A Combined Linear and Nonlinear Preconditioning Technique for Incompressible Navier-Stokes Equations *

Feng-Nan Hwang and Xiao-Chuan Cai

Department of Computer Science, University of Colorado, Boulder, CO 80309, USA
(hwangf@colorado.edu and cai@cs.colorado.edu)

Abstract. We propose a new two-level nonlinear additive Schwarz preconditioned inexact Newton algorithm (ASPIN). The two-level nonlinear preconditioner combines a local nonlinear additive Schwarz preconditioner and a global *linear* coarse preconditioner. Our parallel numerical results based on a lid-driven cavity incompressible flow problem show that the new two-level ASPIN is nearly scalable with respect to the number of processors if the coarse mesh size is fine enough.

1 Introduction

We focus on the parallel numerical solution of large, sparse nonlinear systems of equations arising from the finite element discretization of nonlinear partial differential equations. Such systems appear in many computational science and engineering applications, such as the simulation of fluid flows [8]. In particular, we introduce a nonlinearly preconditioned iterative method that is robust and scalable for solving nonlinear systems of equations. Our approach is based on the inexact Newton method with backtracking technique (INB) [4], which can be briefly described as follows. Let

$$F(x^*) = 0 \tag{1}$$

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) can be computed by first finding an inexact Newton direction $s^{(k)}$ satisfying

$$||F(x^{(k)}) + F'(x^{(k)})s^{(k)}||_2 \leq \eta_k ||F(x^{(k)})||_2,$$

then obtaining $x^{(k+1)}$ with $x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)}$. 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 GMRES. The scalar $\lambda^{(k)}$ is selected using a linesearch technique. Although INB has the desirable property of

* The work was partially supported by the Department of Energy, DE-FC02-01ER25479, and by the National Science Foundation, CCR-0219190, ACI-0072089 and CCF-0305666.

local fast convergence, like other nonlinear iterative methods, INB is very fragile. It converges rapidly for a well-selected set of parameters (for example, certain initial guess, certain range of the Reynolds number Re), but fails to converge due to a change in some parameters. It is often observed that INB converges well at the beginning of the iterations, then suddenly stalls for no apparent reason. In [2, 3, 6] some nonlinear preconditioning techniques were developed, and the convergence of Newton-type methods becomes not sensitive to these unfriendly parameters if INB is applied to a nonlinearly preconditioned system

$$\mathcal{F}(x^*) = 0 \tag{2}$$

instead. Here the word "preconditioner" refers to the fact that systems (1) and (2) have the same solution and the new system (2) is better conditioned, both linearly and nonlinearly. The preconditioner is constructed using a nonlinear additive Schwarz method. To improve the processor scalability, a two-level method was then proposed in [3], which works well if the number of processors is not large. For a large number of processors, the *nonlinear* coarse solver takes too much CPU and communication times. In this paper, we suggest a combined *linear and nonlinear* additive Schwarz preconditioner and show that using a linear coarse solver we can retain the nonlinear robustness and reduce the nonlinear complexity considerably.

2 Nonlinear preconditioning algorithms

In this section, we describe a two-level nonlinear preconditioner based on a combination of local nonlinear additive Schwarz preconditioners and a global linear coarse preconditioner. We restrict our discussion to a two-component system (velocity and pressure) resulting from the finite element discretization of two-dimensional steady-state incompressible Navier-Stokes equations defined on a bounded domain Ω in R^2 with a polygonal boundary Γ:

$$\begin{cases} \boldsymbol{u} \cdot \nabla \boldsymbol{u} - 2\nu \nabla \cdot \epsilon(\boldsymbol{u}) + \nabla p = 0 & \text{in} \quad \Omega, \\ \nabla \cdot \boldsymbol{u} = 0 & \text{in} \quad \Omega, \\ \boldsymbol{u} = \boldsymbol{g} & \text{on} \quad \Gamma, \end{cases} \tag{3}$$

where $\boldsymbol{u} = (u_1, u_2)$ is the velocity, p is the pressure, $\nu = 1/Re$ is the dynamic viscosity, and $\epsilon(\boldsymbol{u}) = 1/2(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)$ is the symmetric part of the velocity gradient. The pressure p is determined up to a constant. To make p unique, we impose an additional condition $\int_\Omega p\, dx = 0$. To discretize (3), we use a stabilized finite element method [5] on a given quadrilateral mesh $\mathcal{T}^h = \{K\}$. Let V^h and P^h be a pair of finite element spaces for the velocity and pressure, given by

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

Here, $C^0(\Omega)$, $L^2(\Omega)$, and $H^1(\Omega)$ are the standard notations with usual meanings in the finite element literature [5]. 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

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

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

$$P_0^h = \{p^h \in P^h : \int_\Omega p \, dx = 0\}.$$

Following [5], 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

$$B(\boldsymbol{u}^h, p^h; \boldsymbol{v}, q) = 0 \qquad \forall (\boldsymbol{v}, q) \in V_0^h \times P_0^h \tag{4}$$

with

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

We use the stabilization parameters δ and τ as suggested in [5]. The stabilized finite element formulation (4) can be written as a nonlinear algebraic system

$$F(x) = 0, \tag{5}$$

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 (4).

2.1 Subdomain partition and one-level nonlinear preconditioner

To define parallel Schwarz type preconditioners, we partition the finite element mesh $\mathcal{T}^h$ introduced in the previous section. Let $\{\Omega_i^h, i = 1, ..., N\}$ be a non-overlapping subdomain partition whose union covers the entire domain Ω and its mesh $\mathcal{T}^h$. We use $\mathcal{T}_i^h$ 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 $\mathcal{T}^h$. Similarly, we use $\mathcal{T}_i^{h,\delta}$ to denote the collection of mesh points in $\Omega_i^{h,\delta}$. Now, we define the subdomain velocity space as

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

and the subdomain pressure space as

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

On the physical boundaries, Dirichlet conditions are imposed according to the original equations (3). On the artificial boundaries, both $\boldsymbol{u} = 0$ and $p = 0$.

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 $3n_i \times 3n$ matrix with values of either 0 or 1, where n and n_i are the total number of mesh points in $\mathcal{T}^h$ and $\mathcal{T}_i^{h,\delta}$, respectively, and $\sum_{i=1}^{N} 3n_i \geq 3n$. Note that for $Q_1 - Q_1$ elements, we have three variables per 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 $F_i : R^{3n} \to R^{3n_i}$ 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 $x \in R^{3n}$, $T_i(x) : R^{3n} \to R^{3n_i}$ is defined as the solution of the following subspace nonlinear systems,

$$F_i(x - R_i^T T_i(x)) = 0, \text{ for } = 1, ..., N. \tag{6}$$

Throughout this paper, we always assume that (6) is uniquely solvable. Using the subdomain mapping functions, we introduce a new global nonlinear function,

$$\mathcal{F}^{(1)}(x) = \sum_{i=1}^{N} R_i^T T_i(x), \tag{7}$$

which we refer to as the nonlinearly preconditioned $F(x)$. The one-level additive Schwarz inexact preconditioned Newton algorithm (ASPIN(1)) is defined as: Find the solution x^* of (5) by solving the nonlinearly preconditioned system,

$$\mathcal{F}^{(1)}(x) = 0, \tag{8}$$

using INB with an initial guess $x^{(0)}$. As shown in [2, 6], an approximation of the Jacobian of $\mathcal{F}^{(1)}$ takes the form

$$\widehat{\mathcal{J}}^{(1)}(x) = \sum_{i=1}^{N} J_i^{-1} J(x), \tag{9}$$

where J is the Jacobian of the original function $F(x)$ and $J_i = R_i J {R_i}^T$.

2.2 A parallel linear coarse component for the nonlinear preconditioner

The one-level ASPIN is robust, but not linearly scalable with respect to the number of processors. Some coarse preconditioner is required to couple the subdomain preconditioners. One such coarse preconditioner is proposed and tested

in [3, 7]. The nonlinear coarse system is obtained by the discretization of original nonlinear partial differential equations on a coarse mesh. Although, in general, solving the coarse systems is easier than the fine systems, a Newton-Krylov-Schwarz method sometimes is not good enough to converge the coarse system. Therefore, ASPIN(1) is used to solve the coarse system in [3, 7]. To evaluate the coarse function at certain point, one needs to solve a set of nonlinear systems of equations. Although the ASPIN(1)-based coarse solver provides good mathematical properties, such as helping speed up the convergence of the linear iterative method, the computational cost to solve many coarse systems is usually high in practice. Numerical experiments [7] show that the ASPIN(1)-based coarse solver works fine only for a moderate number of processors, for a large number of processors, a more efficient coarse solver is needed.

Here 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 only requires the solution of a linear system, and hence the computational cost is reduced considerably. More precisely, we assume there exists a finite element mesh $\mathcal{T}^H$ covering the domain Ω. The two meshes $\mathcal{T}^H$ and $\mathcal{T}^h$ do not have to be nested. For the purpose of parallel computing, the coarse mesh is partitioned into non-overlapping subdomains $\{\Omega_i^H\}$ and overlapping subdomains $\{\Omega_i^{H,\delta}\}$. The corresponding sets of mesh points are denoted by $\{\mathcal{T}_i^H\}$, and $\{\mathcal{T}_i^{H,\delta}\}$. For the simplicity of our software implementation, we assume a non-overlapping partition to be *nested*. In other words, we must have

$$\Omega_i^h = \Omega_i^H$$

for $i = 1, \ldots, N$, even though the corresponding sets of mesh points do not have to be nested; i.e.,

$$\mathcal{T}_i^h \neq \mathcal{T}_i^H.$$

This also means that the same number of processors is used for both the fine and coarse mesh problems. If the overlap is taken into account, in general,

$$\Omega_i^{h,\delta} \neq \Omega_i^{H,\delta}, \quad \text{and} \quad \mathcal{T}_i^{h,\delta} \neq \mathcal{T}_i^{H,\delta}.$$

As in the fine mesh case, we can also define the restriction and extension operators R_i^c and $(R_i^c)^T$ for each coarse subdomain. On the coarse mesh $\mathcal{T}^H$, we can define finite element subspaces similar to the ones defined on the fine meshes, and discretize the original Navier-Stokes equations to obtain a nonlinear system of equations,

$$F^c(x_c^*) = 0, \tag{10}$$

where the coarse solution x_c^* of (10) is determined through a pre-processing step. Similar to the fine mesh, on the coarse subdomains, we obtain the coarse Jacobian submatrices

$$J_i^c = (R_i^c) J^c (R_i^c)^T, i = 1, \ldots, N,$$

where J^c is the Jacobian matrix of the coarse mesh function F^c.

We next define the coarse-to-fine and fine-to-coarse mesh transfer operators. Let $\{\phi_j^H(x), j = 1, \ldots, m\}$ be the finite element basis functions on the coarse mesh, where m is the total number of coarse mesh points in $\mathcal{T}^H$. We define an $3n \times 3m$ matrix I_H^h, the coarse-to-fine extension matrix, as

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

where the block matrix E_i of size $3 \times 3m$ is given by

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

and the row vector $(e_H^h)_i$ of length m is given by

$$(e_H^h)_i = \left[\phi_1^H(x_i), \phi_2^H(x_i), \ldots \phi_m^H(x_i)\right],\ x_i \in \mathcal{T}^h$$

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.

To define the coarse function $T_0 : R^{3n} \to R^{3n}$, we introduce a projection $T^c : R^{3n} \to R^{3m}$ as the solution of the linearize coarse system

$$F^c(x_c^*) + J^c(x_c^*)(T^c(x) - x_c^*) = I_h^H F(x), \tag{11}$$

for any given $x \in R^{3n}$. Note that the left hand side of (11) 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 rewrite (11) 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

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

and its derivative is given by

$$\frac{\partial T_0(x)}{\partial x} = I_H^h(J^c(x_c^*))^{-1} I_h^H J(x). \tag{12}$$

We introduce a new nonlinear function

$$\mathcal{F}^{(2)}(x) = T_0(x) + \sum_{i=1}^{N} R_i^T T_i(x),$$

and combining (12) and (9), we obtain an approximation of Jacobian of $\mathcal{F}^{(2)}$ in the form

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

The two-level additive Schwarz preconditioned inexact Newton algorithm with a linear coarse solver (ASPIN(2)) is defined as: Find the solution x^* of (5) by solving the nonlinearly preconditioned system

$$\mathcal{F}^{(2)}(x) = 0, \tag{13}$$

using INB with an initial guess $x^{(0)}$. Details of ASPIN(2) is 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 by the ASPIN(2) algorithm as follows:

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

1. Find $w_0^{(k)}$ by solving the linearize coarse mesh problem

$$J^c(x_c^*)z_c = I_h^H F(x^{(k)}) \tag{14}$$

using a Krylov-Schwarz method with a left preconditioner, $P^{-1} = \sum_{i=1}^{N} (R_i^c)^T (J_i^c)^{-1} R_i^c$ and the initial guess $z_c = 0$.

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

$$G_i(w) \equiv F_i(x_i^{(k)} - w) = 0 \tag{15}$$

using Newton method with backtracking and the initial guess $w = 0$.

3. Form the global residual

$$\mathcal{F}^{(2)}(x^{(k)}) = I_H^h w_0^{(k)} + \sum_{i=1}^{N} R_i^T w_i^{(k)}.$$

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

Step 3: Evaluate pieces of the Jacobian matrix $\mathcal{J}^{(2)}(x)$ of the preconditioned system that are needed in order to multiply (16) below with a vector in the next step. This includes $J(x^{(k)})$ as well as J_i and its sparse LU factorization.

$$\widehat{\mathcal{J}}^{(2)} = \left\{ I_H^h (J^c(x_c^*))^{-1} I_h^H + \sum_{i=1}^{N} \left[R_i^T (J_i(x^{(k)}))^{-1} R_i \right] \right\} J(x^{(k)}). \tag{16}$$

Step 4: Find an inexact Newton direction $s^{(k)}$ by solving the following Jacobian system approximately using a Krylov subspace method

$$\widehat{\mathcal{J}}^{(2)} s^{(k)} = -\mathcal{F}^{(2)}(x^{(k)}) \tag{17}$$

in the sense that

$$||\mathcal{F}^{(2)}(x^{(k)}) + \widehat{\mathcal{J}}^{(2)}(x^{(k)}) s^{(k)}||_2 \leq \eta_k ||\mathcal{F}^{(2)}(x^{(k)})||_2 \tag{18}$$

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

Step 5: Scale the search direction $s^{(k)} \leftarrow \frac{s_{max}}{||s^{(k)}||_2} s^{(k)}$ if $||s^{(k)}||_2 \geq s_{max}$.

Step 6: Compute a new approximate solution

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

where $\lambda^{(k)}$ is determined by the linesearch technique.

Remark 1. No preconditioning is used in Step 4 of ASPIN(2). In fact, $\widehat{\mathcal{J}}^{(2)}$ can be viewed as the original Jacobian system J preconditioned by a two-level additive Schwarz preconditioner, where the coarse preconditioner $I_H^h(J^c(I_H^h x^{(k)} I_h^H)^{-1} I_h^H$ is approximated by $I_H^h(J^c(x_c^*))^{-1} I_h^H$. Hence, $\widehat{\mathcal{J}}^{(2)}$ is well-conditioned through nonlinear preconditioning as long as $I_H^h x^{(k)} I_h^H$ is close to x_c^*.

Remark 2. Although each component of $\widehat{\mathcal{J}}^{(2)}$ is sparse, $\widehat{\mathcal{J}}^{(2)}$ itself is often dense and expensive to form explicitly. However, if a Krylov subspace method is used to the global Jacobian system (17), only the Jacobian-vector product, $u = \widehat{\mathcal{J}}^{(2)} v$, is required. In a distributed-memory parallel implementation, this operation consists of five phrases:

1. Solve $J^c(x_c^*) z_c = I_h^H v$, using a Krylov-Schwarz method with a left preconditioner, $P^{-1} = \sum_{i=1}^N (R_i^c)^T (J_i^c)^{-1} R_i^c$ and the initial guess $z_c = 0$.
2. Perform the matrix-vector multiply, $w = Jv$, in parallel.
3. On each subdomain, collect the data from the subdomain and its neighboring subdomains, $w_i = R_i w$.
4. Solve $J_i u_i = w_i$ using a sparse direct solver.
5. Send the partial solutions to its neighboring subdomain and take the sum, $u = \sum_{i=1}^N R_i^T u_i + I_H^h z_c$.

3 Numerical results

In this section, we consider a two-dimensional lid-driven cavity flow problem. We used PETSc [1] for the parallel implementation and obtained all numerical results on a cluster of workstations. Only machine independent results are reported. In our implementation, after ordering the mesh points, we numbered unknown nodal values in the order of u_1^h, u_2^h, and p^h at each mesh point. The mesh points were grouped subdomain by subdomain for the purpose of parallel processing. Regular checkerboard partitions were used for our experiments. The number of subdomains was always the same as the number of processors, n_p. At the fine mesh level, the linesearch technique [4] was used for both global and local nonlinear problems. The global nonlinear iteration was stopped if the condition $||\mathcal{F}^{(2)}(x^{(k)})||_2 \leq 10^{-6} ||\mathcal{F}^{(2)}(x^{(0)})||_2$ was satisfied, and the local nonlinear iteration on each subdomain was stopped if the condition $||G_i(w_{i,l}^{(k)})||_2 \leq 10^{-4} ||G_i(w_{i,0}^{(k)})||_2$ is satisfied. Restarted GMRES(200) was used for solving the global Jacobian systems (17). The global linear iteration was stopped if the relative tolerance $||\mathcal{F}^{(2)}(x^{(k)}) + \mathcal{J}^{(2)}(x^{(k)}) s^{(k)}||_2 \leq 10^{-6} ||\mathcal{F}^{(2)}(x^{(k)})||_2$ was satisfied. During local nonlinear iterations, a direct sparse solver, LU decomposition,

was employed for solving each local Jacobian system. At the coarse mesh level, restarted GMRES(200) with a left Schwarz preconditioner was used for solving the coarse systems (14). The stopping criterion for the coarse mesh problem was that the condition $||I_h^H F(x^{(k)}) - J^c(x_c^*)z_c||_2 \leq 10^{-10}||I_h^H F(x^{(k)})||_2$ was satisfied. $\delta = 2$ for both the fine and coarse systems. As suggested in [6], we included the re-scaling of the search direction $s^{(k)}$ in Step 5 if $||s^{(k)}||_2 \geq s_{max}$ to enhance the robustness of ASPIN for solving incompressible flows. This step also reduces the number of line search steps, since the evaluation of nonlinearly preconditioned function is expensive. All numerical results reported here are based on the optimal choice of the parameter s_{max}, which results in the smallest number of global nonlinear iterations.

We first study the effect of the coarse mesh size on the global nonlinear iterations and the global linear iterations of ASPIN(2) for different values of Reynolds number. In this set of numerical experiments, all results are obtained using a fixed fine mesh 128×128 on 16 processors, and the coarse mesh size is varied from 16×16 to 80×80. Table 1 shows that to apply two-level methods on a moderate number of processors, the coarse mesh has to be sufficiently fine, say 40×40 in this case. For this particular case, the numbers of global nonlinear iterations, as well as global linear iterations, are not very sensitive with the increase of Reynolds number. To study the parallel scalability of ASPIN(2) with respect to the number of processors, we use a fixed fine mesh 128×128 and a coarse mesh 40×40. For comparison purposes, we also include the results obtained using ASPIN(1). Table 2 shows that by adding a coarse preconditioner, not only the global linear iterations is reduced significantly as we increase the number of processors from 4 to 64, but also the global nonlinear iterations is improved especially for high Reynolds number flows.

Table 1. ASPIN(2): Varying the coarse mesh size for different values of Reynolds number. Fine mesh: 128×128. The number of processors n_p=16.

Coarse meshes	Re=10^3	Re=3×10^3	Re=5×10^3	Re=8×10^3	Re=10^4
	Number of global nonlinear iterations				
16×16	8	11	11	14	17
20×20	9	9	11	13	14
32×32	8	9	11	10	12
40×40	8	9	9	10	11
64×64	8	10	9	11	11
	Average number of global linear iterations				
16×16	58	74	94	111	122
20×20	50	66	75	89	103
32×32	45	52	59	64	68
40×40	43	50	54	60	60
64×64	42	49	52	55	65

Table 2. ASPIN(1) and ASPIN(2): Varying the number of processors. Fine mesh size: 128×128. Coarse mesh size: 40×40.

n_p	Re=10^3	Re=3×10^3	Re=5×10^3	Re=8×10^3	Re=10^4
	Number of global nonlinear iterations				
ASPIN(1)					
$2 \times 2 = 4$	9	10	13	19	19
$4 \times 4 = 16$	9	12	12	16	18
$8 \times 8 = 64$	10	15	14	19	19
ASPIN(2)					
$2 \times 2 = 4$	9	9	11	10	12
$4 \times 4 = 16$	8	9	9	10	11
$8 \times 8 = 64$	8	9	12	12	14
	Average number of global linear iterations				
ASPIN(1)					
$2 \times 2 = 4$	67	69	71	73	74
$4 \times 4 = 16$	127	128	133	137	140
$8 \times 8 = 64$	395	394	400	497	655
ASPIN(2)					
$2 \times 2 = 4$	33	40	40	40	46
$4 \times 4 = 16$	43	50	54	60	60
$8 \times 8 = 64$	49	62	61	78	79

References

1. S. Balay, K. Buschelman, W. D. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang, *Portable, Extensible, Toolkit for Scientific Computation(PETSc) home page*, http://www.mcs.anl.gov/petsc, 2004.
2. X.-C. Cai and D. E. Keyes, *Nonlinearly preconditioned inexact Newton algorithms*, SIAM J. Sci. Comput., 24 (2002), pp. 183-200.
3. X.-C. Cai, D. E. Keyes, and L. Marcinkowski, *Nonlinear additive Schwarz preconditioners and applications in computational fluid dynamics*, Int. J. Numer. Meth. Fluids, 40 (2002), pp. 1463-1470.
4. J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.
5. L. P. Franca and S. L. Frey, *Stabilized finite element method: II. The incompressible Navier-Stokes equation*, Comput. Methods Appl. Mech. Engrg., 99 (1992), pp. 209-233.
6. F.-N. Hwang and X.-C. Cai, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., (2004), to appear.
7. L. Marcinkowski and X.-C. Cai, *Parallel performance of some two-level ASPIN algorithms*, Lecture Notes in Computational Science and Engineering, ed. R. Kornhuber, R. H. W. Hoppe, D. E. Keyes, J. Periaux, O. Pironneau and J. Xu, Springer-Verlag, Haidelberg, pp. 639-646.
8. J. N. Shadid, R. S. Tuminaro, and 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), pp. 155-185.