PARALLEL FULLY COUPLED SCHWARZ PRECONDITIONERS FOR SADDLE POINT PROBLEMS*

FENG-NAN HWANG† AND XIAO-CHUAN CAI‡

Abstract. We study some parallel overlapping Schwarz preconditioners for solving Stokes-like problems arising from finite element discretization of incompressible flow problems. Most of the existing methods are based on the splitting of the velocity and pressure variables. With the splitting, fast solution methods are often constructed using various fast Poisson solvers for one of the variables. More recently, several papers have investigated the so-called fully coupled approaches in which the variables are not separated. The fully coupled approach has some advantages over the variable splitting method when solving Stokes-like equations with many variables, where a good splitting may be hard to obtain. In this paper we systematically study the parallel scalability of several versions of the fully coupled Schwarz method for both symmetric and nonsymmetric Stokes-like problems. We show numerically that the performance of a two-level method with a multiplicative iterative coarse solver is superior to the other variants of Schwarz preconditioners.

Key words. saddle point problem, two-level Schwarz preconditioning, fully coupled methods, finite element, parallel processing

AMS subject classifications. 65F10, 65N30, 65N55

1. Introduction. We study some parallel fully coupled Schwarz preconditioned iterative methods [36, 42, 43] for solving saddle point problems, which appear in many areas of computational science and engineering [3, 4, 21, 32, 35, 40, 44]. Due to the indefiniteness of saddle point problems the convergence of most iterative methods is often not guaranteed, and when they converge, the convergence rates of the methods are sometimes too slow to be considered practical, especially for large-scale applications. To resolve the situation, finding a good preconditioner is critical. In this paper, we focus on two types of saddle point problems, namely the symmetric Stokes problem discretized with a stabilized finite element method and the more general, nonsymmetric Jacobian system arising in the Newton-Krylov-Schwarz (NKS) algorithm for solving nonlinear incompressible Navier-Stokes equations. In NKS, the Jacobian system has to be solved as a part of the nonlinear iteration, and is often the most expensive part [17, 18, 23, 24, 25, 30]. The performance of Schwarz methods depends on several important parameters, including the overlapping size, the quality of the coarse and subdomain solutions, the coarse mesh size, as well as some physical parameters of the continuous equations. We perform a large number of numerical experiments to understand how these parameters affect the overall parallel performance of Schwarz methods for both symmetric and nonsymmetric saddle point problems. Our investigation is purely numerical, and to the best of our knowledge, the corresponding optimal convergence theory for Stokes-like problems is yet to be established.

Many iterative methods for solving saddle point problems are available, such as Uzawa’s algorithms and their variants [7, 15, 33, 45], multigrid methods [5, 16, 23, 24, 25], Krylov subspace methods with block-type preconditioners [15, 17, 26, 27, 29, 41], and domain decomposition methods [6, 28, 29, 30, 34, 45]. Most of the existing work explicitly uses the
block structure of the discrete problem, 
\[
\begin{pmatrix}
C & B^T \\
B & \alpha M_p
\end{pmatrix}
\begin{pmatrix}
u \\
p
\end{pmatrix}
=
\begin{pmatrix}
f \\
g
\end{pmatrix},
\]
where $C$ is a symmetric positive definite matrix corresponding to a diffusion operator for the Stokes problem, $B$ is a matrix corresponding to a discrete divergence operator, and $M_p$ is a mass pressure matrix with a stabilization parameter $\alpha$. The block structure is obtained by using a variable splitting-based ordering of the unknowns. First, the velocity variables for all mesh points are ordered, followed by the pressure variables for all mesh points. Even though the decoupled block structure is non-physical, i.e. it splits the variables, velocity and pressure, which are supposedly to be physically coupled at each mesh point, such algebraic structure is very useful in the design and analysis of algorithms. The popular pressure projection methods, such as the SIMPLE and SIMPLER algorithms [33], also known as Uzawa’s algorithms, are typical examples of the fully decoupled solution strategy. These methods decouple the discrete system (1.1) into two subsystems, the Schur complement system for the pressure and the Laplacian system for the velocity; then Uzawa’s algorithm can be constructed by solving these two subsystems iteratively. Also based on the block structure of (1.1), the block diagonal preconditioner is constructed by dropping both off-diagonal blocks [16, 18, 27, 29] and the block-triangular preconditioner is obtained by replacing the lower off-diagonal block with a zero block [17, 26, 29, 41].

In contrast to the fully decoupled approaches, the Schwarz preconditioned iterative methods are considered to be the fully coupled approaches in which the velocity and pressure variables are always coupled together throughout the computation. Without being restricted to the block structure, the fully coupled methods can be applied easily to other multi-component indefinite problems, such as the flow control problem [4, 21, 35], which is one of the target applications of our algorithms and software. Furthermore, it is also possible to generalize these preconditioners for the purpose of nonlinear preconditioning, similar to the additive Schwarz preconditioned inexact Newton algorithms [8, 9, 22]. The idea of the fully coupled methods for saddle point problems is not new, and the methods have been studied in the context of multigrid methods [5]. According to the multigrid results [23, 24, 25], the fully coupled methods seem to be a better choice for saddle point problems over the decoupled methods.

In this paper, we consider three types of Schwarz preconditioners. We begin by studying the one-level additive Schwarz preconditioner [28], which is defined by summing up the solutions of the Stokes problem defined on overlapping subdomains with certain artificial boundary conditions. The main advantage of the one-level Schwarz preconditioner is that all subproblems are independent of each other and can be solved in parallel. However, as expected, its convergence rate, in terms of the number of iterations, degenerates when the number of processors is large due to the lack of communication between subdomains. For a fully scalable method, a coarse mesh problem is required. Based on sequential numerical results [28], Klawonn and Pavarino showed that the number of GMRES [39] iterations for the two-level additive Schwarz methods for symmetric indefinite problems (Stokes and linear elasticity) with minimal overlap using exact subdomain and coarse solvers is independent of the mesh size and the number of subdomains, provided that the coarse mesh is sufficiently fine. However, the parallel performance of Schwarz methods for saddle point problems was not studied until the work in [44]. For the Stokes problem on unstructured meshes, Tuminaro et al. compared one-level methods with two-level methods using an inexact LU as the subdomain solver and SuperLU [13], a parallel direct LU solver, as the coarse solver. Their numerical experiments showed that although the two-level method seems to be scalable
terms of the number of GMRES iterations, the computing time increases more than 200% from 64 to 256 processors. A similar problem also arises in the case of nonsymmetric indefinite problems (a thermal convection problem) solved by a GMRES-multigrid method using two meshes, in which Gauss-Seidel is used as the smoother and SuperLU is used as the coarse solver. One solution to the problem of high computing time is to replace SuperLU with a less expensive inexact solver, such as the one-step Schwarz-Richardson method as in [44]. Indeed, total computing time can be saved to a certain extent, but both the number of GMRES iterations and the computing time grow as the number of processors is increased. Therefore, it seems to us that these two coarse solvers are either too expensive or too inexact to be effective. In this paper, we propose and test a parallel preconditioned iterative coarse solver so that the cost of the coarse solve can be controlled easily by adjusting the coarse stopping condition. The preconditioner for the coarse mesh problem is the same as the one-level additive Schwarz preconditioner, except that it is constructed on a partitioned coarse mesh. For the two-level methods, the coarse mesh part of the preconditioner can be incorporated into its local subdomain part either additively or multiplicatively. All components of the two-level method are parallel, including the coarse solver, and the restriction and interpolation operators. We show, through parallel numerical experiments, that the performance of the two-level method with a multiplicative iterative coarse solver is superior to the other two variants of Schwarz preconditioners.

The paper is organized as follows. In Section 2, we describe the Stokes and Jacobian problems arising from finite element discretization of incompressible flow problems. We introduce three types of Schwarz preconditioners in Section 3. Then, in Section 4, we present some numerical results obtained on a parallel computer. Finally, conclusions and some remarks are given in Section 5.

2. Model problems and their stabilized finite element formulations. We first consider the two-dimensional steady-state Stokes problem [20, 37], which can be described as

\[
\begin{aligned}
-\Delta u + \nabla p &= f \quad \text{in} \quad \Omega, \\
\nabla \cdot u &= 0 \quad \text{in} \quad \Omega, \\
\mathbf{u} &= g \quad \text{on} \quad \Gamma,
\end{aligned}
\]

where \( \mathbf{u} = (u_1, u_2)^T \) is the velocity, \( p \) is the pressure, and \( \mathbf{f} = (f_1, f_2)^T \) is the body force. Here we assume that \( \Omega \) is a bounded domain in \( \mathbb{R}^2 \) with a polygonal boundary \( \Gamma \). Since only Dirichlet boundary condition is considered, the pressure \( p \) is determined up to a constant. To make \( p \) unique, we impose the condition \( \int_{\Omega} p \, dx = 0 \).

To discretize (2.1), we use a stabilized finite element method on a given conforming quadrilateral mesh \( \mathcal{T}^h = \{K\} \) [20]. For each element \( K \), we denote by \( h_K \) as its diameter. Let \( V^h \) and \( P^h \) be a pair of finite element spaces for the velocity and pressure given by

\[
V^h = \{ v \in (C^0(\Omega) \cap H^1(\Omega))^2 : v|_K \in Q_1(K)^2, K \in \mathcal{T}^h \}
\]

and

\[
P^h = \{ p \in C^0(\Omega) \cap L^2(\Omega) : p|_K \in Q_1(K), K \in \mathcal{T}^h \}.
\]

Here, \( C^0(\Omega), L^2(\Omega), \) and \( H^1(\Omega) \) are the standard notations with usual meanings in the finite element literature [20]. For simplicity, our implementation uses a \( Q_1 - Q_1 \) finite element (continuous bilinear velocity and pressure). The weighting and trial velocity function spaces \( V_0^h \) and \( V_g^h \) are defined as

\[
V_0^h = \{ v \in V^h : v = 0 \text{ on } \Gamma \} \quad \text{and} \quad V_g^h = \{ v \in V^h : v = g \text{ on } \Gamma \}.
\]
Similarly, let the finite element space $P_0^h$ be both the weighting and trial pressure function spaces:

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

As suggested by [14], the stabilized finite element method for the steady-state Stokes problem reads: Find $u^h \in V_g^h$ and $p^h \in P_0^h$, such that

\begin{equation}
B(u^h, p^h; v, q) = F(v, q) \quad \forall (v, q) \in V_0^h \times P_0^h
\end{equation}

with

$$B(u, p; v, q) = (\nabla u, \nabla v) - (\nabla \cdot v, p) - (\nabla \cdot u, q) - \alpha \sum_{K \in T^h} h_K^2 (-\Delta u + \nabla p, \Delta u + \nabla q)_K$$

and

$$F(v, q) = (f, v) - \alpha \sum_{K \in T^h} h_K^2 (f, \Delta v + \nabla q)_K.$$ 

In [14], Douglas and Wang showed that this method is stable and has the optimal convergence for any choices of positive stabilization parameter $\alpha$. We use a constant of $\alpha = 1$ throughout this paper. The equivalent matrix form of (2.2) can be written as

\begin{equation}
A x = b,
\end{equation}

where $A$ is a large, sparse, symmetric indefinite matrix, and $x$ is a vector corresponding to the nodal values of $u^h = (u^h_1, u^h_2)$ and $p^h$ in (2.2). In our implementation, we number the unknown nodal values in the order of $u^h_1, u^h_2$, and $p^h$ at each mesh point. The mesh points are grouped subdomain by subdomain for the purpose of parallel processing. The subdomain partitioning will be discussed further in the next section. Note that the linear system (2.3) can be written explicitly in the same block-structure form as in (1.1) by rearrangement of the unknown variables in the order of $u^h_1, u^h_2, p^h$. We solve the linear systems (2.3) by a right preconditioned Krylov subspace method, i.e.

\begin{equation}
A M^{-1} y = b, \text{ with } x = M^{-1} y,
\end{equation}

where $M^{-1}$ is called a right preconditioner to be defined in the next section.

In addition to the Stokes problem, we also consider some nonsymmetric indefinite Jacobian systems that we have to solve in order to obtain search directions when using NKS for solving incompressible Navier-Stokes equations. Briefly speaking, NKS is a general framework for solving a large, sparse, nonlinear system of equations, $F(x^*) = 0$, arising from a discretization of nonlinear partial differential equations. As its name suggests, NKS consists of three main components: a Newton-type method as the nonlinear solver, a Krylov subspace method as the linear solver, and a Schwarz-type method as the preconditioner. To be more specific, NKS can be described as follows. Let $x^{(0)}$ be a given initial guess. Assume $x^{(k)}$ is the current approximation of $x^*$. Then a new approximation $x^{(k+1)}$ can be computed by first finding a Newton direction $s^{(k)}$ by solving the following preconditioned Jacobian system approximately,

\begin{equation}
J(x^{(k)}) M_k^{-1} y = -F(x^{(k)}), \text{ with } s^{(k)} = M_k^{-1} y,
\end{equation}
then obtaining the new approximation via \( x^{(k+1)} = x^{(k)} + \lambda^{(k)} g^{(k)} \), where \( J(x) \) is the Jacobian of \( F \) evaluated at \( x^{(k)} \) and \( \lambda^{(k)} \in [0, 1] \) is a damping parameter. In our case, the nonlinear system, \( F(x) = 0 \), is obtained by applying \( Q_1 - Q_1 \) Galerkin least square (GLS) finite element discretization [37] on the mesh \( T^h \) and the corresponding Jacobian \( J \) is constructed by using a multi-colored forward finite difference method [12]. Figure 2.1 is an example showing that the discrete Stokes problem in (2.3) and the Jacobian matrix in (2.5) have similar nonzero structures. The matrix for the discrete Stokes problem has about 3/4 the number of nonzero elements as in the Jacobian matrix. The additional nonzero elements in the Jacobian matrix come from mainly the derivatives of the convection and stabilization terms in GLS. For incompressible Navier-Stokes equations, the Reynolds number (\( Re \)) plays an important role in the behavior of the numerical solution and the convergence of iterative methods. Several numerical difficulties arise due to high \( Re \). It is well known that the convergence radius for a Newton-type method becomes smaller as \( Re \) increases. Newton-type methods often fail to converge when a good initial guess is unavailable. In this paper we will not address the issues related to the robustness of Newton-type methods; interested readers may consult [8, 9, 22, 40]. Instead, we restrict our study to the parallel performance of Schwarz preconditioned Krylov subspace methods for solving the Jacobian systems. Note that the Jacobian systems become more nonsymmetric and ill-conditioned as \( Re \) increases.

3. Fully coupled Schwarz methods. In this section, we introduce several Schwarz preconditioners for the Stokes and Jacobian problems.

3.1. One-level additive Schwarz preconditioner. To define parallel Schwarz type preconditioners we need to partition the finite element mesh \( T^h \) introduced in the previous section. Let \( \{ \Omega^h_i, i = 1, ..., N \} \) be a non-overlapping subdomain partition whose union covers the entire domain \( \Omega \) and its mesh \( T^h \). We use \( T^h_i \) to denote the collection of mesh points in \( \Omega^h_i \). To obtain overlapping subdomains, we expand each subdomain \( \Omega^h_i \) to a larger subdomain \( \Omega^h_i,\delta \) with the boundary \( \partial \Omega^h_i,\delta \). Here \( \delta \) is an integer indicating the level of overlap. We assume that neither \( \Omega^h_i \) nor \( \partial \Omega^h_i,\delta \) cut any elements of \( T^h \). Similarly, we use \( T^h_i,\delta \) to denote the collection of mesh points in \( \Omega^h_i,\delta \).
Now, we define the subdomain velocity space as
\[ V_i^h = \{ v^h \in V^h \cap (H^1(\Omega_i^{h,\delta}))^2 : v^h = 0 \text{ on } \partial \Omega_i^{h,\delta} \} \]
and the subdomain pressure space as
\[ P_i^h = \{ p^h \in P^h \cap L^2(\Omega_i^{h,\delta}) : p^h = 0 \text{ on } \partial \Omega_i^{h,\delta} \setminus \Gamma \}. \]
Both are subspaces of \( V^h \) and \( P^h \), respectively, if all subdomain functions are extended to the whole domain by zero. Note that for \( Q_1 \) - \( Q_1 \) elements, each interior node has three degrees of freedom, two for the velocity and one for the pressure. On the physical boundaries, we impose Dirichlet conditions according to the original equations (2.1). On the artificial boundaries, we assume both \( u = 0 \) and \( p = 0 \). Similar boundary conditions are used in [29]. Although it is still not certain that these boundary conditions are mathematically correct, they work well in practice. The subdomain problem reads as follows: Find \( u_i^h \in V_i^h \) and \( p_i^h \in P_i^h \), such that
\[
(3.1) \quad B(u_i^h, p_i^h; v, q) = F(v, q) \quad \forall (v, q) \in V_i^h \times P_i^h,
\]
which takes the matrix form
\[ A_i x_i = b_i. \]
Note that the right-hand side of (3.1) is not important, since we only use the left-hand side matrix \( A_i \) to define the subdomain preconditioner. The right-hand side is not used at all in the computation.

Let \( R_i : V^h \times P^h \rightarrow V_i^h \times P_i^h \) 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 \((3n_i - 2d_i) \times (3n - 2d)\) matrix with values of either 0 or 1, where \( n \) and \( n_i \) are the total number of mesh points in \( T^h \) and \( T_i^{h,\delta} \), respectively. Similarly, \( r \) and \( r_i \) are total number of mesh points imposed the Dirichlet boundary conditions for velocity in \( T^h \) and \( T_i^{h,\delta} \), respectively. Since our subdomain partition is element-based, \( \sum_{i=1}^N (3n_i - 2r_i) \geq 3n - 2r \). 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 operations, but does involve communication in a distributed memory implementation. Using the restriction operator, we write the one-level additive Schwarz preconditioner (ASM1) in the matrix form as
\[ P^{-1}_{ASM1} = \sum_{i=1}^N R_i^T A_i^{-1} R_i, \]
where \( A_i^{-1} \) is the subspace inverse of \( A_i \). In practice, it is more convenient to obtain \( A_i \) from \( A_i = R_i A_i^T R_i \). We remark that the global-to-local restriction operator \( R_i \) collects the data from neighboring subdomains, while the local-to-global prolongation operator \( R_i^T \) sends partial solutions to neighboring subdomains.

### 3.2. Two-level methods with a parallel coarse preconditioner.

The direct coarse preconditioner proposed in [28] provides good mathematical properties, but is not easy to parallelize. Here we propose a parallel coarse solver, which is the same as the one-level additive Schwarz preconditioned Krylov subspace method discussed in the previous subsection, except that it is constructed on a partitioned coarse mesh. More precisely, we assume there exists a finite element mesh \( T^H \) covering the domain \( \Omega \). The two meshes \( T^H \) and \( T^h \) do not
have to be nested. The coarse mesh problem will help speed up the convergence of the iterative method, but has nothing to do with the accuracy of the discretization, which is determined by the fine mesh $\mathcal{T}^h$ only.

For the purpose of parallel computing, the coarse mesh is partitioned to 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 that we have a nested non-overlapping partition. In other words, we 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.

$$\mathcal{T}_i^H \not\subset \mathcal{T}_i^h.$$

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,

$$\Omega_i^{H,\delta} \neq \Omega_i^{H,\delta} \text{ and } \mathcal{T}_i^{H,\delta} \not\subset \mathcal{T}_i^{h,\delta}.$$

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 mesh, and discretize the original Stokes problem to obtain a linear system of equations

$$A^c x^c = b^c.$$

Note that the vectors $b^c$ and $x^c$ are not used in the computation; only the matrix $A^c$ is used to define our coarse preconditioner. Following a similar argument, on the coarse subdomains, we obtain the coarse submatrices

$$A_i^c, i = 1, \ldots, N.$$

As opposed to strongly elliptic problems, our experiments show that the coarse mesh size for indefinite problems needs to be sufficiently fine to retain the scalability of the algorithm. (see also [11]). Hence, the parallelization of a coarse solver is necessary, and, in general, there are three strategies: (1) A direct exact approach, which performs an LU decomposition of $A^c$ in parallel and then does the forward/backward substitutions using some parallel sparse direct solvers, such as SuperLU_DIST [13]. (2) A direct inexact approach, which replaces $(A_i^c)^{-1}$ by $\sum_{i=1}^N (R_i^c)^T (B_i^c)^{-1} R_i^c$, where $B_i^c$ could be an ILU(k) decomposition of $A_i^c$. (3) An iterative approach, which solves the coarse mesh problem, $A^c x^c = w^c$, by some parallel iterative methods, such as parallel GMRES, with a coarse additive Schwarz preconditioner.

Aitbayev et al. [1] and Tuminaro et al. [44] have studied the performance of some direct exact coarse solvers based on the first approach and inexact domain decomposition coarse solvers based on the second approach. However, they are either too expensive or too inexact to be effective. Instead, in this paper we adopt the third approach, which is something in between the first and the second approaches, and the solution accuracy of the coarse mesh problem can be controlled easily. Now we discuss the parallel iterative coarse solver and the operators that transfer the data between the fine and coarse meshes. We define the coarse preconditioner $(M^c)^{-1}$ in terms of a matrix-vector multiply: For any given coarse mesh vector $w^c$,

$$z^c = (M^c)^{-1} w^c.$$
is understood as an approximate solution of the following preconditioned linear system of equations

$$\begin{align*}
A^c \left[ \sum_{i=1}^{N} (R_i^c)^T (A_i^c)^{-1} R_i^c \right] y^c &= w^c, \\
\sum_{i=1}^{N} (R_i^c)^T (A_i^c)^{-1} R_i^c ) y^c, \\
\end{align*}$$

which satisfies the condition

$$\|w^c - A^c z^c\|_2 \leq \varepsilon_c \|w^c\|_2.$$ 

Note that for any given $w^c$, the computation of $z^c$ can be carried out in parallel using all $N$ processors. $M^c$ is not exactly a matrix, but a preconditioned iterative solver.

We define the coarse-to-fine and fine-to-coarse mesh transfer operators as follows. 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 $T^H$. Let $s$ be the total number of coarse mesh points at which the Dirichlet boundary condition for velocity in $T^H$ is imposed. We define an $(3n - 2r) \times (3m - 2s)$ matrix $I_h^H$, the coarse-to-fine extension operator, 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

$$E_i = \begin{bmatrix}
(e^h_i)_i & 0 & 0 \\
0 & (e^h_i)_i & 0 \\
0 & 0 & (e^h_i)_i
\end{bmatrix},$$

and the row vector $(e^h_i)_i$ of length $m$ is given by

$$(e^h_i)_i = [\phi_1^H(x_i), \phi_2^H(x_i), \ldots \phi_m^H(x_i)], \quad x_i \in T^h$$

for $i = 1, \ldots, n$. A global fine-to-coarse restriction operator $I_h^H$ can be defined as the transpose of $I_h^H$.

Using the coarse preconditioner defined above, we define a two-level additive Schwarz preconditioner (ASM2)

$$P^{-1}_{ASM2} = I_h^H (M^c)^{-1} I_h^H + \sum_{i=1}^{N} R_i^T A_i^{-1} R_i.$$ 

Because the partitioned coarse mesh problem and the partitioned fine mesh problem are allocated to the same processor, the coarse and fine subproblems cannot be solved at the same time. Therefore it is reasonable to consider a multiplicative approach for the coarse mesh problem in order to obtain a faster convergence. Following [31], we define the following hybrid Schwarz method (HSM),

$$P^{-1}_{HSM} = I_h^H (M^c)^{-1} I_h^H + (I - I_h^H (M^c)^{-1} I_h^H A) \left( \sum_{i=1}^{N} R_i^T A_i^{-1} R_i \right),$$ 

which is additive among all fine mesh subdomains, and multiplicative between the fine and coarse preconditioners.

We remark that once the overlapping subdomains and the mesh transfer operators are defined on both coarse and fine meshes, the three parallel overlapping Schwarz preconditioners
just described are easily adapted for solving nonsymmetric Jacobian systems. The only issue now is, how to define, for each Newton iteration, the subdomain problems on both fine and coarse meshes and the coarse mesh problem. We define the matrices associated with the fine mesh subdomain problems and coarse mesh problem in a simple way. For the subdomain problems, we define $J_i$ by $J_i = R_i J(z^{(k)}) R_i^T$. For the coarse problem, on the coarse mesh $\mathcal{T}_H$, we first discretize the original Navier-Stokes equations to obtain a nonlinear system of equations, $F^c(x^c) = 0$. Then the corresponding coarse Jacobian $J^c(x)$ is computed using multi-colored finite differences. Hence at the $i$-th Newton iteration, the coarse mesh matrix takes the form $J^c(\Delta x^c(k))$, where $\Delta x^c$ is the standard injection operator, which restricts the solution from the fine mesh to the coarse mesh. Similar to the fine mesh, on the coarse subdomains, we define the coarse Jacobian submatrices as $J_i^c = (R_i^c) J^c(\Delta x^c(k)) (R_i^c)^T$.

4. Numerical experiments.
In this section, we consider a two-dimensional lid-driven cavity flow problem as a benchmark for evaluating the parallel performance of the three preconditioners introduced in the previous section. A detailed description of the lid-driven cavity problem can be found in [19]. Several parameters in the Schwarz methods need to be specified for achieving optimal performance, in this paper, we focus on the impact of the following parameters on the convergence rate and the overall execution time: (1) the subdomain overlapping size; (2) the subdomain and coarse mesh solution quality; and (3) the coarse mesh size. We also investigate the parallel scalability, which shows how the algorithm behaves as the size of fine mesh and the number of processors grow. The Portable, Extensible Toolkit for Scientific computation (PETSc) [2] is used for the parallel implementation and all numerical results are obtained on a cluster of PCs running Linux. All timings are reported in seconds, and the execution time excludes the time spent on preprocessing steps including the setup of the problem matrix, the construction of the extension matrix, and the decomposition of the meshes. The matrix decomposition is included in the timing. The parameters and other details of the numerical experiments for the Stokes problem are summarized below:

- For the one-level Schwarz method, restarted GMRES(100) is employed for solving the preconditioned linear system (2.4). For the two-level Schwarz methods, including ASM2 and HSM, Flexible GMRES (FGMRES) [38] is applied for solving the preconditioned system while restarted GMRES(100) is used for the coarse mesh problem (3.2). It should be noted that according to our numerical experiences, when standard GMRES was used instead of FGMRES for the two-level Schwarz methods, there was no convergence problem. However, we did observe some loss of solution accuracy if the coarse mesh problem is not solved accurately enough under the same outer stopping criterion. So due to the changes of two-level Schwarz preconditioners with the inexact coarse solver at each of the outer iterations, FGMRES is more suitable than the standard GMRES. We use a zero initial guess for both problems. The stopping criterion for (2.4) is that the condition

$$\left\| b - (AM_k^{-1})(M_k x^{(k)}) \right\|_2 \leq \epsilon_F \|b\|_2$$

is satisfied and we set $\epsilon_F = 10^{-5}$ for all test cases. Similarly, the stopping criterion for the coarse mesh problem (3.2) is

$$\left\| w^c - A^c z^{(k)} \right\|_2 \leq \epsilon_C \|w^c\|_2.$$

Several values of $\epsilon_C$ are tested, ranging from $10^{-1}$ to $10^{-10}$.

- Three uniform checkerboard subdomain partitions are used for our experiments: $2 \times 2$, $4 \times 4$, and $8 \times 8$. The number of subdomains is always the same as the number of processors, $n_p$. 


• Three fine meshes are considered: $128 \times 128$, $256 \times 256$, and $512 \times 512$ and the number of unknowns ranges from 50,000 to 780,000. The coarse mesh is varied from $16 \times 16$ to $80 \times 80$. Since a non-nested coarse mesh is used, the number of processors and the coarse mesh size are not related.

• The overlapping size for the fine mesh is defined as

$$\text{ovlp} = \max \left\{ \frac{L_x'}{2h_K}, \frac{L_y'}{2h_K} \right\}$$

for both interior subdomains and those touching the boundary. Since square elements are used for the test problem, the elemental diameter $h_K$'s are the same and equal to the fine mesh size. $L_x'$ and $L_y'$ are defined here as the side lengths of the overlapping subdomain $\Omega_{x,\delta}^h$ in the $x$-direction and the $y$-direction, respectively. Similarly, $L_x$ and $L_y$ are defined as the side lengths of the non-overlapping subdomain $\Omega_x^h$ in the $x$-direction and the $y$-direction, respectively. We defined the overlapping size $\text{ovlp}^H$ for the coarse mesh problem in the same fashion as above, and used the value $\text{ovlp}^H = 1$ for all test cases.

• Either a direct sparse solver or an inexact LU($k$) solver, with level of fill-in $k = 0, 1, 2, \ldots, 5$, is employed to solve the subdomain problems.

4.1. The effect of the coarse mesh size and inexact coarse solvers. In Table 4.1, we examine the effect of the coarse mesh size on the convergence and execution time of the two-level Schwarz methods. In this set of numerical experiments, we keep the subdomain size fixed and scale up $\sqrt{n_{\text{proc}}}$ and the fine mesh size $h$ by a factor of 64. All subdomain problems are solved by an exact sparse LU decomposition and $\text{ovlp} = 1$. We vary the coarse mesh size from $16 \times 16$ to $80 \times 80$. The tolerance for the coarse iterative solver is $\varepsilon_C = 10^{-10}$. The empty entry (−) in the table indicates that a uniform partitioning for such coarse mesh size is not available. In each row of Table 4.1, for both two-level Schwarz methods, the number of FGMRES iterations is reduced monotonically when the coarse mesh size is increased. However, the smallest number of iterations does not imply the fastest convergence. The two-level methods with a $80 \times 80$ coarse problem are always slower than with a $16 \times 16$ coarse problem, although they only take fewer iterations. This is because solving the larger coarse mesh problem takes a more significant portion of the total time. The optimal coarse mesh size for both methods, based on the optimal execution time, depends on the fine mesh size. Roughly $H \sim 10h$ is needed to achieve the fastest convergence.

Next, we relax the requirement of solution accuracy for the coarse problem, and vary $\varepsilon_C$ from $10^{-1}$ to $10^{-10}$. The ratio of the coarse to fine mesh sizes, $H/h$, is set to be 8. As shown in Table 4.2, the accuracy of the coarse mesh solution does not need to be very high in order to retain the optimal convergence rate of the two-level Schwarz preconditioners. $\varepsilon_C = 10^{-2}$ is sufficient in both cases. Surprisingly, previous work has shown that this is not true for other types of problems such as indefinite elliptic problems [11]. In [11], a theory for ASM2 requires an exact solver on the coarse mesh. Furthermore, the inexact coarse solver saves a significant amount of time in both two-level Schwarz methods, especially when the number of processors is large. In the 64-processor case, the inexact coarse solver with $\varepsilon_C = 10^{-2}$ takes only about half of the time needed for the almost exact solver with $\varepsilon_C = 10^{-10}$.

4.2. The effect of inexact subdomain solvers. We investigate the effect of using ILU($k$) as inexact subdomain solvers. In Table 4.3 we report computational results with varying levels of fill-in in ILU($k$) for a fixed $64 \times 64$ subdomain problem and increasing the number of processors and the fine mesh size. We use a fixed coarse to fine mesh size ratio of $H/h = 8$ for the two-level Schwarz preconditioners. The tolerance for the iterative coarse
Varying the coarse mesh size. Fixed subdomain mesh size $64 \times 64$. All subdomain problems are solved by LU, $\omega = 1$. The tolerance $\varepsilon_C$ for the coarse mesh problem is set to be $10^{-10}$.

<table>
<thead>
<tr>
<th>Fine mesh $(n_p)$</th>
<th>Coarse mesh</th>
<th>$16 \times 16$</th>
<th>$20 \times 20$</th>
<th>$32 \times 32$</th>
<th>$40 \times 40$</th>
<th>$64 \times 64$</th>
<th>$80 \times 80$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM2</td>
<td>FGMRES</td>
<td>25</td>
<td>23</td>
<td>19</td>
<td>17</td>
<td>15</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>2.3</td>
<td>2.4</td>
<td>2.8</td>
<td>3.5</td>
<td>6.4</td>
<td>8.9</td>
</tr>
<tr>
<td>128 $\times$ 128 (4)</td>
<td>FGMRES</td>
<td>33</td>
<td>30</td>
<td>25</td>
<td>22</td>
<td>19</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>3.0</td>
<td>2.9</td>
<td>2.9</td>
<td>3.1</td>
<td>4.7</td>
<td>6.9</td>
</tr>
<tr>
<td>256 $\times$ 256 (16)</td>
<td>FGMRES</td>
<td>59</td>
<td>–</td>
<td>40</td>
<td>35</td>
<td>27</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>6.5</td>
<td>–</td>
<td>5.4</td>
<td>5.2</td>
<td>6.0</td>
<td>7.8</td>
</tr>
</tbody>
</table>

Varying the tolerance $\varepsilon_C$ for the coarse mesh problem. Fixed subdomain problem size: $64 \times 64$. The ratio of coarse to fine mesh sizes: $H/h = 8$. All subdomain problems are solved by exact LU, $\omega = 1$.

<table>
<thead>
<tr>
<th>Fine mesh $(n_p)$</th>
<th>$\varepsilon_C$</th>
<th>$10^{-1}$</th>
<th>$10^{-2}$</th>
<th>$10^{-3}$</th>
<th>$10^{-4}$</th>
<th>$10^{-5}$</th>
<th>$10^{-6}$</th>
<th>$10^{-7}$</th>
<th>$10^{-10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM2</td>
<td>FGMRES</td>
<td>19</td>
<td>19</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>1.9</td>
<td>2.1</td>
<td>2.1</td>
<td>3.0</td>
<td>5.0</td>
<td>6.3</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>128 $\times$ 128 (4)</td>
<td>FGMRES</td>
<td>24</td>
<td>21</td>
<td>17</td>
<td>15</td>
<td>11</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>2.5</td>
<td>2.5</td>
<td>2.4</td>
<td>2.5</td>
<td>3.3</td>
<td>4.6</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>256 $\times$ 256 (16)</td>
<td>FGMRES</td>
<td>59</td>
<td>–</td>
<td>27</td>
<td>24</td>
<td>18</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>Time (sec)</td>
<td>4.7</td>
<td>–</td>
<td>4.0</td>
<td>4.0</td>
<td>4.4</td>
<td>5.4</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>

solver is set to be $\varepsilon_C = 10^{-2}$ and a fixed $\omega = 1$ is used. We vary the level of fill-ins, $k$, from 0 to 5. As shown in Table 4.3, we observe that the number of iterations for ASM1 is more sensitive to the change of $k$ in ILU($k$) and the number of processors (or the fine mesh size) than ASM2 and HSM: the convergence rate of ASM1 can be improved dramatically by increasing $k$, especially when the number of processors is large. On the other hand, for ASM2 and HSM with ILU($k$), $k$ from 0 to 3, the numbers of iterations are nearly independent of the numbers of processors. About the timing, we observe that ASM1 with ILU($k$) is faster than ASM1 with LU only in the 4-processor case. ASM1 with ILU($k$) becomes less efficient as the number of processors increases due to the exceedingly large number of iterations required for convergence. For ASM2 and HSM, the results with optimal execution time are obtained by using $k = 0$ and $k = 1$, respectively, and the time savings are remarkable for both cases compared to the cases using LU as a subdomain solver: ILU($k$) is more than 55% faster than LU.

4.3. The effect of the overlapping size. Table 4.4 shows the effect of the overlapping size for the two-level Schwarz methods with 64 processors. The overlapping size varies from 1 to 16. A fixed $512 \times 512$ fine mesh and a fixed $64 \times 64$ coarse mesh are used. In
FULLY COUPLED SCHWARZ PRECONDITIONERS

4.3. Fully coupled Schwarz preconditioners

Table 4.3

Varying the level of ILU(k) fill-in. Fixed subdomain mesh size: 64 x 64. The ratio of coarse to fine mesh sizes is fixed at H/h = 8. The tolerance $\epsilon_C$ for the iterative coarse solver is set to be $10^{-5}$, $owlp = 1$.

<table>
<thead>
<tr>
<th>Fine mesh (n_p)</th>
<th>ILU(k)</th>
<th>k = 0</th>
<th>k = 1</th>
<th>k = 2</th>
<th>k = 3</th>
<th>k = 4</th>
<th>k = 5</th>
<th>LU</th>
</tr>
</thead>
<tbody>
<tr>
<td>128 x 128 (4)</td>
<td>GMRES</td>
<td>190</td>
<td>98</td>
<td>67</td>
<td>51</td>
<td>44</td>
<td>40</td>
<td>22</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>8.4</td>
<td>3.4</td>
<td>2.2</td>
<td>1.8</td>
<td>1.6</td>
<td>1.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>256 x 256 (16)</td>
<td>GMRES</td>
<td>743</td>
<td>328</td>
<td>163</td>
<td>124</td>
<td>103</td>
<td>90</td>
<td>54</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>37.9</td>
<td>16.5</td>
<td>7.9</td>
<td>5.5</td>
<td>5.0</td>
<td>4.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 x 512 (64)</td>
<td>GMRES</td>
<td>4151</td>
<td>1980</td>
<td>962</td>
<td>628</td>
<td>490</td>
<td>410</td>
<td>168</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>137.8</td>
<td>69.8</td>
<td>35.5</td>
<td>24.6</td>
<td>20.3</td>
<td>18.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4

Varying the overlapping size $owlp$. Fine mesh size: 512 x 512. Coarse mesh size: 64 x 64. $n_p = 64$.

<table>
<thead>
<tr>
<th>$owlp$</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU subdomain solver, almost exact coarse solver, $\epsilon_C = 10^{-10}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM2</td>
<td>FGMRES</td>
<td>27</td>
<td>25</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>6.3</td>
<td>6.0</td>
<td>5.4</td>
<td>5.2</td>
<td>6.9</td>
</tr>
<tr>
<td>HSM</td>
<td>FGMRES</td>
<td>18</td>
<td>16</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>4.4</td>
<td>3.9</td>
<td>3.7</td>
<td>4.5</td>
<td>6.1</td>
</tr>
<tr>
<td>ILU(1) subdomain solver, inexact coarse solver, $\epsilon_C = 10^{-2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM2</td>
<td>FGMRES</td>
<td>25</td>
<td>22</td>
<td>23</td>
<td>26</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>1.3</td>
<td>1.3</td>
<td>1.4</td>
<td>1.8</td>
<td>2.7</td>
</tr>
<tr>
<td>HSM</td>
<td>FGMRES</td>
<td>14</td>
<td>14</td>
<td>15</td>
<td>19</td>
</tr>
<tr>
<td>Time (sec)</td>
<td>0.9</td>
<td>1.0</td>
<td>1.1</td>
<td>1.4</td>
<td>2.1</td>
</tr>
</tbody>
</table>

The table, we observe that the behavior of these two-level Schwarz methods depends on the subdomain and coarse mesh solution quality. Similar to elliptic-type PDEs, when using exact LU subdomain solve and almost exact coarse solve ($\epsilon_C = 10^{-10}$), the number of iterations is reduced monotonically as we increase $owlp$, except in the case of HSM with $owlp = 8$ and 16. However, the saving in computing time is limited. With the optimal $owlp$, the total time taken to solve the problem is about 17% less than the time with the minimal $owlp$. On the other hand, if we relax the quality of solution for both subdomain and coarse problems, the convergence rates of ASM2 and HSM degenerate with $owlp$ increasing, and the optimal execution times are obtained by the use of the small $owlp$. We do not fully understand why the number of iterations increases when $owlp$ becomes larger, although we have observed that this also happens for some indefinite problems, e.g. the Helmholtz problem [10], and we believe the indefiniteness of the problem plays a role here.

4.4. Parallel performance study. Scalability is an important issue in parallel computing, and the issue is more significant when solving large-scale problems with many pro-
processors. To evaluate the parallel scalability of the three Schwarz methods, we adopt the fixed-subdomain-mesh-per-processor scalability as a measurement. An algorithm is considered to be scalable if the computing time remains constant provided that the fine mesh and the number of processors both increase at the same rate, while the subdomain size is fixed. The scalability study of ASM1, ASM2, and HSM is summarized in Table 4.5. Note that the data in the table, including the iteration numbers and execution time for ASM1 except for the single-processor case, is excerpted from the last column of Table 4.3 and for ASM2 and HSM with ASM1 as a coarse solver from the first column and second column of Table 4.3, respectively. For the purpose of comparison, we also include the numerical results for two-level Schwarz methods with two different coarse solvers, namely a redundant LU approach (RLU), which redundantly solves the same coarse mesh problem on all processors and a one-step Schwarz-Richardson method (SR), which is equivalent to the second approach mentioned in Section 3.2. For the case of single processor, the numerical result is obtained by using ILU(1) in conjunction with GMRES. The scaled efficiency shown in Table 4.5 is defined by $\eta = T_1/T_{np}$, where $T_1$ and $T_{np}$ are the computing time obtained with 1 and $n_p$ processors. In the ideal case, $\eta \sim 1$.

From Table 4.5, we observe that ASM1 is not scalable; the number of GMRES iterations grows, roughly, as $\sqrt{n_p}$ and only 6% scaled efficiency is achieved. Comparing the two-level Schwarz methods with different coarse solvers, we find that neither RLU nor SR is as efficient as ASM1. Although the cost per FGMRES for SR is quite low and scalable (see the last three rows of Table 4.6), the solution of SR is not accurate enough to retain the scalability of the method. Also, as shown in Table 4.6, for the small coarse mesh problem, RLU is preferable to ASM1, but as the coarse problem size increases, ASM1 outperforms RLU. ASM1 takes only 10% of the computing time needed for RLU in the case of 64 processors. Next, we compare ASM2 and HSM using ASM1 as the coarse solver. Both ASM2 and HSM are scalable in terms of FGMRES iterations and HSM requires only about half the FGMRES iterations needed for ASM2. Furthermore, HSM is always 20–25% faster than ASM and the scaled efficiency of HSM maintains at least 67% for all cases. While the number of FGMRES iterations is unchanged, the degradation of the scaled efficiency is mainly due to the non-scalable cost per FGMRES iteration. From the first three rows of Table 4.6, we see that the computing time increases as the number of processors grows.

4.5. Schwarz preconditioners for the nonsymmetric saddle point problem. In this section, we study the performance of HSM for the nonsymmetric indefinite Jacobian systems in the NKS algorithm for incompressible Navier-Stokes equations. Table 4.7 shows the average number of FGMRES iterations and the execution time for HSM during a Newton iteration for the two-dimensional lid-driven cavity problem on 64 processors. The stopping criterion for (2.5) is:

$$\|F(x^{(k)}) + (J_k(x^{(k)})M_k^{-1})(M_k s^{(k)\star})\|_2 \leq 10^{-4}\|F(x^{(k)})\|_2.$$ 

The numbers in parentheses are Newton iteration counts required to reduce the initial non-linear residual by a factor of $10^{-6}$, that is, $\|F(x^{(k)})\|_2 \leq 10^{-6}\|F(x^{(0)})\|_2$. The Reynolds number $Re$ is varied from 1 to 1,000. A fixed fine mesh $512 \times 512$ and a fixed coarse mesh $64 \times 64$ are employed. Some key observations from Table 4.7 are made as follows:

1) The number of FGMRES iterations becomes exceedingly large when ILU(k) is used as the subdomain solve. The only exception is the case of ILU(0). This phenomenon is different from the case of symmetric Stokes problems. We suspect that ILU(k) is not stable for the nonsymmetric indefinite saddle point problem, and therefore do not recommend it.

2) HSM with ILU(0) is competitive with HSM with LU only for small $Re$. As $Re$ increases, HSM with LU is found to be 20–40% faster than HSM with ILU(0).
### Table 4.5

Fixed-subdomain-size-per-processor scalability.

<table>
<thead>
<tr>
<th>Fine mesh (nₓ)</th>
<th>FGMRES iterations</th>
<th>Time (sec)</th>
<th>Scaled efficiency η</th>
</tr>
</thead>
<tbody>
<tr>
<td>64x64 (1)</td>
<td>36</td>
<td>0.6</td>
<td>100%</td>
</tr>
<tr>
<td>128x128 (4)</td>
<td>22</td>
<td>1.7</td>
<td>35%</td>
</tr>
<tr>
<td>256x256 (16)</td>
<td>54</td>
<td>3.3</td>
<td>18%</td>
</tr>
<tr>
<td>512x512 (64)</td>
<td>168</td>
<td>9.3</td>
<td>6%</td>
</tr>
</tbody>
</table>

### Table 4.6

Computing time per FGMRES iteration for the iterative and direct coarse solvers.

<table>
<thead>
<tr>
<th>Coarse Mesh Size</th>
<th>nₓ</th>
<th>Ave. Inner iterations</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASM1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16x16</td>
<td>4</td>
<td>5</td>
<td>4x10⁻³</td>
</tr>
<tr>
<td>32x32</td>
<td>16</td>
<td>7</td>
<td>6x10⁻³</td>
</tr>
<tr>
<td>64x64</td>
<td>64</td>
<td>12</td>
<td>2x10⁻²</td>
</tr>
<tr>
<td>Redundant LU</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16x16</td>
<td>4</td>
<td>1</td>
<td>2x10⁻³</td>
</tr>
<tr>
<td>32x32</td>
<td>16</td>
<td>1</td>
<td>2x10⁻²</td>
</tr>
<tr>
<td>64x64</td>
<td>64</td>
<td>1</td>
<td>2x10⁻¹</td>
</tr>
<tr>
<td>One-step SR</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16x16</td>
<td>4</td>
<td>1</td>
<td>6x10⁻⁴</td>
</tr>
<tr>
<td>32x32</td>
<td>16</td>
<td>1</td>
<td>7x10⁻⁴</td>
</tr>
<tr>
<td>64x64</td>
<td>64</td>
<td>1</td>
<td>7x10⁻³</td>
</tr>
</tbody>
</table>
Average number of FGMRES iterations and computing time. HSM for the Jacobian system in the NKS algorithm for Navier-Stokes equations. Reynolds number $Re$ is varied from 1 to 1,000. Fine mesh: $512 \times 512$ and coarse mesh: $64 \times 64$. The tolerance $\epsilon_g$ for the coarse mesh problem is set to be $10^{-2}$, $\epsilon_{ip} = 1$, and $n_p = 64$.

<table>
<thead>
<tr>
<th>$Re$</th>
<th>ILU($k$)</th>
<th>Time (sec)</th>
<th>FGMRES</th>
<th>Time (sec)</th>
<th>FGMRES</th>
<th>Time (sec)</th>
<th>FGMRES</th>
<th>Time (sec)</th>
<th>FGMRES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1</td>
<td>2.4</td>
<td>30.5 (2)</td>
<td>11.1</td>
<td>36.5 (4)</td>
<td>48.0 (7)</td>
<td>6.1 (11)</td>
<td>9.8</td>
<td>66.2 (11)</td>
</tr>
<tr>
<td>100.0</td>
<td>1</td>
<td>3.6</td>
<td>126.0 (2)</td>
<td>15.2</td>
<td>143.5 (4)</td>
<td>199.4 (7)</td>
<td>60.3 (11)</td>
<td>34.4</td>
<td>279.2 (11)</td>
</tr>
<tr>
<td>500.0</td>
<td>1</td>
<td>5.6</td>
<td>701.9 (2)</td>
<td>16.4</td>
<td>827.0 (4)</td>
<td>558.9 (7)</td>
<td>64.9 (11)</td>
<td>42.0</td>
<td>354.5 (11)</td>
</tr>
<tr>
<td>1000.0</td>
<td>1</td>
<td>7.0</td>
<td>163.0 (2)</td>
<td>14.2</td>
<td>174.5 (4)</td>
<td>289.4 (7)</td>
<td>38.9 (11)</td>
<td>38.9</td>
<td>319.5 (11)</td>
</tr>
</tbody>
</table>

(3) The number of FGMRES iterations for HSM with LU depends slightly on $Re$. The higher the Reynolds number is, the more linear and nonlinear iterations it takes to meet the stopping requirement.

5. Concluding remarks. We presented a fully coupled parallel solver with three types of overlapping Schwarz preconditioners for saddle point problems arising from CFD. As shown in the numerical results, additive Schwarz-type preconditioners for the Stokes problem and the elliptic-type problems have some similar convergence properties: the number of GMRES iterations decreases monotonically as the overlapping size increases when LU is used as the subdomain solve; the number of GMRES iterations for ASM1 grows as $\sqrt{n_p}$, and HSM requires roughly half as many FGMRES iterations as ASM2. Our numerical results also showed that the ILU($k$) based inexact subdomain solver is useful only for the symmetric Stokes problem but not for the nonsymmetric indefinite Jacobian systems in NKS due to the possible instability of the incomplete decomposition. The iterative inexact coarse solver is sufficient to retain the optimal convergence rate of the two-level Schwarz preconditioners. In comparison with ASM1 and ASM2, HSM with a iterative coarse solver showed superior parallel performance.

REFERENCES


