Available online at www.sciencedirect.com
SCIENCE DIRECT®

Journal of Computational Physics xxx (2004) xxx–xxx

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier–Stokes equations ☆

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

[a] *Department of Applied Mathematics, University of Colorado, Boulder, CO 80309, USA*
[b] *Department of Computer Science, University of Colorado, Campus Box 430, Boulder, CO 80309-0430, USA*

Received 20 August 2003; received in revised form 1 October 2004; accepted 19 October 2004

Abstract

A nonlinear additive Schwarz preconditioned inexact Newton method (ASPIN) was introduced recently for solving large sparse highly nonlinear systems of equations obtained from the discretization of nonlinear partial differential equations. In this paper, we discuss some extensions of ASPIN for solving steady-state incompressible Navier–Stokes equations with high Reynolds numbers in the velocity–pressure formulation. The key idea of ASPIN is to find the solution of the original system by solving a nonlinearly preconditioned system that has the same solution as the original system, but with more balanced nonlinearities. Our parallel nonlinear preconditioner is constructed using a nonlinear overlapping additive Schwarz method. To show the robustness and scalability of the algorithm, we present some numerical results obtained on a parallel computer for two benchmark problems: a driven cavity flow problem and a backward-facing step problem with high Reynolds numbers. The sparse nonlinear system is obtained by applying a $Q_1 - Q_1$ Galerkin least squares finite element discretization on two-dimensional unstructured meshes. We compare our approach with an inexact Newton method using different choices of forcing terms. Our numerical results show that ASPIN has good convergence and is more robust than the traditional inexact Newton method with respect to certain parameters such as the Reynolds number, the mesh size, and the number of processors.
© 2004 Elsevier Inc. All rights reserved.

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

☆ The research was supported in part by the Department of Energy, DE-FC02-01ER25479, and in part by the National Science Foundation, CCR-0219190, ACI-0072089 and CCF-0305666.
* Corresponding author. Fax: +1 303 492 2844.
E-mail address: cai@cs.colorado.edu (X.-C. Cai).

0021-9991/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2004.10.025

1. Introduction

The focus of this paper is to develop a fast, scalable and robust parallel iterative algorithm and software for solving large, sparse, nonlinear systems of equations arising from the finite element discretization of steady-state incompressible Navier–Stokes equations in the velocity–pressure formulation. Solving Navier–Stokes equations numerically is important, since many applications in computational science and engineering depend on it. Even though years of research have been spent on finding fast and reliable methods for solving Navier–Stokes equations on very fine meshes for a wide range of Reynolds numbers, it remains a difficult computing task due to certain characteristics of the equations that are yet to be fully understood mathematically, such as the boundary layer at high Reynolds numbers. To resolve the details of the solution, high resolution meshes are often required, which implies large condition numbers of the resulting algebraic systems, and also implies the need for a large-scale parallel computer. Several classes of method exist for incompressible Navier–Stokes equations, including projection type methods and multigrid type methods [14,20,31]. In this paper, we focus on the class of Newton methods because of their ease of implementation and applicability to general flows and geometry. Our algorithm does not require the splitting of any variable or operator, and this makes it suitable for many other coupled nonlinear systems of PDEs.

The inexact Newton method (IN) is a popular technique for solving nonlinear systems, since it is easy to implement, general-purpose and has a rapid convergence rate when the initial guess is near the desired solution [11,12,15]. One drawback of IN is that a good enough initial guess is often not easily obtainable, especially for high Reynolds number flows. It is well-known from numerical experience that the radius of convergence for IN becomes smaller as the Reynolds number increases. As a result, IN often diverges, even at moderate Reynolds numbers and with globalization techniques, such as a linesearch technique or a trust region method [11,36]. To overcome this difficulty, several techniques have been proposed for finding a good initial guess. For example, continuation methods are popular choices for solving high Reynolds number flow problems; see [22,24,25] and references therein. In general, continuation methods fall into three categories: parameter continuation [1,23], mesh sequencing [29], and pseudo time stepping [8,26,27]. The advantage of continuation is that the implementation is often relatively easy and robust. On the other hand, we do not always have the criteria for determining the size of the incremental parameter, or the optimal size of a coarse mesh, or the optimal choice of the time increment for each iteration. Alternately, to obtain the convergence of IN for high Reynolds number flow problems, Shadid et al. [34] introduced an algorithm based on an inexact Newton method with backtracking (INB). For theoretical discussions of INB, see [15,16]. The key component of INB is the forcing term, which provides a criterion for determining the accuracy of the Jacobian solver during Newton iterations. Instead of using a constant forcing term throughout the computation, INB becomes more robust and efficient if the forcing term is chosen adaptively based on the nonlinear residual at the previous Newton step. The right combinations of the forcing term and certain discretization parameters may lead to nice convergence even for high Reynolds numbers. However, numerical experiments conducted by us and others [34] have showed that the selection of the forcing terms is quite problem-dependent. In other words, the nonlinear solver may diverge because of a slight change of a problem parameter.

Recently, without employing any of the techniques discussed above, a more robust parallel algorithm, namely a nonlinear additive Schwarz preconditioned inexact Newton method (ASPIN), was introduced [5–7,27] for solving large sparse nonlinear systems of equations arising from the discretization of nonlinear partial differential equations. ASPIN has been shown numerically to be more robust than INB for solving some challenging flow problems such as incompressible Navier–Stokes equations in the velocity–vorticity formulation [5,6] and a full potential flow problem [7]. The key idea of ASPIN is that we find the solution of the original system $F(x) = 0$ by solving a nonlinearly preconditioned system $\mathscr{F}(x) = 0$ that has the same solution, but with more balanced nonlinearities. In this paper, we propose a new version of ASPIN for solving incompressible Navier–Stokes equations in the primitive variable form. The sparse nonlinear system is obtained by using a Galerkin least squares finite element (GLS) discretization [17,18] on two dimensional unstructured meshes. The GLS formulation is derived from the standard Galerkin formulation by adding the square of the residual of the momentum equations in each element. The additional term improves not only the numerical stability of the standard Galerkin method for high Reynolds number flows, but also preserves the accuracy. A re-scaling step is added to the ASPIN algorithm in [5] to make the algorithm more suitable for the primitive variable Navier–Stokes equations, which can be used to model many different flows.

This paper is organized as follows. In the next section, we describe the incompressible Navier–Stokes equations and their discretization using the GLS formulation. Then, Section 3 briefly reviews an inexact Newton method with backtracking for solving nonlinear systems of equations. Section 4 describes the details of ASPIN for the incompressible Navier–Stokes equations. Section 5 presents numerical results obtained on a parallel computer for two benchmark problems: a lid-driven cavity flow problem [20] and a backward-facing step problem [19,21]. We compare our approach with the well-understood, Newton–Krylov–Schwarz (NKS) algorithm [4,27,34]. Finally, conclusions and some remarks are given in Section 6.

2. Incompressible Navier–Stokes equations and Galerkin least squares finite element discretization

Consider two-dimensional steady-state incompressible Navier–Stokes equations in the primitive variable form [22,32]

$$\begin{aligned}
&\boldsymbol{u}\cdot\nabla\boldsymbol{u} - 2\nu\nabla\cdot\epsilon(\boldsymbol{u}) + \nabla p = \boldsymbol{f} \quad \text{in } \Omega,\\
&\nabla\cdot\boldsymbol{u} = 0 \quad \text{in } \Omega,\\
&\boldsymbol{u} = \boldsymbol{g} \quad \text{on } \Gamma_{\mathrm{D}},\\
&\sigma\boldsymbol{n} = \boldsymbol{h} \quad \text{on } \Gamma_{\mathrm{N}},
\end{aligned} \tag{2.1}$$

where $\boldsymbol{u} = (u_1, u_2)^{\mathrm{T}}$ is the velocity, p is the pressure, ν is the dynamic viscosity, and $\epsilon(\boldsymbol{u}) = \frac{1}{2}[(\nabla\boldsymbol{u}) + (\nabla\boldsymbol{u})^{\mathrm{T}}]$ is the symmetric part of the velocity gradient. The Cauchy stress tensor σ is defined as $\sigma = -p\boldsymbol{I} + 2\nu\epsilon(\boldsymbol{u})$, where $\boldsymbol{I}$ is a second-order identity tensor. For simplicity, in this paper the body force $\boldsymbol{f} = 0$. Here we assume that Ω is a bounded domain in R^2 with a polygonal boundary Γ. Two types of boundary conditions are imposed: Dirichlet conditions (Γ_{D}) and Neumann conditions (Γ_{N}). Note that if only the Dirichlet boundary is specified, i.e., $\Gamma_{\mathrm{N}} = \phi$, the pressure p is determined up to a constant. To make p unique, we impose an additional condition

$$\int_{\Omega} p\,\mathrm{d}x = 0.$$

To discretize (2.1), we use a stabilized $Q_1 - Q_1$ finite element method ([22]) on a given conforming quadrilateral mesh $\mathscr{T}_h = \{K\}$. For each element K we use h_K to denote its diameter. Let V^h and P^h be a pair of finite element spaces for the velocity and pressure, given by

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

The weighting and trial velocity function spaces V_0^h and V_g^h are defined as follows:

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

The 2-norm of $\boldsymbol{v}$ is defined as $\|\boldsymbol{v}\|_2 = (\sum_{i=1}^{2}|v_i|^2)^{1/2}$.

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

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

with

$$B(\boldsymbol{u}, p; \boldsymbol{v}, q) = ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u}, \boldsymbol{v}) + (2\nu\epsilon(\boldsymbol{u}), \epsilon(\boldsymbol{v})) - (\nabla \cdot \boldsymbol{v}, p) - (\nabla \cdot \boldsymbol{u}, q) + \sum_{K \in \mathcal{T}_h} ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u} + \nabla p - 2\nu\nabla \cdot \epsilon(\boldsymbol{u}), \tau((\nabla \boldsymbol{v}) \cdot \boldsymbol{v} - \nabla q - 2\nu\nabla \cdot \epsilon(\boldsymbol{v})))_K + (\nabla \cdot \boldsymbol{u}, \delta \nabla \cdot \boldsymbol{v})$$

and

$$F(\boldsymbol{v}, q) = (\boldsymbol{h}, \boldsymbol{v})_{\Gamma_\mathrm{N}}.$$

We use the stabilization parameters δ and τ suggested in [17]. Let $Re_K(\boldsymbol{x}) = \|\boldsymbol{u}(\boldsymbol{x})\|_2 h_K/(12\,\nu)$ be an element Reynolds number, which distinguishes the locally convection-dominated flow ($Re_K(\boldsymbol{x}) \geqslant 1$) from the locally diffusion-dominated flow ($0 \leqslant Re_K(\boldsymbol{x}) < 1$). For convection-dominated elements, we use

$$\delta = \lambda \|\boldsymbol{u}(\boldsymbol{x})\|_2 h_K \quad \text{and} \quad \tau = \frac{h_K}{2\|\boldsymbol{u}(\boldsymbol{x})\|_2}$$

and for diffusion-dominated elements, we use

$$\delta = \frac{\lambda \|\boldsymbol{u}(\boldsymbol{x})\|_2^2 h_K^2}{12\nu} \quad \text{and} \quad \tau = \frac{h_K^2}{6\nu}.$$

Therefore, for convection-dominated regions, τ and δ are O(h); for diffusion-dominated regions, τ and δ are O(h^2). Note that for a fixed mesh, δ is a function of the velocity, and τ is a function of the velocity for convection-dominated regions, and is a constant for diffusion-dominated regions.

Remark 1. Franca and Frey [17] proved that the convergence of the GLS formulation holds for any combination of interpolation functions for velocity and pressure. In the implementation, it is more convenient to use equal-degree polynomials, such as $Q_1 - Q_1$ element, which are usually ruled out in the standard Galerkin formulation because of the violation of the LBB condition.

Remark 2. For simplicity, we consider only rectangular bilinear $Q_1 - Q_1$ elements in this paper. The viscous terms in the GLS formulation $2\nu\nabla \cdot \epsilon(\boldsymbol{u}^h)$ and $2\boldsymbol{v}\nabla \cdot \epsilon(\boldsymbol{v}^h)$ vanish. Therefore, the stabilized term on the left-hand side of the formulation (2.2) is reduced to

$$\sum_K ((\nabla \boldsymbol{u}) \cdot \boldsymbol{u} + \nabla p), \tau((\nabla \boldsymbol{v}) \cdot \boldsymbol{v} - \nabla q)))_K.$$

In this case, the GLS formulation and the Streamline-Upwind/Petrov–Galerkin (SUPG) formulation [3] coincide.

Let x be a vector corresponding to the nodal values of $\boldsymbol{u}^h$ and p^h in (2.2), then the weighting functions $(\boldsymbol{u}^h, p^h)$ and test functions $(\boldsymbol{v}, p)$ can be expressed in terms of finite element basis functions and the nodal values. Substituting these four functions into the finite element weak form (2.2), we can write (2.2) as a nonlinear algebraic system

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

which is often large, sparse and highly nonlinear when the Reynolds number is high. In our implementation, after ordering the mesh points, we number 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. More discussion on the subdomain partitioning will be given later. The components of the function $F(x)$ are ordered in the same fashion. In the rest of the paper, we focus on finding a solution of (2.3), starting from an initial guess $x^{(0)}$.

3. Review of inexact Newton with backtracking

In this section, we review an inexact Newton method with backtracking technique (INB) for solving general nonlinear systems of equations. INB will also serve as the basis of our new preconditioned inexact Newton method.

Let $x^{(0)}$ be a given initial guess, and $x^{(k)}$ the current approximate solution. Then a new approximate solution $x^{(k+1)}$ of (2.3) can be computed by the following steps:

Algorithm 1. (Inexact Newton with Backtracking)

```
Step 1: Find an inexact Newton direction $s^{(k)}$ such that
    $\|F(x^{(k)}) - F'(x^{(k)})s^{(k)}\|_2 \leqslant \eta_k \|F(x^{(k)})\|_2.$
Step 2: Compute a new approximate solution $x^{(k+1)}$ with backtracking
    $x^{(k+1)} = x^{(k)} - \lambda^{(k)} s^{(k)}.$
```

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 methods, such as a Krylov subspace type method, GMRES [33]. If the chosen forcing terms are zero, the algorithm reduces to the exact Newton algorithm. On the other hand, we can determine the forcing terms based on the information obtained from the previous Newton iteration. Two common choices of forcing terms were suggested by Eisenstat and Walker [16]:

- Choice 1. Select any $\eta_0 \in [0,1)$ and for $k = 1,2,\ldots$, choose
$$\eta_k = \frac{\left| \|F(x^{(k)})\|_2 - \|F(x^{(k-1)}) - F'(x^{(k-1)})s^{(k-1)}\|_2 \right|}{\|F(x^{(k-1)})\|_2}. \tag{3.1}$$
- Choice 2. Given $\gamma \in [0,1]$ and $\rho \in (1,2]$, select any $\eta_0 \in [0,1)$ and for $k = 1,2,\ldots$, choose
$$\eta_k = \gamma \left(\frac{\|F(x^{(k)})\|_2}{\|F(x^{(k-1)})\|_2} \right)^{\rho}. \tag{3.2}$$

To avoid the initial η_k becoming too small while the intermediate solution is still far away from a solution, two corresponding safeguards are also needed: For Choice 1 we replace η_k by $\max\{\eta_k, \eta_{k-1}^{\frac{1+\sqrt{5}}{2}}\}$, if $\eta_{k-1}^{\frac{1+\sqrt{5}}{2}} > 0.1$, and for Choice 2 we replace η_k by $\max\{\eta_k, \gamma\eta_{k-1}^{\rho}\}$, if $\eta_{k-1}^{\rho} > 0.1$.

The step length, $\lambda^{(k)} \in [\lambda_{\min}, \lambda_{\max}] \subset (0,1)$, in Step 2 of Algorithm 1 is selected so that
$$f(x^{(k)} - \lambda^{(k)} s^{(k)}) \leqslant f(x^{(k)}) - \alpha\lambda^{(k)} \nabla f(x^{(k)})^{\mathrm{T}} s^{(k)}, \tag{3.3}$$
where the two parameters $\lambda_{\min}$ and $\lambda_{\max}$ are known as safeguards which are required for strong global convergence, the merit function f: $R^n \to R$ is defined as $\|F(x)\|_2^2/2$, and the parameter α is used to assure that the reduction of f is sufficient. Here, a linesearch technique [11] is employed to determine the step length $\lambda^{(k)}$.

The Jacobian matrix F' is a key component in INB. In our implementation, we form F' as a sparse matrix using a multi-colored forward finite difference method [9]. Some other techniques are also available for approximating the Jacobian matrix, such as automatic differentiation and symbolic differentiation. We refer to Chapter 7 of [30] for an overview of these methods. Alternatively, a matrix-free implementation of Newton–Krylov can be used to avoid the explicit computation of F'. Interested readers may refer to [27] for a complete survey of matrix-free Newton–Krylov methods as well as their applications in computational physics.

4. ASPIN algorithm for incompressible Navier–Stokes equations

When using INB directly in (2.3), very good results have been observed for the many cases. However, when the Reynolds number is high, INB sometimes converges nicely, but sometimes, it does not converge for some reasons that are not completely understood yet. In the rest of this section, we re-formulate the nonlinear system (2.3) in the form of preconditioning, and then apply INB to the nonlinearly preconditioned system.

4.1. Nonlinear additive Schwarz preconditioning

We introduce the nonlinear preconditioner by defining four key components: the velocity and pressure spaces associated with subdomains; the restriction and interpolation operators; the subdomain nonlinear functions; and the subdomain correction operators. There are different ways to define a nonlinear preconditioner, but here we consider only the additive Schwarz framework. We will show numerically in Section 5 that nonlinear additive Schwarz [10] is an excellent nonlinear preconditioner in the sense that it enhances the robustness of INB for a wide range of Reynolds numbers and reduces the number of both linear and nonlinear iterations.

To apply an overlapping domain decomposition method [35,37], we first partition the flow domain Ω into non-overlapping subdomains Ω_i, $i = 1,\ldots,N$. Then, we expand each subdomain Ω_i to obtain a larger overlapping subdomain Ω'_i with the boundary $\partial\Omega'_i$. We assume that $\Omega_i \subset \Omega'_i$ and $\partial\Omega'_i$ does not cut any elements of $\mathscr{T}_h$.

Now, we define the subdomain velocity space as

$$V^h_i = \{\boldsymbol{v}^h \in V^h \cap (H^1(\Omega'_i))^2 : \boldsymbol{v}^h = 0 \text{ on } \partial\Omega'_i \setminus \Gamma_{\mathrm{N}}\}$$

and the subdomain pressure space as

$$P^h_i = \{p^h \in P^h \cap L^2(\Omega'_i) : p^h = 0 \text{ on } \partial\Omega'_i \setminus \Gamma\}.$$

Both are subspaces of V^h and P^h, respectively, if we extend all subdomain functions to the whole domain by zero. See Fig. 1 for an example of a subdomain mesh. Note that for $Q_1 - Q_1$ elements, we have three degrees of freedom per interior node, two for the velocity and one for the pressure.

Let n be the total number of degrees of freedom associated with the space $V^h_g \times P^h_0$ and n_i be the total number of degrees of freedom associated with the subspace $V^h_i \times P^h_i$. Because of the overlap, we usually have $\sum_{i=1}^{N} n_i \geqslant n$.

Let $R_{\Omega'_i}$: $V^h_g \times P^h_0 \to V^h_i \times P^h_i$ be a restriction operator, which returns all degrees of freedom (both velocity and pressure) associated with the subspace $V^h_i \times P^h_i$. $R_{\Omega'_i}$ is an $n_i \times n$ matrix with values either 0 or 1. The multiplication of $R_{\Omega'_i}$ with a vector does not involve any arithmetic operations, but does involve communication in a distributed-memory parallel implementation. Then, the interpolation operator $R^{\mathrm{T}}_{\Omega'_i}$ can be defined as the transpose of $R_{\Omega'_i}$.

Using the restriction operator, we define the subdomain nonlinear function $F_{\Omega'_i} : R^n \to R^{n_i}$ as

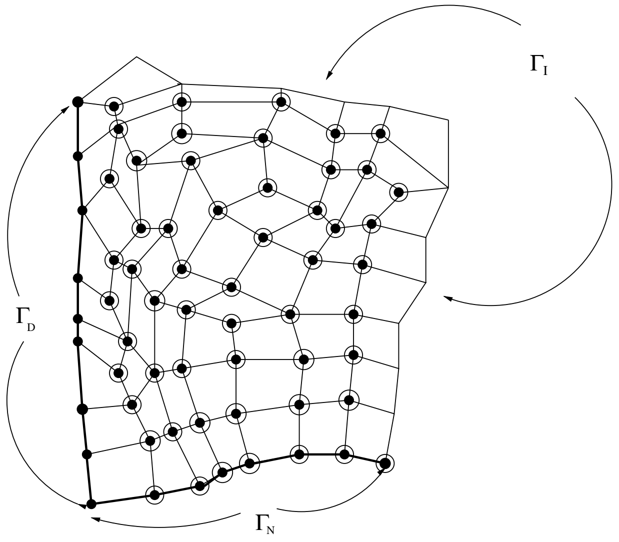


Fig. 1. Subdomain velocity and pressure space. (•) denote pressure degrees of freedom and (○) denote velocity degrees of freedom. Homogenous Dirichlet boundary conditions are imposed on the interior boundary Γ_I for both $\boldsymbol{u}$ and p. On Γ_D only $\boldsymbol{u}$ is given.

$$F_{\Omega'_i} = R_{\Omega'_i} F.$$

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

$$F_{\Omega'_i}(x - R^{\mathrm{T}}_{\Omega'_i} T_i(x)) = 0, \quad \text{for} = 1, \ldots, N. \tag{4.1}$$

We impose here Dirichlet or Neumann conditions according to the original Eqs. (2.1) on the physical boundaries. On artificial boundaries, we assume both $\boldsymbol{u} = 0$ and $p = 0$. Similar boundary conditions were used in [28] for the Stokes equations.

Throughout this paper, we assume that (4.1) is uniquely solvable. Using the subdomain mapping functions, we introduce a new global nonlinear function

$$\mathscr{F}(x) = \sum_{i=1}^{N} R^{\mathrm{T}}_{\Omega'_i} T_i(x), \tag{4.2}$$

where $\mathscr{F}(x)$ is called the nonlinearly preconditioned $F(x)$, and define a nonlinearly preconditioned system

$$\mathscr{F}(x) = 0. \tag{4.3}$$

For strong elliptic problems, it can be shown that the nonlinearly preconditioned system and the original system have the same solution [5]. But for the incompressible Navier–Stokes equations, we can not apply the theory of [5] directly. Alternatively, we will verify numerically, in Section 5.3.1, that the preconditioned nonlinear system (4.3) has the same solution as that of the original system (2.3).

4.2. Computing the Jacobian of the preconditioned system

Several techniques are available for the construction and solving of the Jacobian problems. Some methods are matrix-free, and some need the explicit construction of the matrix. In our implementation, we choose to construct the Jacobian of $\mathscr{F}(x)$ semi-explicitly. Since $\mathscr{F}(x)$ is defined implicitly in (4.1) and (4.2), the Jacobian calculation is not as straightforward as that of $F(x)$. Here, we describe a computable form of the Jacobian matrix for the preconditioned system as suggested in [5].

Consider subdomain Ω_i'. Let $(\Omega_i')^c$ be the complement of Ω_i' in the domain Ω. We write $x = (x_i, x_i^c)$, where $x_i = R_{\Omega_i'}x$ and $x_i^c = R_{(\Omega_i')^c}x$. Here $R_{(\Omega_i')^c}$ is a restriction matrix defined similarly as $R_{\Omega_i'}$. From the definition (4.1) of $T_i(x)$, we have

$$F_{\Omega_i'}(x_i - T_i(x_i, x_i^c), x_i^c) = 0. \tag{4.4}$$

Let $y_i = x_i - T_i$ and $z_i = (y_i, x_i^c)$. Then the global function can be rewritten as a function of z_i:

$$F(z_i) = F(y_i, x_i^c).$$

Furthermore, we write the Jacobian matrix of the original nonlinear function $F(x)$ in the form of

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

or, in terms of y_i and x_i^c as

$$J = \left(\frac{\partial F}{\partial y_i}\right) R_{\Omega_i'} + \left(\frac{\partial F}{\partial x_i^c}\right) R_{(\Omega_i')^c},$$

since subdomains Ω_i' and $(\Omega_i')^c$ do not overlap.

Similarly, let $J_{\Omega_i'}(z_i)$ be the Jacobian of the nonlinear function $F(\cdot)$ restricted to the subdomain Ω_i', i.e.,

$$J_{\Omega_i'}(z_i) = \left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)_{n_i\times n_i}.$$

Suppose that we want to evaluate the Jacobian of the preconditioned function, denoted as $\mathscr{J}$, at the kth Newton iteration, $x^{(k)} = \left(x_i^{(k)}, (x_i^c)^{(k)}\right)$. Using (4.2), the Jacobian $\mathscr{J}$ can be calculated as follows:

$$\mathscr{J} \equiv \mathscr{F}\prime = \sum_{i=1}^{N} R_{\Omega_i'}^{\mathrm{T}}\left(\frac{\partial T_i}{\partial x}\right).$$

We next derive an approximate formula for each $\partial T_i/\partial x$. Now, taking the partial derivative of Eq. (4.4) with respect to x_i, we find that

$$\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)\left(\frac{\partial y_i}{\partial x_i}\right) = 0,$$

or, more precisely,

$$\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)\left(I_{\Omega_i'} - \frac{\partial T_i}{\partial x_i}\right) = 0.$$

Assuming $\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)$ is nonsingular, we obtain that

$$\left(\frac{\partial T_i}{\partial x_i}\right) = I_{\Omega_i'} = \left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)^{-1}\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right). \tag{4.5}$$

Here, $I_{\Omega_i'}$ is the $n_i \times n_i$ identity matrix. Next, we take the partial derivative of (4.4) with respect to x_i^c to obtain

$$-\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)\left(\frac{\partial T_i}{\partial x_i^c}\right) + \left(\frac{\partial F_{\Omega_i'}}{\partial x_i^c}\right) = 0.$$

Solving the above equation for $\left(\frac{\partial T_i}{\partial x_i^c}\right)$ yields

$$\left(\frac{\partial T_i}{\partial x_i^c}\right) = \left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)^{-1}\left(\frac{\partial F_{\Omega_i'}}{\partial x_i^c}\right). \tag{4.6}$$

Note that

$$\left(\frac{\partial T_i}{\partial x}\right)_{n_i\times n} = \left(\frac{\partial T_i}{\partial x_i}\right)R_{\Omega_i'} + \left(\frac{\partial T_i}{\partial x_i^c}\right)R_{(\Omega_i')^c}. \tag{4.7}$$

By substituting (4.5) and (4.6) into (4.7), we have

$$\begin{aligned}\left(\frac{\partial T_i}{\partial x}\right) &= \left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)^{-1}\left[\left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)R_{\Omega_i'} + \left(\frac{\partial F_{\Omega_i'}}{\partial x_i^c}\right)R_{(\Omega_i')^c}\right] = \left(\frac{\partial F_{\Omega_i'}}{\partial y_i}\right)^{-1}R_{\Omega_i'}\left[\left(\frac{\partial F}{\partial y_i}\right)R_{\Omega_i'} + \left(\frac{\partial F}{\partial x_i^c}\right)R_{(\Omega_i')^c}\right]\\ &= (J_{\Omega_i'})^{-1}R_{\Omega_i'}J.\end{aligned} \tag{4.8}$$

Summing up (4.8) for all subdomains and evaluating x at $x^{(k)}$, we have a formula for the Jacobian of the nonlinearly preconditioned function

$$\mathscr{J} = \sum_{i=1}^{N}\left[R_{\Omega_i'}^{\mathrm{T}}(J_{\Omega_i'}(z_i))^{-1}R_{\Omega_i'}\right]J(z_i)\Big|_{z_i=(x_i^{(k)}-T_i(x^{(k)}),(x_i^c)^{(k)})}. \tag{4.9}$$

Although (4.9) is computable, in practice, it is more convenient to use the following approximation suggested in [5]:

$$\widehat{\mathscr{J}} = \sum_{i=1}^{N}\left[R_{\Omega_i'}^{\mathrm{T}}(J_{\Omega_i'}(z))^{-1}R_{\Omega_i'}\right]J(z)\Big|_{z=x^{(k)}}. \tag{4.10}$$

Remark 3. Note that the subdomain preconditioner $(J_{\Omega_i'})^{-1}$ corresponds to the solution of a discrete Stokes-like problem using GLS formulation with homogenous Dirichlet boundary conditions for both the velocity and pressure. In our implementation, $J_{\Omega_i'}$ is obtained from $J_{\Omega_i'} = R_{\Omega_i'}JR_{\Omega_i'}^{\mathrm{T}}$, where J is constructed using multi-colored forward finite difference methods.

Remark 4. Although each component of $\widehat{\mathscr{J}}$ is sparse, $\widehat{\mathscr{J}}$ itself is often dense and expensive to form explicitly. However, if a Krylov subspace method is used to solve the Jacobian problem, only the Jacobian-vector product, $u = \widehat{\mathscr{J}}v$, is required. In a distributed-memory parallel implementation, this operation consists of four phrases:

1. Perform the matrix-vector multiply, $w = Jv$, in parallel.
2. On each subdomain, collect the data from the subdomain and its neighboring subdomains, $w_i = R_{\Omega_i'}w$.
3. Solve $J_{\Omega_i'}u_i = w_i$ using a sparse direct solver.
4. Send and receive partial solutions to and from its neighboring subdomains and then compute the sum, $u = \sum_i^N R_{\Omega_i'}^{\mathrm{T}}u_i$.

4.3. Details of ASPIN

Summarizing the discussions in Sections 4.1 and 4.2, we here describe the complete ASPIN algorithm for solving incompressible Navier–Stokes equations. 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 by the ASPIN algorithm as follows:

Algorithm 2. (Additive Schwarz Preconditioned Inexact Newton)

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

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

$$G_{\Omega_i'}(w) \equiv F_{\Omega_i'}\left(x^{(k)} - R^{\mathrm{T}}_{\Omega_i'} w\right) = 0,$$

   using INB with the initial guess $w = 0$.

2. Form the global residual

$$\mathscr{F}(x^{(k)}) = \sum_{i=1}^{N} R^{\mathrm{T}}_{\Omega_i'} w_i^{(k)}.$$

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

Step 2: Evaluate pieces of the Jacobian matrix $\mathscr{J}(x)$ of the preconditioned system that are needed in order to multiply (4.11) below with a vector in the next step. This includes $J(x^{(k)})$ as well as $J_{\Omega_i'}$ and its sparse LU factorization.

$$\widehat{\mathscr{J}} = \sum_{i=1}^{N} \left[R^{\mathrm{T}}_{\Omega_i'} (J_{\Omega_i'}(x^{(k)}))^{-1} R_{\Omega_i'}\right] J(x^{(k)}). \tag{4.11}$$

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

$$\widehat{\mathscr{J}} s^{(k)} = \mathscr{F}(x^{(k)}),$$

in the sense that

$$\|\mathscr{F}(x^{(k)}) - \widehat{\mathscr{J}}(x^{(k)}) s^{(k)}\|_2 \leqslant \eta_k \|\mathscr{F}(x^{(k)})\|_2,$$

for some $\eta_k \in [0, \eta_{\max}]$ for some $\eta_{\max} < 1$ independent of k.

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 described in (3.3).

Remark 5. At the kth global nonlinear iteration, local subdomain nonlinear systems

$$G_{\Omega_i'}(w) = 0, \quad i = 1, \ldots, N$$

need to be solved. We solve these subsystems using Newton's method. During local nonlinear iterations, a direct sparse solver, LU decomposition, is employed for solving each local Jacobian system.

Remark 6. No preconditioning is used in Step 3 of Algorithm 2. In fact, $\widehat{\mathscr{J}}$ can be viewed as the original Jacobian system J preconditioned by a one-level additive Schwarz preconditioner. Hence, $\widehat{\mathscr{J}}$ is well-conditioned through nonlinear preconditioning as long as the number of subdomains is not very large.

Remark 7. As suggested by Dennis and Schnabel in [11, p. 129], we include a re-scaling of the search direction $s^{(k)}$ in Step 4 if $\|s^{(k)}\|_2 \geqslant S_{\max}$. 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. In the next section, we will see 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 8. One key issue of the backtracking technique is the selection of the merit function. For the global nonlinear problem, we use $f(x) = \|\mathscr{F}(x)\|_2^2/2$; for the local nonlinear problem, we use $f(x) = \|F_{\Omega_i'}(x)\|_2^2/2$.

5. Numerical results

In this section, we present a few numerical results using ASPIN to show its convergence properties, robustness with respect to high Reynolds numbers, and parallel scalability. We also compare the results with those obtained using a standard Newton–Krylov–Schwarz algorithm [4,27]. A lid-driven cavity flow problem [20] and a backward-facing step problem [19] are considered here as benchmarks to evaluate the performance of the algorithms. The implementation uses PETSc [2], and all numerical results are obtained on a cluster of distributed-memory workstations. Double precision is used throughout the computation. A zero initial guess is used for all test cases. Only machine-independent results are reported. Timing results will be reported in a future paper based on an optimized version of the current software.

5.1. Selection of parameters for ASPIN

ASPIN is a collection of several nested linear and nonlinear solvers and many stopping parameters are involved. We summarize the parameters selected in our numerical experiments as follows:

- The global nonlinear iteration is stopped if the condition,

  $$\|\mathscr{F}(x^{(k)})\|_2 \leqslant \varepsilon_{\text{global-nonlinear}} \|\mathscr{F}(x^{(0)})\|_2,$$

  is satisfied. We set $\varepsilon_{\text{global-nonlinear}} = 10^{-6}$ for all test cases, and the success of ASPIN is declared when the above condition is satisfied.
- GMRES is used for solving the global Jacobian systems. The global linear iteration is stopped if the condition,

  $$\|\mathscr{F}(x^{(k)}) - \widehat{\mathscr{J}}(x^{(k)})s^{(k)}\|_2 \leqslant \varepsilon_{\text{global-linear}} \|\mathscr{F}(x^{(k)})\|_2,$$

  is satisfied. We vary $\varepsilon_{\text{global-linear}}$ from 10^{-3} to 10^{-6} and find that the global Jacobian systems need to be solved with a certain degree of accuracy, specifically no larger than 10^{-4}, to guarantee the robustness and efficiency of ASPIN for some cases. We also observe that less than 8% of linear iterations can be saved for most cases using larger $\varepsilon_{\text{global-linear}}$. Therefore, to assure the convergence of ASPIN over a wide range of applications, we select $\varepsilon_{\text{global-linear}} = 10^{-6}$ for all test cases.
- The local nonlinear iteration on each subdomain is stopped if the condition,

  $$\|G_{\Omega_i'}(w_{i,l}^{(k)})\|_2 \leqslant \varepsilon_{\text{local-nonlinear}} \|G_{\Omega_i'}(w_{i,0}^{(k)})\|_2,$$

  is satisfied, or if the maximum local nonlinear iteration of 25 is reached. We test different $\varepsilon_{\text{local-nonlinear}}$ ranging from 10^{-4} to 10^{-6} and find that the numbers of ASPIN iterations are nearly independent of $\varepsilon_{\text{local-nonlinear}}$, but the number of local nonlinear iterations per global function evaluation, which is one of the expensive steps in the algorithm, can be reduced by choosing a loose tolerance. Hence, $\varepsilon_{\text{local-nonlinear}} = 10^{-4}$ is set for all test cases.

- Regular checkerboard partitions are used for our experiments. The number of subdomains is always the same as the number of processors, n_p.
- The overlapping size is defined as

$$\text{ovlp} = \max\left\{\frac{H\prime_x - H_x}{2h}, \frac{H\prime_y - H_y}{2h}\right\}$$

for both interior subdomains and subdomains touching the boundary. Rectangular elements are used for the two benchmark problems. H'_x and H'_y are defined here as the side lengths of the overlapping subdomain Ω'_i in x-direction and y-direction, respectively. Similarly, H_x and H_y are defined as the side lengths of the non-overlapping subdomain Ω_i in x-direction and y-direction, respectively. A graphic example is presented in Fig. 2. We use ovlp = 2 for all test cases, except in Table 9, where we study the effect of using different overlapping sizes.
- The local subdomain Jacobian matrix $G'_{\Omega'_i}$, and the global Jacobian matrix J are formed using multi-colored forward finite differences. The finite difference parameter is set at 10^{-8}.
- We use the standard backtracking technique for both global and local nonlinear problems. The parameters associated with INB are: $\alpha = 10^{-4}$, $\lambda_{\min} = 0.1$ and $\lambda_{\max} = 0.5$.

5.2. A Newton–Krylov–Schwarz algorithm

We compare our algorithm with an inexact Newton based algorithm applied directly to the original nonlinear system (2.3). There are several parallel Newton–Krylov algorithms including Newton–Krylov-Multigrid [13,31] and Newton–Krylov–Schwarz (NKS) [4,27,34]. Because of the similar data structure, we compare only with an NKS algorithm. NKS has three main components: a Newton method as the nonlinear solver, a Krylov subspace method as the linear solver, and a Schwarz-type method as the preconditioner. For the nonlinear solver, the INB described in Section 3 is employed. Three choices of the forcing term η_k are tested. Choices 1 and 2 are given by (3.1) and (3.2). We denote the constant $\eta_k = 10^{-6}$ as

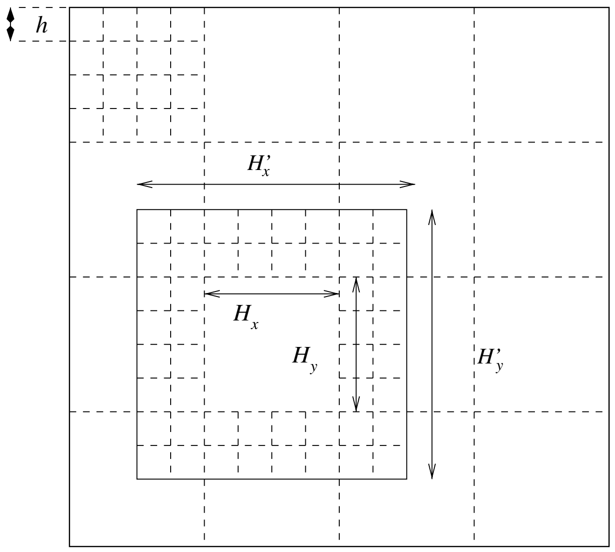


Fig. 2. A sample mesh partition with ovlp = 2 on a rectangular mesh.

Table 1
Parameters for Choices 1 and 2

For both Choices 1 and 2	
Initial forcing term η_0	0.01
Maximum forcing term $\eta_{\max}$	0.9
For Choice 2 only	
Power ρ	2.0
Multiplication factor γ	0.9

Choice 0. For the linear solver, we apply GMRES for solving each Jacobian system. To accelerate the convergence of GMRES, we choose a one-level additive Schwarz method as a right preconditioner: At each Newton iteration, find a Newton direction $s^{(k)}$ to satisfy

$$\|F(x^{(k)}) - (J(x^{(k)})M_k^{-1})(M_k s^{(k)})\|_2 \leqslant \eta_k \|F(x^{(k)})\|_2, \tag{5.1}$$

where $M_k^{-1} = \sum_{i=1}^{N} R_{\Omega_i'}^{\mathrm{T}} J_{\Omega_i'}^{-1}(x^{(k)}) R_{\Omega_i'}$. Note that right preconditioning preserves the l_2-norm so that the preconditioned system (5.1) changes neither the linear residual norm nor the function norm. We use the same partition and overlap as in the corresponding ASPIN algorithm. The list of parameters for Choices 1 and 2 appears in Table 1 [15,16]. We declare the convergence of INB if the condition,

$$\|F(x^{(k)})\|_2 \leqslant \varepsilon_{\text{nonlinear}} \|F(x^{(0)})\|_2$$

is satisfied. Here $\varepsilon_{\text{nonlinear}} = 10^{-6}$ for all test cases. Otherwise, we claim that NKS fails. This happens if the maximum nonlinear iteration of 100 is reached, or the backtracking fails.

5.3. Test 1: lid-driven cavity flow

We consider the incompressible lid-driven cavity flow defined on the unit square. The flow domain and boundary conditions are shown in Fig. 3. Because the lid velocity $V = 1$, and the length of the lid $L = 1$, the Reynolds number for this problem is $1/\nu$. Since only Dirichlet boundary condition is specified for the velocity, the pressure is determined up to a constant. To make the pressure unique, we set its value at the lower right corner to zero. We run the test for three uniform meshes 64×64, 128×128 and 256×256. The subdomains are obtained by partitioning the mesh uniformly as shown in Fig. 2. For this test case, we consider 2×2 and 4×4 subdomain partitions. As mentioned before, the number of processors is the same as the number of subdomains.

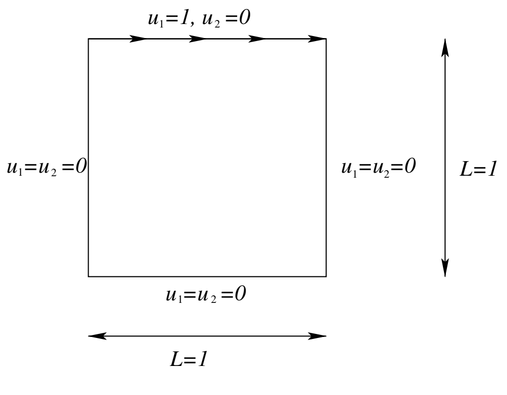


Fig. 3. Test 1: a lid-driven cavity flow problem.

Table 2
Nonlinear residuals and quantitative comparisons of the numerical solutions obtained by two methods, NKS and ASPIN

Mesh sizes	$Re = 10^3$	$Re = 5 \times 10^3$	$Re = 10^4$
$\Vert F(x^*_{\mathrm{ASPIN}})\Vert_2$			
128×128	4.64×10^{-11}	5.58×10^{-13}	3.48×10^{-13}
256×256	1.30×10^{-9}	2.98×10^{-9}	1.20×10^{-11}
$\Vert x^*_{\mathrm{NKS}} - x^*_{\mathrm{ASPIN}}\Vert_2/\Vert x^*_{\mathrm{NKS}}\Vert_2$			
128×128	1.75×10^{-7}	1.11×10^{-7}	3.22×10^{-9}
256×256	6.39×10^{-7}	4.62×10^{-9}	2.58×10^{-7}

5.3.1. Numerical verification of the computed solutions

We claim that two systems of equations are equivalent if they have the same solution. To show that the original system and the preconditioned system are equivalent is trivial for the case of linear preconditioning, but not so for the case of nonlinear preconditioning. In [5], Cai and Keyes proved the equivalence of two nonlinear systems, the original system and nonlinearly preconditioned system, under certain assumptions. However, as mentioned before, applying the equivalence theorem in [5] for a general nonlinear system $F(x) = 0$, like (2.3), is not straightforward since it is difficult to check whether the assumptions hold or not for our cases. Instead, we verify numerically that the solution of the preconditioned system (4.3) is also the solution of the original nonlinear system (2.3).

Let x^*_{NKS} and x^*_{ASPIN} be the numerical solutions of the original and preconditioned nonlinear systems, respectively. Here, the reference solution x^*_{NKS} is obtained by using NKS for $Re = 10^3$ and for $Re = 5 \times 10^3$ and 10^4 the original nonlinear system is solved by NKS with a zeroth-order Reynolds number based continuation technique [23], since NKS itself fails to converge for such high Reynolds numbers. x^*_{ASPIN} is obtained by using ASPIN. The relative tolerance for both NKS and ASPIN are set at 10^{-10}. To show that x^*_{ASPIN} is close to x^*_{NKS}, we evaluate $F(x)$ at $x = x^*_{\mathrm{ASPIN}}$ for several Reynolds numbers and mesh sizes, and the values are given in Table 2. The nonlinear residuals in all cases are of order 10^{-9} or smaller, which implies that x^*_{ASPIN} is a solution of $F(x) = 0$ for each case. Also, in the same table we show the relative errors in l_2-norm, $\Vert x^*_{\mathrm{NKS}} - x^*_{\mathrm{ASPIN}}\Vert_2/\Vert x^*_{\mathrm{NKS}}\Vert_2$. All values are relatively small; we believe that the minor differences are due to the use of preconditioner-dependent stopping conditions. Hence, these quantitative comparisons confirm our assertion that nonlinear preconditioning does not alter the solution of the nonlinear system. Figs. 4 and 5 are two typical sets of solution plots. Clearly, the streamlines in Fig. 4 and the pressure elevations in Fig. 5 obtained from solving two systems for $Re = 10^4$ on a 256×256 mesh are almost indistinguishable.

5.3.2. Choices of S_{max}

The choice of $S_{\max}$ in the re-scaling step is critical to both the robustness and the efficiency of ASPIN. Here, the "feasible" values of $S_{\max}$ are those values for which ASPIN converges. In general, the range of feasible values of $S_{\max}$ depends on some physical parameters, such as the Reynolds number, the mesh size and the subdomain size. The optimal $S_{\max}$ is determined empirically so that ASPIN takes the smallest number of global nonlinear iterations in several successful runs. Fig. 6 shows the history of nonlinear residuals of ASPIN with different $S_{\max}$ for $Re = 10^4$. Note that ASPIN fails to converge when $S_{\max} > 3$ in this case. From the plot, we see that although small $S_{\max}$ enhances the robustness of ASPIN, it slows down the overall convergence. The number of ASPIN iterations is decreased as we increase $S_{\max}$. Up to 50% nonlinear iterations can be saved if we choose the optimal $S_{\max}$. Table 3 presents the feasible intervals and the optimal values for $S_{\max}$ in the cases of a 128×128 mesh on 16 processors for $Re = 10^3$, 5×10^3 and 10^4. The data in the table is obtained by first trying a modest number of $S_{\max}$ values to assure the convergence of ASPIN for each case, then estimating the approximate feasible intervals and the

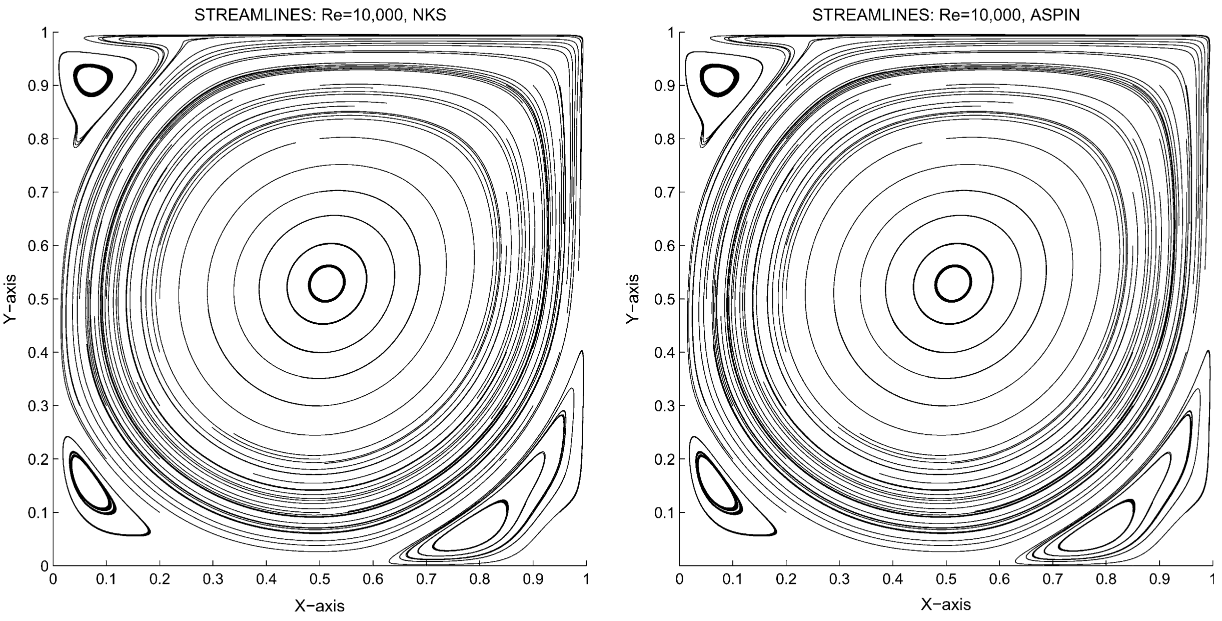
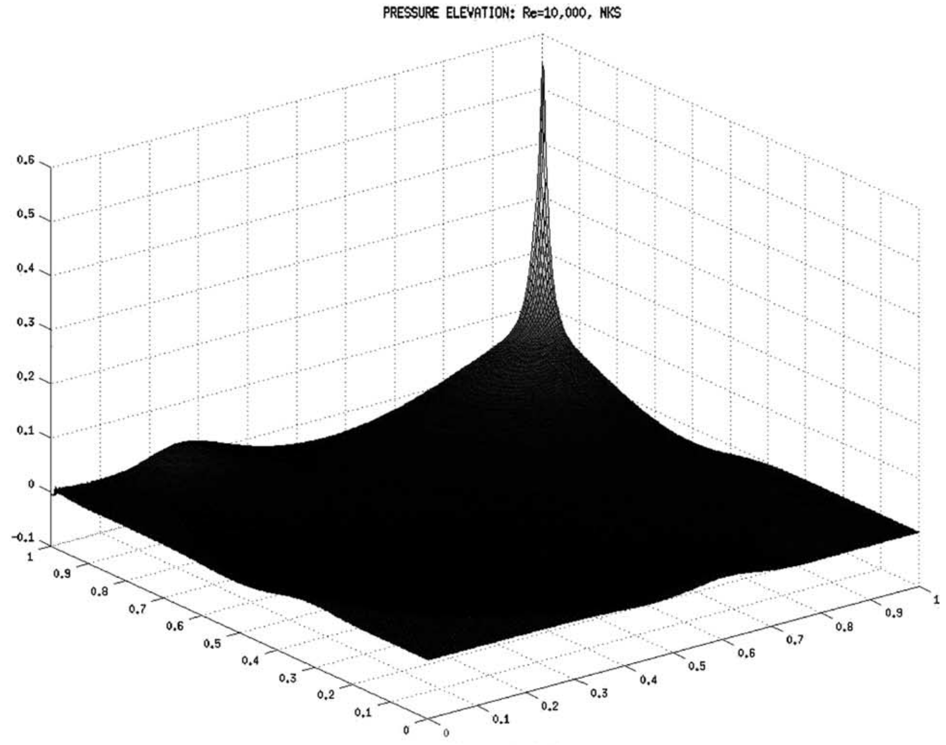


Fig. 4. Streamlines. The original (left) and preconditioned (right) nonlinear system.

optimal values of $S_{\max}$ from these results. As shown in Table 3, the range of feasible $S_{\max}$ is wider for a low Reynolds number flow than for a high Reynolds number flow. The sudden failure of ASPIN occurs for $S_{\max}$ values greater than the optimal $S_{\max}$ in the cases of the two largest Reynolds numbers. In other words, for high Reynolds number flows, choosing a feasible value of $S_{\max}$ to assure the convergence of ASPIN is more difficult. In that case, it is safer to select small enough $S_{\max}$, compared to the first step length $\|s^{(1)}\|_2$ to guarantee the convergence of ASPIN at the first trial, and then the performance of ASPIN can be improved by increasing $S_{\max}$ gradually. Similar phenomenon is also observed in the case of choosing a good relaxation parameter for the Successive Over Relaxation (SOR) method. The convergence of SOR typically deteriorates rapidly to the right of the optimal relaxation parameter than to the left; consequently, it is usually better to choose a small relaxation parameter. It should be noted that all numerical results for ASPIN presented later in this section are obtained by using the optimal values of $S_{\max}$ for each problem.

5.3.3. Comparison with NKS

In Figs. 7 and 8, we compare the nonlinear residual history of ASPIN with those of NKS with three different choices of forcing terms. We run ten tests for Reynolds numbers ranging from 10^3 to 10^4, with an increment of 10^3. All results are obtained by using a 128×128 mesh on 16 (4×4) processors. We see that nonlinear residuals of NKS with all choices of forcing terms behave similarly. Except for a few cases with low Reynolds number, NKS nonlinear residuals stagnate around 10^{-3} without any progress after about the first 15 iterations. All of them fail to converge after 100 iterations. Different choices of forcing terms do not help much in this particular set of tests. We should note that the success of NKS using these two adaptive forcing terms on the lid-driven cavity problem up to $Re = 10^4$ has been reported in [34]. Several parameters in NKS need to be well-tuned in order for the method to converge for all applications. By comparing our implementation with [34], we find that the differences in the selections of some algorithmic parameters, such as quality of subdomain solve (exact or inexact), the number of subdomains, the degree of overlap, and the choices of some discretization parameters, such as the stabilization parameter, τ, may affect the convergence



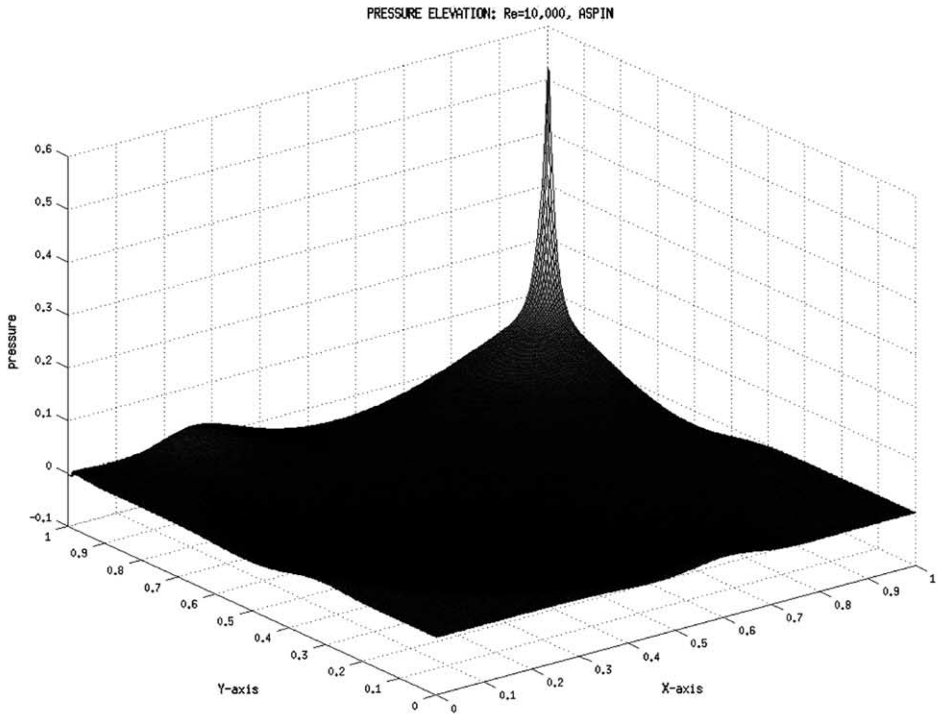


Fig. 5. Pressure elevations. The original (top) and preconditioned (bottom) nonlinear system.

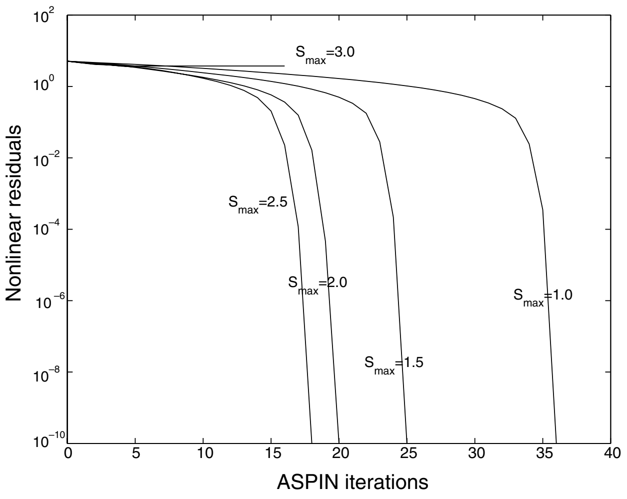


Fig. 6. History of nonlinear residuals of ASPIN with different values of $S_{\max}$. A 128×128 mesh is used on 16 processors, $Re = 10^4$. ASPIN with $S_{\max} = 3.0$ is terminated at an earlier iteration because of the failure of backtracking.

of NKS. As we will see in the next test case, NKS with adaptive forcing terms seems sensitive to the change in the number of processors (or subdomains). It may be interesting to study how these factors affect the convergence and the performance of NKS. On the other hand, ASPIN converges for the whole range of Reynolds numbers. Furthermore, we find that ASPIN preserves the local quadratic convergence of Newton when the intermediate solution is near the desired solution.

5.3.4. Scalability of ASPIN

Scalability is an important issue in parallel computing, and the issue becomes even more significant when we solve large scale problems with many processors. Table 4 shows that ASPIN iterations are nearly independent of mesh size; the nonlinear iteration numbers change up or down by small fractions when we increase the mesh size from a coarse mesh, 64×64, to a fine mesh, 256×256 on 16 (4×4) processors. However, the average number of GMRES iterations increases quite a bit when we increase the mesh size. Next, we fix the mesh size and vary the number of processors. In Table 5, we see that the number of ASPIN iterations does not change much, while the average number of GMRES iterations increases a lot when we increase the number of processors from 4 to 16 on a fixed 128×128 mesh. The increase in GMRES iteration numbers is not unexpected, since we do not have a coarse space in the preconditioner.

Table 3
Feasible values of $S_{\max}$ and optimal values of $S_{\max}$. A 128×128 mesh on 16 processors for $Re = 10^3$, 5×10^3 and 10^4. Note that the optimal values of $S_{\max}$ may or may not be unique. For example, several optimal values are in the interval [300, 450] for $Re = 10^3$, while only single optimal values are found for $Re = 5 \times 10^3$ and 10^4

Re	$\|s^{(1)}\|_2$	Feasible values for $S_{\max}$	Optimal value for $S_{\max}$
10^3	694	[2, 600]	[300, 450]
5×10^3	156	[2, 25]	25
10^4	57	[1, 2.5]	2.5

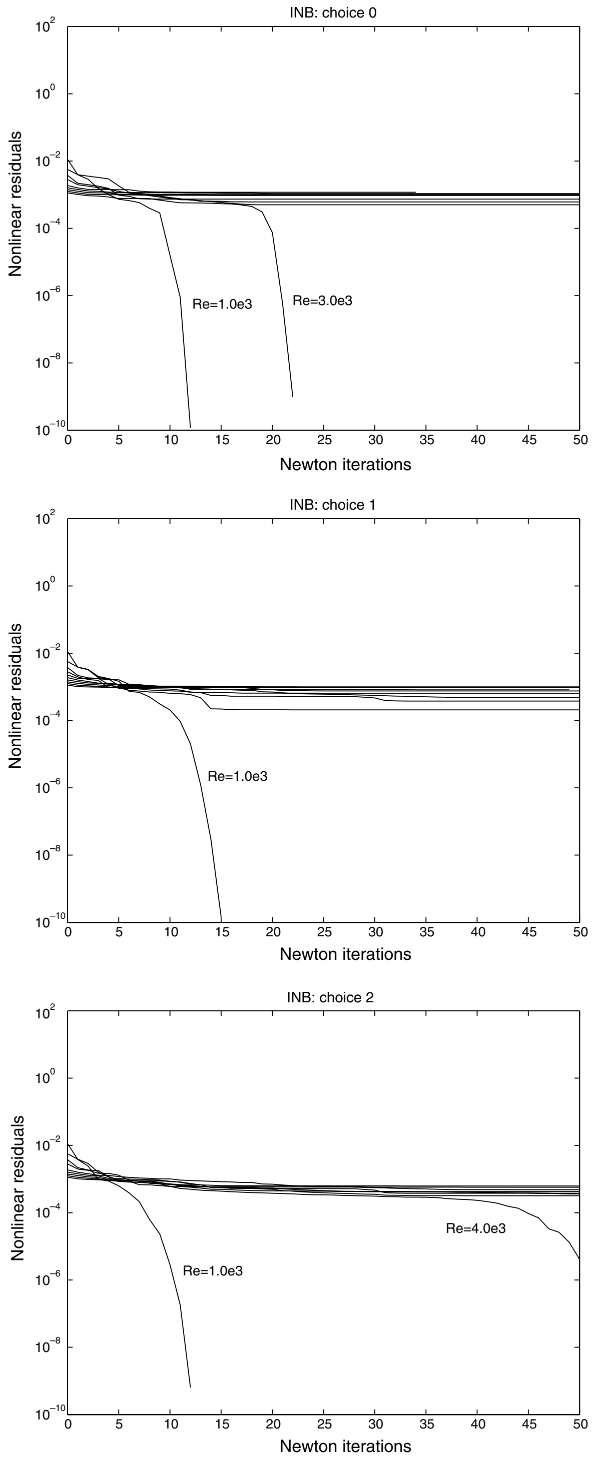


Fig. 7. History of nonlinear residuals. INB with three different forcing terms. Only converged cases are labelled.

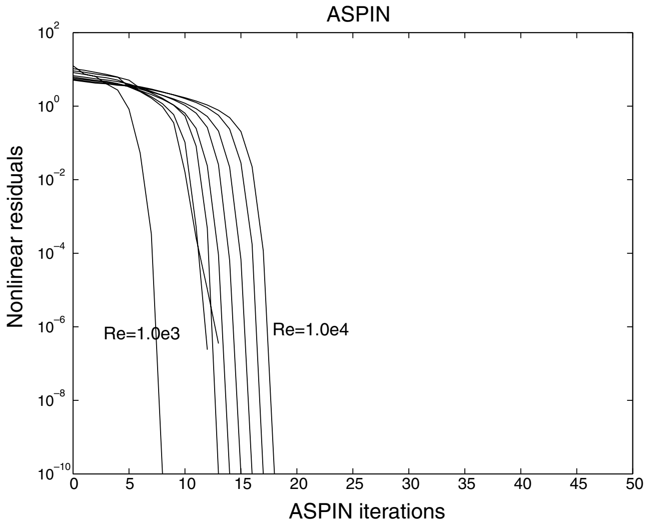


Fig. 8. History of nonlinear residuals. ASPIN converges in all ten test cases.

5.3.5. *Solving the subdomain nonlinear problems*

Fig. 9 shows the numbers of subdomain nonlinear iterations required for different subdomains in Step 1 of Algorithm 2. In this test case, we partition the domain into 2×2 subdomains and number them, first from bottom to top, and then from left to right. Note that two subdomains, Ω_2 and Ω_4, touch the moving

Table 4
Lid-driven cavity problem: different mesh sizes on 16 processors

Mesh sizes	$Re = 10^3$	$Re = 3\times 10^3$	$Re = 5\times 10^3$	$Re = 8\times 10^3$	$Re = 10^4$
Number of ASPIN iterations					
64×64	11	12	15	17	18
128×128	9	13	11	16	18
256×256	12	13	14	18	15
Average number of GMRES iterations					
64×64	86	88	92	97	97
128×128	129	128	131	137	140
256×256	190	188	194	197	197

Table 5
Lid-driven cavity problem: different number of processors on a 128×128 mesh

n_p	$Re = 10^3$	$Re = 3\times 10^3$	$Re = 5\times 10^3$	$Re = 8\times 10^3$	$Re = 10^4$
Number of ASPIN iterations					
$2\times 2 = 4$	11	10	13	19	19
$4\times 4 = 16$	8	13	11	16	18
Average number of GMRES iterations					
$2\times 2 = 4$	67	69	71	73	74
$4\times 4 = 16$	129	128	131	137	140

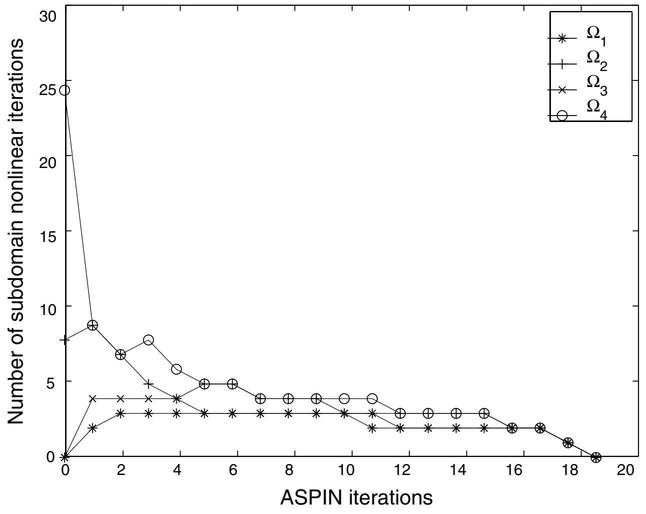


Fig. 9. Numbers of subdomain nonlinear iterations required for different subdomains in Step 1 of Algorithm 2. A 128×128 mesh is tested on 4 processors, and $Re = 10^4$.

lid. ASPIN converges in 19 iterations. We observe that at the beginning of ASPIN iterations, ASPIN has difficulty solving the subdomain nonlinear problem by INB with a zero initial guess on the subdomain Ω_4, where a boundary layer and singularity are present for $Re = 10^4$. INB starts to stall after about 15 iterations and the nonlinear residual cannot be reduced to the order of 10^{-4} on Ω_4 because the size of the subdomain problem is too large. We may resolve this situation by increasing the number of subdomains; there are no unsuccessful subdomain nonlinear iterations observed as we use 4×4 subdomains to solve the same problem. After 7 ASPIN iterations, the nonlinearities of each subdomain problem become more balanced, and only about up to 4 Newton iterations are needed to solve each subdomain problem.

5.4. Test 2: flow passing a backward-facing step

We consider another benchmark problem often used to test the correctness and performance of numerical algorithms. For the Eq. (2.1) defined on a long channel $[0,30] \times [-0.5,0.5]$, no-slip conditions are imposed on the top and bottom walls, as well as the lower half of the left wall, i.e., $u_1 = u_2 = 0$. Detailed geometry and boundary condition information appears in Fig. 10. A fully developed parabolic velocity profile is specified at the inlet boundary, which is given by $u_1(y) = 24y(0.5 - y)$ for $0 \leqslant y \leqslant 0.5$. Here, we have a maximum velocity of 1.5 and an average velocity of $u_{\text{ave}} = 1$ at the inlet boundary. The Reynolds number for this problem is defined as $Re = V_{\text{ave}}L/\nu$, where V_{ave} is the average velocity at the inlet boundary (here,

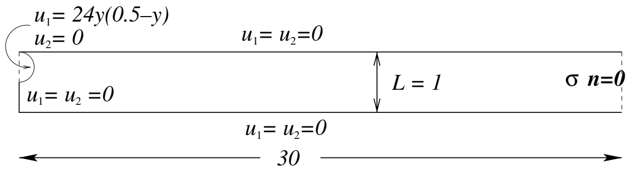


Fig. 10. Test 2: a backward-facing step problem.

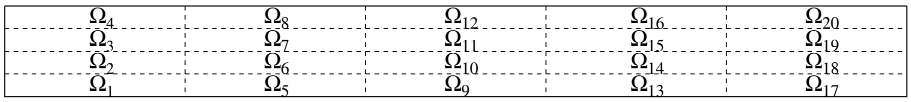


Fig. 11. Backward facing step problem: subdomain numbering.

$V_{\text{ave}} = 1$) and L is the channel height (here, $L = 1$). We apply two uniform meshes (600×20 and 1200×40). Two subdomain partitions are considered: 10 (5×2) and 20 (5×4) processors. The subdomain ordering for the 20-processor case is shown in Fig. 11.

5.4.1. Scalability of NKS and ASPIN

First, we look at the performance of NKS with three choices of forcing terms when we vary the number of processors and the mesh size. In Table 6, unlike the lid-driven cavity flow problem, we observe that certain choices of forcing term do enhance the robustness of NKS in some cases. For example, NKS with Choice 2 converges for all tested Reynolds numbers on 10 (5×2) and 20 (5×4) processors, while NKS with Choice 0 fails to converge when the Reynolds number is higher than 5×10^2. Furthermore, the average number of GMRES iterations for NKS with Choice 2 is quite small compared with Choice 0 when the convergence can be achieved. However, NKS seems quite sensitive to the changes in the number of processors: The number of Newton iterations for Choice 2 nearly doubles when we increase the number of processors from 10 to 20 in the cases of high Reynolds number. Meanwhile, the number of successes for Choice 1 is reduced from 5 to 2 when we use a stronger preconditioner for the Jacobian system. In Table 7, we see that refining the mesh increases neither the number of Newton iterations nor the average number of GMRES iterations significantly. For high Reynolds numbers, such as $Re = 7 \times 10^2$ and 8×10^2, NKS with Choice 1 and 2 on a fine mesh requires even fewer numbers of nonlinear iterations than on a coarse mesh.

Next, we study the scalability of ASPIN for the backward-facing step problem. Table 8 shows how the number of ASPIN iterations and the average number of GMRES iterations change when we increase the mesh size from a coarse mesh, 600×20, to a fine mesh, 1200×40 on 20 (5×4) processors. We see that the average number of ASPIN iterations remains the same in the case of $Re = 100$. However, if $Re = 800$,

Table 6
Backward-facing step problem: comparison of NKS with three choices of forcing terms. The number of Newton iterations and the average number of GMRES iterations for different Reynolds numbers. 1200×40 elements on 5×2 and 5×4 processors. "–" indicates a failure of convergence

Choice number:	0		1		2	
n_p	5×2	5×4	5×2	5×4	5×2	5×4
Number of nonlinear iterations						
$Re = 1 \times 10^2$	6	6	7	7	6	6
$Re = 5 \times 10^2$	–	–	–	20	14	28
$Re = 6 \times 10^2$	–	–	17	29	19	37
$Re = 7 \times 10^2$	–	–	–	44	22	50
$Re = 8 \times 10^2$	–	–	–	43	27	57
Average number of GMRES iterations						
$Re = 1 \times 10^2$	62	119	20	44	30	48
$Re = 5 \times 10^2$	–	–	–	23	14	25
$Re = 6 \times 10^2$	–	–	13	29	12	24
$Re = 7 \times 10^2$	–	–	–	35	14	25
$Re = 8 \times 10^2$	–	–	–	31	19	28

Table 7
Backward-facing step problem: comparison of NKS with three choices of forcing terms. The number of Newton iterations and the average number of GMRES iterations for different Reynolds numbers. 600×20 and 1200×40 meshes on 20 (5×4) processors. "–" indicates a failure of convergence

Choice number:	0		1		2	
Mesh sizes	600×20	1200×40	600×20	1200×40	600×20	1200×40
Number of nonlinear iterations						
$Re = 1 \times 10^2$	6	6	7	7	6	6
$Re = 5 \times 10^2$	–	–	26	20	22	28
$Re = 6 \times 10^2$	–	–	35	29	33	37
$Re = 7 \times 10^2$	–	–	61	44	63	50
$Re = 8 \times 10^2$	–	–	57	43	72	57
Average number of GMRES iterations						
$Re = 1 \times 10^2$	62	119	41	44	43	48
$Re = 5 \times 10^2$	–	–	22	23	17	25
$Re = 6 \times 10^2$	–	–	28	29	16	24
$Re = 7 \times 10^2$	–	–	23	35	20	25
$Re = 8 \times 10^2$	–	–	23	31	20	28

Table 8
Backward-facing step problem: different mesh sizes on 20 (5×4) processors

Mesh sizes	$Re = 10^2$	$Re = 5 \times 10^2$	$Re = 6 \times 10^2$	$Re = 7 \times 10^2$	$Re = 8 \times 10^2$
Number of ASPIN iterations					
600×20	5	9	15	21	18
1200×40	5	18	19	28	39
Average number of GMRES iterations					
600×20	91	93	94	99	98
1200×40	118	127	132	134	136

Table 9
Backward-facing step problem: varying the overlapping size on a 120×40 mesh

ovlp	$Re = 10^2$	$Re = 5 \times 10^2$	$Re = 6 \times 10^2$	$Re = 7 \times 10^2$	$Re = 8 \times 10^2$
Number of ASPIN iterations					
2	5	18	19	23	39
4	5	12	16	15	26
6	5	9	11	12	13
Average number of GMRES iterations					
2	118	127	132	142	130
4	83	88	90	97	94
6	67	77	80	83	84

the iteration number doubles as we increase the mesh size. The average number of GMRES iterations increases for all values of Reynolds number as expected. In Table 9, we observe that increasing the overlapping size can reduce both ASPIN iterations and GMRES iterations when the Reynolds number is high. Table 10 shows that the number of ASPIN iterations is not sensitive to the number of processors, while the

Table 10
Backward-facing step problem: different number of processors with the same mesh 1200×40

n_p	$Re = 10^2$	$Re = 5 \times 10^2$	$Re = 6 \times 10^2$	$Re = 7 \times 10^2$	$Re = 8 \times 10^2$
Number of ASPIN iterations					
$5 \times 2 = 10$	5	12	20	27	35
$5 \times 4 = 20$	5	18	19	28	39
Average number of GMRES iterations					
$5 \times 2 = 10$	62	68	69	71	72
$5 \times 4 = 20$	118	127	132	134	130

Table 11
Backward facing step problem: total number of subdomain nonlinear iterations; 1200×40 mesh, 5×4 subdomains on 20 processors

Subdomain	$Re = 100$	$Re = 500$	$Re = 700$	$Re = 800$
Ω_1	8	38	74	109
Ω_2	11	46	85	127
Ω_3	13	61	124	182
Ω_4	13	67	147	238
Ω_5	5	40	78	115
Ω_6	5	43	78	111
Ω_7	5	43	77	109
Ω_8	5	38	78	118
Ω_9	4	18	42	59
Ω_{10}	4	25	45	63
Ω_{11}	4	24	46	71
Ω_{12}	4	18	45	67
Ω_{13}	4	15	26	38
Ω_{14}	4	17	36	54
Ω_{15}	4	15	39	56
Ω_{16}	4	15	26	38
Ω_{17}	6	28	51	74
Ω_{18}	6	29	52	76
Ω_{19}	6	28	55	75
Ω_{20}	6	28	51	74

average GMRES iterations increase a lot when we increase the number of processors from 10 to 20 on a fixed 1200×40 mesh.

5.4.2. Solving the subdomain nonlinear problems

In Table 11, we compare the numbers of Newton iterations for solving the subdomain nonlinear problems. In this test case, we partition the domain into 5×4 subdomains and number them naturally, first from bottom to top, and then from left to right. See Fig. 11. Note that the inlet boundary is shared by two subdomains Ω_3 and Ω_4; the outlet boundary is shared by four subdomains, Ω_{17}, Ω_{18}, Ω_{19} and Ω_{20}. From Fig. 12, we observe that for high Reynolds number flows there are two singularities within subdomains from Ω_1 to Ω_4 and the pressure changes significantly in subdomains Ω_3 and Ω_4 compared with others such as the subdomains from Ω_{15} to Ω_{20}. As expected, more Newton iterations are needed in the subdomains Ω_3 and Ω_4, about four times as many as in other smooth regions.

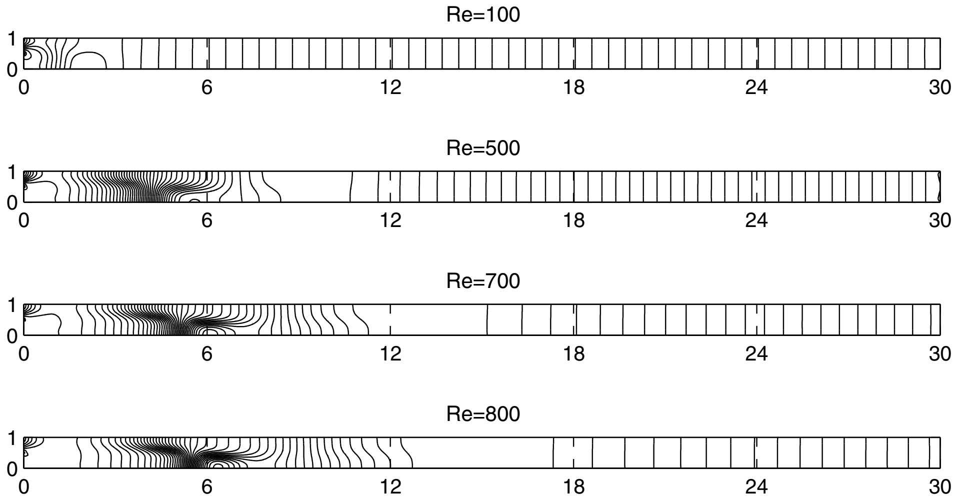


Fig. 12. Backward-facing step problem: pressure contours for different Reynolds numbers. The solutions are obtained using ASPIN on a 1200×40 mesh and 5×4 processors.

6. Conclusions and remarks

Finding a fast, robust and scalable solver for incompressible Navier–Stokes equations is one of the key research areas in computational fluid dynamics. Several important progresses have been made in this area, such as [34]. In this paper, we developed a fully parallel nonlinearly preconditioned inexact Newton method for solving incompressible Navier–Stokes equations in the primitive variable form. The nonlinear preconditioner is constructed using the overlapping additive Schwarz domain decomposition method. A PETSc based parallel software package was developed and tested for the two-dimensional incompressible Navier–Stokes equations discretized with a stabilized $Q_1 - Q_1$ finite element method. From the numerical experiments on two benchmark problems including a lid-driven cavity flow problem and a backward-facing step problem, we concluded that the new method is more robust than the commonly used inexact Newton method with backtracking for high Reynolds number flows. Some parallel scalability results were also given for a moderate number of processors. Finally, it should be noted that since the global function evaluation of ASPIN is much more expensive than that of NKS, ASPIN is intended for problems that NKS fails to converge or experiences unacceptably slow convergence.

Acknowledgements

We thank the PETSc group of the Argonne National Laboratory for their software support of the research, and L.P. Franca and S.F. McCormick for many helpful discussions. We also thank the anonymous referees for their valuable suggestions.

References

[1] V.F. DE Almeida, J.J. Derby, Construction of solution curves for large two-dimensional problems of steady-state flows of incompressible fluids, SIAM J. Sci. Comput. 22 (2000) 285–311.

[2] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Users Manual, ANL-95/11 – Revision 2.1.5, Argonne National Laboratory, 2002.

[3] 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. Meth. Appl. Mech. Eng. 32 (1982) 199–259.

[4] X.-C. Cai, W.D. Gropp, D.E. Keyes, R.G. Melvin, D.P. Young, Parallel Newton–Krylov–Schwarz algorithms for the transonic full potential equation, SIAM J. Sci. Comput. 19 (1998) 246–265.

[5] X.-C. Cai, D.E. Keyes, Nonlinearly preconditioned inexact Newton algorithms, SIAM J. Sci. Comput. 24 (2002) 183–200.

[6] X.-C. Cai, D.E. Keyes, L. Marcinkowski, Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics, Int. J. Numer. Meth. Fluids 40 (2002) 1463–1470.

[7] X.-C. Cai, D.E. Keyes, D.P. Young, A nonlinear additive Schwarz preconditioned inexact Newton method for shocked duct flows, in: Proceedings of the 13th International Conference on Domain Decomposition Methods, France, October 9–12, 2000.

[8] T.S. Coffey, C.T. Kelley, D.E. Keyes, Pseudo-transient continuation and differential-algebraic equations, SIAM J. Sci. Comput. 25 (2003) 553–569.

[9] T.F. Coleman, J.J. Moré, Estimation of sparse Jacobian matrices and graph coloring problem, SIAM J. Numer. Anal. 20 (1983) 209–243.

[10] M. Dryja, W. Hackbusch, On the nonlinear domain decomposition method, BIT (1997) 296–311.

[11] J. Dennis, R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear EquatioSIAM, SIAM, Philadelphia, 1996.

[12] R.S. Dembo, S.C. Eisenstat, T. Steihaug, Inexact Newton methods, SIAM J. Numer. Anal. 19 (1982) 400–408.

[13] M. Dumett, P. Vassilevski, C.S. Woodward, A multigrid method for nonlinear unstructured finite element elliptic equations, SIAM J. Sci. Comput. 2003, submitted.

[14] H.C. Elman, V.E. Howle, J.N. Shadid, R.S. Tuminaro, A parallel block multi-level preconditioner for the 3D incompressible Navier–Stokes equations, J. Comput. Phy. 187 (2003) 504–523.

[15] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Opt. 4 (1994) 393–422.

[16] S.C. Eisenstat, H.F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM J. Sci. Comput. 17 (1996) 16–32.

[17] L.P. Franca, S.L. Frey, Stabilized finite element method: II. The incompressible Navier–Stokes equation, Comput. Meth. Appl. Mech. Eng. 99 (1992) 209–233.

[18] L.P. Franca, S.L. Frey, T.J.R. Hughes, Stabilized finite element method: I. Application to the advective–diffusive model, Comput. Meth. Appl. Mech. Eng. 95 (1992) 253–276.

[19] D.K. Gartling, A test problem for outflow boundary conditions- flow over a backward-facing step, Int. J. Numer. Meth. Fluids 11 (1990) 953–967.

[20] 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.

[21] P.M. Gresho, D.K. Gartling, J.R. Torczynski, K.A. Cliffe, K.H. Winters, T.J. Garratt, A. Spence, J.W. Goodrich, Is the steady viscous incompressible two-dimensional flow over a backward facing step at Re = 800 stable, Int. J. Numer. Meth. Fluids 17 (1993) 501–541.

[22] M.D. Gunzburger, Finite Element Methods for Viscous Incompressible Flows, Academics Press, New York, 1989.

[23] M.D. Gunzburger, J. Peterson, Predictor and steplength selection in continuation methods for the Navier–Stokes equations, Comput. Math. Appl. 22 (1991) 73–81.

[24] S.K. Hannani, M. Stanislas, P. Dupont, Incompressible Navier–Stokes computation with SUPG and GLS formulations – a comparison study, Comput. Meth. Appl. Mech. Eng. 124 (1995) 153–170.

[25] D. Hendriana, L.J. Bethe, On upwind methods for parabolic finite elements in incompressible flows, Int. J. Numer. Meth. Eng. 47 (2000) 317–340.

[26] C.T. Kelley, D.E. Keyes, Convergence analysis of pseudo-transient continuation, SIAM J. Sci. Comput. 35 (1998) 508–523.

[27] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357–397.

[28] A. Klawonn, L. Pavarino, Overlapping Schwarz methods for mixed linear elasticity and Stokes problems, Comput. Meth. Appl. Meth. Eng. 165 (1998) 233–245.

[29] W. Layton, H.K. Lee, J. Peterson, Numerical solution of the stationary Navier–Stokes equations using a multilevel finite element method, SIAM J. Sci. Comput. 20 (1998) 1–12.

[30] J. Nocedal, S.J. Wright, Numerical Optimization, Springer-Verlag, New York, 1999.

[31] M. Pernice, M.D. Tocci, A multigrid-preconditioned Newton–Krylov method for the incompressible Navier–Stokes equations, SIAM J. Sci. Comput. 23 (2001) 398–418.

[32] J.N. Reddy, D.K. Gartling, The Finite Element Method in Heat Transfer and Fluid Dynamics, CRC Press, Florida, 2000.

[33] Y. Saad, M.H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comp. 7 (1986) 856–869.

[34] 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.
[35] B.F. Smith, P. Bjørstad, W.D. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.
[36] 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.
[37] A. Toselli, O. Widlund, Domain Decomposition Methods – Algorithms and Theory, Springer, 2004.