INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 0000; **00**:1–19
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/nme

Fully implicit Lagrange-Newton-Krylov-Schwarz algorithms for boundary control of unsteady incompressible flows

Haijian Yang[1], Ernesto E. Prudencio[2] and Xiao-Chuan Cai[1*†]

[1]*Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA*
[2]*Institute for Computational Engineering and Sciences, University of Texas at Austin, Austin, TX 78712, USA*

SUMMARY

We develop a parallel fully implicit domain decomposition algorithm for solving optimization problems constrained by time dependent nonlinear partial differential equations. In particular, we study the boundary control of unsteady incompressible Navier-Stokes equations. After an implicit discretization in time, a fully coupled sparse nonlinear optimization problem needs to be solved at each time step. The class of full space Lagrange-Newton-Krylov-Schwarz (LNKSz) algorithms is used to solve the sequence of optimization problems. Among optimization algorithms, the fully implicit full space approach is considered to be the easiest to formulate and the hardest to solve. We show that LNKSz, with a one-level restricted additive Schwarz preconditioner, is an efficient class of methods for solving these hard problems. To demonstrate the scalability and robustness of the algorithm, we consider several problems with a wide range of Reynolds numbers and time step sizes, and we present numerical results for large scale calculations involving several millions unknowns obtained on machines with more than one thousand processors. Copyright © 0000 John Wiley & Sons, Ltd.

Received . . .

KEY WORDS: Schwarz preconditioner; parallel computing; nonlinear constrained optimization; inexact Newton; flow control; unsteady incompressible Navier-Stokes equations

1. INTRODUCTION

Flow control and optimization has many important applications in science and engineering [21]. Finding the right control is computationally expensive, especially when the flow is unsteady, and it often requires the use of large scale parallel computers. In this paper we investigate a parallel fully implicit domain decomposition method for the boundary control of unsteady incompressible Navier-Stokes flows. The most important issues under consideration are (1) whether the algorithm scales well when the number of processors is large; (2) whether the algorithm is stable and converges well with relatively large time steps; and (3) if the algorithm is robust with respect to some of physical parameters, such as the Reynolds number. Many constrained optimization problems can be written as

$$\begin{cases} \min\limits_{\mathbf{x}\in\mathbf{W}} & \mathcal{F}(\mathbf{x}) \\ \quad \text{s.t.} & \mathbf{C}(\mathbf{x}) = \mathbf{0} \in \mathbf{Y}, \end{cases} \tag{1}$$

where $\mathbf{W}$ and $\mathbf{Y}$ are normed spaces, $\mathbf{W}$ is the space of optimization variables, $\mathcal{F} : \mathbf{W} \to \mathbb{R}$ is the objective functional and $\mathbf{C} : \mathbf{W} \to \mathbf{Y}$ represents the constraints. When the constraints in (1)

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

are partial differential equations (PDE), there are two families of approaches for their numerical solution, generally speaking: optimize-then-discretize (OTD) and discretize-then-optimize (DTO). As far as we know, no approach has a clear advantage over the other [21]. We choose to use the DTO approach in this paper. We focus on the class of one-level Lagrange-Newton-Krylov-Schwarz (LNKSz) algorithms, introduced in [30], which aims for the robust, efficient and scalable parallel numerical solution of PDE-constrained optimization problems. Its application to some problems (1) with steady-state constraints has been already studied in [29, 30, 31].

In this paper we extend the LNKSz approach to problems where the constraints are initial boundary value problems defined for $t \in [t_0, t_f]$ over a bounded Lipschitz domain Ω, for given $t_f > t_0 \geqslant 0$. The discretization in time results in $k_{\text{max}} \geqslant 1$ successive time steps $\Delta t^{(k)} \equiv t^{(k+1)} - t^{(k)} > 0$, $k = 0, 1, \ldots, k_{\text{max}} - 1$, with $t^{(0)} = t_0$ and $t^{(k_{\text{max}})} = t_f$, and the discretization in space results in a mesh $\Omega_h^{(k)}$ of characteristic mesh size $h > 0$. For simplicity, we assume that both Ω and Ω_h do not change over time. There are two basic alternatives to consider. In the first alternative one aims for the solution of the entire problem at once for the whole time interval $[t_0, t_f]$, solving the finite dimensional optimization problem

$$\left\{ \begin{array}{rl} \min\limits_{\mathbf{x} \in \mathbf{W}_{k_{\text{max}},h}} & \mathcal{F}_{k_{\text{max}},h}(\mathbf{x}) \\ \text{s.t.} & \mathbf{C}_{k_{\text{max}},h}(\mathbf{x}) = \mathbf{0} \in \mathbf{Y}_{k_{\text{max}},h}, \end{array} \right. \tag{2}$$

where $\mathbf{W}_{k_{\text{max}},h} = \mathbb{R}^{k_{\text{max}} \times n_h}$ and $\mathbf{Y}_{k_{\text{max}},h} = \mathbb{R}^{k_{\text{max}} \times m_h} = \mathbf{Y}^*_{k_{\text{max}},h}$. This approach, however, can be too expensive computationally, even for the most modern massively parallel computers currently available. In the second alternative, one replaces (2) by a sequence of k_{max} similar subproblems, each having an objective function similar to the objective of the full formulation (2), except that it is now defined on the shorter time interval $[t^{(k)}, t^{(k+1)}]$, $k = 0, 1, \ldots, k_{\text{max}} - 1$. Each subproblem can be written as

$$\left\{ \begin{array}{rl} \min\limits_{\mathbf{x} \in \mathbf{W}_h^{(k)}} & \mathcal{F}_h^{(k)}(\mathbf{x}) \\ \text{s.t.} & \mathbf{C}_h^{(k)}(\mathbf{x}) = \mathbf{0} \in \mathbf{Y}_h^{(k)}. \end{array} \right. \tag{3}$$

where $\mathbf{W}_h^{(k)} = \mathbb{R}^{n_h}$ and $\mathbf{Y}_h^{(k)} = \mathbb{R}^{m_h} = \mathbf{Y}_h^{(k)*}$. Ignoring the sub- and super-scripts, we write the associated Lagrangian functional $\mathcal{L} : \mathbf{W} \times \mathbf{Y}^* \to \mathbb{R}$ as

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \equiv \mathcal{F}(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathbf{C}(\mathbf{x}) \rangle_{\mathbf{Y}}, \quad \forall\, (\mathbf{x}, \boldsymbol{\lambda}) \in \mathbf{W} \times \mathbf{Y}^*, \tag{4}$$

where $\mathbf{Y}^*$ is the adjoint space of $\mathbf{Y}$, $\langle \cdot, \cdot \rangle_{\mathbf{Y}}$ denotes the duality pairing and variables $\boldsymbol{\lambda}$ are called Lagrange multipliers or adjoint variables. In many cases it is possible to prove that, if $\hat{\mathbf{x}}$ is a (local) solution of (1) then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ such that $(\hat{\mathbf{x}}, \hat{\boldsymbol{\lambda}})$ is a critical point of $\mathcal{L}$. So, under sufficient smoothness assumptions, one proves that a solution has to necessarily solve a system of equations, called Karush-Kuhn-Tucker (KKT) condition or optimality system. Each subproblem (3) becomes then similar in size to a steady-state optimization problem, although time derivative terms will change the KKT Jacobian pattern w.r.t. steady-state Jacobian. In the context of control problems, this approach is often referred to as the suboptimal approach [4, 5, 27].

We briefly mention a few related publications that partially motivated our current work. A class of Lagrange-Newton-Krylov-Schur algorithms (LNKSr) is introduced in [6, 7] in which four block factorization based preconditioners, as well as some globalization techniques and heuristics, are proposed and tested. LNKSr attempts to transform the problem of finding a good preconditioner for the KKT problem to the problem of finding a good preconditioner for the linearized forward operator [30]. In [30, 31] the Schur type preconditioner is replaced by an overlapping Schwarz method which has a better asymptotic convergence rate. Note that these papers do not deal with time dependent control problems. In [1, 3] active feedback controls are used to enhance the stability in a 2D channel flow. Although the technique is very effective, the control is obtained by ‘experience’, not by computation. In other words, the control is not optimal. In [33], Ravindran proposes a sequential quadratic programming method (SQP) and time domain decomposition for time-dependent optimal control problems. At each time step the semi-implicit linear quadratic subproblem arising during the

SQP iteration is solved by using the corresponding first order necessary condition of the optimality condition, and some block iterative methods, such as the block Jacobi method and the successive over-relaxation method. Because of the limitations of these linear solvers, these methods can not be used for large scale parallel computation. Since the method is semi-implicit, it does not allow the use of large time steps. In [19, 20], Gunzburger and Manservisi present a gradient method for the time-dependent optimal control problem associated with the tracking of the velocity of a Navier-Stokes flow in a bounded two-dimensional domain through the adjustment of a distributed control. The authors focused on the mathematical analysis and showed that there are some time step size restrictions in the gradient method.

In this paper, we propose and investigate a class of parallel full space SQP LNKSz algorithms for the time-dependent boundary control of two-dimensional incompressible Navier-Stokes equations. The approach is a one-shot fully implicit method and is unconditionally stable with large time steps. In LNKSz, a Lagrangian functional is first formed using the objective function and all the constraints, and then differentiated to obtain a coupled KKT system of nonlinear equations consisting of all the state variables, the controls variables and the multipliers. These nonlinear systems are large, sparse and extremely difficult to solve. Inexact Newton method with line search is then applied to solve these large nonlinear systems. At each Newton iteration the linearized KKT system is solved with a one-level restricted additive Schwarz preconditioned Krylov subspace method. The critically important component of LNKSz is the restricted additive Schwarz preconditioner which was first introduced for scalar elliptic problems and is extended to the coupled systems in this paper. We show numerically that the restricted additive Schwarz preconditioner performs quite well for the fully coupled problem and on parallel machines with more than 1000 processors.

The rest of the paper is organized as follows. In Section 2, we present the unsteady flow control problem and a fully implicit discretization scheme. Section 3 describes the LNKSz algorithm. Section 4 is devoted to numerical experiments and the parallel performance of LNKSz. Final conclusions are given in Section 5.

2. FULLY IMPLICIT DISCRETIZATION OF THE UNSTEADY BOUNDARY CONTROL PROBLEM

Let $\Omega \subset \mathbb{R}^2$ be a bounded domain in the plane representing a region where a homogeneous incompressible Newtonian fluid is passing through during the time interval $[t_0, t_f]$. Let $\Gamma = \partial\Omega$ be the boundary of Ω, $\boldsymbol{\nu}$ the outward unit vector on Γ, $t \in [t_0, t_f]$ the time variable, $(x, y) \in \overline{\Omega}$ indicate a position in the domain, $\mathbf{v}(t, x, y) = (v_1(t, x, y), v_2(t, x, y))$ the fluid velocity, ω the fluid vorticity and $\mathbf{f} = (f_1, f_2)$ a given external force. Assume an initial velocity field $\mathbf{v}_0 = (v_{0,1}, v_{0,2})$ is given at $t = 0$, the corresponding initial vorticity field is then defined as

$$\omega_0 = \text{curl } \mathbf{v}_0 = -\frac{\partial v_{0,1}}{\partial y} + \frac{\partial v_{0,2}}{\partial x}.$$

The velocity-vorticity formulation of the incompressible Navier-Stokes equations consists of the following equations [32]

$$\begin{cases} -\Delta v_1 - \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ -\Delta v_2 + \dfrac{\partial \omega}{\partial x} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \dfrac{\partial \omega}{\partial t} - \dfrac{1}{Re}\Delta\omega + v_1\dfrac{\partial \omega}{\partial x} + v_2\dfrac{\partial \omega}{\partial y} - \text{curl } \mathbf{f} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \mathbf{v} - \mathbf{v}_D & = & \mathbf{0} & \text{on } [t_0, t_f] \times \Gamma, \\ \omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} & = & 0 & \text{on } [t_0, t_f] \times \Gamma, \\ \mathbf{v}(t_0, x, y) - \mathbf{v}_0 & = & \mathbf{0} & \text{in } \overline{\Omega}, \\ \omega(t_0, x, y) + \dfrac{\partial v_1}{\partial y}(t_0, x, y) - \dfrac{\partial v_2}{\partial x}(t_0, x, y) & = & 0 & \text{in } \overline{\Omega}, \end{cases} \tag{5}$$

where Re is the Reynolds number, curl $\mathbf{f} = -\partial f_1/\partial y + \partial f_2/\partial x$. The velocity profile $\mathbf{v}_D$ given on Γ is assumed to satisfy the principle of mass conservation; i.e.,

$$\int_\Gamma \mathbf{v}_D(t) \cdot \boldsymbol{\nu} \, d\Gamma = 0 \quad \text{for all} \quad t \in [t_0, t_f].$$

We assume $\mathbf{f} = \mathbf{0}$ for simplicity. In optimal control problems, the objective of interest is represented as a cost functional to be minimized. We first describe the optimal control problem, followed by the suboptimal control case. We refer to [15, 18, 19, 20, 23, 24] and references therein for details about the function spaces and the existence of an optimal solution of the optimal control problem.

Let us denote the state space by $\mathbf{S} = \{(v_1, v_2, \omega)\}$, the state variable by $\mathbf{s} = (s_1, s_2, s_3) = (v_1, v_2, \omega)$, the control space by $\mathbf{U} = \{(u_1, u_2)\}$, and the control variable is $\mathbf{u} = (u_1, u_2)$. The control is applied over the time interval $[t_0, t_f]$, either to track a desired flow field or to reduce the size of wake spread in the flow domain. It is also desirable that the least possible amount of control is applied. Given an initial velocity profile, the optimization problem consists on finding $(v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$\min_{(\mathbf{s},\mathbf{u}) \in \mathbf{S}\times\mathbf{U}} \mathcal{F}(\mathbf{s}, \mathbf{u}) = \int_{t_0}^{t_f} \theta(\mathbf{s}) \, dt + \frac{\gamma}{2} \int_{t_0}^{t_f} \int_{\Gamma_u} \|\mathbf{u}\|_2^2 \, d\Gamma \, dt. \tag{6}$$

is achieved subject to the constraints

$$\left\{ \begin{array}{lcll} -\Delta v_1 - \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ -\Delta v_2 + \dfrac{\partial \omega}{\partial x} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \dfrac{\partial \omega}{\partial t} - \dfrac{1}{Re}\Delta\omega + v_1 \dfrac{\partial \omega}{\partial x} + v_2 \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \mathbf{v} - \mathbf{u} & = & \mathbf{0} & \text{on } [t_0, t_f] \times \Gamma_u, \\ \mathbf{v} - \mathbf{v}_D & = & \mathbf{0} & \text{on } [t_0, t_f] \times \Gamma_c, \\ \omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} & = & 0 & \text{on } [t_0, t_f] \times \Gamma, \\ \mathbf{v}(t_0, x, y) - \mathbf{v}_0 & = & \mathbf{0} & \text{in } \overline{\Omega}, \\ \omega(t_0, x, y) + \dfrac{\partial v_1}{\partial y}(t_0, x, y) - \dfrac{\partial v_2}{\partial x}(t_0, x, y) & = & 0 & \text{in } \overline{\Omega}, \\ \displaystyle\int_{\Gamma_u} \mathbf{v} \cdot \boldsymbol{\nu} \, d\Gamma & = & 0 & \text{in } [t_0, t_f], \end{array} \right. \tag{7}$$

where $\theta(\mathbf{s})$ is a given application specific function of the state variables, $\Gamma = \Gamma_u \cup \Gamma_c$ is the boundary where Γ_u is part of the boundary on which the control is applied and Γ_c is part of the boundary on which the control is not applied, and $\gamma > 0$ is a constant parameter used to adjust the relative importance of the control norms in achieving the minimization, thus indirectly constraining their magnitudes. We remark that the last constraint in (7) is necessary for the consistency with the physical law of mass conservation. In other words, the control $\mathbf{u} = (u_1, u_2)$ can not be any control, it must belong to the space of functions satisfying the principle the mass conservation. Similarly to [30, 31], in this paper we study tangential boundary control problems. In other words, the control $\mathbf{u}$ is allowed to be just tangential to , i.e., $\mathbf{u} \cdot \boldsymbol{\nu} = 0$ on Γ_u, and the velocity $\mathbf{v}$ is assumed to satisfy $\int_{\Gamma_c} \mathbf{v} \cdot \boldsymbol{\nu} \, d\Gamma = 0$. The optimal control problems we consider can be described in a general manner: seek the boundary control $\mathbf{u}$ and state pair $(\mathbf{v}, \omega)$ such that the cost functional $\mathcal{F}$ is minimized subject to the constraints where the flow field satisfies the Navier-Stokes equations (7) over $[t_0, t_f]$.

For the description of a suboptimal control problem, we will utilize a second order backward differentiation formula for time discretization [22]. As described in the Introduction, the sequence of subproblems' objectives should be the same as the objective aimed with the optimal control problem (6)-(7). For $k = 0, 1, \ldots, k_{\text{max}} - 1$, let us denote the state space by $\mathbf{S}^{(k)} = \{(v_1^{(k)}, v_2^{(k)}, \omega^{(k)})\}$, the state variables by $\mathbf{s}^{(k)} = (v_1^{(k)}, v_2^{(k)}, \omega^{(k)})$, the control space by $\mathbf{U}^{(k)} = \{\mathbf{u}^{(k)}\}$ and the control variables by $\mathbf{u}^{(k)} = (u_1^{(k)}, u_2^{(k)})$. Let $\Delta t^{(k)} \equiv t^{(k+1)} - t^{(k)}$. Following [33] (not with respect to time discretization, though), a suboptimal control problem for the optimization problem is to find $(v_1^{(k+1)}, v_2^{(k+1)}, \omega^{(k+1)}, u_1^{(k+1)}, u_2^{(k+1)})$, for $k = 0, 1, \ldots, k_{\text{max}} - 1$, such that

$$\begin{aligned} \min_{(\mathbf{s}^{(k+1)},\mathbf{u}^{(k+1)})\in\mathbf{S}^{(k+1)}\times\mathbf{U}^{(k+1)}} \mathcal{F}^{(k+1)}(\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}) &= \Delta t^{(k)}\theta(\mathbf{s}^{(k+1)}) \\ &+ \frac{\gamma}{2}\Delta t^{(k)} \int_{\Gamma_u} \|\mathbf{u}^{(k+1)}\|_2^2 d\Gamma \end{aligned} \tag{8}$$

is achieved subject to the constraints

$$\begin{cases} -\Delta v_1^{(k+1)} - \dfrac{\partial\omega^{(k+1)}}{\partial y} = 0 & \text{in } \Omega, \\ -\Delta v_2^{(k+1)} + \dfrac{\partial\omega^{(k+1)}}{\partial x} = 0 & \text{in } \Omega, \\ \dfrac{1}{\Delta t^{(k)}}\left[\dfrac{3}{2}\omega^{(k+1)} - 2\omega^{(k)} + \dfrac{1}{2}\omega^{(k-1)}\right] & \\ \quad -\dfrac{1}{Re}\Delta\omega^{(k+1)} + v_1^{(k+1)}\dfrac{\partial\omega^{(k+1)}}{\partial x} + v_2^{(k+1)}\dfrac{\partial\omega^{(k+1)}}{\partial y} = 0 & \text{in } \Omega, \\ \mathbf{v}^{(k+1)} - \mathbf{u}^{(k+1)} = \mathbf{0} & \text{on } \Gamma_u, \\ \mathbf{v}^{(k+1)} - \mathbf{v}_D^{(k+1)} = \mathbf{0} & \text{on } \Gamma_c, \\ \omega^{(k+1)} + \dfrac{\partial v_1^{(k+1)}}{\partial y} - \dfrac{\partial v_2^{(k+1)}}{\partial x} = 0 & \text{on } \Gamma, \\ \displaystyle\int_{\Gamma_u} \mathbf{v}^{(k+1)} \cdot \boldsymbol{\nu}\, d\Gamma = 0, & \end{cases} \tag{9}$$

where, just for $k = 0$, the third constraint in (9) is replaced by the following equation

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

It should be noted that tangential control problems automatically satisfy the last constraint in (9), making the number of constraint equations equal to the number of state variables. For simplicity, we rewrite the suboptimal control problem (8)-(9) as

$$\begin{cases} \min\limits_{(\mathbf{s}^{(k+1)},\mathbf{u}^{(k+1)})\in\mathbf{S}^{(k+1)}\times\mathbf{U}^{(k+1)}} & \mathcal{F}^{(k+1)}(\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}) \\ \text{s.t.} & \mathbf{C}^{(k+1)}(\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}) = \mathbf{0} \in \mathbf{Y}^{(k+1)} \end{cases} \tag{10}$$

for the time interval $[t^{(k)}, t^{(k+1)}]$, $k = 0, 1, \ldots, k_{\text{max}} - 1$. We discretize the objective function and the constrains with a standard second-order five-point finite difference method on a uniform mesh in space. In order to simplify notations, we continue to use $\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}, \mathcal{F}^{(k+1)}$ and $\mathbf{C}^{(k+1)}$ to denote variables and values at nodes (ordered in some fashion). The Lagrangian functional $\mathcal{L}^{(k+1)} : \mathbf{s}^{(k+1)} \times \mathbf{u}^{(k+1)} \times \mathbf{Y}^{(\mathbf{k+1})^*} \to \mathbb{R}$ associated with the discrete version of (10) is defined by

$$\mathcal{L}^{(k+1)}(\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}, \lambda^{(k+1)}) = \mathcal{F}^{(k+1)} + \left\langle \lambda^{(k+1)}, \mathbf{C}^{(k+1)} \right\rangle \tag{11}$$

for $k = 0, 1, \ldots, k_{\text{max}} - 1$. Here $\lambda^{(k+1)}$ is a vector of Lagrange multipliers, and $\langle \cdot, \cdot \rangle$ represents the standard scalar product.

Let n_s, n_u, n_λ denote the number of unknowns with respect to the state variables, the control variables and the Lagrange multipliers, respectively, $N \equiv n_s + n_u + n_\lambda$ and $X^{(k+1)} \equiv (\mathbf{x}^{(k+1)}, \lambda^{(k+1)}) \equiv (\mathbf{s}^{(k+1)}, \mathbf{u}^{(k+1)}, \lambda^{(k+1)}) \in \mathbb{R}^N$. Then, for $k = 0, 1, \ldots, k_{\text{max}} - 1$, the KKT system obtained by differentiating (11) becomes

$$\begin{aligned} G^{(k+1)}(X^{(k+1)}) &= \begin{pmatrix} \nabla\mathcal{L}_{\mathbf{x}}^{(k+1)} \\ \nabla\mathcal{L}_{\lambda}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \nabla\mathcal{F}^{(k+1)} + \nabla\mathbf{C}^{(k+1)}\lambda^{(k+1)} \\ \mathbf{C}^{(k+1)} \end{pmatrix} \\ &= \begin{pmatrix} \nabla\mathcal{F}_{\mathbf{s}}^{(k+1)} + \nabla_{\mathbf{s}}\mathbf{C}^{(k+1)}\lambda^{(k+1)} \\ \nabla\mathcal{F}_{\mathbf{u}}^{(k+1)} + \nabla_{\mathbf{u}}\mathbf{C}^{(k+1)}\lambda^{(k+1)} \\ \mathbf{C}^{(k+1)} \end{pmatrix} = 0, \end{aligned} \tag{12}$$

where $G^{(k+1)}: \mathbb{R}^N \to \mathbb{R}^N$, $\nabla \mathcal{L}_{\mathbf{x}}^{(k+1)}$ denotes the gradient of $\mathcal{L}^{(k+1)}$ w.r.t the state and the control variables, $\nabla \mathcal{L}_{\lambda}^{(k+1)}, \nabla \mathcal{F}_{\mathbf{s}}^{(k+1)}$ and $\nabla \mathcal{F}_{\mathbf{u}}^{(k+1)}$ are defined in a similar way, $\nabla \mathbf{C}^{(k+1)}$ denotes the Jacobian of $\mathbf{C}^{(k+1)}$, and $\nabla_{\mathbf{s}} \mathbf{C}^{(k+1)}, \nabla_{\mathbf{u}} \mathbf{C}^{(k+1)}$ denote the Jacobian of $\mathbf{C}^{(k+1)}$ w.r.t the state and the control variables, respectively. Moreover, let $\varphi(\mathbf{u})$ be the discretization of $\int_{\Gamma_u} \|\mathbf{u}\|_2^2 \, d\Gamma$, then (12) can be rewritten as

$$G^{(k+1)}(X^{(k+1)}) = \begin{pmatrix} \Delta t^{(k)} \nabla_{\mathbf{s}} \theta^{(k+1)} + \nabla_{\mathbf{s}} \mathbf{C}^{(k+1)} \lambda^{(k+1)} \\ \frac{\gamma}{2} \Delta t^{(k)} \nabla_{\mathbf{u}} \varphi^{(k+1)} + \nabla_{\mathbf{u}} \mathbf{C}^{(k+1)} \lambda^{(k+1)} \\ \mathbf{C}^{(k+1)} \end{pmatrix} = 0,$$

where $\nabla_{\mathbf{u}} \varphi^{(k+1)}$ denotes the gradient of $\varphi(\mathbf{u}^{(k+1)})$ w.r.t the control variable $\mathbf{u}^{(k+1)}$. The Jacobian matrix of $G^{(k+1)}$ is the transpose of the Hessian of the Lagrangian $\mathcal{L}^{(k+1)}$, and has the following structure

$$\begin{pmatrix} \Delta t^{(k)} \nabla_{\mathbf{ss}} \theta^{(k+1)} + \nabla_{\mathbf{ss}} \mathbf{C}^{(k+1)} \lambda^{(k+1)} & 0 & \nabla_{\mathbf{s}} \mathbf{C}^{(k+1)} \\ 0 & \frac{\gamma}{2} \Delta t^{(k)} \nabla_{\mathbf{uu}} \varphi^{(k+1)} & \nabla_{\mathbf{u}} \mathbf{C}^{(k+1)} \\ \nabla_{\mathbf{s}} \mathbf{C}^{(k+1)} & \nabla_{\mathbf{u}} \mathbf{C}^{(k+1)} & 0 \end{pmatrix},$$

which is symmetric indefinite under sufficient smoothness assumptions and can be computed by a finite difference approximation. The zero block on the diagonal is problematic for our preconditioning algorithm and to make our approach work, we actually switch the first and the third rows without switching the first and the third columns in the above matrix [30]. This switch destroys the symmetry but allows good convergence of a preconditioned iterative method for non-symmetric systems. The preconditioning algorithm will be further discussed later in the paper.

In optimal control problems one tries to obtain the control $\mathbf{u}$ for the entire time interval $[t_0, t_f]$ and thus needs to solve a system of $N \times k_{\max}$ nonlinear equations. Suboptimal approaches, on the other hand, solve a sequence of $k_{\max}$ problems similar to the original problem, but each problem seeks a control only for the time interval $[t^{(k)}, t^{(k+1)}]$, $k = 0, 1, \ldots, k_{\max} - 1$. Each problem (10) then becomes similar in size to a steady-state optimization problem, although the time derivative term changes the KKT Jacobian pattern at each Newton iteration. In [33], Ravindran introduced a SQP method and time domain decomposition for time-dependent optimal control problems, using a first order time discretization based on a backward differentiation formula. The approach is semi-implicit and does not allow large time steps. As discussed earlier in this section, our method is a fully implicit second-order backward differentiation formula and we will show numerically that it allows large time steps. The main issue with fully implicit methods is that a large nonlinear system has to be solved at each time step. Next we discuss a class of full space one-level LNKSz methods for solving the discretized optimization problems.

3. FULL SPACE LAGRANGE-NEWTON-KRYLOV-SCHWARZ METHODS

There are two major families of Newton techniques for the solution of discretized KKT systems. The family of reduced space methods [16, 17, 25, 26, 36], which usually involves three basic steps: (1) the reformulation of the global problem into a much smaller set of equations (the so-called reduced system) defined only on the interface; (2) solve the reduced problem iteratively; (3) solve the global problem using the solution of the interface problem. Within each step, there is parallelism that can be explored, but the three steps have to be carried out sequentially. The family of full space methods [6, 7, 30, 31], in which all equations are solved simultaneously. As computers become more powerful in processing speed and memory capacity, full space methods seem to become more attractive due to their increased degree of parallelism and better scalability. There are many challenges, though, some of them related to the KKT Jacobian: it is usually ill-conditioned and indefinite, a property known to slow down Krylov solvers, and its order is more than twice bigger than the size of the forward problem. Therefore, a key element of a successful parallel full space approach is the preconditioner for the linearized KKT system: it has to be able to substantially reduce the condition number of the KKT Jacobian and, at the same time, provide the scalability for massively parallel computing.

LNKSz was introduced in [30, 31] for steady state problems. When the number of processors is small (such as 64) the one-level method works fine. A rather complicated two-level method, with a problem dependent pollution removing coarse solver, is necessary when the number of processors is slightly larger (such as 128). In this paper, we target unsteady problems, as it turns out the time step parameter Δt plays a very important role in the performance of LNKSz. Even with relatively large time step sizes, the one-level method scales quite well with a large number of processors. More will be discussed in the Numerical Experiments section.

In the following we describe the one-level LNKSz. For each $k = 0, 1, \ldots, k_{\text{max}} - 1$, from the Lagrangian functional (11) we obtain a KKT system (12), which can be solved with an inexact Newton method [13, 14]. Let the initial guess $X_0^{(k+1)}$ be the solution of the previous time step, and $X_n^{(k+1)}$ the current approximate solution, we find the next solution $X_{n+1}^{(k+1)}$ as

$$X_{n+1}^{(k+1)} = X_n^{(k+1)} + \alpha_n^{(k+1)} S_n^{(k+1)}, \quad n = 0, 1, ... \tag{13}$$

where $\alpha_n^{(k+1)}$ is the steplength and $S_n^{(k+1)}$ is the search direction obtained by solving the Jacobian system approximately using a Krylov subspace method such that

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

where $(M_n^{(k+1)})^{-1}$ is an overlapping Schwarz preconditioner [35, 37] and the KKT matrix $J_n^{(k+1)} = J_n^{(k+1)}(X_n^{(k+1)})$ is computed by a finite difference approximation. The accuracy of the Jacobian solver is determined by the two linear tolerances $\eta_r, \eta_a \geq 0$. The steplength $\alpha_n^{(k+1)}$ is determined by a backtracking linesearch procedure (see, e.g., section 6.3 in [12]). The stopping condition for the nonlinear iteration (13) is

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

where $\varepsilon_r, \varepsilon_a \geq 0$ are nonlinear tolerances.

In order to define the one-level Schwarz preconditioner, we need to obtain an overlapping partition of $\Omega \in \mathbb{R}^2$. We first divide Ω into non-overlapping subdomains Ω_i, $i = 1, \ldots, N_s$, and then expand each Ω_i to Ω_i^δ, i.e., $\Omega_i \subset \Omega_i^\delta \subset \Omega$. The overlap $\delta > 0$ is defined as the distance between $\partial\Omega_i^\delta$ and $\partial\Omega_i$, in the interior of Ω. For boundary subdomains we simply cut off the part outside Ω. More precisely, we assume $H_x^{'} \times H_y^{'}$ is the size of Ω_i^δ, $H_x \times H_y$ is the size of Ω_i, and h_x (or h_y) is the mesh size in the x-direction (or y-direction). Then the overlap δ is defined as $(H_x^{'} - H_x)/(2h_x)$ (or $(H_y^{'} - H_y)/(2h_y)$).

Recall that N is the total number of unknowns associated with Ω. Let N_i be the number of unknowns associated with Ω_i^δ. Let J be the $N \times N$ sparse matrix of the Jacobian system

$$JS = -G. \tag{14}$$

We define the $N_i \times N$ matrix R_i^δ as follows: its element $(R_i^\delta)_{l_1,l_2}$ is either 1 if the integer indices $1 \leq l_1 \leq N_i$ and $1 \leq l_2 \leq N$ are related to unknowns defined at the same grid point and this grid point belongs to Ω_i^δ or 0 otherwise. The multiplication of R_i^δ with a $N \times 1$ vector generates a shorter $N_i \times 1$ vector by discarding all components corresponding to grid points outside Ω_i^δ. The $N_i \times N$ matrix R_i^0 is similarly defined, with the difference that its application to a $N \times 1$ vector also zeroes all those components corresponding to grid points on $\Omega_i^\delta \setminus \Omega_i$. We denote by J_i the $N_i \times N_i$ subdomain matrix given by

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

We remark that the matrix J_i contains all variables associated with the subdomain Ω_i^δ including all three field variables, the state variables, the control variables and the Lagrange multipliers. There is no further splitting of the fields within the preconditioner. This is quite different from most of the approaches in the literature [16] in which different sub-preconditioners are employed for different field variables.

We assume 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 [10]

$$M_{RAS}^{-1} = \sum_{i=1}^{N_s} (R_i^0)^T B_i^{-1} R_i^\delta. \tag{15}$$

Various inexact additive Schwarz preconditioners can be constructed by replacing the matrices B_i with convenient and inexpensive to compute matrices, such as those obtained with incomplete factorizations. In this paper we employ the LU factorization. Note that in (15) R_i^0 is a restriction to the non-overlapping subdomain. The performance of the entire algorithm depends critically on the condition number $\kappa(JM_{RAS}^{-1})$, which depends on the mesh size, the number of subdomains N_s, the overlap size δ, the time step size Δt, the Reynolds number, amongst others. Some theoretical work for the additive Schwarz method for linear unsteady problems can be found in [8].

It should be noted that the convergence behavior of Newton's method for unsteady problems is quite different than the behavior for steady state problems. For unsteady problems, the initial guesses are usually much closer to the desired solution than the steady case, and the Jacobian solvers require lower accuracy. That is, when compared to one time step of unsteady problems, steady state problems require more linear iterations as well as more Newton iterations to converge.

4. NUMERICAL EXPERIMENTS

We implement the algorithms described in the previous sections using the Portable, Extensible Toolkit for Scientific computing (PETSc) library of Argonne National Laboratory [2]. The numerical tests are carried out on an IBM BlueGene/L using up to 1024 computing nodes. Each node has 512MB of memory. In our numerical experiments we deal with two-dimensional rectangular domains $\Omega = (0, L_1) \times (0, L_2), L_2 \le L_1$. All notations related to the geometry of the computation domain is depicted in Figure 1. We consider two problems: a cavity flow problem and a backward-facing step flow problem.

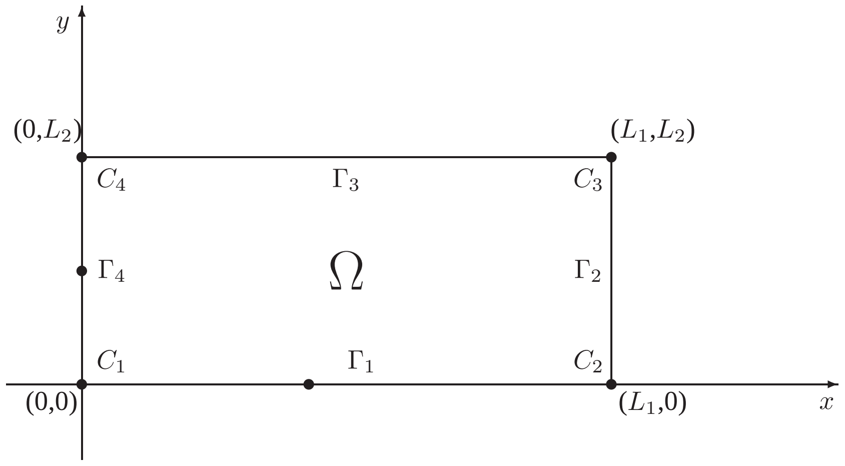

Figure 1. Rectangular domain $\Omega = (0, L_1) \times (0, L_2)$ involved in our numerical experiments.

4.1. *Details of numerical approaches*

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 second order finite difference method on a uniform nonstaggered mesh. Similarly to [30, 31], all derivative terms in the constraints are discretized with a second order scheme, including the ω boundary condition.

In order to form the algebraic system of nonlinear discretized equations, we need to order the unknowns and the corresponding functions. The unknowns are ordered mesh point by mesh point, in contrast to physical variable by physical variable as usually required by other methods. The mesh points are ordered subdomain by subdomain, for the purpose of parallel processing. The ordering of the subdomains is not important since we use additive methods whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting during the sparse LU method (used in our experiments), at each mesh point, the corresponding functions are ordered as [30]

$$(\nabla_{\lambda_1}\mathcal{L}, \nabla_{\lambda_2}\mathcal{L}, \nabla_{\lambda_3}\mathcal{L}, \nabla_{u_1}\mathcal{L}, \nabla_{u_2}\mathcal{L}, \nabla_{v_1}\mathcal{L}, \nabla_{v_2}\mathcal{L}, \nabla_{\omega}\mathcal{L}) \tag{16}$$

while the corresponding unknowns are ordered as

$$(v_1, v_2, \omega, u_1, u_2, \lambda_1, \lambda_2, \lambda_3). \tag{17}$$

Because the orderings for the unknowns and for the function components are different, the Jacobian matrix is nonsymmetric and so we use a nonsymmetric iterative method GMRES [34].

The Jacobian matrix is constructed approximately using a finite difference method [11]. To solve the Jacobian systems we use restarted GMRES with an absolute (relative) tolerance equal to 10^{-10} (10^{-6}), a restart parameter equal to 90 and a maximum number of iterations equal to 5,000. Regarding the one-level additive Schwarz preconditioner, the number of subdomains is equal to the number of processors, and the extended subdomain problems have zero Dirichlet interior boundary conditions and are solved with a sparse direct method. All subdomains Ω_i and Ω_i^δ are rectangular and made up of integral number of mesh cells. Line search is performed with cubic backtracking. For Newton iterations, the maximum allowed number is 300 and the absolute (relative) stopping tolerance is 10^{-10} (10^{-6}). Simulation problems are solved with the one-level Newton-Krylov-Schwarz algorithm (NKS) [9]. We do not use Reynolds continuation in any of the algorithms.

Throughout this section, “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 RAS preconditioned GMRES iterations per Newton iteration, and “Run” is the total computing time in seconds.

4.2. A cavity flow problem

In this subsection we consider a tangential boundary flow control problem whose objective is to make the flow velocity field $\mathbf{v}$ on $\Omega \times [t_0, t_f]$ to get as close as possible to a target velocity profile $\mathbf{v}_{ss}$ given on $\Omega \times [t_0, t_f]$ as well [33]. The tangential control u is applied everywhere on $\Gamma \times [t_0, t_f]$ in order to drive $\mathbf{v}$. More specifically, we want to find $(v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$\min_{(\mathbf{s},\mathbf{u})\in\mathbf{S}\times\mathbf{U}} \mathcal{F}(\mathbf{s},\mathbf{u}) = \frac{\beta}{2}\int_{t_0}^{t_f}\int_{\Omega}\left[\mathbf{v}(t,x,y) - \mathbf{v}_{ss}\right]^2\, d\Omega\, dt + \frac{\gamma}{2}\int_{t_0}^{t_f}\int_{\Gamma}\|\mathbf{u}\|_2^2\, d\Gamma\, dt \tag{18}$$

is achieved subject to the constraints (7) with a square domain $\Omega = (0,1)\times(0,1)$, $t_0 = 0$, $t_f = 0.5$, initial condition $\mathbf{v}_0 = 0$ and target velocity

$$\mathbf{v}_{ss}(t,x,y) = \begin{pmatrix} sin(2\pi y + \pi t)(cos(2\pi x) - 1) \\ 2sin(2\pi x)sin(\pi y + \pi t)sin(\pi y) \end{pmatrix}.$$

As discussed in the Introduction, we numerically solve the corresponding sub-optimal control problems rather than the full optimal one. In [33] the parameters in the cost functional (18) were taken as $\beta = 10^3$ and $\gamma = 1$, and only very coarse meshes were considered. Here we fix $\beta = 10^3$ and test LNKSz with different values of objective function parameter γ, Reynolds number Re, *constant* time step Δt, characteristic mesh size h and RAS overlap size δ. As will be shown by our experiments, the problem indeed becomes much harder to solve as the mesh gets larger, i.e., as h decreases.

First, we investigate the influence of γ on the efficiency of the sub-optimal control. For different values of γ and fixed $Re = 20$, 32×32 grid, $t_f = 0.5$, $\Delta t = 0.1$ (i.e., there are 5 time steps), and $\delta = 6$, Figure 2 shows the error

$$\|\mathbf{v} - \mathbf{v}_{ss}\|_h = \left(\int_{\Omega_h} (\mathbf{v} - \mathbf{v}_{ss})^2 \, d\Omega_h \right)^{1/2},$$

while Figure 3 shows the two-norm of the boundary control velocity. One can see in Figure 2 that, for lower values of γ, the error $\|\mathbf{v} - \mathbf{v}_{ss}\|_h$ is smaller. This happens because smaller values of γ diminishes the contribution of the control norm to the objective function, thus allowing bigger controls to be applied, as shown by Figure 3. In other words, the control is more efficient as γ decreases. Similar results were obtained in references [20, 33].

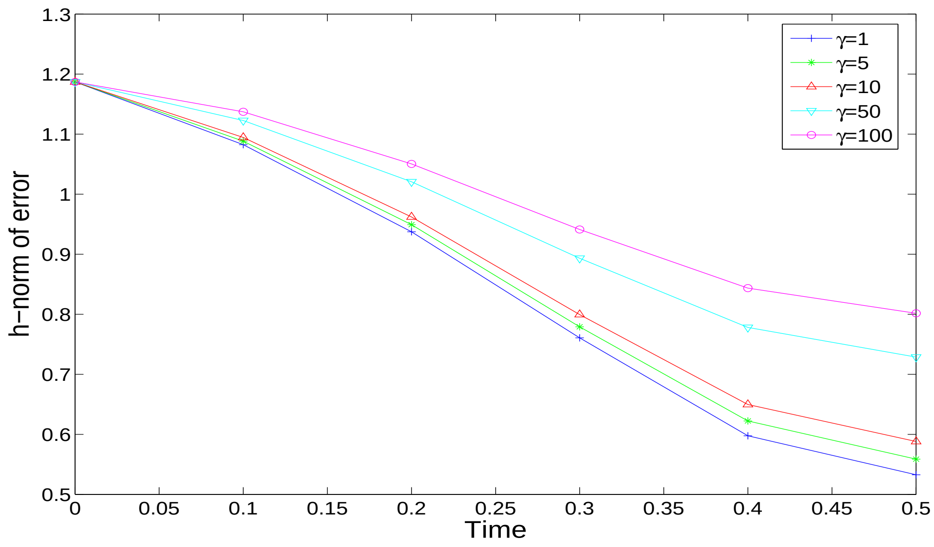

Figure 2. Cavity flow control problem (18)-(7): h-norm of the difference between the controlled flow and the target flow for different values of the parameter γ and fixed $Re = 20$, 32×32 grid, $t_f = 0.5$, $\Delta t = 0.1$ (i.e, there are 5 time steps) and overlap $\delta = 6$. The case $\gamma = 1$ is also reported in Figures 4-5.

Figures 4 and 5 present the velocity field of the controlled and target flows at several different times, for $\gamma = 1$ and all the other parameters having the values reported in the last paragraph. As shown in the figures, the controlled flow is qualitative close (presence of two vortices) to the optimal flow pattern at $t = 0.3$, and such closeness improves at $t = 0.4$ and $t = 0.5$. We remark that the evolution of the controlled and target flow fields in this paper are similar to that of [33].

From now on, in this subsection, we investigate how different parameters impact the performance of LNKSz. Table I presents some results with different stopping conditions for the Jacobian solver. It is clear that the absolute convergence tolerance η_a and the relative convergence tolerance η_r have to be small enough to keep LNKSz converge. In the following tests we choose $\eta_a = 10^{-10}$ and $\eta_r = 10^{-6}$.

Table II shows the effect of γ on the performance of LNKSz, for fixed $Re = 200$, 512×512 grid, $t_f = 0.5$, $\Delta t = 0.05$ (i.e., there are 10 time steps), and $\delta = 8$. The number of Newton iterations in the first time step is 23, 16, 3 and 2 for the respective values of $\gamma = 5, 10, 50, 100$. We can see that, as γ increases, the average numbers of Newton and GMRES iterations become smaller and the total computing time decreases. In other words, the control problem is more difficult to solve for smaller γ values. On the other hand, a larger γ also decreases the accuracy between the control flow $\mathbf{v}$ and the target flow $\mathbf{v}_{ss}$, which is more important for the control problem. So, in many of the following tests we choose a relatively small $\gamma = 5$.

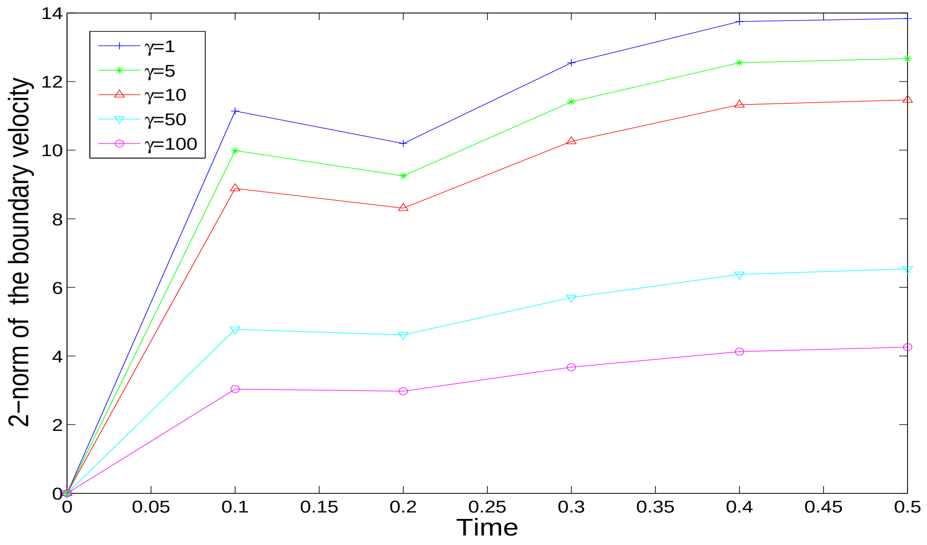

Figure 3. Cavity flow control problem (18)-(7): two-norm of the tangential boundary control velocity for different values of the objective function parameter γ and fixed $Re = 20$, 32×32 grid, $t_f = 0.5$, $\Delta t = 0.1$ (i.e, there are 5 time steps) and overlap $\delta = 6$.

Table I. Cavity flow control problem (18)-(7): effect of different stopping conditions of the linear iteration on the performance of LNKSz, for fixed $Re = 200$, 512×512 grid, $t_f = 0.5$, $\Delta t = 0.05$, and overlap $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. η_a is the absolute convergence tolerance and η_r is the relative convergence tolerance of the linear iteration. " $*$ " means divergence of GMRES.

Grid size	N_p	η_a	η_r	IN	RAS	Run
	256	10^{-10}	10^{-8}	5.4	207	1419
	256	10^{-10}	10^{-6}	5.4	155	1179
512×512	256	10^{-10}	10^{-4}	5.5	119	1049
	256	10^{-6}	10^{-4}		$*$	
	256	10^{-4}	10^{-2}		$*$	

Table II. Cavity flow control problem (18)-(7): effect of the parameter γ on the performance of LNKSz, for fixed $Re = 200$, 512×512 grid, $t_f = 0.5$, $\Delta t = 0.05$, and overlap $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

γ	5			10			50			100		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	5.4	93	5076	4.3	78	3839	2.1	72	1828	2.0	71	1747
128	5.4	118	2749	4.3	108	2108	2.1	87	936	2.0	85	904
256	5.4	133	1353	4.3	123	1035	2.1	106	470	2.0	105	446
512	5.4	201	991	4.3	178	731	2.1	142	313	2.0	138	293
1024	5.4	209	582	4.3	187	432	2.1	149	182	2.0	142	170

Table III shows the effect of δ on the performance of LNKSz, for fixed $\gamma = 5$, $Re = 200$, 512×512 grid, $t_f = 0.5$ and $\Delta t = 0.05$ (i.e., there are 10 time steps). From the numbers of Newton and GMRES iterations, we see that in general the algorithm converges better as the overlap increases. However, a larger overlap also increases the inter-processor communication and the overall computing time. The moderate overlaps $\delta = 4$ and 6 provide a good compromise among computing time, Newton iterations and GMRES iterations. The algorithm performs quite well for up at least 1024 processors. We have chosen $\delta = 6$ for all subsequent tests reported in this subsection.

Table III. Cavity flow control problem (18)-(7): effect of overlap size δ on the performance of LNKSz, for fixed $Re = 200$, 512×512 grid, and $t_f = 0.5$, $\Delta t = 0.05$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

δ	2			4			6			8		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	5.4	206	4641	5.4	135	4427	5.4	110	4505	5.4	93	5076
128	5.4	271	2621	5.4	192	2519	5.4	134	2508	5.4	118	2749
256	5.4	294	1132	5.4	213	1140	5.4	155	1179	5.4	133	1353
512	5.4	443	792	5.4	284	746	5.4	223	829	5.4	201	991
1024	5.4	470	374	5.4	332	423	5.4	242	477	5.4	209	582

Table IV shows the effect of Δt on the performance of fully implicit LNKSz, for fixed $\gamma = 5$, $Re = 200$, 512×512 grid, $t_f = 0.5$, and $\delta = 6$. Every tested time step was kept constant throughout the algorithm run. LNKSz converges for all time steps and is unconditionally stable. We also see that the time step impacts the convergence rate of LNKSz: as Δt decreases, the average number of Newton iterations and GMRES iterations become smaller, while the total computing time increases. This behavior is somehow expected since smaller values of Δt cause the velocity field of the previous time step to become a better initial guess for the Newton method at a current time step. Once the time step becomes too small, however, such initial guess contribution to the convergence of Newton iterates "stalls" and the "intrinsic" nonlinearity of the problem is revealed. The algorithm allows large time steps if short computing time is the target.

Table IV. Cavity flow control problem (18)-(7): effect of the time steps on the performance of LNKSz, for fixed $\gamma = 5$, $Re = 200$, 512×512 grid, $t_f = 0.5$, and $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

Δt	0.1			0.05			0.025			0.0125		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	8.6	138	3909	5.4	110	4505	4.4	102	6995	4.0	106	12970
128	8.6	165	2215	5.4	134	2508	4.4	126	3924	4.0	128	7233
256	8.6	214	1142	5.4	155	1179	4.4	141	1798	4.0	142	3302
512	8.6	266	743	5.4	223	829	4.4	202	1249	4.0	203	2291
1024	8.6	306	451	5.4	242	477	4.4	215	702	4.0	217	1291

Table V summarizes the effect of the mesh size on the performance of LNKSz, for fixed $\gamma = 5$, $Re = 200$, $t_f = 0.5$, $\Delta t = 0.05$ (i.e., there are 10 time steps), and $\delta = 6$. When the mesh is 1024×1024 and $N_p = 64, 128, 256$, the test is not carried out because of the lack of memory on the particular supercomputers used. On the nonlinear level, it is clear that the number of Newton iterations per time step is independent of the number of processors, while it increases as the mesh is refined. For the Jacobian solver, however, the number of GMRES iterations grows as N_p increases. This is expected with any one-level methods. The total computing time decreases at a reasonably good rate when we increase N_p from 64 to 1024, and the rate improves for larger meshes.

In order to study the impact of the Reynolds number on the performance of LNKSz, we increase it to $Re = 400$ and keep all other values as before. Results are summarized in Table VI. Comparing it to Table V, we can see that the average number of Newton iterations became slightly larger, while the average number of GMRES iterations and the total computing time became smaller. On

Table V. Cavity flow control problem (18)-(7): effect of the mesh size on the performance of LNKSz, for fixed $\gamma = 5$, $Re = 200$, $t_f = 0.5$, $\Delta t = 0.05$, and $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1024 \times 1024 \times 8 = 8,388,608$ degrees of freedom.

Mesh	128×128			256×256			512×512			1024×1024		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.6	40	112	4.5	63	641	5.4	110	4505	$*$	$*$	$*$
128	3.6	59	80	4.5	87	395	5.4	134	2508	$*$	$*$	$*$
256	3.6	62	50	4.5	98	212	5.4	155	1179	$*$	$*$	$*$
512	3.6	67	39	4.5	162	185	5.4	223	829	9.5	479	9791
1024	3.6	69	29	4.5	182	127	5.4	242	477	9.5	522	4985

the whole, LNKSz for the cavity flow control problem is not sensitive to Re in the tested range of Reynolds numbers.

Table VI. Cavity flow control problem (18)-(7): effect of the mesh size on the performance of LNKSz, for fixed $\gamma = 5$, $Re = 400$, $t_f = 0.5$, $\Delta t = 0.05$, and $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1024 \times 1024 \times 8 = 8,388,608$ degrees of freedom.

Mesh	128×128			256×256			512×512			1024×1024		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.6	39	111	4.5	60	629	5.6	104	4543	$*$	$*$	$*$
128	3.6	56	79	4.5	72	365	5.6	129	2554	$*$	$*$	$*$
256	3.6	58	48	4.5	90	203	5.6	145	1176	$*$	$*$	$*$
512	3.6	61	37	4.5	134	164	5.6	201	797	9.8	399	8827
1024	3.6	63	28	4.5	144	108	5.6	215	453	9.8	465	4687

Next we change $\gamma = 1$ and investigate again the effect of mesh size and Re on the performance of LNKSz. Tables VII and VIII summarizes results for $Re = 200$ and 400 respectively, for fixed $t_f = 0.5$, $\Delta t = 0.05$ (i.e., there are 10 time steps), and $\delta = 6$. Similarly to Tables V and VI, tests are not carried out with 64, 128 and 256 processors and 1024×1024 mesh because of the lack of memory. By decreasing γ from 5 to 1, the average number of Newton iterations becomes larger, and the increase of the number of GMRES iterations is moderate. LNKSz converges and the computing time scalability is good for large meshes.

Table VII. Cavity flow control problem (18)-(7): effect of the mesh size on the performance of LNKSz, for fixed $\gamma = 1$, $Re = 200$, $t_f = 0.5$, $\Delta t = 0.05$, and $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1024 \times 1024 \times 8 = 8,388,608$ degrees of freedom.

Mesh	128×128			256×256			512×512			1024×1024		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	4.5	45	144	6.5	69	957	21.3	140	19960	$*$	$*$	$*$
128	4.5	66	106	6.5	100	610	21.3	157	10654	$*$	$*$	$*$
256	4.5	69	66	6.5	115	335	21.3	220	5765	$*$	$*$	$*$
512	4.5	75	51	6.5	193	302	21.3	275	3793	64.4	644	83622
1024	4.5	80	39	6.5	233	219	21.3	338	2417	64.4	625	39249

Finally, Figure 6 shows the speedup and the total computing time for 64, 128, 256, 512 and 1024 processors, and fixed $\gamma = 1$, $Re = 200$, 512×512 grid, $t_f = 0.5$, $\Delta t = 0.05$ (i.e., there are 10 time steps), and $\delta = 6$. When N_p increases from 64 to 1024, the total computing time decreases at a

Table VIII. Cavity flow control problem (18)-(7): effect of the mesh size on the performance of LNKSz, for fixed $\gamma = 1$, $Re = 400$, $t_f = 0.5$, $\Delta t = 0.05$, and $\delta = 6$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1024 \times 1024 \times 8 = 8,388,608$ degrees of freedom.

Mesh	128×128			256×256			512×512			1024×1024		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	4.8	45	154	6.9	66	1003	22.4	137	20455	$*$	$*$	$*$
128	4.8	64	111	6.9	93	626	22.4	155	11187	$*$	$*$	$*$
256	4.8	67	70	6.9	113	353	22.4	197	5650	$*$	$*$	$*$
512	4.8	68	52	6.9	160	282	22.4	243	3633	66.8	529	76517
1024	4.8	72	39	6.9	193	202	22.4	302	2322	66.8	548	36509

reasonably good rate, which indicates that LNKSz has a good speedup at least for this range of numbers of processors.

4.3. *A backward-facing step flow problem*

In this subsection, we present computational results for a backward-facing step channel flow defined on a computational domain $\Omega = (0,6) \times (0,1)$; i.e., $L_1 = 6$ and $L_2 = 1$. Let $\Gamma_{1,a} = \{(x,y) \in \Gamma_1 : 0 < x \le L_2\}$, $\Gamma_{1,b} = \Gamma_1 \setminus \Gamma_{1,a}$, $\Gamma_{4,a} = \{(x,y) \in \Gamma_4 : \frac{L_2}{2} \le y < L_2\}$ and $\Gamma_{4,b} = \Gamma_4 \setminus \Gamma_{4,a}$. We define $v_{in} = 8(L_2 - y)(y - \frac{L_2}{2})cos(t)$ and $v_{out} = y(L_2 - y)cos(t)$, then the backward-facing step simulation problem consists of the following equations

$$\left\{\begin{array}{lcll}
-\Delta v_1 - \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\
-\Delta v_2 + \dfrac{\partial \omega}{\partial x} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\
\dfrac{\partial \omega}{\partial t} - \dfrac{1}{Re}\Delta\omega + v_1\dfrac{\partial \omega}{\partial x} + v_2\dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\
v_1 & = & 0 & \text{on } [t_0, t_f] \times \Gamma_s, \\
v_1 & = & v_{in} & \text{on } [t_0, t_f] \times \Gamma_{4,a}, \\
v_1 & = & v_{out} & \text{on } [t_0, t_f] \times \Gamma_2, \\
\dfrac{\partial v_1}{\partial \boldsymbol{\nu}} & = & 0 & \text{on } [t_0, t_f] \times \Gamma_{1,a}, \\
v_2 & = & 0 & \text{on } [t_0, t_f] \times (\Gamma \setminus \Gamma_{4,b}), \\
\dfrac{\partial v_2}{\partial \boldsymbol{\nu}} & = & 0 & \text{on } [t_0, t_f] \times \Gamma_{4,b}, \\
\omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} & = & 0 & \text{on } [t_0, t_f] \times \Gamma, \\
\mathbf{v}(t_0, x, y) - \mathbf{v}_0 & = & \mathbf{0} & \text{in } \overline{\Omega}, \\
\omega(t_0, x, y) + \dfrac{\partial v_1}{\partial y}(t_0, x, y) - \dfrac{\partial v_2}{\partial x}(t_0, x, y) & = & 0 & \text{in } \overline{\Omega},
\end{array}\right. \tag{19}$$

where $\Gamma_s = \{C_1\} \cup \Gamma_{1,b} \cup \overline{\Gamma}_3 \cup \Gamma_{4,b}$, $\boldsymbol{\nu}$ is the unit outward normal vector along Γ. And the backward-facing step control problem consists of finding $(v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$\min_{(\mathbf{s},\mathbf{u}) \in \mathbf{S} \times \mathbf{U}} \mathcal{F}(\mathbf{s}, \mathbf{u}) = \frac{\beta}{2}\int_{t_0}^{t_f}\int_{\Omega} \omega^2 \, d\Omega + \frac{\gamma}{2}\int_{t_0}^{t_f}\int_{\Gamma_u} \|\mathbf{u}\|_2^2 \, d\Gamma \, dt \tag{20}$$

is achieved subject to the constraints

$$\begin{cases} -\Delta v_1 - \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ -\Delta v_2 + \dfrac{\partial \omega}{\partial x} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \dfrac{\partial \omega}{\partial t} - \dfrac{1}{Re}\Delta \omega + v_1 \dfrac{\partial \omega}{\partial x} + v_2 \dfrac{\partial \omega}{\partial y} & = & 0 & \text{in } [t_0, t_f] \times \Omega, \\ \mathbf{v} - \mathbf{u} & = & 0 & \text{on } [t_0, t_f] \times \Gamma_u, \\ \mathbf{u} \cdot \boldsymbol{\nu} & = & 0 & \text{on } [t_0, t_f] \times \Gamma_u, \\ v_1 & = & 0 & \text{on } [t_0, t_f] \times (\Gamma_{1,b} \cup \overline{\Gamma}_3), \\ v_1 & = & v_{in} & \text{on } [t_0, t_f] \times \Gamma_{4,a}, \\ v_1 & = & v_{out} & \text{on } [t_0, t_f] \times \Gamma_2, \\ v_2 & = & 0 & \text{on } [t_0, t_f] \times (\Gamma \setminus \Gamma_u), \\ \omega + \dfrac{\partial v_1}{\partial y} - \dfrac{\partial v_2}{\partial x} & = & 0 & \text{on } [t_0, t_f] \times \Gamma, \\ \mathbf{v}(t_0, x, y) - \mathbf{v}_0 & = & \mathbf{0} & \text{in } \overline{\Omega}, \\ \omega(t_0, x, y) + \dfrac{\partial v_1}{\partial y}(t_0, x, y) - \dfrac{\partial v_2}{\partial x}(t_0, x, y) & = & 0 & \text{in } \overline{\Omega}, \end{cases} \tag{21}$$

where $\Gamma_u = \Gamma_{4,b} \cup \{C_1\} \cup \Gamma_{1,a}$.

In all cases of this subsection, for the backward-facing step simulation problem and the backward-facing step control problem we use the following initial velocity:

$$v_{0,1}(x, y) = y(L_2 - y) + \begin{cases} \dfrac{L_2}{16} y & \text{if } 0 \leqslant y \leqslant \dfrac{L_2}{2}, \\ \dfrac{L_2}{16}(L_2 - y) & \text{if } \dfrac{L_2}{2} \leqslant y \leqslant L_2, \end{cases}$$

and

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

The flow domain is $\Omega = (0, 6) \times (0, 1)$, i.e., $L_1 = 6$ and $L_2 = 1$. The time domain is $[0, 1]$. Similarly to [33], the parameters in the cost functional (20) are $\beta = 10$ and $\gamma = 1$, respectively. NKS is used to solve the corresponding simulation problem and LNKSz is used to solve the corresponding control problem.

First, we present results for the backward-facing step simulation. Table IX shows the effect of Δt on the performance of fully implicit one-level NKS, for fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\delta = 8$. The number of time steps are 5, 10, 20 and 40, respectively. On the nonlinear level, the number of Newton iterations decreases as the time step size is reduced (as expected), and is independent of the number of processors. The total computing time increases as the time step size is reduced. For the linear solver, the number of GMRES iterations increases as the number of processors increases (again, as expected from the convergence theory of one-level domain decomposition methods [35, 37]).

Next, we solve the backward-facing step flow simulation problem on several different meshes and fixed $t_f = 1$, $\Delta t = 0.1$ (i.e., there are 10 time steps) and $\delta = 8$. The results are summarized in Tables X and XI for $Re = 200$ and $Re = 400$, respectively. We see that the number of nonlinear iterations per time step does not change much with respect to the mesh size and is independent of the number of processors. For the linear solver, the number of linear iterations grows with the number of processors, as expected from the convergence theory of one-level domain decomposition methods. As the Reynolds number increases, both the number of Newton iterations and the total computing time increase, although GMRES iterations decrease. In other words, NKS for the backward-facing step flow simulation problem is a little sensitive to the Reynolds number.

In the following, we switch to the control problem. Table XII shows the effect of δ on the performance of LNKSz, for fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\Delta t = 0.1$ (i.e., there are 10 time steps). From the number of Newton and GMRES iterations, we see that in general the algorithm

Table IX. Backward-facing step simulation problem (19): effect of the time steps on the performance of NKS, with fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

Δt	0.2			0.1			0.05			0.025		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.0	47	47	2.3	48	72	2.1	49	132	2.0	45	246
128	3.0	56	28	2.3	59	43	2.1	58	78	2.0	54	146
256	3.0	59	15	2.3	62	24	2.1	63	44	2.0	60	83
512	3.0	86	13	2.3	85	20	2.1	84	36	2.0	83	69
1024	3.0	91	9	2.3	92	13	2.1	90	24	2.0	87	45

Table X. Backward-facing step simulation problem (19): effect of the mesh size on the performance of NKS, for fixed $Re = 200$, $t_f = 1$, $\Delta t = 0.1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1536 \times 256 \times 3 = 1,179,648$ degrees of freedom.

Mesh	384×64			768×128			1536×256		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	2.4	33	18	2.3	48	72	2.3	67	379
128	2.4	42	13	2.3	59	43	2.3	89	207
256	2.4	45	8	2.3	62	24	2.3	97	104
512	2.4	49	7	2.3	85	20	2.3	157	82
1024	2.4	51	6	2.3	92	13	2.3	168	48

Table XI. Backward-facing step simulation problem (19): effect of the mesh size on the performance of NKS, for fixed $Re = 400$, $t_f = 1$, $\Delta t = 0.1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1536 \times 256 \times 3 = 1,179,648$ degrees of freedom.

Mesh	384×64			768×128			1536×256		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	2.7	30	20	2.5	44	76	2.4	65	391
128	2.7	39	14	2.5	57	46	2.4	88	215
256	2.7	42	9	2.5	60	26	2.4	93	106
512	2.7	47	8	2.5	81	21	2.4	146	81
1024	2.7	46	6	2.5	88	14	2.4	161	49

converges better as the overlap increases. On the other hand, a larger overlap also increases the inter-processor communication and the overall computing time. A moderate overlap, $\delta = 8$, provides the best results in terms of combined robustness (converges well for all values of N_p) and run time. When the overlap is too small ($\delta = 4, 6$), the tests with $N_p = 512, 1024$ are not convergent.

Table XIII shows the effect of Δt on the performance of LNKSz, for fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\delta = 8$. On the nonlinear level, the number of Newton iterations decreases as Δt is reduced, and is independent of the number of processors. However, the computing time increases when Δt is reduced. For the linear solver, the number of GMRES iterations increases as the number of processors increases.

Next, we solve the backward-facing step flow control problem on several meshes as well. Table XIV summarizes the effect of the mesh size on the performance of LNKSz, for fixed $Re = 200$, $t_f = 1$, $\Delta t = 0.1$ (i.e., there are 10 time steps), and $\delta = 8$. When the mesh is 1536×256 and $N_p = 64$, the test is not carried out because of the lack of memory. The tests with the mesh

Table XII. Backward-facing step control problem (20)-(21): effect of overlap size δ on the performance of LNKSz, for fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\Delta t = 0.1$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

δ	4			6			8			10		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.1	155	877	3.1	116	879	3.1	80	924	3.1	67	1054
128	3.1	264	576	3.1	181	593	3.1	131	638	3.1	126	874
256	3.1	342	353	3.1	232	362	3.1	147	354	3.1	123	440
512	$*$	$*$	$*$	$*$	$*$	$*$	3.1	269	392	3.1	166	374
1024	$*$	$*$	$*$	$*$	$*$	$*$	3.1	293	244	3.1	262	318

Table XIII. Backward-facing step control problem (20)-(21): effect of the time steps on the performance of LNKSz, for fixed $Re = 200$, 768×128 grid, $t_f = 1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds.

Δt	0.2			0.1			0.05			0.025		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.0	93	470	3.1	80	924	2.2	83	1323	2.1	81	2533
128	3.0	146	327	3.1	131	638	2.2	150	974	2.1	146	1850
256	3.0	157	178	3.1	147	354	2.2	163	536	2.1	156	1007
512	3.0	283	197	3.1	269	392	2.2	314	622	2.1	279	1104
1024	3.0	306	122	3.1	293	244	2.2	344	393	2.1	333	740

1536×256 and $N_p = 512, 1024$ are not convergent for $\delta = 8$. So we choose $\delta = 10$ in these two cases.

Table XIV. Backward-facing step control problem (20)-(21): effect of the mesh size on the performance of LNKSz, for fixed $Re = 200$, $t_f = 1$, $\Delta t = 0.1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1536 \times 256 \times 8 = 3,145,728$ degrees of freedom. Cases "(*)" denote that the tests with the mesh 1536×256 and $N_p = 512, 1024$ are not convergent for $\delta = 8$. So $\delta = 10$ in these two cases.

Mesh	384×64			768×128			1536×256		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.2	53	226	3.1	80	924	$*$	$*$	$*$
128	3.2	65	188	3.1	131	638	3.0	275	3532
256	3.2	70	105	3.1	147	354	3.0	347	1997
512	3.2	84	92	3.1	269	392	$(*)$3.0	809	3035
1024	3.2	95	66	3.1	293	244	$(*)$3.0	819	1664

We then increase the Reynolds number to $Re = 400$ and show the results in Table XV. The time step size is $\Delta t = 0.1$ and the number of time steps is 10. Overlap is $\delta = 8$. When the mesh is 1536×256 and $N_p = 64$, the test is not carried out because the lack of memory. The tests with the mesh 1536×256 and $N_p = 512, 1024$ are not convergent. So we increase the overlap to $\delta = 10$ in the case of $N_p = 512$, and to $\delta = 12$ in the case of $N_p = 1024$. With a larger Reynolds number, the number of Newton iterations and the total computing time are both larger, but the number of GMRES iterations is smaller. Compared with the cavity flow control problem, LNKSz for the backward-facing step flow control problem is more sensitive to the Reynolds number.

Table XV. Backward-facing step control problem (20)-(21): effect of the mesh size on the performance of LNKSz, for fixed $Re = 400$, $t_f = 1$, $\Delta t = 0.1$, and $\delta = 8$. "N_p" is the number of processors, "IN" is the average number of inexact Newton iterations per time step, "RAS" is the (rounded) average number of RAS preconditioned GMRES iterations per Newton iteration, and "Run" is the (rounded) total computing time in seconds. The biggest control problem has $1536 \times 256 \times 8 = 3,145,728$ degrees of freedom. Cases "(*)" denote that the tests with the mesh 1536×256 and $N_p = 512, 1024$ are not convergent. So $\delta = 10$ in the case of $N_p = 512$, and $\delta = 12$ in the case of $N_p = 1024$.

Mesh	384×64			768×128			1536×256		
N_p	IN	RAS	Run	IN	RAS	Run	IN	RAS	Run
64	3.4	56	243	3.3	87	1018	$*$	$*$	$*$
128	3.4	67	201	3.3	126	664	3.1	279	3686
256	3.4	72	113	3.3	155	390	3.1	354	2095
512	3.4	90	101	3.3	280	428	$(*)$3.1	782	3045
1024	3.4	95	70	3.3	300	266	$(*)$3.1	699	1925

In Figure 7 we show the two norm of the vorticity history for the backward-facing step flow control problem and the simulation problem, for fixed $Re = 200$, 64×32 grid, $t_f = 1$, $\Delta t = 0.2$ (i.e., there are 5 time steps), and $\delta = 6$. The norm of the vorticity with the control is always less than that without the control.

The experiments in this subsection show that control problems are computationally more demanding, in terms of the total number of nonlinear iterations, the average number of the linear iterations per Newton iteration and the total computing time, than the corresponding simulations problems. From Tables XII, XIV and XV, we understand that, in order to make LNKSz converge, it is necessary to use relatively large overlaps. Interestingly, we see that the number of Newton iterations decreases as the mesh is refined.

5. CONCLUSIONS

Boundary control of unsteady incompressible flows is a challenging and very expensive computational problem. In this paper, we introduced and numerically studied a family of domain decomposition based, fully parallel, fully implicit, one-shot approach for the sub-optimal control of unsteady Navier-Stokes flows in two-dimensional physical space. The main workhorse of this approach is a Lagrange-Newton-Krylov-Schwarz method which is proven to be quite robust in the sense that it converges rapidly for a wide range of flow and mesh parameters. The fully implicit time discretization is unconditionally stable and allows large time steps. The scalability of the method was demonstrated by running the software successfully on computers with more than 1000 processors and for problems with millions of degrees of freedom. Our future research includes the extension of the methods to two or more levels so that even higher resolution problems can be solved efficiently on computers with a larger number of processors.

ACKNOWLEDGEMENT

This research was supported in part by DOE under DE-FC-02-06ER25784, and in part by NSF under grants CCF-0634894, and CNS-0722023, and DMS-0913089. The author HJY was also supported in part by the Planned Science and Technology Project of Hunan Province under 2011GK3135. The authors would like to thank the anonymous referees for their valuable suggestions to improve the paper.

REFERENCES

1. Aamo OM, Krstić M, Bewley TR. Control of mixing by boundary feedback in 2D channel flow. *Automatica* 2003; **39**:1597–1606.
2. Balay S, Buschelman K, Gropp WD, Kaushik D, Knepley M, McInnes LC, Smith BF, Zhang H. *PETSc Users Manual*. Argonne National Laboratory, 2009.

3. Balogh A, Liu W-J, Krstić M. Stabililty enhancement by boundary control in 2-D channel flow. *IEEE Transactions on Automatic Control* 2001; **46**:1696–1711.
4. Bewley TR. Flow control: new challenges for a new renaissance. *Progress in Aerospace Sciences* 2001; **37**:21–58.
5. Bewley TR, Temam R, Ziane M. A general framework for robust control in fluid mechanics. *Physica D* 2000; **138**:360–392.
6. 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.
7. 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.
8. Cai X-C. Additive Schwarz algorithms for parabolic convection-diffusion equations. *Numerische Mathematik* 1990; **60**:41–62.
9. Cai X-C, Gropp W.D, Keyes D.E, Melvin R.G, Young DP. Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation. *SIAM Journal on Scientific Computing* 1998; **19**:246–265.
10. Cai X-C, Sarkis M. A restricted additive Schwarz preconditioner for general sparse linear systems. *SIAM Journal on Scientific Computing* 1999; **21**:92–797.
11. Coleman TF, Moré JJ. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis* 1983; **20**:187–209.
12. Dennis JE, Schnabel RB. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM. Philadelphia, 1996.
13. Eisenstat SC, Walker HF. Globally convergent inexact Newton method. *SIAM Journal on Optimization* 1994; **4**:393–422.
14. Eisenstat SC, Walker HF. Choosing the forcing terms in an inexact Newton method. *SIAM Journal on Scientific Computing* 1996; **17**:16–32.
15. Fursikov A, Gunzburger MD, Hou LS. Boundary value problems and optimal boundary control for the Navier-Stokes system: the two-dimensional case. *SIAM Journal on Control and Optimization* 1998; **36**:852–894.
16. Ghattas O, Bark J-H. Optimal control of two- and three-dimensional incompressible Navier-Stokes flows. *Journal of Computational Physics* 1997; **136**:231–244.
17. Ghattas O, Orozco CE. *A parallel reduced Hessian SQP method for shape optimization*. in Multidisciplinary Design Optimization: State of the Art, N. M. Alexandrov and M. Y. Hussaini, eds., SIAM, Philadelphia, 1997.
18. Girault V, Raviart P. *The Finite Element Method for Navier-Stokes Equations: Theory and Algorithms*, Springer, New York, 1986.
19. Gunzburger MD, Manservisi S. The velocity tracking problem for Navier-Stokes flow with boundary control. *SIAM Journal on Numerical Analysis* 2000; **39**:594–634.
20. 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.
21. Gunzburger MD. *Perspectives in Flow Control and Optimization*. SIAM, First ed., 2003.
22. Hairer E, Nørsett S, Wanner G. *Solving Ordinary Differential Equations I*. Springer-Verlag, 1993.
23. Hou LS, Ravindran SS. A penalized Neumann control approach for solving an optimal Dirichlet control problem for the Navier-Stokes equations. *SIAM Journal on Control and Optimization* (1998); **36**:1795–1814.
24. Ito K, Ravidran SS. Optimal control of thermally convected fluid flows. *SIAM Journal on Scientific Computing* 1998; **199**:1847–1869.
25. Kunisch K, Sachs EW. Reduced SQP methods for parameter identification problems. *SIAM Journal on Numerical Analysis* 1992; **29**:1793–1820.
26. Kupfer F-S, Sachs EW. Numerical solution of a nonlinear parabolic control problem by a reduced SQP method. *Computational Optimization and Applications* 1992; **1**:113–135.
27. Moin P, Bewley TR. Feedback control of turbulence, *Applied Mechanics Reviews* 1994; **47**:3–13.
28. Nocedal J, Wright SJ. *Numerical Optimization*. Springer-Verlag, New York, First ed., 2000.
29. Prudencio E. *Parallel Fully Coupled Lagrange-Newton-Krylov-Schwarz Algorithms and Software for Optimization Problems Constrained by Partial Differential Equations*. PhD thesis, Department of Computer Science, University of Colorado at Boulder, 2005.
30. 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.
31. 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.
32. Quartapelle L. *Numerical Solution of the Incompressible Navier-Stokes Equations*. International Series of Numerical Mathematics, Vol. 113, Birkhäuser Verlag, 1996.
33. 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.
34. Saad Y. *Iterative Methods for Sparse Linear Systems*. SIAM, Second ed., 2003.
35. Smith B, Bjørstad P, Gropp W. *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge University Press, 1996.
36. Schulz V. Solving discretized optimization problems by partially reduced SQP methods. *Computing and Visualization in Science* 1998; **1**:83–96.
37. Toselli A, Widlund O. *Domain Decomposition Methods-Algorithms and Theory*. Springer, Berlin, 2005.

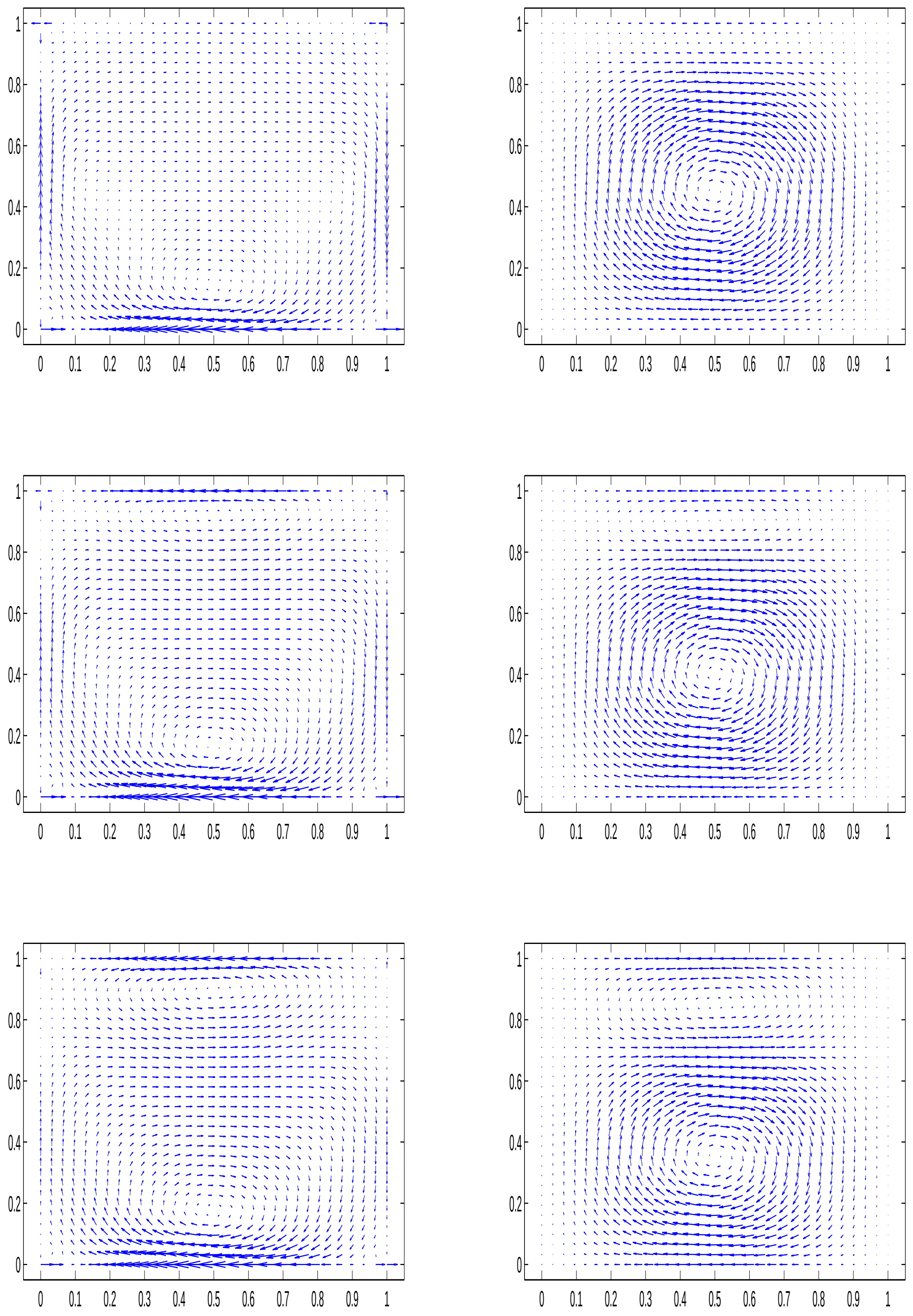

Figure 4. Cavity flow control problem (18)-(7): the velocity field of the controlled (left column) and target (right column) flows at several different times, for fixed $\gamma = 1$, $Re = 20$, 32×32 grid, $t_f = 0.5$, $\Delta t = 0.1$ (i.e, there are 5 time steps) and overlap $\delta = 6$. The first, second and third rows correspond to $t = 0.1$, $t = 0.2$ and $t = 0.3$, respectively. A quantitative analysis of the difference between the two flows is reported in Figure 2.

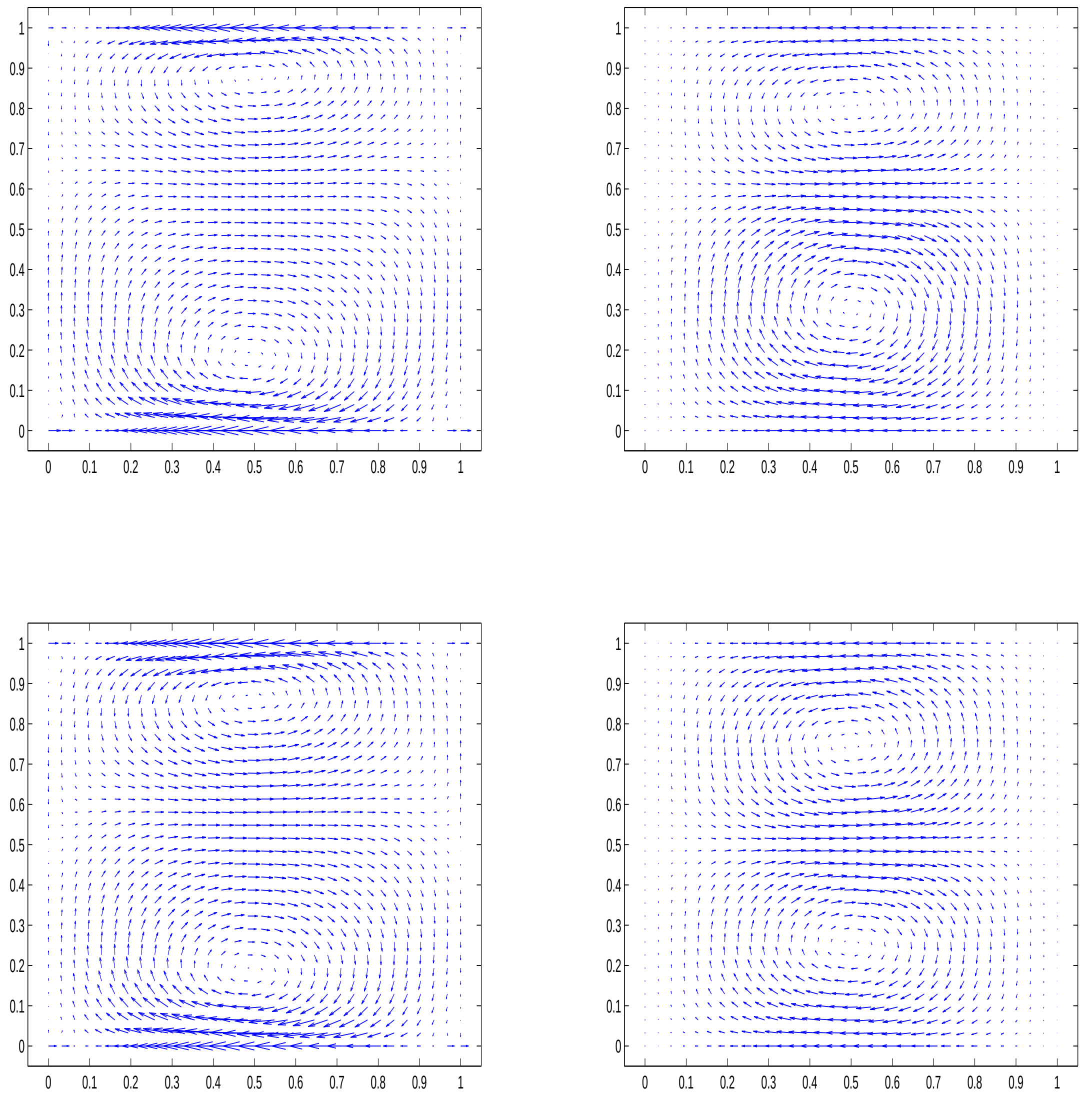

Figure 5. Cavity flow control problem (18)-(7): the velocity field of the controlled (left column) and target (right column) flows at several different times, for fixed $\gamma = 1$, $Re = 20$, 32×32 grid, $t_f = 0.5$, $\Delta t = 0.1$ (i.e, there are 5 time steps) and overlap $\delta = 6$. The first and second rows correspond to $t = 0.4$ and $t = 0.5$, respectively. A quantitative analysis of the difference between the two flows is reported in Figure 2.

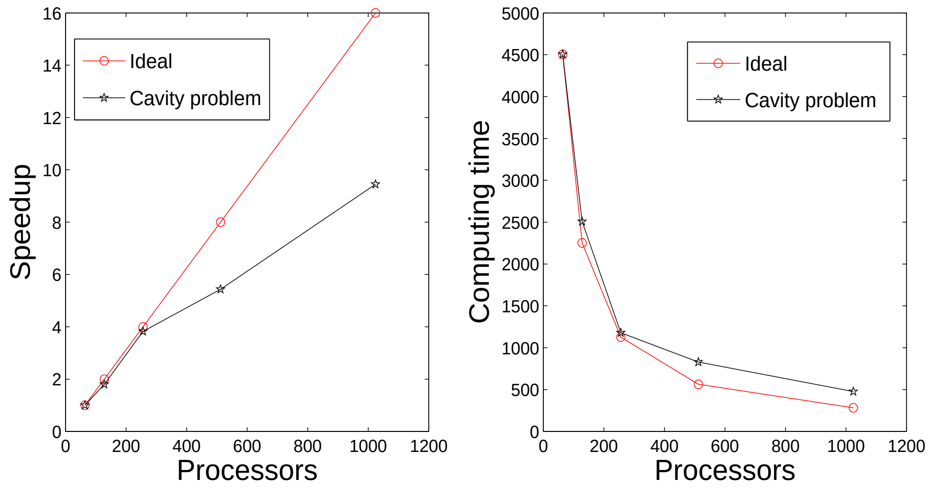

Figure 6. Cavity flow control problem (18)-(7): the speedup and the total computing time for 64, 128, 256, 512 and 1024 processors, and fixed $\gamma = 1$, $Re = 200$, 512×512 grid, $t_f = 0.5$, $\Delta t = 0.05$ (i.e., there are 10 time steps), and $\delta = 6$.

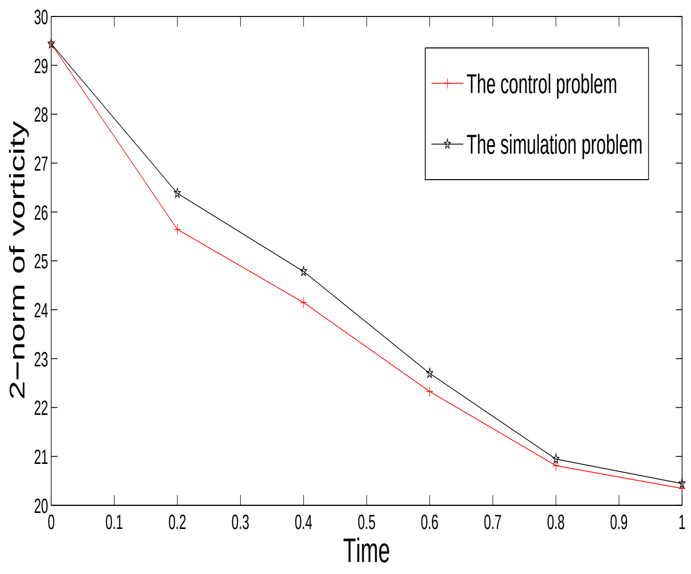

Figure 7. Two-norm of vorticity history for the backward-facing step control problem (20)-(21) and the simulation problem (19), for fixed $Re = 200$, 64×32 grid, $t_f = 1$, $\Delta t = 0.2$ (i.e., there are 5 time steps), and $\delta = 6$.