{d}x = 0$, is imposed to make p unique. To discretize (2.1), we use a stabilized finite element method [2,9] on a given quadrilateral mesh $\mathscr{T}^h = \{K\}$. For each element K, we use h_K to denote its element diameter. Let V^h and P^h be a pair of finite element spaces for the velocity and pressure given by

$$V^h = \{\boldsymbol{v} \in (C^0(\Omega) \cap H^1(\Omega))^2 : \boldsymbol{v}|_K \in Q_1(K)^2, K \in \mathscr{T}^h\},$$
$$P^h = \{p \in C^0(\Omega) \cap L^2(\Omega) : p|_K \in Q_1(K), K \in \mathscr{T}^h\}.$$

Here $C^0(\Omega)$, $L^2(\Omega)$, and $H^1(\Omega)$ are the standard notations with the usual meanings in the finite element literature [11,19]. For simplicity, our implementation uses a $Q_1 - Q_1$ element (continuous bilinear velocity and pressure). The weighting and trial velocity function spaces V_0^h and V_g^h are

$$V_0^h = \{\boldsymbol{v} \in V^h : \boldsymbol{v} = 0 \text{ on } \Gamma\} \quad \text{and}$$
$$V_g^h = \{\boldsymbol{v} \in V^h : \boldsymbol{v} = g \text{ on } \Gamma\}.$$

Similarly, let the finite element space P_0^h be both the weighting and trial pressure function spaces:

$$P_0^h = \left\{p \in P^h : \int_\Omega p \,\mathrm{d}x = 0\right\}.$$

Following [9], the stabilized finite element method for steady-state incompressible Navier–Stokes equations reads: find $\boldsymbol{u}^h \in V_g^h$ and $p^h \in P_0^h$, such that

$$B(\boldsymbol{u}^h, p^h; \boldsymbol{v}, q) = 0 \quad \forall (\boldsymbol{v}, q) \in V_0^h \times P_0^h \tag{2.2}$$

with

$$\begin{aligned} B(\boldsymbol{u}, p; \boldsymbol{v}, q) &= ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u}, \boldsymbol{v}) + (2\nu\epsilon(\boldsymbol{u}), \epsilon(\boldsymbol{v})) - (\nabla \cdot \boldsymbol{v}, p) - (\nabla \cdot \boldsymbol{u}, q) \\ &\quad + \sum_{K \in \mathscr{T}^h} ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u} + \nabla p, \tau((\nabla \boldsymbol{v}) \cdot \boldsymbol{v} - \nabla q))_K + (\nabla \cdot \boldsymbol{u}, \delta \nabla \cdot \boldsymbol{v}). \end{aligned}$$

We use the stabilization parameters δ and τ as suggested in [9]. The stabilized finite element formulation (2.2) can be written as a nonlinear algebraic system

$$F(x) = 0, \tag{2.3}$$

which is often large, sparse, and highly nonlinear when the value of Reynolds number is large. The vector x corresponds to the nodal values of $\boldsymbol{u}^h = (u_1^h, u_2^h)$ and p^h in (2.2).

2.2. Subdomain partition, and one-level nonlinear preconditioner

To define parallel Schwarz-type preconditioners [22,23], we partition the finite element mesh $\mathscr{T}^h$ introduced in the previous section. Let $\{\Omega_i^h, i = 1, \ldots, N\}$ be a non-overlapping subdomain partition whose union covers the entire domain Ω and its mesh $\mathscr{T}^h$. We use $\mathscr{T}_i^h$ to denote the collection of mesh points in Ω_i^h. To obtain overlapping subdomains, we expand each subdomain Ω_i^h to a larger subdomain $\Omega_i^{h,\delta}$ with the boundary $\partial\Omega_i^{h,\delta}$. Here δ is an integer indicating the degree of overlap. We assume that $\partial\Omega_i^{h,\delta}$ does not cut any elements of $\mathscr{T}^h$. Similarly, we use $\mathscr{T}_i^{h,\delta}$ to denote the collection of mesh points in $\Omega_i^{h,\delta}$, including $\partial\Omega_i^{h,\delta}$.

Now, we define the subdomain velocity space as

$$V_i^h = \{v^h \in V^h \cap (H^1(\Omega_i^{h,\delta}))^2 : v^h = 0 \text{ on } \partial\Omega_i^{h,\delta}\}$$

and the subdomain pressure space as

$$P_i^h = \{p^h \in P^h \cap L^2(\Omega_i^{h,\delta}) : p^h = 0 \text{ on } \partial\Omega_i^{h,\delta} \setminus \Gamma\}.$$

On the physical boundaries, we impose Dirichlet conditions according to the original (2.1). On the artificial boundaries, we assume both $\boldsymbol{u} = 0$ and $p = 0$. Similar boundary conditions were used in [15].

Let $R_i : V^h \times P^h \to V_i^h \times P_i^h$ be a restriction operator, which returns all degrees of freedom (both velocity and pressure) associated with the subspace $V_i^h \times P_i^h$. R_i is an $(3n_i - 2r_i) \times (3n - 2r)$ matrix with values of either 0 or 1, where n and n_i are the total number of mesh points in $\mathscr{T}^h$ and $\mathscr{T}_i^{h,\delta}$, respectively, and r and r_i are the total number of mesh points at which the Dirichlet boundary condition for velocity in $\mathscr{T}^h$ and $\mathscr{T}_i^{h,\delta}$ is imposed, respectively. Since the element-based partitioning is used, $\sum_{i=1}^N (3n_i - 2r_i) \geqslant (3n - 2r)$. Note that for $Q_1 - Q_1$ elements, we have three variables per interior mesh point, two for the velocity and one for the pressure. Then, the interpolation operator R_i^{T} can be defined as the transpose of R_i. The multiplication of R_i (and R_i^{T}) with a vector does not involve any arithmetic operation, but does involve communication in a distributed memory parallel implementation. Using the restriction operator, we define the subdomain nonlinear function $F_i : R^{3n-2r} \to R^{3n_i-2r_i}$ as

$$F_i = R_i F.$$

We next define the subdomain mapping functions, which in some sense play the role of subdomain preconditioners. For any given $x \in R^{3n-2r}$, $T_i(x) : R^{3n-2r} \to R^{3n_i-2r_i}$ is defined as the solution of the following subspace nonlinear systems,

$$F_i(x - R_i^{\mathrm{T}} T_i(x)) = 0 \quad \text{for } i = 1, \ldots, N. \tag{2.4}$$

Throughout this paper, we assume that Eq. (2.4) is uniquely solvable. Using the subdomain mapping functions, we introduce a new global nonlinear function,

$$\mathscr{F}^{(1)}(x) = \sum_{i=1}^N R_i^{\mathrm{T}} T_i(x), \tag{2.5}$$

to which we refer as the one-level nonlinearly preconditioned $F(x)$. As shown in [4,13], an approximation of the Jacobian of $\mathscr{F}^{(1)}$ takes the form

$$\widehat{\mathscr{J}}^{(1)}(x) = \sum_{i=1}^N R_i^{\mathrm{T}} J_i^{-1} R_i J(x), \tag{2.6}$$

where J is the Jacobian of the original function $F(x)$ and $J_i = R_i J R_i^{\mathrm{T}}$. The one-level additive Schwarz inexact preconditioned Newton (ASPIN) algorithm is defined as: find the solution x^* of (2.3) by solving the nonlinearly preconditioned system,

$$\mathscr{F}^{(1)}(x) = 0, \tag{2.7}$$

using INB with an initial guess $x^{(0)}$. It is noted that $\widehat{\mathscr{J}}^{(1)}$ can be viewed as the original Jacobian J preconditioned by a one-level additive Schwarz preconditioner. Hence,

according the classical Schwarz theory, $\widehat{\mathscr{J}}^{(1)}$ is well-conditioned only for the case with a small number of processors. In other words, the number of linear iterations for solving the global Jacobian system is expected to increase as the number of processors increases.

2.3. A linear coarse component for the nonlinear preconditioner

The one-level ASPIN is robust, but not linearly scalable with respect to the number of processors. A coarse preconditioner is required to couple the local subdomain preconditioners. One such coarse preconditioner, which is proposed and tested in [5,17], is based on a nonlinear coarse problem. To be more specific, the nonlinear coarse system can be defined as follows. Consider the nonlinear coarse system, $F^c(x_c^*) = 0$, where F^c is the discretization of original nonlinear PDEs on a coarse mesh $\mathscr{T}^H$ covering the domain Ω. The solution x_c^* is assumed to be uniquely determined and available through a preprocessing step. We define the coarse-to-fine and fine-to-coarse mesh transfer operators. Let $\{\phi_j^H(\xi), j = 1, \ldots, m\}$ be the finite element basis functions on the coarse mesh, where m is the total number of coarse mesh points in $\mathscr{T}^H$. We define an $(3n-2r)\times(3m-2q)$ matrix I_H^h, the coarse-to-fine extension matrix, as

$$I_H^h = [E_1 E_2 \cdots E_n]^{\mathrm{T}},$$

where the block matrix E_i of size $3\times(3m-2q)$ is given by

$$E_i = \begin{bmatrix} (e_H^h)_i^v & 0 & 0 \\ 0 & (e_H^h)_i^v & 0 \\ 0 & 0 & (e_H^h)_i^p \end{bmatrix}$$

and the row vector $(e_H^h)_i^v$ of length $m-q$ for the velocity is given by

$$(e_H^h)_i^v = \left[\phi_1^H(\xi_i), \phi_2^H(\xi_i), \ldots, \phi_m^H(\xi_i)\right], \quad \xi_i \in \mathscr{T}^h \setminus \Gamma$$

for $i = 1, \ldots, n-r$. Similarly, the row vector $(e_H^h)_i^p$ of length m for the pressure is given by

$$(e_H^h)_i^p = \left[\phi_1^H(\xi_i), \phi_2^H(\xi_i), \ldots, \phi_m^H(\xi_i)\right], \quad \xi_i \in \mathscr{T}^h$$

for $i = 1, \ldots, n$. A global coarse-to-fine extension operator I_H^h can be defined as the transpose of I_h^H. Then, the coarse function is defined as

$$S_0(x) = S^c(x) - x_c^*, \tag{2.8}$$

where $y = S^c(x): R^{3n-2r} \to R^{3m-2q}$ is the solution of the nonlinear coarse system,

$$F^c(y) = I_h^H F(x). \tag{2.9}$$

Note that to evaluate the coarse function $S_0(x)$ at any point, one needs to solve the nonlinear system of equations defined in (2.9). In general, this coarse system is easier to solve than the fine mesh system, but a Newton–Krylov–Schwarz method is sometimes not good enough to converge the system. Therefore, as suggested in [5,17], the one-level ASPIN is used to solve the coarse system. Although the one-level ASPIN-based coarse preconditioner provides good mathematical properties, such as acceleration of the convergence of the linear iterative method, in practice, the computational cost of solving many coarse systems is usually high. Numerical experiments [17] show that the one-level ASPIN based coarse preconditioner works fine only for a small number of processors; for a large number of processors, a more efficient coarse preconditioner is needed.

In this paper, we introduce a new coarse system, which is linear, and the system is constructed by a linearization of the nonlinear coarse system mentioned above, using a Taylor approximation. The coarse function evaluation requires only the solution of a linear system, and hence the computational cost is reduced considerably compared to the algorithm of [5].

To define the coarse function $T_0: R^{3n-2r} \to R^{3m-2q}$, we introduce a projection $T^c: R^{3n-2r} \to R^{3m-2q}$ as the solution of the linearized coarse system of (2.9)

$$F^c(x_c^*) + J^c(x_c^*)(T^c(x) - x_c^*) = I_h^H F(x), \tag{2.10}$$

for any given $x \in R^{3n-2r}$. Note that the left-hand side of (2.10) is a first-order Taylor approximation of $F^c(x)$ at the exact coarse-mesh solution, x_c^*. Since $F^c(x_c^*) = 0$, we can rewrite (2.10) as

$$T^c(x) = x_c^* + (J^c(x_c^*))^{-1} I_h^H F(x),$$

provided that $J^c(x_c^*)$ is nonsingular. It is easy to see that $T^c(x^*)$ can be computed without knowing the exact solution x^* of F, and $T^c(x^*) = x_c^*$. Then the coarse function can be defined as

$$T_0(x) = T^c(x) - T^c(x^*) = (J^c(x_c^*))^{-1} I_h^H F(x)$$

and its Jacobian is given by

$$\frac{\partial T_0(x)}{\partial x} = (J^c(x_c^*))^{-1} I_h^H J(x). \tag{2.11}$$

We now define a new two-level additive nonlinearly preconditioned function

$$\mathscr{F}^{(2)}(x) = I_H^h T_0(x) + \sum_{i=1}^{N} R_i^{\mathrm{T}} T_i(x), \tag{2.12}$$

and by combining (2.11) and (2.6), we obtain an approximation of the Jacobian of $\mathscr{F}^{(2)}$ in the form

$$\widehat{\mathscr{J}}^{(2)}(x) = \left\{ I_H^h (J^c(x_c^*))^{-1} I_h^H + \sum_{i=1}^{N} \left[R_i^{\mathrm{T}} (J_i(x))^{-1} R_i \right] \right\} J(x).$$

The two-level Schwarz preconditioned inexact Newton algorithm with a linear coarse solver is defined as: find the solution x^* of (2.3) by solving the nonlinearly preconditioned system

$$\mathscr{F}^{(2)}(x) = 0, \tag{2.13}$$

using INB with an initial guess $x^{(0)}$.

Remark 1. In the linear case, $F(x)=0$, where $F(x)=Ax-b$, each component of $\mathscr{F}^{(2)}$ can be written down explicitly as $T_0=(A^c)^{-1}I_h^H(Ax-b)$ and $T_i=(A_i)^{-1}R_i(Ax-b)$. Here $(A^c)^{-1}$ and A_i^{-1} are the subspace inverses of $A^c=I_h^HAI_H^h$ and $A_i=R_iAR_i^{\mathrm{T}}$, respectively. Hence,

$$\mathscr{F}^{(2)}(x)=\left\{I_H^h(A^c)^{-1}(I_h^H)+\left(\sum_{i=1}^{N}R_i^{\mathrm{T}}A_i^{-1}R_i\right)\right\}(Ax-b)=0.$$

This is indeed a two-level additive Schwarz preconditioned linear system.

The details of the two-level ASPIN are given below. Let $x^{(0)}$ be an initial guess and $x^{(k)}$ the current approximate solution. Then a new approximate solution $x^{(k+1)}$ can be computed with the two-level ASPIN algorithm as follows:

Algorithm 1 (*Two-level ASPIN*)

Step 1: Evaluate the nonlinear residual $\mathscr{F}^{(2)}(x)$ at $x^{(k)}$ through the following steps:

(1) Find $w_0^{(k)}=T_0(x^{(k)})$ by solving the linear coarse problem,

$$J^c(x_c^*)z_c=I_h^HF(x^{(k)}). \tag{2.14}$$

(2) Find $w_i^{(k)}=T_i(x^{(k)})$ by solving in parallel, the local nonlinear systems,

$$G_i(w)\equiv F_i(x^{(k)}-w)=0. \tag{2.15}$$

(3) Form the global residual

$$\mathscr{F}^{(2)}(x^{(k)})=I_H^hw_0^{(k)}+\sum_{i=1}^{N}R_i^{\mathrm{T}}w_i^{(k)}.$$

Step 2: Check the stopping condition on $\|\mathscr{F}^{(2)}(x^{(k)})\|_2$. If $\|\mathscr{F}^{(2)}(x^{(k)})\|_2$ is small enough, stop, otherwise, continue.

Step 3: Find an inexact Newton direction $s^{(k)}$ by solving the following Jacobian system approximately using a Krylov subspace method

$$\widehat{\mathscr{J}}^{(2)}s^{(k)}=-\mathscr{F}^{(2)}(x^{(k)}) \tag{2.16}$$

in the sense that

$$\|\mathscr{F}^{(2)}(x^{(k)})+\widehat{\mathscr{J}}^{(2)}(x^{(k)})s^{(k)}\|_2\leqslant\eta_k\|\mathscr{F}^{(2)}(x^{(k)})\|_2 \tag{2.17}$$

for some $\eta_k\in[0,1)$.

Step 4: Scale the search direction $s^{(k)}\leftarrow\frac{s_{\max}}{\|s^{(k)}\|_2}s^{(k)}$ if $\|s^{(k)}\|_2\geqslant s_{\max}$.

Step 5: Compute a new approximate solution

$$x^{(k+1)}=x^{(k)}+\lambda^{(k)}s^{(k)},$$

where $\lambda^{(k)}$ is a damping parameter determined by the standard backtracking procedure.

Remark 2. No preconditioning is used in the solution of the linear system in Step 3 of Algorithm 1. In fact, $\widehat{\mathscr{J}}^{(2)}$ can be viewed as the Jacobian matrix for the original function, J preconditioned by a two-level additive Schwarz preconditioner, where the coarse part of the preconditioner $I_H^h(J^c(I_h^Hx^{(k)}))^{-1}I_h^H$ is approximated by $I_H^h(J^c(x_c^*))^{-1}I_h^H$. Hence, $\widehat{\mathscr{J}}^{(2)}$ is well-conditioned through nonlinear preconditioning as long as x_c^* is close to $I_h^Hx^{(k)}$.

Remark 3. Although each component of $\widehat{\mathscr{J}}^{(2)}$ is sparse, $\widehat{\mathscr{J}}^{(2)}$ itself is often dense and expensive to form explicitly. However, if a Krylov subspace method is applied to the global Jacobian system (2.16), only the Jacobian-vector product is required. For example, in a distributed-memory parallel implementation, the operation, $u=\widehat{\mathscr{J}}^{(2)}v$, consists of eight phases:

(1) Perform the matrix-vector multiply, $w=Jv$, in parallel.
(2) On each subdomain, do the following three steps:
  (a) Collect the data from the subdomain and its neighboring subdomains, $w_i=R_iw$.
  (b) Solve $J_ix_i=w_i$ using a sparse direct solver, and
  (c) Send/receive the partial solutions to/from its neighboring subdomain, $\hat{x}_i=R_i^{\mathrm{T}}x_i$.
(3) Restrict $y^c=I_h^Hw$.
(4) Solve the coarse linear system, $J^c(x_c^*)z^c=y^c$.
(5) Extend $y=I_H^hz^c$.
(6) Compute the sum, $u=\sum_{i=1}^{N}\hat{x}_i+y$.

Remark 4. As suggested by Dennis and Schnabel of [8, p. 129], we include a re-scaling of the search direction $s^{(k)}$ in Step 4 if $\|s^{(k)}\|_2\geqslant s_{\max}$ before entering the backtracking step. In general, the purposes of this step length constraint are to avoid very large steps during the calculation and to prevent the intermediate solution from leaving the domain of our interest. The scalar $s_{\max}$ is provided by the user. The optimal choices of $s_{\max}$ are determined empirically in practices and often depend on some physical parameter such as the Reynolds number, the mesh size and the number of processors. In [13], numerical results show that the re-scaling step plays an important role in enhancing the robustness of ASPIN for solving incompressible Navier–Stokes equations, especially when Re is high. With careful choices of $s_{\max}$, the efficiency of ASPIN can be improved as well.

Remark 5. The linear coarse problem,

$$J^c(x_c^*)x^c=y^c,$$

has to be solved with different right-hand side vectors y^c as part of nonlinear and linear iterations, for example, in the first sub-step of Step 1 of Algorithm 1 and Step 4 in Remark 3. Since in our algorithm the coarse matrix remains unchanged throughout the global nonlinear iterations as well as the global linear iterations, an LU decomposition based direct solver rather than a parallel iterative methods is a better choice. The coarse matrix needs to be decomposed only once at the beginning of the iterations, and

two matrices, upper and lower triangular matrices are stored. Then we perform the forward and backward substitutions at each coarse solve.

3. Numerical results

In this section, we consider a two-dimensional lid-driven cavity flow problem as a benchmark for evaluating the parallel performance of the new two-level ASPIN. The detailed description of the lid-driven cavity flow problem can be found in [10]. The Reynolds number (Re) for this test problem is defined as $Re = 1/v$, where v is the dynamic viscosity in (2.1). Our main focus is on the impact of the coarse mesh size and the subdomain overlapping size on the convergence rate and the overall execution time. In addition, we investigate the linear and nonlinear scalability of ASPIN for different values of the Reynolds number.

3.1. Implementation details and parameter selections

We use the Portable, Extensible Toolkits for Scientific computations (PETSc) [1] for the parallel implementation and obtain all numerical results on a cluster of workstations. In our implementation, after ordering the mesh points, we number the unknown nodal values in the order of u_1^h, u_2^h, and p^h at each mesh point. The mesh points are grouped subdomain by subdomain for the purpose of parallel processing. Regular uniform checkerboard partitions, 2×2, 4×4, and 8×8, are used for our experiments. The number of subdomains is always the same as the number of processors, n_p. Three fine meshes are considered: 64×64, 128×128, and 256×256. The total number of unknowns ranges from 12 K to 190 K. The coarse mesh size is varied from 16×16 to 80×80. Since non-nested coarse mesh is used, n_p and the coarse mesh size are not related. The overlapping size for the fine mesh is defined as $ovlp = \max\{(L_x' - L_x)/2h^K, (L_y' - L_y)/2h^K\}$ for both interior subdomains and those touching the boundary. Since square elements are used for the test problem, the elemental diameter h^Ks are the same and equal to the fine mesh size. L_x' and L_y' are defined here as the side lengths of the overlapping subdomain $\Omega_i^{h,\delta}$ in the x-direction and the y-direction, respectively. Similarly, L_x and L_y are defined as the side lengths of the non-overlapping subdomain Ω_i^h in the x-direction and the y-direction, respectively. At the fine mesh level, we use INB with the zero vectors as the initial guesses for the global nonlinear system (2.5) and the local nonlinear systems (2.15). The original Jacobian J as well as the local Jacobian matrices are constructed approximately by a multicolored forward finite difference method. During local nonlinear iterations, the LU decomposition with the forward and backward substitutions is employed for solving each local Jacobian system. The global nonlinear iteration is stopped if the condition

$$\|\mathscr{F}^{(2)}(x^{(k)})\|_2 \leqslant 10^{-6}\|\mathscr{F}^{(2)}(x^{(0)})\|_2$$

is satisfied, and the success of the two-level ASPIN is declared if the above condition is satisfied. The local nonlinear iteration on each subdomain is stopped if the condition

$$\|G_i(w_{i,l}^{(k)})\|_2 \leqslant 10^{-4}\|G_i(w_{i,0}^{(k)})\|_2$$

is satisfied. We use GMRES with no preconditioning for solving the global Jacobian system (2.16). The global linear iteration is stopped if the condition

$$\|\mathscr{F}^{(2)}(x^{(k)}) + \widehat{\mathscr{J}}^{(2)}(x^{(k)})s^{(k)}\|_2 \leqslant 10^{-4}\|\mathscr{F}^{(2)}(x^{(k)})\|_2$$

is satisfied. At the coarse mesh level, the redundant LU approach is used for solving the coarse linear systems. In this approach, each processor performs the LU decomposition of the same coarse matrix in parallel, then the forward and backward substitutions are done sequentially at each stage of the coarse solve. All numerical results reported here are based on the optimal choices of $s_{\max}$ that result in the fastest overall convergence in terms of the computing time for each case.

3.2. The effect of the coarse mesh size

In Table 1, we study the effect of the coarse mesh size on the global nonlinear and linear iterations and the computing time of the two-level ASPIN for $Re = 10^4$. In this set of numerical experiments, we keep the subdomain mesh size fixed and scale up the total fine mesh size and the number of processors. The coarse mesh size is varied from 16×16 to 80×80 and the maximum ratio of fine-to-coarse mesh size is limited to 1/2. Cases in which the coarse mesh sizes are equal to or larger than a half of the fine mesh size are not tested (marked as "–" in Table 1). The label "**" in the table indicates that the 8×8 uniform partitioning of the 20×20 coarse mesh is not available. From the cases in Table 1, we observe that by increasing the coarse mesh size, we reduce not only the average number of global linear iterations significantly as we increase the number of processors from 4 to 64, but also the number of global nonlinear iterations, especially in the 64-processor case. Roughly $H \sim 8/3h$ is needed to achieve the fastest convergence in terms of the computing time. Also, although not shown in the table, we note that the optimal coarse mesh size depends somewhat on the Reynolds number. The coarse mesh size for low-Reynolds number flows does not need to be as fine as it does for the high Reynolds number flows. For example, only $H \sim 4h$ is required for optimal performance in the cases of $Re = 10^3$ and 5×10^4.

3.3. The effect of the overlapping size

Table 2 shows the effect of the overlapping size on the two-level ASPIN. We vary the overlapping size from 2 to 5. From the table, we find that similar to elliptical-dominated PDEs, the average number of global linear iterations decreases monotonically as the overlapping size increases.

Table 1
Varying the coarse mesh size for $Re = 10^4$. Fixed subdomain mesh size: 32×32. $ovlp = 2$. The coarse problem is solved by the redundant LU approach. NGNI: the number of global nonlinear iterations. ANGLI: the average number of global linear iterations. $s^0_{\max}$: the optimal size of $s_{\max}$

Fine (n_p)	Coarse	16×16	20×20	32×32	40×40	64×64	80×80
64×64 (4)	NGNI	14	11	11	–	–	–
	ANGLI	52.6	42.3	28.3	–	–	–
	Time (s)	182.3	150.1	*136.0*	–	–	–
	$s^0_{\max}$	1.0	1.5	1.5	–	–	–
128×128 (16)	NGNI	19	16	13	10	11	–
	ANGLI	102.1	80.1	51.5	47.2	42.3	–
	Time (s)	444.5	246.3	181.8	*150.6*	162.2	–
	$s^0_{\max}$	2.0	3.0	3.0	5.0	4.0	–
256×256 (64)	NGNI	22	**	17	17	14	10
	ANGLI	238.0	**	81.5	68.0	61.4	66.1
	Time (s)	408.5	**	250.5	250.4	236.2	*215.8*
	$s^0_{\max}$	4.0	**	5.0	5.0	7.0	18.0

Table 2
Varying the overlapping size, $ovlp$, for $Re = 10^4$. Fixed subdomain mesh size: 32×32. The coarse problem is solved by the redundant LU approach. NGNI: the number of global nonlinear iterations. ANGLI: the average number of global linear iterations

Fine/coarse (n_p)	$ovlp$	2	3	4	5
$64 \times 64/32 \times 32$ (4)	NGNI	11	11	10	10
	ANGLI	28.3	26.0	25.7	24.8
	Time (s)	136.0	*130.6*	134.4	147.3
$128 \times 128/40 \times 40$ (16)	NGNI	10	11	11	11
	ANGLI	47.2	42.5	40.9	39.3
	Time (s)	*150.6*	169.8	180.5	195.2
$256 \times 256/80 \times 80$ (64)	NGNI	10	10	11	13
	ANGLI	66.1	65.1	57.0	50.8
	Time (s)	*215.8*	242.4	282.7	336.6

However, except in the case of the smallest problem with four processors, the total computing time of the two-level ASPIN increases due mainly to the increase in the size of nonlinear subdomain problems. Therefore, in general, our recommendation is to use small overlap ($ovlp = 2$) for the two-level ASPIN.

3.4. Parallel performance study

To evaluate the parallel performance, we use the so-called fixed-subdomain-size-per-processor scalability. Using this metric, we study how the algorithm behaves as the number of unknowns and processors increase simultaneously while the subdomain mesh size is kept a constant. The scalability study of the two-level ASPIN for different values of Reynolds number is summarized in Table 3. For the purpose of comparison, we also include the results obtained using the one-level ASPIN. The scaled efficiency η shown in Table 3 is defined by $\eta = T_4/T_{n_p}$, where T_4 and T_{n_p} are the execution times obtained by using 4 and n_p processors, respectively. In an ideal case, $\eta \sim 1$.

From Table 3, we see that although the one-level ASPIN is robust, it is not scalable either nonlinearly or linearly.

Table 3
Fixed-subdomain-size-per-processor scalability. $ovlp = 2$ for all cases

Re	Fine/coarse (n_p)	NGNI	ANGLI	Time (s)	η (%)
One-level ASPIN					
1×10^3	$64 \times 64/$ – (4)	7	41.9	72.2	100.0
	$128 \times 128/$ – (16)	8	114.2	125.0	57.8
	$256 \times 256/$ – (64)	10	815.7	256.4	28.2
5×10^3	$64 \times 64/$ – (4)	10	46.3	131.6	100.0
	$128 \times 128/$ – (16)	14	118.9	204.1	64.5
	$256 \times 256/$ – (64)	19	897.0	535.7	24.6
10^4	$64 \times 64/$ – (4)	12	49.1	201.1	100.0
	$128 \times 128/$ – (16)	18	129.7	312.9	64.3
	$256 \times 256/$ – (64)	19	872.9	558.2	36.0
Two-level ASPIN					
1×10^3	$64 \times 64/20 \times 20$ (4)	7	23.7	75.0	100.0
	$128 \times 128/32 \times 32$ (16)	9	31.4	84.4	89.0
	$256 \times 256/64 \times 64$ (64)	8	40.4	98.4	76.2
5×10^3	$64 \times 64/32 \times 32$ (4)	9	26.4	104.4	100.0
	$128 \times 128/40 \times 40$ (16)	9	40.7	102.4	90.3
	$256 \times 256/64 \times 64$ (64)	10	59.4	172.6	60.5
10^4	$64 \times 64/32 \times 32$ (4)	11	28.3	136.0	100.0
	$128 \times 128/40 \times 40$ (16)	10	47.2	150.6	90.3
	$256 \times 256/80 \times 80$ (64)	10	66.1	215.8	63.0

The average number of global linear iterations grows significantly as n_p is increased and in the 64 processor case, only 36% scaled efficiency is achieved. On the other hand, the two-level ASPIN with a minimum overlap and a sufficiently fine coarse mesh is more scalable with respect to the Reynolds number, the mesh size, and the number of processors. Linear and nonlinear iterations for the two-level ASPIN are not especially sensitive to the increase of those factors. Furthermore, the two-level ASPIN is always two to three times faster than the one-level ASPIN for the case of 64 processors and the scaled efficiency of the two-level ASPIN maintains at least 60% for all cases. To understand the degradation of the scaled efficiency in the two-level ASPIN, we measure some key components in the algorithm. This includes the two-level nonlinearly preconditioned function evaluation, both for the local subdomain part in (2.15) and the global coarse part in (2.14), the

Table 4
Timing: breakdown of each component in the two-level ASPIN. $Re = 10^4$. The numbers are in the following order, partial time (s) spent on each component, average time per call (s), and partial time/total time (%)

Fine/coarse (n_p)	Function evaluation		Form J	Solve J^c
	Coarse	Subdomains		
$64 \times 64/32 \times 32$ (4)	0.1, 0.008, 0.1	106.6, 8.8, 78.4	22.8, 2.1, 16.8	2.7, 0.008, 2.0
$128 \times 128/40 \times 40$ (16)	0.2, 0.01, 0.1	115.5, 9.6, 76.8	21.7, 2.2, 14.4	7.0, 0.01, 4.7
$256 \times 256/80 \times 80$ (64)	1.0, 0.06, 0.5	131.9, 9.4, 61.1	24.6, 2.4, 11.4	48.4, 0.06, 22.4

construction of the Jacobian for the original function, which is needed in Step (1) of Remark 3, and the coarse solution required as part of preconditioned GMRES iterations in Step (4) of Remark 3. Table 4 presents the breakdown of the timing results for the case of $Re = 10^4$. As expected, the most time-consuming component in the two-level ASPIN is the local subdomain part of the global function evaluation, taking over 60% of the total computing time for each case, since each processor needs to solve several nonlinear subdomain systems. Exact LU decomposition for solving the local Jacobian system and the construction of the local Jacobian matrices using multicolored finite differences both take a lot of time. However, these two operations are purely local and do not involve any communications so that time spent in this stage remains nearly constant as the number of processor is increased. On the other hand, the linear coarse part of the global function evaluation (the first column of Table 4) is very efficient, taking less than 0.5% of the total computing time. From the same table, we also find that the only non-scalable component in the two-level ASPIN is the coarse solution required in part of the preconditioned GMRES iterations. It is expected that the computing time spent on this component will be dominant when a large-scale problem is solved using a large number of processors. Replacing a redundant direct coarse solver by a parallel direct solver, such as SuperLU_Dist [7] may be a solution. However, the performance of a parallel direct solver depends heavily on the ordering and pivoting strategies. How these factors affect the overall performance of the two-level ASPIN needs to be investigated further.

4. Concluding remarks

We presented a new two-level ASPIN algorithm and its application to incompressible Navier–Stokes equations. The two-level nonlinear preconditioner is constructed by using a local nonlinear overlapping Schwarz domain decomposition method and a global linear coarse solver. We obtained some encouraging numerical results for a moderate number of processors. We show that the new two-level ASPIN maintains fast convergence and robustness properties of the one-level ASPIN. In addition, if the coarse mesh size is fine enough, the new algorithm provides better nonlinear and linear scalability with respect to the number of processors. To show the applicability of ASPIN for larger problems, more applications with complex geometry need to be tested using larger numbers of processors.

References

[1] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, Portable, Extensible Toolkit for Scientific Computation (PETSc), 2005, homepage: <www.mcs.anl.gov/petsc>.
[2] A.N. Brooks, T.J.R. Hughes, Streamline upwind/Petrov–Galerkin formulations for convective dominated flows with particular emphasis in the incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 32 (1982) 199–259.
[3] W.L. Briggs, V.E. Henson, S.F. McCormick, A Multigrid Tutorial, SIAM, Philadelphia, 2000.
[4] X.-C. Cai, D.E. Keyes, Nonlinearly preconditioned inexact Newton algorithms, SIAM J. Sci. Comput. 24 (2002) 183–200.
[5] X.-C. Cai, D.E. Keyes, L. Marcinkowski, Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics, Int. J. Numer. Methods Fluids 40 (2002) 1463–1470.
[6] X.-C. Cai, L. Marcinkowski, P. Vassilevski, An element agglomeration nonlinear additive Schwarz preconditioned Newton method for unstructured finite element problems, Appl. Math. 50 (2005) 247–275.
[7] J.W. Demmel, J.R. Gilbert, X.S. Li, SuperLU Users' Guide, Lawrence Berkeley National Laboratory, 2003.
[8] J. Dennis, R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations, SIAM, Philadelphia, 1996.
[9] L.P. Franca, S.L. Frey, Stabilized finite element method: II. The incompressible Navier–Stokes equations, Comput. Methods Appl. Mech. Engrg. 99 (1992) 209–233.
[10] U. Ghia, K.N. Ghia, C.T. Shin, High-*Re* solution for incompressible flow using the Navier–Stokes equations and the multigrid method, J. Comput. Phys. 48 (1982) 387–411.
[11] M.D. Gunzburger, Finite Element Methods for Viscous Incompressible Flows, Academic Press, New York, 1989.
[12] F.-N. Hwang, X.-C. Cai, Improving robustness and parallel scalability of Newton method through nonlinear preconditioning, in: R. Kornhuber, R.H.W. Hoppe, D.E. Keyes, J. Periaux, O. Pironneau, J. Xu (Eds.), Lecture Notes in Computational Science and Engineering, Springer-Verlag, Heidelberg, 2004, pp. 201–208.
[13] F.-N. Hwang, X.-C. Cai, A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier–Stokes equations, J. Comput. Phys. 204 (2005) 666–691.
[14] F.-N. Hwang, X.-C. Cai, A combined linear and nonlinear preconditioning technique for incompressible Navier–Stokes equations, in: J. Dongarra, K. Madsen, J. Wasniewski (Eds.), Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, 2006, pp. 313–322.
[15] A. Klawonn, L.F. Pavarino, Overlapping Schwarz method for mixed linear elasticity and Stokes problems, Comput. Methods Appl. Mech. Engrg. 165 (1998) 233–245.
[16] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: A survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.

[17] L. Marcinkowski, X.-C. Cai, Parallel performance of some two-level ASPIN algorithms, in: R. Kornhuber, R.H.W. Hoppe, D.E. Keyes, J. Periaux, O. Pironneau, J. Xu (Eds.), Lecture Notes in Computational Science and Engineering, Springer-Verlag, Heidelberg, 2004, pp. 639–646.
[18] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.
[19] J.N. Reddy, D.K. Gartling, The Finite Element Method in Heat Transfer and Fluid Dynamics, CRC Press, Florida, 2000.
[20] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.
[21] J.N. Shadid, R.S. Tuminaro, H.F. Walker, An inexact Newton method for fully coupled solution of the Navier–Stokes equations with heat and mass transport, J. Comput. Phys. 137 (1997) 155–185.
[22] B. Smith, P. Bjørstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.
[23] A. Toselli, O. Widlund, Domain Decomposition Methods – Algorithms and Theory, Springer, Berlin, 2005.
[24] R.S. Tuminaro, H.F. Walker, J.N. Shadid, On backtracking failure in Newton-GMRES methods with a demonstration for the Navier–Stokes equations, J. Comput. Phys. 180 (2002) 549–558.