INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 0000; **00**:1–18
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld

Parallel fully implicit two-grid methods for distributed control of unsteady incompressible flows

Haijian Yang and Xiao-Chuan Cai*†

Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA

SUMMARY

In this paper we investigate some fully coupled parallel two-grid Lagrange-Newton-Krylov-Schwarz (LNKSz) algorithms for the suboptimal distributed control of unsteady incompressible flows governed by the Navier-Stokes equations. The algorithms include two major parts: a two-grid Newton method for the nonlinear part of the problem and a two-level Schwarz preconditioner for the linear part of the problem. Most of the existing approaches for distributed control problems are based on the so-called reduced space method which is easier to implement but may have convergence issues in some situations. In the full space approach we couple the state variables, the control variables, and the adjoint variables in a single large system of nonlinear equations. The coupled system is considerably more ill-conditioned than its sub-systems, however, with the powerful two-grid approach, we are able to solve these difficult systems efficiently on large scale parallel computers. We show numerically that such an approach is scalable in the sense that the number of Newton iterations and the number of linear iterations are both nearly independent of the grid size, the number of processors, and the Reynolds numbers. We present numerical experiments for some suboptimal control problems obtained on supercomputers with more than two thousand processors. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: PDE constrained optimization; multilevel Schwarz preconditioner; parallel computing; two-grid inexact Newton; distributed control; unsteady incompressible Navier-Stokes equations

1. INTRODUCTION

Flow optimal control problems have attracted substantial interests in recent years due to their wide ranging applications [16, 24]. Flow control problems can be described in many different forms such as flow matching, vorticity minimization, viscous drag minimization, avoiding hot spots, stabilization enhancement, mixing maximization, and so on. In this paper we focus on the distributed control problem, which is to control the fluid flow by computing the external force applied to the flow. Such problems are extremely demanding in terms of computational resources. Popular approaches for solving unsteady flow control problems are explicit or semi-implicit methods, both have limitations on the time step size imposed by the Courant-Friedrichs-Lewy (CFL) condition. For many applications, it is desirable to use algorithms that allow large time steps that are determined by the desired accuracy, but not the stability condition. Moreover, the algorithm is also required to be robust with respect to some of physical parameters, such as the Reynolds number. Hence, the focuses of the paper are algorithms that allow large time steps, are robust with respect to physical

*Correspondence to: Xiao-Chuan Cai, Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA.
†E-mail: cai@cs.colorado.edu

parameters, and scalable on machines with a large number of processors. Many attempts have been made in the past few years to mathematically understand the flow control problems and to numerically solve the flow control problems in various forms; see e.g., [13, 14, 15, 19, 25, 26]. In [20], a boundary control problem is first transformed into a distributed control problem and then the distributed control problem is solved by a class of multigrid methods. In [15], a gradient method is proposed for the numerical solution of a time-dependent distributed control problem associated with the tracking of the velocity of a Navier-Stokes flow in a bounded two-dimensional domain. In [18], a numerically inexpensive globalization strategy of sequential quadratic programming methods for the unsteady distributed control problem is investigated. Although these numerical algorithms are stable and reliable for solving various unsteady distributed control problems, there are some restrictions on the time step size due to the semi-implicit nature of the algorithms. Flow optimal control problems are computationally expensive. In this paper we propose a class of fully implicit algorithms for the distributed control of unsteady incompressible flows. Since we use a fully implicit scheme that is suitable for large scale supercomputers, the CFL condition can be completely relaxed. We show numerically that the proposed method is stable and converges well with relatively large times steps, and it is robust with respect to some of the physical parameters, such as the Reynolds number.

The class of full space Lagrange-Newton-Krylov type algorithms was introduced for boundary control of incompressible flows [4, 5, 22, 23, 31, 32]. The methods include two parts: a Lagrange-Newton method for the nonlinear system obtained from the optimization problem and a Krylov subspace method for the Jacobian system arising from the Newton method. For the class of nonlinear constrained optimization problems, the scalability of the method requires mesh independent convergence of the outer Newton iterations. To fix this issue, we use a grid sequencing method which employs an interpolated coarse grid solution as the initial guess for the fine grid system. Our experiments show that this strategy provides a good improvement of the overall method in terms of the total computing time and Newton iterations. In other words, the impact of grid sequencing is restricted to the nonlinear solver part of the algorithm, and its influence to the linear solver part is very small.

The rest of the paper is organized as follows. In Section 2, we present the unsteady distributed control problems and introduce a fully implicit discretization scheme. Section 3 is devoted to the main components and features of LNKSz and then describes the details of the two-level Schwarz preconditioner. Some numerical results are given in Section 4. We end the paper with some concluding remarks in Section 5.

2. MATHEMATICAL MODEL AND DISCRETIZATION

In this section, we first describe a general distributed control problem governed by the unsteady incompressible Navier-Stokes equations, and then introduce a fully implicit discretization for the suboptimal control problem.

We consider the unsteady incompressible Navier-Stokes equations in the velocity-vorticity formulation:

$$-\Delta v_1 - \frac{\partial \omega}{\partial y} = 0 \quad \text{in } [0,T]\times\Omega, \tag{1}$$

$$-\Delta v_2 + \frac{\partial \omega}{\partial x} = 0 \quad \text{in } [0,T]\times\Omega, \tag{2}$$

$$\frac{\partial \omega}{\partial t} - \frac{1}{Re}\Delta\omega + v_1\frac{\partial \omega}{\partial x} + v_2\frac{\partial \omega}{\partial y} - \text{curl }\mathbf{f} = 0 \quad \text{in } [0,T]\times\Omega, \tag{3}$$

where Ω is a bounded domain in $\mathbb{R}^2$, and T denotes the final time. In the above equations the velocity field $\mathbf{v} = (v_1, v_2)$ and the vorticity ω are the state variables, $\mathbf{f} = (f_1, f_2)$ is the external force, curl $\mathbf{f} = -\partial f_1/\partial y + \partial f_2/\partial x$, and Re is the Reynolds number. With some given initial and boundary conditions, the equations (1)-(3) are often referred to as the simulation or the forward problem.

In the distributed control problem we try to find an external force $\mathbf{f}$ over the whole or part of Ω in order to achieve the goal [15, 18]

$$\min \mathcal{F}(\mathbf{v},\omega,\mathbf{f}) = \frac{1}{2}\int_0^T \mathcal{G}(\mathbf{v},\omega)\,dt + \frac{\gamma}{2}\int_0^T \int_{\Omega_f} \|\mathbf{f}\|_2^2\,d\Omega\,dt \tag{4}$$

subject to the constraints (1), (2), and (3) with the following boundary and initial conditions

$$\begin{cases} \mathbf{v}-\mathbf{v}_D & = \mathbf{0} \quad \text{on } [0,T]\times\Gamma, \\ \omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} & = 0 \quad \text{on } [0,T]\times\Gamma, \\ \mathbf{v}(0,x,y) - \mathbf{v}_0 & = \mathbf{0} \quad \text{in } \overline{\Omega}, \\ \omega(0,x,y) + \dfrac{\partial v_1}{\partial y}(0,x,y) - \dfrac{\partial v_2}{\partial x}(0,x,y) & = 0 \quad \text{in } \overline{\Omega}. \end{cases} \tag{5}$$

Here, Γ is the boundary of Ω, $\Omega_f \subseteq \Omega$ is the control domain, $\mathbf{v}_D$ and $\mathbf{v}_0$ are given velocities, $\mathcal{G}(\mathbf{v},\omega)$ is the goal of the optimal control problem. $\gamma > 0$ is a regularization parameter used to adjust the relative importance of the control norms in achieving the minimization, thus indirectly constraining their magnitudes.

For solving optimization problems constrained by time dependent nonlinear partial differential equations, it typically requires a combination of a discretization technique in space and time with an optimization method. We follow the discretize-then-optimize approach in this paper, namely we first fully discretize the differential equations and then apply an optimization method to solve the discrete finite dimensional problem. The computational cost for solving the problem is enormous, even for the latest massively parallel computers it is a very difficult job to solve the unsteady control problem at once for the whole time interval $[0,T]$, we therefore replace the original full-time-interval problem by a sequence of suboptimal problems [27], which are similar to the original problem but only defined on the time interval $[t^{(k-1)}, t^{(k)}]$, $k = 1,\ldots,k_{\max}$, with $t^{(0)} = 0$ and $t^{(k_{\max})} = T$. Note that in the suboptimal approach the sequence of subproblems' objectives are the same as the objective of the original control problem. We refer interested readers to the papers [2, 3, 21] on the suboptimal approach. On each time interval, we write

$$\min \mathcal{F}^{(k)}(\mathbf{v},\omega,\mathbf{f}) = \frac{1}{2}\int_{t^{(k-1)}}^{t^{(k)}} \mathcal{G}(\mathbf{v},\omega)\,dt + \frac{\gamma}{2}\int_{t^{(k-1)}}^{t^{(k)}} \int_{\Omega_f} \|\mathbf{f}\|_2^2\,d\Omega\,dt \tag{6}$$

which is subject to the constraints (1)-(3) and the boundary conditions (5) defined on the short time interval $[t^{(k-1)}, t^{(k)}]$. The initial condition is taken as the final solution from the previous time step, except in the first time interval in which the given initial condition in (5) is available.

For the time discretization, by using a second-order backward differentiation formula [17] with a uniform step size $\Delta t \equiv t^{(k)} - t^{(k-1)}$, we have

$$\min \mathcal{F}^{(k)}(\mathbf{v}^{(k)},\omega^{(k)},\mathbf{f}^{(k)}) = \Delta t\,\mathcal{G}(\mathbf{v}^{(k)},\omega^{(k)}) + \frac{\gamma}{2}\Delta t \int_{\Omega_f} \|\mathbf{f}^{(k)}\|_2^2\,d\Omega \tag{7}$$

with the constraints:

$$\begin{cases} -\Delta v_1^{(k)} - \dfrac{\partial \omega^{(k)}}{\partial y} & = 0 \quad \text{in } \Omega, \\ -\Delta v_2^{(k)} + \dfrac{\partial \omega^{(k)}}{\partial x} & = 0 \quad \text{in } \Omega, \\ \dfrac{1}{\Delta t}\left[\dfrac{3}{2}\omega^{(k)} - 2\omega^{(k-1)} + \dfrac{1}{2}\omega^{(k-2)}\right] - \dfrac{1}{Re}\Delta\omega^{(k)} & \\ \quad + v_1^{(k)}\dfrac{\partial \omega^{(k)}}{\partial x} + v_2^{(k)}\dfrac{\partial \omega^{(k)}}{\partial y} - \text{curl}\,\mathbf{f}^{(k)} & = 0 \quad \text{in } \Omega, \\ \mathbf{v}^{(k)} - \mathbf{v}_D^{(k)} & = \mathbf{0} \quad \text{on } \Gamma, \\ \omega^{(k)} + \dfrac{\partial v_1^{(k)}}{\partial y} - \dfrac{\partial v_2^{(k)}}{\partial x} & = 0 \quad \text{on } \Gamma. \end{cases} \tag{8}$$

Note that, for the first time step (i.e., $k = 1$), we use the following first-order backward Euler method to substitute the third constraint in (8):

$$\frac{1}{\Delta t}(\omega^{(k)} - \omega^{(k-1)}) - \frac{1}{Re}\Delta\omega^{(k)} + v_1^{(k)}\frac{\partial\omega^{(k)}}{\partial x} + v_2^{(k)}\frac{\partial\omega^{(k)}}{\partial y} - \text{curl } \mathbf{f}^{(k)} = 0.$$

A major advantage of the fully implicit method is that the time step size Δt is not constrained by a CFL condition, which is often required by explicit or semi-implicit techniques.

For the spatial discretization, we use a second-order five-point finite difference method on a uniform mesh. Let us write the suboptimal problem as

$$\begin{cases} \min & \mathcal{F}_h^{(k)}(\mathbf{x}) \\ \text{s.t.} & \mathbf{C}_h^{(k)}(\mathbf{x}) = \mathbf{0}, \end{cases} \tag{9}$$

where $\mathbf{x} = (\mathbf{v}, \omega, \mathbf{f})$. In fact, (9) can be viewed as a steady state control problem. Those interested in the details of the spatial disctrization can read [22, 23].

By introducing the Lagrange multipliers λ with respect to the state and control variables, we define the following Lagrangian functional

$$\mathcal{L}^{(k)}(\mathbf{x}, \lambda) \equiv \mathcal{F}_h^{(k)}(\mathbf{x}) + \big(\lambda, \mathbf{C}_h^{(k)}(\mathbf{x})\big). \tag{10}$$

Let $X \equiv (\mathbf{x}, \lambda)$. Then, for $k = 1, \ldots, k_{\text{max}}$, the KKT system obtained by differentiating (10) becomes

$$G^{(k)}(X) = \begin{pmatrix} \nabla_{\mathbf{x}}\mathcal{L}^{(k)}(\mathbf{x}, \lambda) \\ \nabla_{\lambda}\mathcal{L}^{(k)}(\mathbf{x}, \lambda) \end{pmatrix} = 0. \tag{11}$$

The optimality system (11) is a large, nonlinear, coupled, and muti-components system, which is much more complicated than the corresponding simulation problem. In the next section we introduce an iterative method that is capable of dealing with the high nonlinearity of the system and also the severe ill-conditionness of its Jacobian matrix.

3. TWO-GRID NEWTON AND TWO-LEVEL SCHWARZ PRECONDITIONERS

In this section, we introduce a parallel scalable solution algorithm for solving (11). After many numerical experiments, we observe that, because of the high nonlinearity, the traditional Newton method doesn't work well for this class of problems, no matter how accurately or inaccurately we solve the Jacobian system. However, if we borrow the coarse space (which is originally used to build the two-level preconditioner for the Jacobian matrix) and use it as a coarse Newton solver, then the nonlinear iteration suddenly become acceptable. Below we first discuss the class of full space Lagrange-Newton-Krylov-Schwarz (LNKSz) method and then focus on the two-grid Newton method.

For each time step $k = 1, \ldots, k_{\text{max}}$, the nonlinear system (11) is solved by an inexact Newton method, and the Newton step is computed by

$$\begin{cases} J_n^{(k)} S_n^{(k)} = -G^{(k)}(X_n^{(k)}), \\ X_{n+1}^{(k)} = X_n^{(k)} + \alpha_n^{(k)} S_n^{(k)}, \quad n = 0, 1, \ldots., \end{cases} \tag{12}$$

where $\alpha_n^{(k)}$ is the steplength determined by a linesearch procedure [9, 10], the Jacobian matrix $J_n^{(k)} = J_n^{(k)}(X_n^{(k)})$ is computed by a finite difference approximation, and the initial guess $X_0^{(k)}$ is the solution of the previous time step. Note that at the first time step, we choose the initial condition (i.e., at $t = 0$) as the initial guess. We continue the iteration (12) until the following convergence criterion is satisfied

$$||G^{(k)}(X_{n+1}^{(k)})|| \le \max\{\varepsilon_r ||G^{(k)}(X_0^{(k)})||, \varepsilon_a\},$$

where ε_r (ε_a) is the relative (absolute) solver tolerance for the Newton iteration. A Krylov subspace method is applied to approximately solve the following right-preconditioned linear system

$$\|G^{(k)}(X_n^{(k)}) + J_n^{(k)}(M_n^{(k)})^{-1}(M_n^{(k)}S_n^{(k)})\| \le \max\{\eta_r\|G^{(k)}(X_n^{(k)})\|, \eta_a\},$$

where $(M_n^{(k)})^{-1}$ is the one-level or two-level Schwarz preconditioner, and η_r (η_a) is the relative (absolute) solver tolerance for the linear iteration. We will refer to the above method as the one-level or two-level LNKSz method.

When using the above Newton's method to solve the nonlinear system (11), one big problem is the deterioration of the convergence rate (i.e., The number of Newton iterations to satisfy the stopping condition.) when the grid is refined, specially for the first time step, since in this case the initial guess is not good enough for Newton iterations. It turns out the classical idea of grid sequencing works quite well in this situation.

In order to use the grid-sequencing method, we assume there are two grids covering Ω, a coarse grid and a fine grid, and we assume there is a coarse to fine grid interpolation operator $\mathcal{I}_H^h$. We construct the optimization problem on the coarse and fine grids, respectively. We first use the one-level LNKSz to solve the nonlinear problem on the coarse grid with the initial guess obtained as a restriction of the fine-grid solution from the previous timestep. Of course, at the first time step, we choose the initial condition as the initial guess. Then, we interpolate the solution to the next fine grid and use it as an initial guess for the nonlinear problem on that grid. Moreover, the optimization problem on the fine grid is solved by the two-level LNKSz. We refer to this LNKSz method combined with the grid-sequencing technique as a two-grid LNKSz method.

We remark that the two purposes of the coarse grid are (1) as a part of grid-sequencing technique for the Newton iteration; (2) as a part of the two-level Schwarz preconditioner for solving the linear Jacobian problem.

In the following, we define the Schwarz preconditioners. First, we define the one-level additive Schwarz method. More precisely, we first partition Ω into non-overlapping subdomains Ω_i, $i = 1, \dots, N_s$. Then each subdomain Ω_i is extended with $\delta > 0$ layers of grid points to a larger subdomain Ω_i^δ that overlaps with its neighbors. Subdomain boundaries that coincide with the physical boundary are not extended. Suppose the total number of unknowns associated with Ω is N and let N_i be the number of unknowns in Ω_i^δ. The Jacobian matrix J is an $N \times N$ sparse matrix in the system

$$JS = -G. \tag{13}$$

A restriction operator R_i^δ is an $N_i \times N$ matrix that maps a vector defined on the entire domain to a smaller vector defined on the subdomain Ω_i^δ by discarding all components corresponding to mesh points outside Ω_i^δ. Specifically, R_i^0 is also an $N_i \times N$ matrix that is similarly defined, with the difference that its application to a $N \times 1$ vector also zeroes all those components corresponding to mesh points outside Ω_i. The subdomain matrix is an $N_i \times N_i$ matrix that is defined as

$$J_i = R_i^\delta\ J\ (R_i^\delta)^T.$$

In general, it is difficult to prove theoretically that the matrix J is nonsingular for a particular flow problem, but in our experiments, J is indeed nonsingular. We also assume that J_i is nonsingular and denote by B_i^{-1} either the inverse of or a preconditioner for J_i. The one-level restricted additive Schwarz (RAS) preconditioner for J is defined as [7]

$$M_{RAS}^{-1} = (R_1^0)^T B_1^{-1} R_1^\delta + \cdots + (R_{N_s}^0)^T B_{N_s}^{-1} R_{N_s}^\delta. \tag{14}$$

In this paper the matrices B_i^{-1} are obtained by LU or ILU factorization. More details of this case will be discussed in the numerical experiments section of this paper.

Next, we define the multiplicative type two-level Schwarz preconditioner. Let I denote the identity operator, $\mathcal{I}_H^h$ be a linear interpolation operator from the coarse grid to the fine grid, and $\mathcal{I}_h^H$ be a restriction operator from the fine grid to the coarse grid. Here, we let $\mathcal{I}_h^H = (\mathcal{I}_H^h)^T$. Similar to the Jacobian matrix J defined on the fine grid, there is also the Jacobian matrix J_c defined on the coarse grid. Then, the multiplicative type two-level Schwarz preconditioner can be defined as [29, 30]

$$M^{-1} = \Big(I - (I - M_{RAS}^{-1}J)(I - M_c^{-1}J)(I - M_{RAS}^{-1}J)\Big)J^{-1} \tag{15}$$

where $M_c^{-1} = \mathcal{I}_H^h J_c^{-1} \mathcal{I}_h^H$. The preconditioner (15) is additive among all fine or coarse mesh subdomains, and multiplicative between the fine and coarse preconditioners. In fact, the preconditioner (15) can be seen symbolically as a V-cycle multigrid algorithm [6].

Note that, in the two-level Schwarz preconditioner (15), a linear problem on the coarse grid $J_c v_0 = b_0$ needs to be solved. In our applications, the problem is too large for direct methods. We use a restarted GMRES, preconditioned with a one-level additive Schwarz method to solve the coarse problem, using the same number of subdomains (and therefore processors) as on the fine grid. Similar to the one-level preconditioner on the fine grid, we introduce a one-level preconditioner on the coarse grid

$$B_c^{-1} = \sum_{i=1}^{N_s} (R_{c,i}^0)^T B_{c,i}^{-1} R_{c,i}^{\delta_c}.$$

Here $B_{c,i}$ is the restriction of J_c on the subdomain $\Omega_i^{\delta_c}$, and $R_{c,i}^{\delta_c}$ and $R_{c,i}^0$ are the restriction operators on the coarse grids defined on $\Omega_i^{\delta_c}$ and Ω_i, respectively. $e^H = J_c^{-1} r^H$ is computed by approximately solving the following problem

$$\|r^H - J_c B_c^{-1} x_c'\| \le \max\{\eta_r^c \|r^H\|, \eta_a^c\},$$

where η_r^c (η_a^c) is the relative (absolute) solver tolerance for the linear iteration on the coarse grid. The purpose of the coarse grid problem is for the coarse grid correction to help the fine grid problem. Hence, we can solve the coarse problem with a much larger error tolerance than the fine problem if it can save computational cost while still is an effective preconditioner for (15). More discussion will be given in the numerical experiments section of the paper. When an iterative method is used for solving the coarse grid problem, the overall preconditioner is an iterative procedure. In other words, the preconditioner changes from iteration to iteration. Hence, in this paper, we use GMRES for all one-level cases and FGMRES for all two-level and two-grid cases [28].

4. NUMERICAL EXPERIMENTS

Our algorithms are implemented based on the Portable Extensible Toolkit for Scientific computation (PETSc) [1]. All computations are performed on an IBM BlueGene/L supercomputer.

4.1. Test cases

We consider two model problems: a flow matching problem [11, 14, 15, 18] and a backward-facing step flow control problem [19, 25, 27].

In the flow matching problem we attempt to have the velocity field agree with a desired flow in the domain Ω. In other words, we compute an external force $\mathbf{f}$ over the interior of Ω such that the corresponding velocity field is as close to the given velocity field $\mathbf{v}_{ss}$ as possible, and at the same time satisfy the constraints. Thus the problem under consideration is to find $(v_1, v_2, \omega, f_1, f_2)$ such that the minimization

$$\min \mathcal{F}(\mathbf{v}, \mathbf{f}) = \frac{1}{2}\int_0^T \int_\Omega \|\mathbf{v} - \mathbf{v}_{ss}\|_2^2 \, d\Omega \, dt + \frac{\gamma}{2}\int_0^T \int_\Omega \|\mathbf{f}\|_2^2 \, d\Omega \, dt \tag{16}$$

is achieved subject to the constraints (1), (2), and (3) with the boundary and initial conditions as follows:

$$\begin{cases} \mathbf{v}-\mathbf{v}_D & = \mathbf{0} \quad \text{on } [0,T]\times\Gamma, \\ \omega+\dfrac{\partial v_1}{\partial y}-\dfrac{\partial v_2}{\partial x} & = 0 \quad \text{on } [0,T]\times\Gamma, \\ \mathbf{v}(0,x,y)-\mathbf{v}_0 & = \mathbf{0} \quad \text{in } \overline{\Omega}, \\ \omega(0,x,y)+\dfrac{\partial v_1}{\partial y}(0,x,y)-\dfrac{\partial v_2}{\partial x}(0,x,y) & = 0 \quad \text{in } \overline{\Omega}, \end{cases} \tag{17}$$

where Γ is the boundary of the domain $\Omega=(0,1)\times(0,1)$, $T=1$. Let $\Gamma=\Gamma_1\cup\Gamma_2\cup\Gamma_3\cup\Gamma_4$ where $\Gamma_1,\Gamma_2,\Gamma_3$ and Γ_4 are disjoint portions of the boundary Γ of the domain Ω, $C_1\cup C_2\cup C_3\cup C_4$ its corners. The geometry for the flow matching problem is shown in Figure 1.

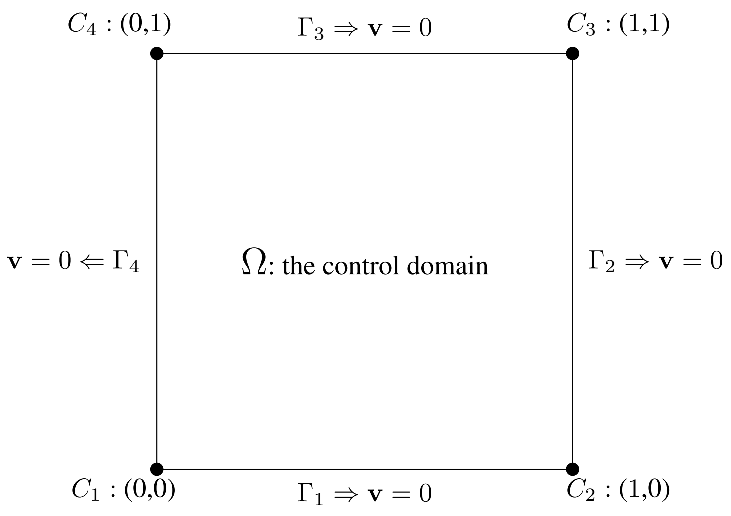
C_4 : (0,1)
$\Gamma_3 \Rightarrow \mathbf{v}=0$
C_3 : (1,1)
$\mathbf{v}=0 \Leftarrow \Gamma_4$
Ω: the control domain
$\Gamma_2 \Rightarrow \mathbf{v}=0$
C_1 : (0,0)
$\Gamma_1 \Rightarrow \mathbf{v}=0$
C_2 : (1,0)

Figure 1. Square domain $\Omega=(0,1)\times(0,1)$ for the flow matching problem.

Similar to [18], the boundary condition is $\mathbf{v}_D=0$ and the initial condition is chosen as

$$\mathbf{v}_0=e\begin{pmatrix} (\cos(2\pi x)-1)\sin(2\pi y) \\ -(\cos(2\pi y)-1)\sin(2\pi x) \end{pmatrix},$$

where e is the Euler number, and the target velocity is given by

$$\mathbf{v}_{ss}(t,x,y)=\begin{pmatrix} \varphi_y(t,x,y) \\ -\varphi_x(t,x,y) \end{pmatrix},$$

where φ is defined by the following function

$$\varphi(t,x,y)=(1-x)^2(1-y)^2\left(1-\cos(2\pi tx)\right)\left(1-\cos(2\pi ty)\right).$$

The second test case is a backward-facing step flow control problem, in which we minimize the vorticity of the flow by computing an external force in a subdomain. We apply the control to the subdomain Ω_f and in the non-control subdomain $\Omega\backslash\Omega_f$ the external force $\mathbf{f}$ is chosen as 0. Let $\Omega=(0,6)\times(0,1)$, $\Omega_f=(0,1)\times(0,0.5)$, $T=1$, $\Gamma_2=\{(x,y)\in\Gamma : 0<y<1, x=6\}$, $\Gamma_4=\{(x,y)\in\Gamma : 0<y<1, x=0\}$, and $\Gamma_{4,a}=\{(x,y)\in\Gamma_4 : 0.5\le y<1\}$. Then the problem consists of finding (v_1,v_2,ω,f_1,f_2) such that

$$\min \mathcal{F}(\omega,\mathbf{f})=\frac{1}{2}\int_0^T\int_\Omega \omega^2\,d\Omega\,dt+\frac{\gamma}{2}\int_0^T\int_{\Omega_f}\|\mathbf{f}\|_2^2\,d\Omega\,dt \tag{18}$$

is achieved subject to the constraints (1), (2), and (3) with the following boundary and initial conditions:

$$\begin{cases} v_1 = v_{in} & \text{on } [0,T] \times \Gamma_{4,a}, \\ v_1 = v_{out} & \text{on } [0,T] \times \Gamma_2, \\ v_1 = 0 & \text{on } [0,T] \times \Gamma_u, \\ v_2 = 0 & \text{on } [0,T] \times \Gamma, \\ \omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} = 0 & \text{on } [0,T] \times \Gamma, \\ \mathbf{v}(0,x,y) - \mathbf{v}_0 = \mathbf{0} & \text{in } \overline{\Omega}, \\ \omega(0,x,y) + \dfrac{\partial v_{0,1}}{\partial y} - \dfrac{\partial v_{0,2}}{\partial x} = 0 & \text{in } \overline{\Omega}, \end{cases} \tag{19}$$

where $\Gamma_u = \Gamma \backslash (\Gamma_{4,a} \cup \Gamma_2)$. The geometry is shown in Figure 2.

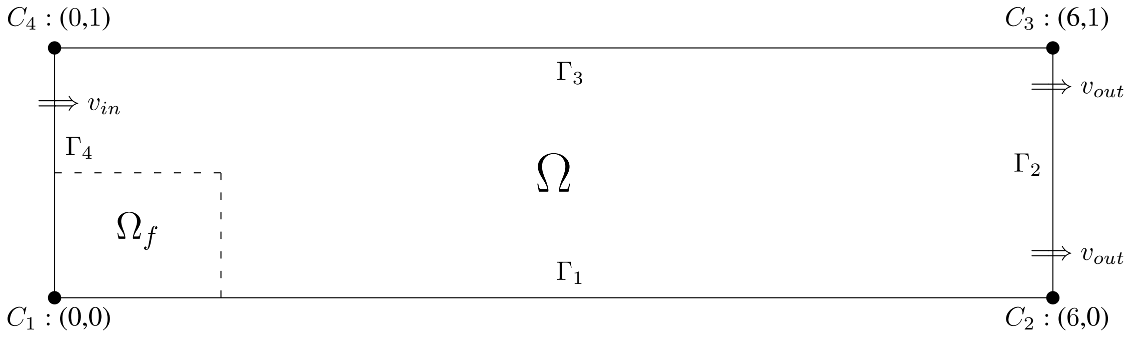

Figure 2. Rectangular domain $\Omega = (0,6) \times (0,1)$ for the backward-facing step channel flow, $\Omega_f = (0,1) \times (0,0.5)$ is the control domain.

At the inflow boundary, a parabolic velocity profile $v_{in} = 8(1-y)(y-\frac{1}{2})\cos(t)$ is imposed. At the outflow boundary, $v_{out} = y(1-y)\cos(t)$ is applied. The following initial velocity is defined by $\mathbf{v}_0 = (v_{0,1}, v_{0,2})$ with

$$v_{0,1} = \begin{cases} y(1-y) + \dfrac{1}{16}y & \text{if } 0 \leqslant y \leqslant \dfrac{1}{2}, \\ y(1-y) + \dfrac{1}{16}(1-y) & \text{if } \dfrac{1}{2} \leqslant y \leqslant 1, \end{cases}$$

and

$$v_{0,2}(x,y) = 0.$$

4.2. *Details of numerical experiments*

For the time discretization we apply the second-order backward differentiation formula as described in Section 2. For the spatial discretization we use a five-point finite difference method on a uniform mesh. In order to form the algebraic system generated from finite difference equations obtained for each of the mesh points, we need to order both the unknowns and the corresponding functions. In contrast to ordering the unknowns component-wise that is usually required by other methods, we order the unknowns mesh point by mesh point and the mesh points are ordered subdomain by subdomain for the purpose of parallel processing. The unknowns at each mesh point are ordered in the order of $v_1, v_2, \omega, f_1, f_2, \lambda_1, \lambda_2, \lambda_3$, and the corresponding functions are ordered in the order of $\nabla_{\lambda_1}\mathcal{L}, \nabla_{\lambda_2}\mathcal{L}, \nabla_{\lambda_3}\mathcal{L}, \nabla_{f_1}\mathcal{L}, \nabla_{f_2}\mathcal{L}, \nabla_{v_1}\mathcal{L}, \nabla_{v_2}\mathcal{L}, \nabla_{\omega}\mathcal{L}$.

In the experiments, we compare the following algorithms which are introduced in Section 3:

- One-level LNKSz: one-level additive Schwarz is used as the preconditioner, and inexact Newton is carried out on the fine grid;
- Two-level LNKSz: two-level multiplicative Schwarz is used as the preconditioner, and inexact Newton is carried out on the fine grid;
- Two-grid LNKSz: two-level multiplicative Schwarz is used as the preconditioner on the fine grid, inexact Newton is used on the coarse grid to generate the initial guess for the inexact Newton on the fine grid.

The Jacobian matrices are constructed approximately using a multi-colored finite difference method, i.e., we take the following numerical differentiation procedure to evaluate J^k, which is defined by

$$J_{i,j}^{(k)} = \frac{G_i^{(k)}(X_j+\alpha) - G_i^{(k)}(X_j-\alpha)}{2\alpha},$$

where $0 < \alpha \leqslant 1$ is a small constant. Many such calculations at points not related to each other can be carried out at the same time by using the multi coloring technique [8].

The size of the coarse grid H is taken as $4h$, where h is the size of the fine grid. GMRES(90) and FGMRES(90) are used to solve the linear system at each Newton step on the coarse and the fine grids, respectively. There are several nested iterative procedures in the proposed algorithms, and each requires a proper stopping condition. We use 10^{-10} (10^{-6}) as the absolute (relative) condition for all linear and nonlinear solves, except for the linear coarse solve of the two-level preconditioner, for which we use 10^{-2} (10^{-1}) as the absolute (relative) condition for the flow matching problem (16)-(17) and 10^{-4} (10^{-2}) as the absolute (relative) condition for the backward-facing step control problem (18)-(19), respectively. The subdomain problems are solved with a sparse LU or ILU factorization. For Newton iterations, line search is performed with cubic backtracking [9, 10]. Note that the Reynolds continuation is not used in any of the algorithms.

Scalability is an important issue in parallel computing, and the issue is more significant when solving large-scale problems with many processors. To evaluate the parallel performance of the proposed methods, we consider the following parallel efficiency:

$$E_f = \frac{N_{p,1} \times T_1}{N_{p,2} \times T_2},$$

where T_1 and T_2 are the execution times obtained by running the parallel code with $N_{p,1}$ and $N_{p,2}$ processors ($N_{p,1} \leq N_{p,2}$), respectively. We also report speedup defined as

$$\text{Speedup} = \frac{T_1}{T_2}.$$

Throughout this paper, "N_p" stands for the number of processors which is the same as the number of subdomains, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the average number of the preconditioned GMRES iterations per Newton iteration, and "RUN" is the total computing time in seconds, which includes the total CPU time for solving all the nonlinear systems. We also use the following notations:

- "$--$": for some cases that the tests are not carried out because of the lack of memory;
- "$++$": for some cases that the tests are not convergent because of the divergence of GMRES;
- "δ": the distance between $\partial\Omega_i^\delta$ and $\partial\Omega_i$ on the fine grid in terms of the number of mesh cells, and δ_c for the coarse grid.

4.3. *Comparing one-level and two-level Schwarz preconditioning*

We first study the application of the one-level and two-level LNKSz to the flow matching problem (16)-(17). In particular, we look at the performance of the algorithms with respect to the change of some of the physical and algorithmic parameters.

In Table I, we show the performance of the one-level and two-level LNKSz when the number of processors and the size of the overlap change. For these experiments we use a fix 512×512 mesh, $\gamma = 0.1$, $Re = 200$, ILU(3), and $\Delta t = 0.1$. First we observe that the number of Newton iterations is completely independent of the number of processors and the overlapping size in both one-level and two-level approaches. However, the number of GMRES iterations changes a lot. In the one-level tests, if the overlap is too small, GMRES sometimes fails to converge. Even though the number of GMRES iterations goes down as the overlap increases, but the total compute doesn't always decrease when δ increases. For each fixed processor count, there is an optimal choice of δ. In the two-level tests, we fix the coarse mesh as 128×128. The performance is considerably better than the one-level method in terms of both the number of iterations and the total compute time, and $\delta = 2$ produces the best timing results for all processor counts.

Table I. Effect of overlap size δ for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\Delta t = 0.1$ (i.e., there are 10 time steps). δ_c = 2 and the coarse subdomain solve is ILU(3). In the one-level method, the fine mesh is 512×512. In the two-level method, the coarse mesh is 128×128 and the fine mesh is 512×512. "$++$" means the divergence of GMRES.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
		$\delta = 1$			$\delta = 2$			$\delta = 3$	
					One-level				
64	2.8	771.8	2712.9	2.8	709.5	2599.6	2.8	663.6	2557.5
128	2.8	904.1	1614.2	2.8	787.3	1491.9	2.8	735.9	1494.7
256	2.8	1028.4	938.1	2.8	896.2	893.3	2.8	819.1	874.7
512	2.8	1262.3	595.1	2.8	1098.5	581.8	2.8	1009.4	585.4
1024		$++$		2.8	1164.8	335.2	2.8	1020.6	330.7
2048		$++$		2.8	1465.3	220.6	2.8	1275.1	240.0
					Two-level				
64	2.8	56.7	1324.4	2.8	38.4	1026.0	2.8	38.1	1033.3
128	2.8	59.9	728.8	2.8	38.8	573.1	2.8	39.3	595.7
256	2.8	58.7	405.6	2.8	40.1	323.5	2.8	40.0	334.5
512	2.8	60.6	258.1	2.8	41.6	212.5	2.8	41.1	219.3
1024	2.8	62.2	162.0	2.8	43.5	136.8	2.8	42.1	141.3
2048	2.8	72.0	120.5	2.8	45.5	93.1	2.8	47.2	102.0

The per processor performance of LNKSz depends heavily on how the subdomain problems are solved. In the following set of the tests, we compare several different subdomain solvers based a sparse LU factorization, and sparse incomplete LU factorizations with varying level of fill-ins. In the tests, we use a fixed mesh 512×512. In Table II, we summarize the results with different number of processors and level of fill-ins. As expected, the two-level method outperforms the one-level method. ILU with small level of fill-in offers the best results in terms of the total compute time.

Table III shows the effect of Δt on the performance of the one-level and two-level LNKSz, for fixed $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\delta = 2$. For the scaling studies, we experiment with several different time steps, and report the total computing time, the nonlinear iteration counts per time step, and the average GMRES iterations per Newton step. As the timestep size Δt is increased, the number of iterations for the one-level method increases, while those for the two-level method are fairly stable. Note that LNKSz converges well with small and large time steps.

Table IV summarizes the impact of the fine mesh size on the performance of LNKSz, for fixed $\gamma = 0.1$, $Re = 200$, $\Delta t = 0.1$ (i.e., there are 10 time steps), and $\delta = 2$. We see that the number of nonlinear iterations per time step does not change with respect to the fine mesh size and is independent of the number of processors. For the linear solver, the number of linear iterations

Table II. Effect of the different subdomain solvers for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $\gamma = 0.1$, $Re = 200$, 512×512 grid, $\delta = 2$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). "$--$" means not enough memory.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
		ILU(2)			ILU(4)			LU	
					One-level				
64	2.8	907.4	2952.0	2.8	584.1	2471.6	2.8	247.1	3549.9
128	2.8	1005.0	1703.9	2.8	682.9	1484.3	2.8	383.1	2109.2
256	2.8	1124.4	980.7	2.8	784.8	891.0	2.8	493.9	1009.8
512	2.8	1295.9	614.4	2.8	986.3	595.0	2.8	765.8	692.2
1024	2.8	1284.0	324.9	2.8	1054.4	344.8	2.8	878.6	364.3
2048	2.8	1645.9	219.8	2.8	1353.9	233.4	2.8	1279.7	269.0
					Two-level				
64	2.8	47.1	1079.2	2.8	37.8	1158.6		$--$	
128	2.8	47.7	605.2	2.8	38.9	647.3	2.8	36.3	1565.9
256	2.8	47.8	336.8	2.8	39.9	359.0	2.8	37.3	610.5
512	2.8	48.4	220.4	2.8	42.0	230.9	2.8	38.3	333.8
1024	2.8	49.3	139.4	2.8	43.3	143.7	2.8	38.8	164.3
2048	2.8	52.0	101.3	2.8	47.8	103.2	2.8	49.6	121.8

Table III. Effect of the time steps for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\delta = 2$.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
		$\Delta t = 0.1$			$\Delta t = 0.05$			$\Delta t = 0.025$	
					One-level				
64	2.8	709.5	2599.6	2.4	455.4	2949.1	2.1	310.7	3877.8
128	2.8	787.3	1491.9	2.4	520.6	1728.0	2.1	342.4	2188.5
256	2.8	896.2	893.3	2.4	601.2	1042.2	2.1	378.5	1259.9
512	2.8	1098.5	581.8	2.4	715.5	657.1	2.1	460.5	802.8
1024	2.8	1164.8	335.2	2.4	781.8	390.0	2.1	519.5	490.6
2048	2.8	1465.3	220.6	2.4	1046.3	276.7	2.1	709.0	342.1
					Two-level				
64	2.8	38.4	1026.0	2.4	38.0	1624.2	2.1	36.8	2806.1
128	2.8	38.8	573.1	2.4	38.5	894.7	2.1	36.8	1521.5
256	2.8	40.1	323.5	2.4	39.4	497.0	2.1	37.9	837.2
512	2.8	41.6	212.5	2.4	41.0	310.8	2.1	39.2	510.2
1024	2.8	43.5	136.8	2.4	42.9	197.2	2.1	41.7	314.2
2048	2.8	45.5	93.1	2.4	45.2	133.5	2.1	43.8	214.7

grows with the the number of processors, which is expected from the convergence theory of one-level domain decomposition methods. From Table IV, we also observe: (1) for the linear solver, the number of iterations for the one-level LNKSz is much larger than that for the two-level LNKSz, and the number of iterations for the one-level LNKSz grows rapidly with the mesh refinement while the two-level LNKSz is fairly stable; (2) compared with the one-level LNKSz, the total computing time of the two-level LNKSz is much smaller, specially for the cases of 1024×1024 grid. The two-level method is clearly more attractive for large scale calculations.

In order to study the impact of the Reynolds number on the performance of LNKSz, in the next experiment we increase Re to 400 and keep all other values unchanged. Results are summarized in

Table IV. Effect of the mesh size for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $\gamma = 0.1$, $Re = 200$, ILU(3), $\delta = 2$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). The total degrees of freedom for the two grids are $2,097,152$ and $8,388,608$, respectively. "$--$" means not enough memory.

N_p	IN	RAS	RUN	IN	RAS	RUN
	512×512			1024×1024		
			One-level			
64	2.8	709.5	2599.6		$--$	
128	2.8	787.3	1491.9	2.8	2179.1	14972.7
256	2.8	896.2	893.3	2.8	2343.1	8066.1
512	2.8	1098.5	581.8	2.8	2666.7	4882.2
1024	2.8	1164.8	335.2	2.8	2610.2	2454.2
2048	2.8	1465.3	220.6	2.8	3520.9	1797.0
			Two-level			
64	2.8	38.4	1026.0		$--$	
128	2.8	38.8	573.1		$--$	
256	2.8	40.1	323.5	2.8	58.6	1737.8
512	2.8	41.6	212.5	2.8	59.0	994.7
1024	2.8	43.5	136.8	2.8	61.4	602.7
2048	2.8	45.5	93.1	2.8	63.5	464.4

Table V. Comparing it to Table IV, we see that the average number of Newton iterations, the average number of linear iterations and the total computing time become larger.

Table V. Effect of a larger Reynolds number for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $\gamma = 0.1$, $Re = 400$, ILU(3), $\delta = 2$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). The total degrees of freedom for the two grids are $2,097,152$ and $8,388,608$, respectively. "$--$" means not enough memory.

N_p	IN	RAS	RUN	IN	RAS	RUN
	512×512			1024×1024		
			One-level			
64	3.0	747.9	2920.8		$--$	
128	3.0	857.0	1728.1	3.0	2103.2	15492.5
256	3.0	985.7	1044.8	3.0	2361.4	8783.9
512	3.0	1291.8	726.0	3.0	2839.5	5560.5
1024	3.0	1337.8	409.3	3.0	3102.6	3109.6
2048	3.0	1738.6	278.3	3.0	4221.1	2296.7
			Two-level			
64	3.0	50.2	1347.5		$--$	
128	3.0	51.7	764.0		$--$	
256	3.0	53.6	433.6	3.0	70.9	2177.5
512	3.0	56.0	283.8	3.0	70.0	1251.8
1024	3.0	58.8	187.7	3.0	72.5	750.8
2048	3.0	58.9	130.7	3.0	75.9	569.3

The difficulty of the control problem changes with the regularization parameter γ. Table VI shows the effect of γ on the performance of the one-level and two-level LNKSz, for fixed $Re = 200$, 512×512 grid, $\Delta t = 0.1$ (i.e., there are 10 time steps), ILU(3), and $\delta = 2$. We observe that, as γ decreases, the average numbers of Newton and linear iterations become larger and the total computing time increases. In other words, the control problem is more difficult to solve for smaller

γ values. In practice the value of γ can not be too small since the size of a control is limited by technological constraints, if the size of the control (measured in an appropriate norm) is not a priori constrained to be within some specified bounds, optimal controls found as solutions of the optimization problem are usually unbounded, and therefore not physically realizable. More discussions about this issue can be found in references [12, 15, 19, 27].

Table VI. Effect of the parameter γ for the flow matching problem (16)-(17) by using the one-level and two-level LNKSz. $Re = 200$, 512×512 grid, ILU(3), $\delta = 2$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). "$++$" means the divergence of GMRES.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
		$\gamma = 1$			$\gamma = 0.1$			$\gamma = 0.01$	
					One-level				
64	2.7	596.7	2148.6	2.8	709.5	2599.6	3.1	1014.4	3996.9
128	2.7	691.6	1279.3	2.8	787.3	1491.9	3.1	945.7	1954.8
256	2.7	793.0	769.0	2.8	896.2	893.3	3.1	1046.4	1141.4
512	2.7	983.1	505.8	2.8	1098.5	581.8	3.1	995.6	587.5
1024	2.7	1044.7	292.0	2.8	1164.8	335.2	3.1	1091.6	349.7
2048	2.7	1357.3	197.8	2.8	1465.3	220.6		$++$	
					Two-level				
64	2.7	31.7	872.4	2.8	38.4	1026.0	3.1	65.2	1756.5
128	2.7	31.1	477.8	2.8	38.8	573.1	3.1	67.5	996.1
256	2.7	32.4	268.5	2.8	40.1	323.5	3.1	70.1	564.8
512	2.7	33.0	173.0	2.8	41.6	212.5	3.1	77.4	436.0
1024	2.7	34.7	113.2	2.8	43.5	136.8	3.1	80.7	257.3
2048	2.7	37.3	78.4	2.8	45.5	93.1	3.1	79.7	160.3

Figures 3 and 4 present the velocity field of the controlled and target flows at several different times. The target flow at $t = 0$ is not at rest and its direction of motion is anti-clockwise, while the trajectory of the flow that we want to control opposites to the target flow, resulting in increasing the difficulty of the control problem. By using distributed control, we can control the velocity in the each point of the domain and find that this kind of control can be effective for the flow matching problem. From Figures 3 and 4, we can see that the controlled flow keeps the opposite of the target flow at $t = 0.2$ and $t = 0.4$. But, with the help of the distributed control, the trajectory of the fluid becomes the same as the target velocity at $t = 0.6$, and the controlled flow is further improvement to the optimal flow pattern at $t = 0.8$ and $t = 1.0$. Figure 5 presents the computed external force at several different times. As shown in Figure 5, the quiver of the computing external force is also anti-clockwise from beginning to end, in order to make the controlled flow as close to the target flow as possible.

We next consider the parallel scalability issue of the one-level and two-level methods. Figure 6 shows the speedup and the total computing time for the one-level and two-level LNKSz, with a fixed $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\Delta t = 0.1$ (i.e., there are 10 time steps). The number of processors is varied from 64 to 2048 and the time for the case of $N_{p,1} = 64$ is taken as a reference timing. When N_p increases from 64 to 2048, the total computing time decreases at a reasonably good rate, which indicates that LNKSz has a good speedup for this range of processor counts. Observing from Figure 6, we highlight that: in comparison with the one-level method, the total computing time of the two-level method is much smaller.

Figure 7 shows the efficiency and the average number of linear iterations for the one-level and two-level LNKSz, with a fixed $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\Delta t = 0.1$ (i.e., there are 10 time steps). The left figure is for the parallel efficiency and the right figure is for the average number of linear iterations. The one-level and two-level methods have similar parallel efficiency, which drops to about 35% when the number of processors reaches 2048. The number of linear iterations increases a lot for the one-level method, and stay more or less the same for the two-level method.

In summary, for the flow matching problem, the two-level method results in a very sharp reduction in the number of linear iterations and a good reduction in compute time and is much more effective than the one-level method. The experiments in the subsection also show that the two-level LNKSz has an excellent linear and nonlinear convergence and is more robust than the one-level LNKSz with respect to certain parameters such as the Reynolds number, the parameter γ, the mesh size, and the number of processors.

4.4. *The two-grid inexact Newton method*

In this subsection, we present some numerical results by using the two-grid inexact Newton method in which a coarse grid is used in the nonlinear solver for generating a better initial guess and also in the linear solver for generating a part of the two-level Schwarz preconditioner. We report results for the flow matching problem (16)-(17) and the backward-facing step flow control problem (18)-(19). We show that the numerical behavior of the two-grid approach is better than that of the one-level and two-level approaches reported in the previous subsection. For all the numerical tests in this subsection, the subdomain solver is fixed to LU and the regularization parameter is set to $\gamma = 0.1$.

4.4.1. The flow matching problem. First, we present results for the flow matching problem (16)-(17) by using one-level, two-level, and two-grid methods. In this subsection, the overlapping sizes of the coarse grid and the fine grid are $\delta_c = 2$ and $\delta = 2$, respectively, and the relative (absolute) solver tolerance $\eta_r^c = 10^{-1}$ ($\eta_a^c = 10^{-2}$) is set for the linear iteration on the coarse grid of two-level proconditioners.

Table VII shows some results obtained with the three methods when we increase the number of processors from 64 to 2048 and refine the fine grid size from 512×512 to 1024×1024. It is clear that the performance of the two-grid method is better than that of the one-level and two-level methods, and the two-grid LNKSz takes fewer number of inexact Newton iterations. Table VIII shows the effect of Δt on the performance of the two-grid method. On the nonlinear solver, the number of Newton iterations is independent of Δt and the number of processors. For the linear solver, the number of linear iterations stays near a constant as Δt changes. Also, the computing time increases when Δt is reduced since the total number of time steps is larger. From Tables VII and VIII, we see that the two-grid LNKSz converges well for different time steps and mesh sizes, and is unconditionally stable, which is a big improvement in terms of Newton iterations and the computing time over the single grid method.

Similarly to the two-level method, comparing to the results of the one-level preconditioner, the performance of the two-grid method is much better. As we increase the number of processors from 64 to 2048, for the two-grid method the average number of linear iterations per Newton step stays small, while for the one-level method the average number increases quickly. Similar results are observed in the terms of the computing time. More importantly, the performance of the two-grid method is better than that of the two-level method with respect to Newton iterations and the computing time.

4.4.2. The backward-facing step flow control problem. In the following, we consider the backward-facing step control problem (18)-(19) and discuss some more details of the two-grid method. In the one-level method, the overlapping size is $\delta = 6$. In the two-level and two-grid methods, the overlapping sizes of the coarse grid and the fine grids are $\delta_c = 4$ and $\delta = 6$, respectively. The relative (absolute) solver tolerance $\eta_r^c = 10^{-2}$ ($\eta_a^c = 10^{-4}$) is set for the linear iteration on the coarse grid of two-level proconditioners for this case.

First, we compare the one-level, two-level and two-grid methods for the backward-facing step control problem (18)-(19) in Table IX. Note that, the one-level method doesn't converge when $N_p = 1024$, which is caused by the divergence of GMRES. Moreover, we note that: (1) for the linear solver, the number of linear iterations for the one-level LNKSz is much larger than that for the two-level and two-grid methods; (2) for the nonlinear solver, the numbers of Newton iterations for the

Table VII. A comparison of the three methods for the flow matching problem (16)-(17) with respect to the fine grid size. $Re = 200$, LU, $\delta_c = 2$, $\delta = 2$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). "$--$" means not enough memory.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
	One-level			Two-level			Two-grid		
				512×512 grid					
64	2.8	247.1	3549.9		$--$			$--$	
128	2.8	383.1	2109.2	2.8	36.2	1565.9	2.0	48.6	1330.5
256	2.8	493.9	1009.8	2.8	37.3	610.5	2.0	49.4	528.1
512	2.8	765.8	692.2	2.8	38.3	333.8	2.0	52.0	296.4
1024	2.8	878.6	364.3	2.8	38.8	164.3	2.0	51.8	149.8
2048	2.8	1279.7	1269.0	2.8	49.6	121.8	2.0	74.6	119.3
				1024×1024 grid					
256	2.8	691.5	6560.3		$--$			$--$	
512	2.8	1046.9	4331.0	2.8	46.3	2037.5	2.0	53.8	1603.1
1024	2.8	1230.4	2149.0	2.8	47.2	855.1	2.0	54.5	692.3
2048	2.8	1942.6	1643.3	2.8	49.6	557.9	2.0	56.7	433.2

Table VIII. Effect of the time steps for the flow matching problem (16)-(17) by using the two-grid method. $Re = 200$, 512×512 grid, LU, $\delta_c = 2$, $\delta = 2$, $T = 1$. The tests with processors $N_p = 64$ are not carried out because of the lack of memory.

N_p	IN	RAS	RUN	IN	RAS	RUN	IN	RAS	RUN
	$\Delta t = 0.1$			$\Delta t = 0.05$			$\Delta t = 0.025$		
128	2.0	48.6	1330.5	2.0	49.2	2592.4	2.0	48.4	5060.4
256	2.0	49.4	528.1	2.0	48.9	1003.3	2.0	48.3	1953.5
512	2.0	52.0	296.4	2.0	52.3	554.9	2.0	51.9	1063.9
1024	2.0	51.8	149.8	2.0	51.6	270.3	2.0	52.6	516.1
2048	2.0	74.6	119.3	2.0	75.3	220.1	2.0	75.4	413.3

one-level and two-level methods are also larger than that for the two-grid method; and (3) compared with the one-level and two-level methods, the total computing time for the two-grid method is much smaller. When the Reynolds number increases from 200 to 400, for the one-level and two-level methods, the average number of Newton iterations and the total computing time become larger. With the help of grid-sequencing, the convergence of the two-grid method is less sensitive to the Reynolds number. Based on the results of Table IX, it is clear that the two-grid method is better than the others.

An important implementation detail to consider in designing two-grid LNKSz is to balance the quality of the initial guess for the fine grid Newton iterations and the computing time spent on the coarse grid nonlinear solver. In Table X, we present a comparison of the computing time for the two-level and two-grid methods. In this table, we report the total time spent on the Newton iterations at several time steps, the time spent on the coarse grid Newton iterations, and the percentage between these two values. We observe that the cost of the coarse grid Newton iterations is very small compared with the total computational cost. It is important to note that the coarse grid has to be sufficiently fine so that the coarse solution has a reasonable accuracy, otherwise, it won't be able to provide a good initial guess for the fine grid nonlinear solver.

One of the difficulties in the nonlinear solver is the choice of the initial guess. In Figure 8, we show the nonlinear residual history by using three different methods at the first time step (i.e., $k = 1$). One can see that the nonlinear system is difficult to solve by using the one-level or two-level

Table IX. A comparison of the three methods for the backward-facing step control problem (18)-(19). 768×128 grid, LU, $\delta_c = 4$, $\delta = 6$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). " $++$" means the divergence of GMRES.

N_p	Method	IN	RAS	RUN	IN	RAS	RUN
		Re=200			Re=400		
64	One-level	3.2	165.4	1370.4	3.7	158.9	1557.5
64	Two-level	3.2	20.4	1342.8	3.7	19.2	1528.0
64	Two-grid	2.1	18.7	898.2	2.0	18.0	836.4
256	One-level	3.2	531.3	795.5	3.7	632.9	1052.3
256	Two-level	3.2	27.4	479.9	3.7	27.1	560.1
256	Two-grid	2.1	25.5	317.5	2.0	26.1	313.2
1024	One-level		$++$			$++$	
1024	Two-level	3.2	66.3	314.3	3.7	67.9	376.9
1024	Two-grid	2.1	64.2	208.5	2.0	68.5	209.8

Table X. A comparison of the computing time for the backward-facing step control problem (18)-(19). $Re = 400$, 768×128 grid, LU, $\delta_c = 4$, $\delta = 6$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). The heading "Timestep(k)" represents the time step k, "Time" is the total time spent on the the Newton iteration at the time step k, "Coarse time" is the time spent on the Newton iteration on the coarse solver as a fraction of total time at the time step k, and "Percent(%)" is the percentage of "Coarse time"/"Time".

N_p	Timestep(k)	Time	Coarse time	Percent(%)	Time
		Two-grid			Two-level
64	$k = 1$	110.0	3.87	3.52%	458.9
64	$k = 2$	80.0	2.39	2.99%	117.0
64	$k = 5$	82.5	2.50	3.03%	118.0
64	$k = 10$	84.7	2.51	2.96%	119.0
256	$k = 1$	38.6	1.71	4.43%	172.8
256	$k = 2$	29.7	0.99	3.33%	41.4
256	$k = 5$	30.0	1.04	3.43%	41.6
256	$k = 10$	30.8	1.06	3.44%	42.3
1024	$k = 1$	23.3	1.37	5.88%	115.1
1024	$k = 2$	20.6	0.68	3.30%	28.1
1024	$k = 5$	21.2	0.72	3.39%	28.4
1024	$k = 10$	21.5	0.74	3.44%	30.8

method. In fact, it takes 11 iterations for the one-level or two-level method to converge. By using the two-grid method only 3 Newton iterations are required to satisfy the desired stopping condition.

To see the major difference among the three methods, in Figure 9 we show the linear, nonlinear iterations, and the total computing time for the backward-facing step control problem (18)-(19) on a 768×128 grid and 256 processors for 10 time steps. The most difficult part of the computation is at the first time step. The one-level and two-level methods take lots of Newton iterations to finally find the solution, while with the help of grid-sequencing, the two-grid method converges quickly. Note that, in the upper left drawing of Figure 9, since the Newton iterations for the one-level and two-level methods are exactly the same, the corresponding lines coincide with each other. Similar results appear in the upper right drawing of Figure 9, where the numbers of GMRES iterations for the two-level and two-grid methods are very close to each other.

5. CONCLUSIONS

In this paper, we developed a family of parallel, fully implicit, fully coupled, two-grid algorithms for the distributed suboptimal control of unsteady incompressible flows governed by the Navier-Stokes equations. With the help of the two-grid Newton method and the two-level multiplicative-type Schwarz preconditioner, we showed numerically that our proposed algorithms have a fast and robust convergence and the rate of convergence is nearly independent of the number of unknowns of the problem, the time steps, the number of processors, and the Reynolds numbers. Good results were obtained for solving a flow matching problem and a backward-facing step flow problem with millions of unknowns and on a parallel machine with up to 2048 processors. Our future research includes the extension of the methods to some more complicated models, such as the optimal control of thermally convected fluid flows [12, 19].

ACKNOWLEDGEMENT

This research was supported in part by DOE under grant DE-SC0001774, and in part by NSF under grant DMS-0913089. H. Yang was also supported in part by the TianYuan Special Funds of NSFC under grant 11126196, in part by HNNSF under grant 12JJ4002, and by the Planned Science and Technology Project of Hunan Province under grant 2011GK3135. The authors would like to thank the anonymous referees for their valuable suggestions to improve the paper.

REFERENCES

1. Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley M, McInnes LC, Smith BF, Zhang H. *PETSc Users Manual*. Argonne National Laboratory: Illinois, 2011.
2. Bewley TR, Temam R, Ziane M. A general framework for robust control in fluid mechanics. *Physica D* 2000; **138**:360–392.
3. Bewley TR. Flow control: new challenges for a new renaissance. *Progress in Aerospace Sciences* 2001; **37**:21–58.
4. Biros G, Ghattas O. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization, part I: The Krylov-Schur solver. *SIAM Journal on Scientific Computing* 2005; **27**:687–713.
5. Biros G, Ghattas O. Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization, part II: The Lagrange-Newton solver and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing* 2005; **27**:714–739.
6. Briggs WL, Henson VE, McCormick SF. *A Multigrid Tutorial*. 2nd ed. SIAM, 2000.
7. Cai X-C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **21**:92–797.
8. Coleman TF, Moré JJ. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis* 1983; **20**:187–209.
9. Dennis JE, Schnabel RB. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM: Philadelphia, 1996.
10. Eisenstat SC, Walker HF. Globally convergent inexact Newton method. *SIAM Journal on Optimization* 1994; **4**:393–422.
11. Gunzburger M, Hou L, Svobodny T. Analysis and finite element approximation of optimal control problems for the stationary Navier-Stokes equations with distributed and Neumann controls. *Mathematics of Computation* 1991; **57**:123–151.
12. Gunzburger M, Hou L, Svobodny T. The approximation of boundary control problems for fluid flows with an application to control by heating and cooling. *Computers & Fluids* 1993; **22**:239–251.
13. Gunzburger M, Manservisi S. The velocity tracking problem for Navier-Stokes flows with bounded distributed control. *SIAM Journal on Control and Optimization* 1999; **37**:1913–1945.
14. Gunzburger MD, Manservisi S. The velocity tracking problem for Navier-Stokes flow with boundary control. *SIAM Journal on Numerical Analysis* 2000; **39**:594–634.
15. Gunzburger MD, Manservisi S. Analysis and approximation of the velocity tracking problem for Navier-Stokes flows with distributed control. *SIAM Journal on Numerical Analysis* 2000; **37**:481–1512.
16. Gunzburger MD. *Perspectives in Flow Control and Optimization*. 1st ed. SIAM, 2003.
17. Hairer E, Nørsett S, Wanner G. *Solving Ordinary Differential Equations I*. Springer-Verlag, 1993.
18. Hintermüller M, Hinze M. Globalization of SQP-methods in control of the instationary Navier-Stokes equations. *Mathematical Modelling and Numerical Analysis* 2002; **36**:725–746.
19. Ito K, Ravidran SS. Optimal control of thermally convected fluid flows. *SIAM Journal on Scientific Computing* 1998; **19**:1847–1869.
20. Manservisi S. An extended domain method for optimal boundary control for Navier-Stokes equations. *International Journal of Numerical Analysis and Modeling* 2007; **4**:584-607.
21. Moin P, Bewley TR. Feedback control of turbulence. *Applied Mechanics Reviews* 1994; **47**:3–13.
22. Prudencio E, Byrd R, Cai X-C. Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems. *SIAM Journal on Scientific Computing* 2006; **27**:1305–1328.

23. Prudencio E, Cai X-C. Parallel multilevel restricted Schwarz preconditioners with pollution removing for PDE-constrained optimization. *SIAM Journal on Scientific Computing* 2007; **29**:964–985.
24. Quartapelle L. *Numerical Solution of the Incompressible Navier-Stokes Equations*. International Series of Numerical Mathematics, Vol. 113, Birkhäuser Verlag, 1996.
25. Ravindran SS. Adaptive reduced-order controllers for a thermal flow system. *SIAM Journal on Scientific Computing* 2002; **23**:1925–1943.
26. Ravindran SS. Adaptive reduced-order controllers for a thermal flow system using proper orthogonal decomposition. *SIAM Journal on Scientific Computing* 2002; **23**:1924–1942.
27. Ravindran SS. Numerical approximation of optimal control of unsteady flows using SQP and time decomposition. *International Journal for Numerical Methods in Fluids* 2004; **45**:21–42.
28. Saad Y. *Iterative Methods for Sparse Linear Systems*. 2nd ed. SIAM: Philadelphia, 2003.
29. Smith B, Bjørstad P, Gropp W. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.
30. Toselli A, Widlund O. *Domain Decomposition Methods-Algorithms and Theory*. Springe: Berlin, 2005.
31. Yang H, Prudencio E, Cai X-C. Fully implicit Lagrange-Newton-Krylov-Schwarz algorithms for boundary control of unsteady incompressible flows. *International Journal for Numerical Methods in Engineering* (2012); **91**:644–665.
32. Yang H, Cai X-C. Scalable parallel algorithms for boundary control of thermally convective flows. *13th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing*, 2012.

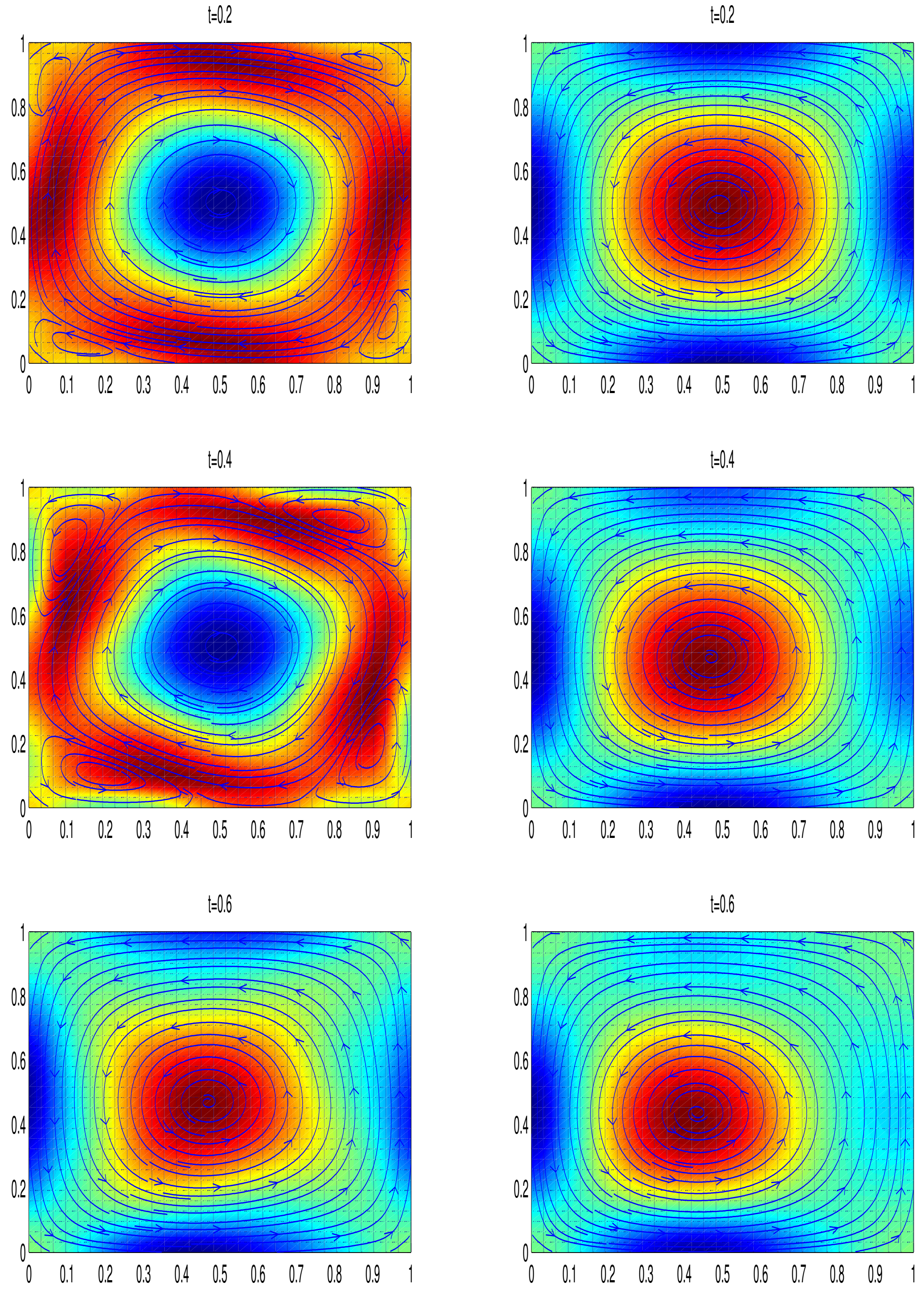

Figure 3. The streamslice and contour drawing for flow matching problem (16)-(17): the velocity field of the controlled (left column) and target (right column) flows at several different times, for fixed $\gamma = 0.01$, $Re = 200$, $T = 1.0$, and $\Delta t = 0.2$ (i.e, there are 5 time steps). The first, second and third rows correspond to $t = 0.2$, $t = 0.4$ and $t = 0.6$, respectively. In the figure, the colored part is the contour drawing for the vorticity ω and the streamline part is for the velocity field $\mathbf{v}$.

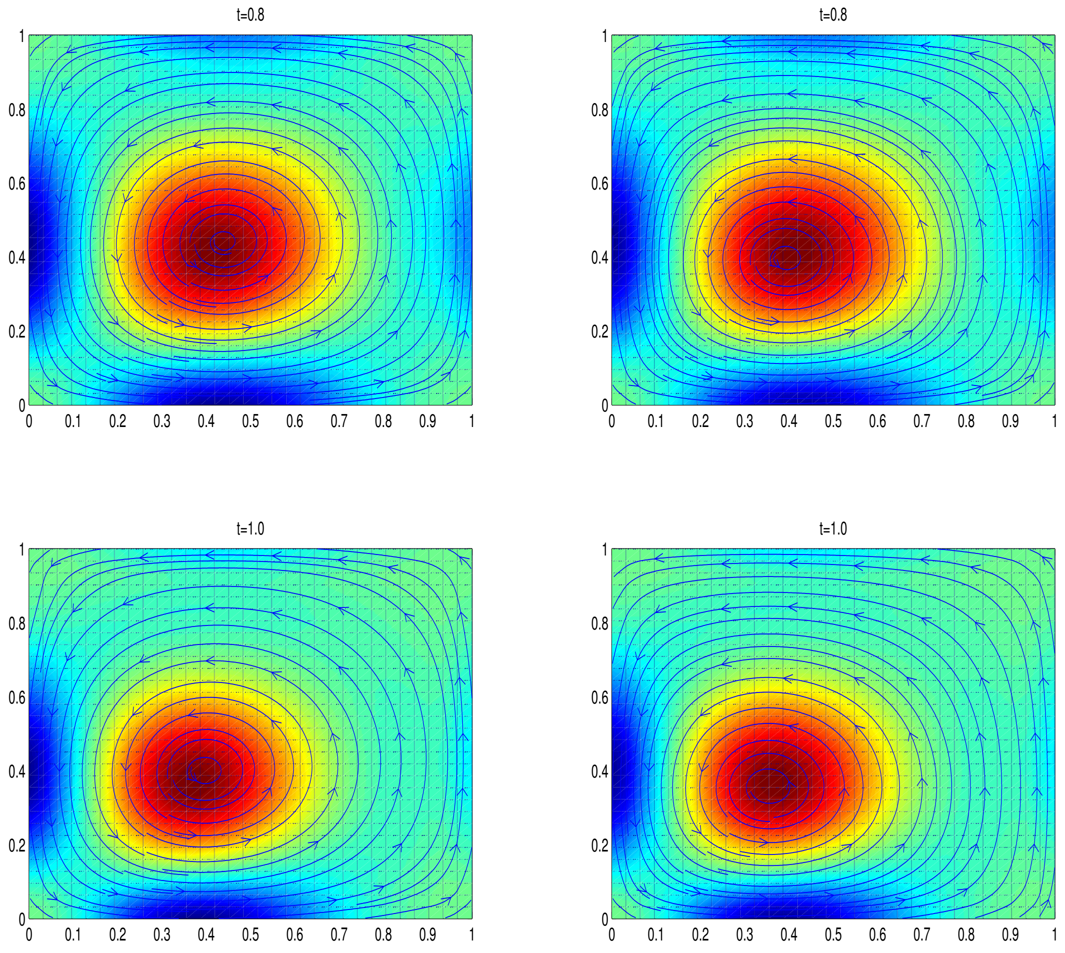

Figure 4. The streamslice and contour drawing for flow matching problem (16)-(17): the velocity field of the controlled (left column) and target (right column) flows at several different times, for fixed $\gamma = 0.01$, $Re = 200$, and $\Delta t = 0.2$ (i.e, there are 5 time steps). The first and second rows correspond to $t = 0.8$ and $t = 1.0$, respectively. In the figure, the colored part is the contour drawing for the ω component and the streamline part is for the velocity field $\mathbf{v}$.

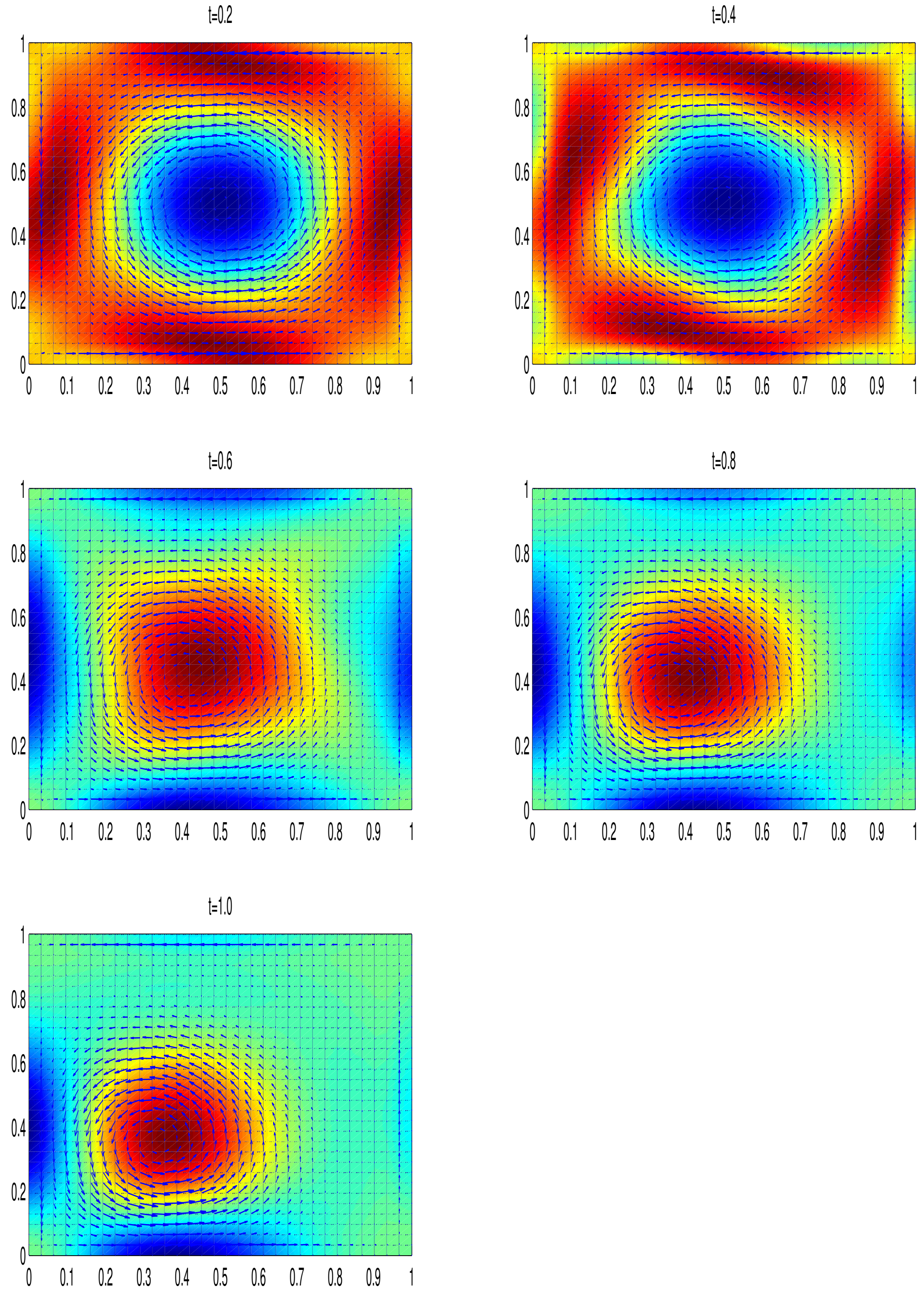

Figure 5. The quiver and contour drawing for flow matching problem (16)-(17): the computing external force $\mathbf{f}$ at several different times, for fixed $\gamma = 0.01$, $Re = 200$, and $\Delta t = 0.2$ (i.e, there are 5 time steps). The first row corresponds to $t = 0.2$ and 0.4. The second row corresponds to $t = 0.6$ and 0.8. The third row corresponds to $t = 1.0$. In the figure, the colored part is the contour drawing for curl $\mathbf{f}$ and the quiver part is for the computing external force $\mathbf{f}$.

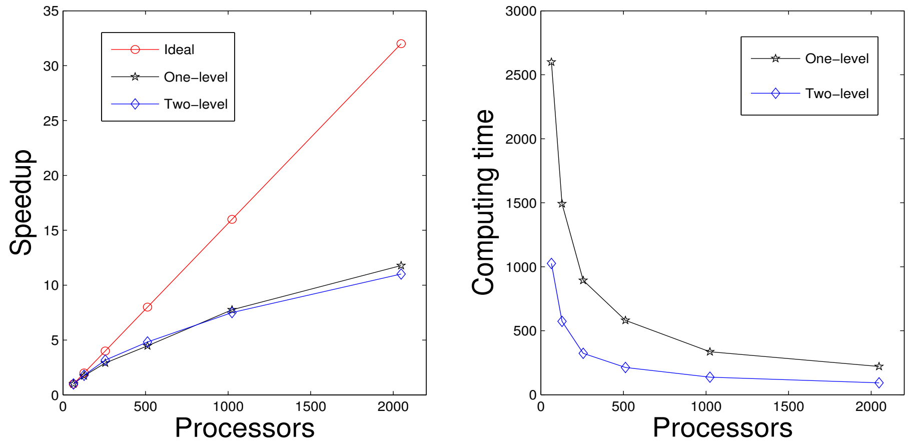

Figure 6. The speedup and the total computing time for the flow matching problem (16)-(17). $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\Delta t = 0.1$ (i.e., there are 10 time steps).

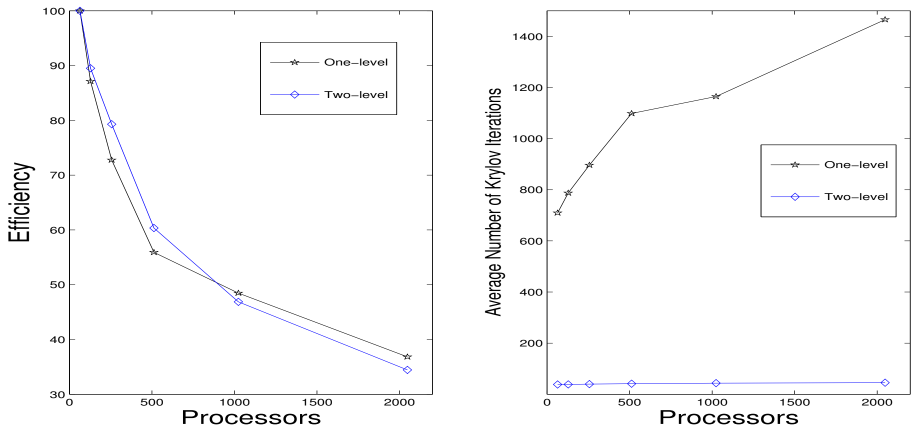

Figure 7. The efficiency and the average number of linear iterations for the flow matching problem (16)-(17). $\gamma = 0.1$, $Re = 200$, 512×512 grid, ILU(3), and $\Delta t = 0.1$ (i.e., there are 10 time steps)

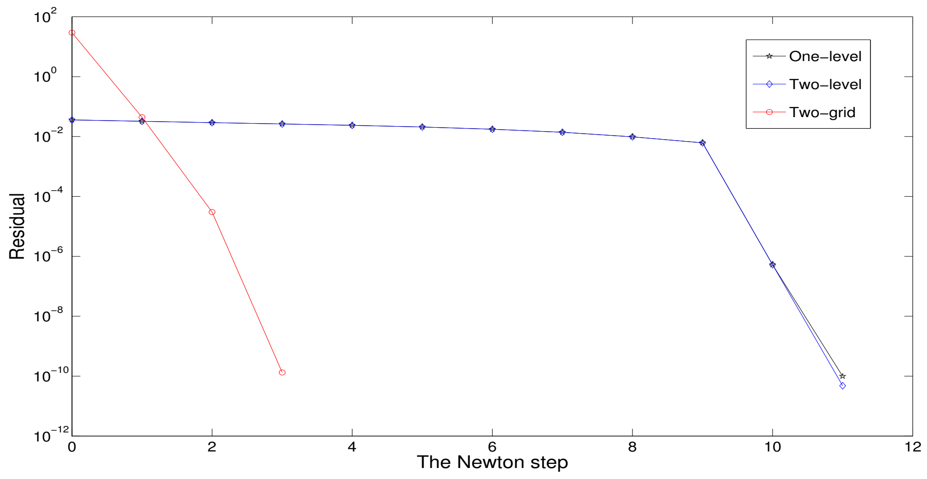

Figure 8. Nonlinear residual history for the backward-facing step control problem (18)-(19) using different methods for the first time step, for fixed $Re = 200$, 768×128 grid, 64 processors, LU, and $\Delta t = 0.1$ (i.e., there are 10 time steps).

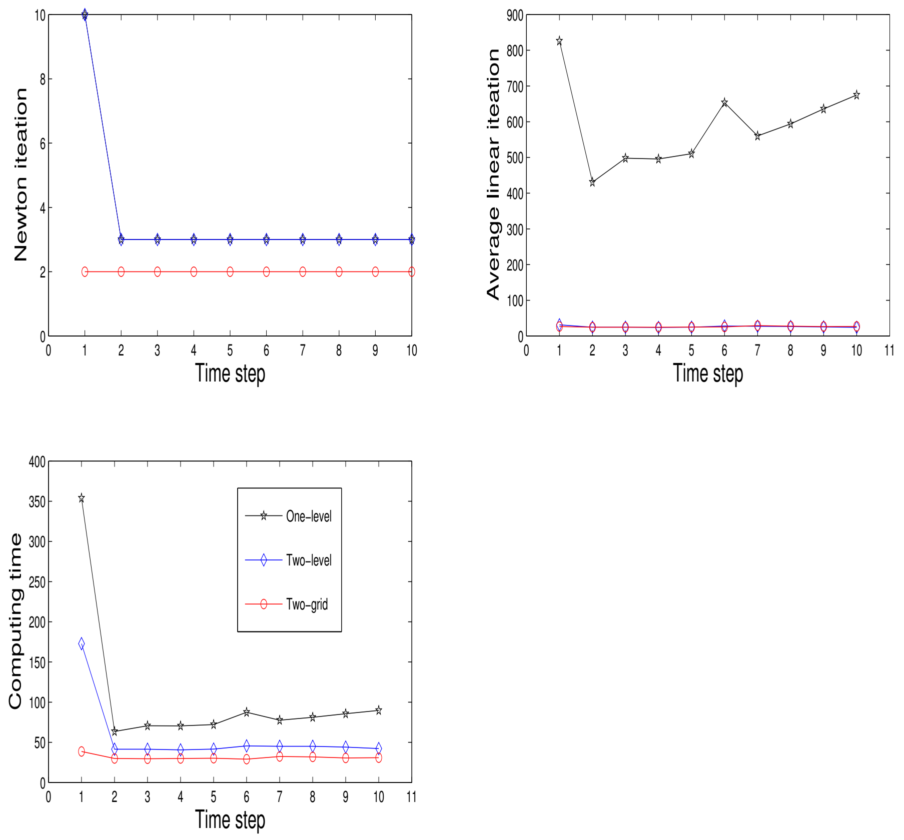

Figure 9. A comparison of the linear, nonlinear iterations, and the computing time for the backward-facing step control problem (18)-(19) by using three methods, for fixed $Re = 400$, 768×128 grid, 256 processors, LU, and $\Delta t = 0.1$ (i.e., there are 10 time steps).