SIAM J. SCI. COMPUT.
Vol. 27, No. 4, pp. 1305–1328

PARALLEL FULL SPACE SQP LAGRANGE–NEWTON–KRYLOV–SCHWARZ ALGORITHMS FOR PDE-CONSTRAINED OPTIMIZATION PROBLEMS*

ERNESTO E. PRUDENCIO†, RICHARD BYRD‡, AND XIAO-CHUAN CAI‡

Abstract. Optimization problems constrained by nonlinear partial differential equations have been the focus of intense research in scientific computing lately. Current methods for the parallel numerical solution of such problems involve sequential quadratic programming (SQP), with either reduced or full space approaches. In this paper we propose and investigate a class of parallel full space SQP Lagrange–Newton–Krylov–Schwarz (LNKSz) algorithms. In LNKSz, a Lagrangian functional is formed and differentiated to obtain a Karush–Kuhn–Tucker (KKT) system of nonlinear equations. An inexact Newton method with line search is then applied. At each Newton iteration the linearized KKT system is solved with a Schwarz preconditioned Krylov subspace method. We apply LNKSz to the parallel numerical solution of some boundary control problems of two-dimensional incompressible Navier–Stokes equations. Numerical results are reported for different combinations of Reynolds number, mesh size, and number of parallel processors. We also compare the application of the LNKSz method to flow control problems against the application of the Newton–Krylov–Schwarz (NKSz) method to flow simulation problems.

Key words. nonlinear partial differential equations constrained optimization, inexact Newton method, domain decomposition, Schwarz preconditioners, parallel computing, flow control, incompressible Navier–Stokes equations

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65N22, 65N55, 65Y05, 76N05, 76D55, 90C06, 90C90, 93C20

DOI. 10.1137/040602997

1. Introduction.

In this paper we describe a general framework for solving optimization problems in interaction with nonlinear partial differential equations (PDEs). The focus is on how to adapt state-of-the-art PDE solvers to the requirements of optimization methods, while allowing for an efficient parallel implementation. Our method treats the differential equations as equality constraints. In order to demonstrate its effectiveness, the problem of optimizing fluid flows modeled by incompressible Navier–Stokes equations on two-dimensional domains is considered. Among optimization problems with constraints, those constrained by nonlinear PDEs are very challenging, both mathematically and computationally. Examples of such problems are inverse, optimal design, and optimal control problems. In inverse problems some parameters of the governing equations of the system behavior are not known and must be estimated by analysis of experimental system output data [3, 34, 45]. Optimal design problems usually refer to problems where the variable is the shape of a domain and one has to find the best shape that minimizes or maximizes an objective function

*Received by the editors January 8, 2004; accepted for publication (in revised form) June 8, 2005; published electronically January 6, 2006.
http://www.siam.org/journals/sisc/27-4/60299.html

†Advanced Computations Department, Stanford Linear Accelerator Center, Menlo Park, CA 94025 (prudenci@slac.stanford.edu). The research of this author was supported in part by the National Science Foundation, CCR-0219190, ACI-0072089, and ACI-0305666.

‡Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (richard@cs.colorado.edu, cai@cs.colorado.edu). The research of the second author was supported in part by the National Science Foundation, CCR-0219190, and by Army Research Office contract DAAD19-02-1-0407. The research of the third author was supported in part by the Department of Energy, DE-FC02-01ER25479, and in part by the National Science Foundation, CCR-0219190, ACI-0072089, and ACI-0305666.

[35]. In optimal control problems one usually searches for the best feasible control variables, such as boundary values or external forces that minimize or maximize a certain system behavior, such as turbulence [4, 6, 10, 23, 43]. In this paper, we consider only boundary control problems, which refer to the control of the system through boundary conditions. In general, a control problem can be described as

$$\begin{cases} \text{Find controls } \mathbf{u} \in \mathbf{U} \text{ and states } \mathbf{s} \in \mathbf{S} \\ \text{such that } \mathcal{F}(\mathbf{s}, \mathbf{u}) \text{ is minimized,} \\ \text{subject to } \mathbf{C}(\mathbf{s}, \mathbf{u}) \geqslant \mathbf{0}. \end{cases}$$

Here $\mathbf{S}$ and $\mathbf{U}$ are called state and control spaces, respectively; the state variables $\mathbf{s}$ represent the state of the system being controlled; the control variables $\mathbf{u}$ represent the means one has to control the system; the objective or cost functional $\mathcal{F} : \mathbf{S} \times \mathbf{U} \to \mathbb{R}$ to be minimized (or maximized) represents the reason why one wants to control the system; and constraints $\mathbf{C}(\mathbf{s}, \mathbf{u}) \geqslant \mathbf{0}$ represent the system behavior and other constraints imposed on state and control variables.

In flow control problems [16, 26, 28, 29], typical state variables are velocity, pressure, temperature, and internal energy. Control variables might be defined on the boundary or inside the flow domain and can be the temperature, velocity, external force, etc. The objective can be, for instance, minimization of turbulence or maximization of mixing. The constraints are given by the physical equations that must be obeyed by the system being controlled, e.g., Navier–Stokes equations, and by eventual bounds on the state and control variables. The discipline of flow control comprises many other disciplines, e.g., fluid mechanics, nonlinear PDEs, optimization algorithms, discretization techniques, numerical methods, high-performance computing, control engineering, microelectromechanical systems, and nanoelectromechanical systems [7]. In addition, the control of fluid flows has many applications as well, e.g., internal combustion engines (efficiency and emissions), heat transfer, chemical reactors, and aerodynamic surfaces [40].

For finite dimensional equality constrained optimization problems there are two major families of techniques: reduced space sequential quadratic programming (SQP) methods and full space SQP methods. Reduced space methods have been the method of choice until recently since they require much less memory, even though many subiterations are needed to converge the outer iterations and the parallel scalability is less ideal. As more powerful computer systems with lots of memory and many processors become available, full space methods seem to be more appropriate due to their increased scalability. One such method, Lagrange–Newton–Krylov–Schur (LNKSr), was introduced in [8, 9], where four block factorization based preconditioners, as well as globalization techniques and heuristics, are proposed and tested. In this paper we replace the Schur-type preconditioner with an overlapping Schwarz method, which has a better asymptotic convergence rate and is easier to convert to a nonlinear preconditioner [13, 14, 31] for highly nonlinear problems.

The rest of the paper is organized as follows. Section 2 discusses the numerical solution of optimal control problems with equality constraints. Section 3, the core of this paper, presents the full space SQP *Lagrange–Newton–Krylov–Schwarz* (LNKSz) method for the parallel numerical solution of such problems and also briefly describes some current methods that have already appeared in the literature. Section 4 presents some boundary flow control problems, which are then solved in section 5 with LNKSz. Numerical experiments are performed and analyzed for different combinations of Reynolds number, mesh size, and number of parallel processors. Final conclusions are given in section 6.

2. Optimal control with equality constraints. In this paper we focus on optimal control problems with equality constraints:

$$\left\{ \begin{array}{ll} \min\limits_{(\mathbf{s},\mathbf{u})\in\mathbf{S}\times\mathbf{U}} & \mathcal{F}(\mathbf{s},\mathbf{u}) \\ \text{subject to} & \mathbf{C}(\mathbf{s},\mathbf{u}) = \mathbf{0} \in \mathbf{Y}, \end{array} \right. \tag{2.1}$$

where $\mathbf{S}$, $\mathbf{U}$, and $\mathbf{Y}$ are normed spaces, $\mathcal{F} : \mathbf{S} \times \mathbf{U} \to \mathbb{R}$ is the objective function, $\mathbf{C} : \mathbf{S} \times \mathbf{U} \to \mathbf{Y}$ represents the constraints, and $\mathbf{C}(\cdot,\cdot) = \mathbf{0}$ is referred to as state equations or a forward problem. The Lagrangian functional $\mathcal{L} : \mathbf{S} \times \mathbf{U} \times \mathbf{Y}^* \to \mathbb{R}$ associated with (2.1) is defined by

$$\mathcal{L}(\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \equiv \mathcal{F}(\mathbf{s},\mathbf{u}) + \langle \boldsymbol{\lambda}, \mathbf{C}(\mathbf{s},\mathbf{u})\rangle_{\mathbf{Y}} \quad \forall\ (\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \in \mathbf{S} \times \mathbf{U} \times \mathbf{Y}^*, \tag{2.2}$$

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.

When the constraints are PDEs over a domain Ω, the discretization necessary for the solution of (2.1) can occur at two different points in the logical development of an algorithm. In the first case, the *optimize-then-discretize* (OTD) approach, one demonstrates that if $(\hat{\mathbf{s}},\hat{\mathbf{u}})$ is a (local) solution of (2.1), then there exist Lagrange multipliers $\hat{\boldsymbol{\lambda}}$ such that $(\hat{\mathbf{s}},\hat{\mathbf{u}},\hat{\boldsymbol{\lambda}})$ is a critical point of $\mathcal{L}$. So, under sufficient smoothness assumptions, one obtains, as necessary condition for a solution, a system of equations, called the Karush–Kuhn–Tucker (KKT) or optimality system, which is then discretized, generating a finite dimensional system of nonlinear equations [27, 32, 41].

In the second case, the *discretize-then-optimize* (DTO) approach, one begins by creating a mesh Ω_h of characteristic size $h > 0$ and then discretizing problem (2.1), obtaining a finite dimensional equality constrained optimization problem with $\mathbf{S} = \mathbb{R}^{n_{s,h}}$, $\mathbf{U} = \mathbb{R}^{n_{u,h}}$, and $\mathbf{Y} = \mathbb{R}^{m_h} = \mathbf{Y}^*$. Under sufficient smoothness conditions the KKT system becomes $\nabla\mathcal{L}_h(\hat{\mathbf{s}},\hat{\mathbf{u}},\hat{\boldsymbol{\lambda}}) = \mathbf{0} \in \mathbb{R}^{n_{s,h}+n_{u,h}+m_h}$. The theory for finite dimensional constrained optimization problems guarantees, under appropriate assumptions, the existence of such Lagrange multipliers $\hat{\boldsymbol{\lambda}}$. It should be pointed out that the discrete KKT systems from both approaches are *not* necessarily the same.

No approach, however, has a clear advantage over the other [26]. In both cases, the derivation of the Lagrangian w.r.t. state (control, Lagrange multiplier) variables results in the adjoint (control, state, respectively) equations. OTD usually demands the weak formulation of the state PDEs and allows the use of different meshes and techniques (e.g., boundary conditions for infinite domains, or shock-wave capturing schemes for the case of flows) for the state and adjoint equations. DTO, on the other hand, guarantees a consistency between the gradients of the objective function and of the Lagrangian functional and is also suited for the use of automatic differentiation.

From now on, we work only with the DTO approach. For simplicity, let us omit the symbols "h" and "$\hat{\ }$" and use the notation $N \equiv n_s + n_u + m$ and $\mathbf{X} \equiv (\mathbf{x},\boldsymbol{\lambda}) \equiv (\mathbf{s},\mathbf{u},\boldsymbol{\lambda}) \in \mathbb{R}^N$. The KKT system becomes

$$\mathbf{F}(\mathbf{X}) \equiv \left(\begin{array}{c} \nabla_{\mathbf{x}}\mathcal{L} \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L} \end{array} \right) = \left(\begin{array}{c} \nabla\mathcal{F} + \boldsymbol{\nabla}\mathbf{C}^T\boldsymbol{\lambda} \\ \mathbf{C}(\mathbf{s},\mathbf{u}) \end{array} \right) = \left(\begin{array}{c} \nabla_{\mathbf{s}}\mathcal{F} + \boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^T\boldsymbol{\lambda} \\ \nabla_{\mathbf{u}}\mathcal{F} + \boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C}^T\boldsymbol{\lambda} \\ \mathbf{C}(\mathbf{s},\mathbf{u}) \end{array} \right) = \mathbf{0}, \tag{2.3}$$

where $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$, $\nabla_{\mathbf{x}}\mathcal{L}$ denotes the gradient of $\mathcal{L}$ w.r.t. state and control variables, with similar meaning holding for $\nabla_{\boldsymbol{\lambda}}\mathcal{L}$, $\nabla_{\mathbf{s}}\mathcal{F}$, and $\nabla_{\mathbf{u}}\mathcal{F}$, while $\boldsymbol{\nabla}\mathbf{C}$ denotes the Jacobian of $\mathbf{C}$ and $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ and $\boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C}$ denote the Jacobian of $\mathbf{C}$ w.r.t. state and control variables, respectively. In (2.3), we refer to $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ as the linearized forward operator.

System (2.3) can be solved with an inexact Newton method [21, 22]. Given an initial guess $\mathbf{X}^{(0)}$, at each iteration $k = 0, 1, 2, \ldots$, an approximate solution

$$\mathbf{p}^{(k)} \equiv \left(\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)}\right) \equiv \left(\mathbf{p}_{\mathbf{s}}^{(k)}, \mathbf{p}_{\mathbf{u}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)}\right)$$

of the linearized KKT system

$$\mathbf{K}^{(k)}\ \mathbf{p}^{(k)}\ =\ -\,\mathbf{F}^{(k)} \tag{2.4}$$

is computed, where $\mathbf{K}^{(k)} = \boldsymbol{\nabla}\mathbf{F}(\mathbf{X}^{(k)})$ and $\mathbf{F}^{(k)} = \mathbf{F}(\mathbf{X}^{(k)})$. The KKT matrix $\mathbf{K}^{(k)}$ is the transpose of the Hessian of the Lagrangian $\mathcal{L}$, is symmetric indefinite under sufficient smoothness assumptions, and can be computed by a finite difference approximation. System (2.4) can also be written as (all terms should be understood as evaluated at $\mathbf{X}^{(k)}$)

$$\begin{bmatrix} \boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)} & \boldsymbol{\nabla}\mathbf{C}^{(k)^T} \\ \boldsymbol{\nabla}\mathbf{C}^{(k)} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{p}_{\mathbf{x}}^{(k)} \\ \mathbf{p}_{\boldsymbol{\lambda}}^{(k)} \end{pmatrix} = - \begin{pmatrix} \nabla_{\mathbf{x}}\mathcal{L}^{(k)} \\ \mathbf{C}^{(k)} \end{pmatrix}.$$

Thus, if $\boldsymbol{\nabla}\mathbf{C}^{(k)}$ has full rank and $\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}$ is positive definite in the tangent space of the constraints (i.e., $\mathbf{d}^T[\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}]\mathbf{d} > 0$ for all $\mathbf{d} \neq \mathbf{0}$ such that $[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{d} = \mathbf{0}$), then we can interpret $(\mathbf{p}_{\mathbf{x}}^{(k)}, \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ as being the *unique* solution and Lagrange multipliers of

$$\begin{cases} \min\limits_{\mathbf{p_x} \in \mathbf{S}\times\mathbf{U}} & \frac{1}{2}\mathbf{p}_{\mathbf{x}}^T[\boldsymbol{\nabla}^2_{\mathbf{xx}}\mathcal{L}^{(k)}]\mathbf{p}_{\mathbf{x}} + \nabla_{\mathbf{x}}\mathcal{L}^{(k)^T}\mathbf{p}_{\mathbf{x}} \\ \text{subject to} & \boldsymbol{\nabla}\mathbf{C}^{(k)}\mathbf{p}_{\mathbf{x}} + \mathbf{C}^{(k)} = \mathbf{0} \in \mathbf{Y}. \end{cases}$$

This interpretation justifies the use of the terminology *sequential quadratic programming* (SQP) for methods involving (2.4); see [36]. Alternatively, one can interpret $(\mathbf{p}_{\mathbf{x}}^{(k)}, \boldsymbol{\lambda}^{(k)} + \mathbf{p}_{\boldsymbol{\lambda}}^{(k)})$ as the solution and Lagrange multipliers for the quadratic programming (QP) using, in its objective function, $\nabla\mathcal{F}^{(k)}$ instead of $\nabla_{\mathbf{x}}\mathcal{L}^{(k)}$.

After approximately solving (2.4), one may use a globalization method such as *line search* or *trust region* [44]. In this study we focus on a line search approach, where the next iterate is $\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}$ and the step length $\alpha^{(k)}$ is selected by backtracking until the sufficient decrease condition

$$\phi(\mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}) \leqslant \phi(\mathbf{X}^{(k)}) + \alpha^{(k)}c_1\nabla\phi(\mathbf{X}^{(k)})^T\mathbf{p}^{(k)} \tag{2.5}$$

is satisfied. Here ϕ is a merit function, and c_1 is a constant satisfying $0 < c_1 < 1/2$. For constrained optimization, merit functions such as l_1 or augmented Lagrangian [5, 33] are most commonly used. In contrast to the standard merit function $\|\mathbf{F}(\mathbf{X})\|_2^2/2$, which is commonly used for systems of nonlinear equations, these merit functions try to balance the sometimes conflicting goals of reducing the objective function and satisfying the constraints [36]. We use the augmented Lagrangian $\phi_{AL} : \mathbb{R}^N \to \mathbb{R}$ in our experiments in this paper. Given a penalty parameter $\rho > 0$, it is defined by

$$\phi_{AL}(\mathbf{X}; \rho) = \mathcal{L}(\mathbf{s}, \mathbf{u}, \boldsymbol{\lambda}) + \frac{\rho}{2}\ \|\mathbf{C}(\mathbf{s}, \mathbf{u})\|_2^2 \quad \forall\ \mathbf{X} = (\mathbf{s}, \mathbf{u}, \boldsymbol{\lambda}) \in \mathbf{S} \times \mathbf{U} \times \mathbf{Y}.$$

At iteration k, $\rho = \rho^{(k)}$ must be such that we obtain descent directions $\mathbf{p}^{(k)}$, i.e.,

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)}; \rho^{(k)}))^T\mathbf{p}^{(k)} = (\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)} + \rho^{(k)}\ \mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0. \tag{2.6}$$

Since $\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} = -\|\mathbf{C}^{(k)}\|_2^2$ for an exact step, it is reasonable to expect

$$\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)} < 0 \tag{2.7}$$

for approximate steps if $\mathbf{C}^{(k)} \neq \mathbf{0}$ and the tolerances for the Krylov subspace method are small enough and the preconditioner is good enough to guarantee that the Krylov subspace method does not stop by achieving the maximum allowed number of iterations. If (2.7) does not hold, we can continue the Krylov iterations, with eventual smaller tolerances, until it does. Once (2.7) holds, we then use a fairly common strategy where we demand that $\rho^{(k)}$ satisfy

$$(\nabla\phi_{AL}(\mathbf{X}^{(k)};\rho^{(k)}))^T\mathbf{p}^{(k)} \leqslant \frac{\rho^{(k)}}{2}\ \mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)},$$

that is,

$$\rho^{(k)} \geqslant \overline{\rho}^{(k)} = -2\frac{(\nabla\mathcal{L}^{(k)})^T\mathbf{p}^{(k)}}{\mathbf{C}^{(k)^{\mathbf{T}}}[\boldsymbol{\nabla}\mathbf{C}^{(k)}]\mathbf{p}_{\mathbf{x}}^{(k)}}.$$

We then choose

$$\rho^{(k)} = \max\{\overline{\rho}^{(k)}, \rho^{(k-1)}\},$$

where $\rho^{(-1)} > 0$ is a given positive value.

However, if $\mathbf{C}^{(k)} = \mathbf{0}$, then there is no way to guarantee a descent direction. This is a fundamental issue with line search methods. Some algorithms handle this by modifying the Hessian to make it positive definite on the null space of $\boldsymbol{\nabla}\mathbf{C}^{(k)}$, but for problems of the size we are considering there is no efficient way to check positive definiteness.

In all tests described in this paper, (2.7) held for every step generated, as long as we made the absolute Krylov tolerance small enough that at least one Krylov iteration was performed. In addition, it is worth noting that, in each run, the value of $\rho^{(k)}$ became fixed before 60% of the iterations had been made. Thus the heuristic merit parameter updating strategy described above appeared to work well for the examples of this paper.

3. Parallel full space SQP Lagrange–Newton–Krylov–Schwarz. In this section we briefly review some existing techniques and then present a new parallel algorithm based on a Newton method with line search and an overlapping Schwarz preconditioner for solving (2.1). The new algorithm will be referred to as the Lagrange–Newton–Krylov–Schwarz (LNKSz) algorithm. All these methods differentiate themselves by the way (2.4) is solved.

3.1. Brief review of existing methods. In this subsection we assume that the number of constraints is equal to the number of state variables, i.e.,

$$m = n_s, \tag{3.1}$$

in (2.1); that is, $\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}$ is a square matrix of dimension m. Situation (3.1) usually happens after the discretization of PDE-constrained optimization problems that involve well-posed boundary value problems as constraints. To explain the existing techniques

[8, 9] we need the following block-LU factorization:

$$\mathbf{K}^{(k)} = \begin{bmatrix} \boldsymbol{\nabla}^2_{\mathbf{ss}}\mathcal{L}\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^{-1} & \mathbf{0} & \mathbf{I} \\ \boldsymbol{\nabla}^2_{\mathbf{su}}\mathcal{L}\boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^{-1} & \mathbf{I} & \mathbf{L}^T \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C} & \boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_2 & \boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^T \end{bmatrix}, \tag{3.2}$$

where $\mathbf{H}_1 = \boldsymbol{\nabla}^2_{\mathbf{uu}}\mathcal{L} - \mathbf{L}^T\mathbf{H}_2 - \boldsymbol{\nabla}^2_{\mathbf{su}}\mathcal{L}\mathbf{L}$, $\mathbf{H}_2 = \boldsymbol{\nabla}^2_{\mathbf{us}}\mathcal{L} - \boldsymbol{\nabla}^2_{\mathbf{ss}}\mathcal{L}\mathbf{L}$, and $\mathbf{L} = \boldsymbol{\nabla}_{\mathbf{s}}\mathbf{C}^{-1}\boldsymbol{\nabla}_{\mathbf{u}}\mathbf{C}$. The matrix $\mathbf{H}_1$ is the Schur complement for $\mathbf{p}^{(k)}_{\mathbf{u}}$ and is called the reduced Hessian of the Lagrangian.

Full space SQP methods solve the entire system of linear equations (2.4) simultaneously, while reduced space SQP methods divide the problem into smaller ones, first solving for the n_u control variables only and then solving for state variables and Lagrange multipliers. Newton reduced space SQP (N-RSQP) methods basically correspond to the block-LU factorization (3.2). Quasi-Newton reduced space SQP (QN-RSQP) methods drop second order terms from the right-hand side of the equations in N-RSQP and also substitute $\mathbf{H}_1$ by quasi-Newton approximations, usually via BFGS [36].

Reduced methods have the advantage of requiring less memory. However, N-RSQP demands too many solutions of systems of m linear equations per outer iteration, while in QN-RSQP the number of outer iterations tends to increase too much as n_u grows. Full space methods try to overcome the lack of parallel scalability of reduced approaches, but they also present difficulties, not only because a descent direction is not guaranteed but also due to the linearized KKT system: it is more than twice the size of the forward problem, usually indefinite (a property known to slow down Krylov solvers), and very ill conditioned. One such method, Lagrange–Newton–Krylov–Schur (LNKSr), was presented in [8, 9], where four preconditioners based on the block factorization (3.2), as well as globalization techniques and heuristics (such as maintaining a BFGS approximation of $\mathbf{H}_1$ and using a QN-RSQP step in case the original LNKSr computed step fails), are proposed and tested. LNKSr attempts to transform the problem of finding a good preconditioner for the KKT matrix to the problem of finding a good preconditioner for the linearized forward operator.

3.2. Lagrange–Newton–Krylov–Schwarz. As mentioned earlier, the key element of a successful full space approach is the preconditioning of the Jacobian of the KKT system, which is indefinite and extremely ill conditioned. A good preconditioner has to be able to substantially reduce the condition number and, at the same time, provide good scalability so that the potential of massively parallel computers can be realized. The Schur complement preconditioner used in LNKSr is an *operator-splitting* type of technique, in which a sequential block elimination step is needed to form the Schur complement w.r.t. the control variable. In contrast to operator-splitting, Schwarz-type preconditioners are *fully coupled* in the sense that all variables are treated equally and the partition is based completely on the physical domain Ω. Because there is no need to eliminate any variables from the system, there is one fewer sequential step in the preconditioning process. Another advantage of LNKSz method is that it does not demand assumption (3.1). With LNKSz we can, for instance, deal directly with full [30] boundary control problems, where an equation like (4.6) is explicitly added to the constraints.

Schwarz preconditioners can be used in one-level or multilevel approaches and, in each case, with a combination of additive and/or multiplicative algorithms [39, 42]. In this paper we deal only with one-level additive algorithms [20]. Let $\Omega \subset \mathbb{R}^2$ be a bounded open domain on which the control problem is defined. We consider only

simple box domains with uniform mesh of characteristic size h here. To obtain the overlapping partition, we first partition the domain into nonoverlapping subdomains Ω_j^0, $j = 1, \ldots, N_s$. Then we extend each subdomain Ω_j^0 to a larger subdomain Ω_j^δ, i.e., $\Omega_j^0 \subset \Omega_j^\delta$. Here $\delta > 0$ indicates the size of the overlap. Only simple box decomposition is considered in this paper; i.e., all the subdomains Ω_j^0 and Ω_j^δ are rectangular and made up of an integral number of fine mesh cells. For boundary subdomains, we simply cut off the part that is outside Ω. Let $H > 0$ denote the characteristic diameter of subdomains Ω_j. Let N, N_j^0, and N_j denote the number of degrees of freedom associated with Ω, Ω_j^0, and Ω_j^δ, respectively. Let $\mathbf{K}$ be an $N \times N$ matrix of a linear system that needs to be solved during the numerical solution process of the differential problem. Let d indicate the degree of freedom per mesh point. For simplicity let us assume that d is the same throughout the entire mesh. We define the $N_j \times N$ matrix $\mathbf{R}_j^\delta$ as follows: Its $d \times d$ block element $(\mathbf{R}_j^\delta)_{\ell_1,\ell_2}$ is either (a) an identity block if the integer indices $1 \leqslant \ell_1 \leqslant N_j/d$ and $1 \leqslant \ell_2 \leqslant N/d$ are related to the same mesh point and this mesh point belongs to $\Omega_j^{'}$ or (b) a zero block otherwise. The multiplication of $\mathbf{R}_j^\delta$ with an $N \times 1$ vector generates a smaller $N_j \times 1$ vector by discarding all components corresponding to mesh points outside $\Omega_j^{'}$. The $N_j \times N$ matrix $\mathbf{R}_j^0$ is similarly defined, with the difference that its application to an $N \times 1$ vector also zeros all those components corresponding to mesh points on $\Omega_j^{'} \setminus \Omega_j$. We denote by $\mathbf{K}_j$ the $N_j \times N_j$ subdomain matrix given by

$$\mathbf{K}_j = \mathbf{R}_j^\delta \, \mathbf{K} \, \left(\mathbf{R}_j^\delta\right)^T.$$

Let $\mathbf{B}_j^{-1}$ be either an inverse of $\mathbf{K}_j$ or a preconditioner for $\mathbf{K}_j$. The classical one-level additive Schwarz preconditioner $\mathbf{B}_{asm}^{-1}$ for $\mathbf{K}$ is defined as

$$\mathbf{B}_{asm}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^\delta\right)^T \mathbf{B}_j{}^{-1} \mathbf{R}_j^\delta.$$

In addition to this standard additive Schwarz method (ASM) described above, we also consider the newly introduced restricted version (RAS) of the method [11, 15]. For some applications, the restricted version requires less communication time since one of the restriction or extension operations does not involve any overlap. The RAS preconditioner is defined as

$$\mathbf{B}_{ras}^{-1} = \sum_{j=1}^{N_s} \left(\mathbf{R}_j^\delta\right)^T \mathbf{B}_j{}^{-1} \mathbf{R}_j^0.$$

Some numerical comparisons of the ASM and RAS are presented later in the paper.

When the Schwarz preconditioner is applied to symmetric positive definite systems arising from the discretization of elliptical problems defined in $H_0^1(\Omega)$, the condition number κ of the preconditioned system satisfies $\kappa \leqslant C\,(1 + H/\delta)\,/H^2$, where C is independent of h, H, δ, and the shapes of Ω and Ω_j^δ [39]; that is, a Schwarz preconditioned Krylov subspace method is expected to have the following properties:

(3.3) The number of iterations grows approximately proportional to $1/H$;

(3.4) If δ is maintained proportional to H, the number of iterations is bounded independently of h and H/h (a parameter related to the number of degrees of freedom of each subproblem);

(3.5) The convergence gets better as δ is increased.

Theoretically, results (3.3)–(3.5) may not be applied immediately to Krylov subspace methods, e.g., GMRES [24, 38], for the solution of indefinite linearized KKT systems. Nonetheless, we carefully examine all the properties in our numerical experiments. In particular, let $\bar{l}$ be the average number of Schwarz preconditioned GMRES iterations per linearized KKT system. We look for the following scalability properties:

(3.6) For fixed h and δ, $\bar{l}$ increases as H decreases;

(3.7) For fixed H and δ, $\bar{l}$ is not very sensitive to the mesh refinement;

(3.8) For fixed h and H, $\bar{l}$ decreases as δ increases.

4. Boundary control of incompressible Navier–Stokes flows. In this section we discuss the boundary control of the two-dimensional steady-state incompressible Navier–Stokes equations in the velocity-vorticity formulation. The velocity is denoted by $\mathbf{v} = (v_1, v_2)$ and the vorticity by ω. Let Ω be an open and bounded polygonal domain in the plane, with $\Gamma = \partial\Omega$ its boundary and $\boldsymbol{\nu}$ the unit outward normal vector along Γ. Let $\mathbf{f}$ be a given external force defined in Ω. An example of a flow simulation problem consists of finding (v_1, v_2, ω) such that

$$
\left\{
\begin{array}{lcll}
-\Delta v_1 - \dfrac{\partial \omega}{\partial x_2} & = & 0 & \text{in } \Omega, \\
-\Delta v_2 + \dfrac{\partial \omega}{\partial x_1} & = & 0 & \text{in } \Omega, \\
-\Delta \omega + Re\ v_1 \dfrac{\partial \omega}{\partial x_1} + Re\ v_2 \dfrac{\partial \omega}{\partial x_2} - Re\ \text{curl}\ \mathbf{f} & = & 0 & \text{in } \Omega, \\
\mathbf{v} - \mathbf{v}_D & = & \mathbf{0} & \text{on } \Gamma, \\
\omega + \dfrac{\partial v_1}{\partial x_2} - \dfrac{\partial v_2}{\partial x_1} & = & 0 & \text{on } \Gamma,
\end{array}
\right. \tag{4.1}
$$

where Re is the Reynolds number, $\mathbf{v}_D$ is a prescribed boundary velocity satisfying $\int_\Gamma \mathbf{v}_D \cdot \boldsymbol{\nu}\, d\Gamma = 0$, and $\text{curl}\ \mathbf{f} = -\frac{\partial f_1}{\partial x_2} + \frac{\partial f_2}{\partial x_1}$. If $\mathbf{v} \in C^3(\Omega) \times C^3(\Omega)$, then [18]

$$
\omega + \frac{\partial v_1}{\partial x_2} - \frac{\partial v_2}{\partial x_1} = 0 \quad \text{in } \Omega \tag{4.2}
$$

and

$$
\nabla \cdot \mathbf{v} = 0 \quad \text{in } \Omega. \tag{4.3}
$$

These two equations show the consistency of the two-dimensional velocity-vorticity formulation of incompressible flows, since (4.2) is the formal definition of the vorticity and (4.3) is related to the physical law of mass conservation.

An example of a boundary control problem consists of finding $(v_1, v_2, \omega, u_1, u_2)$ such that the minimization

$$
\min_{(\mathbf{s},\mathbf{u}) \in \mathbf{S} \times \mathbf{U}} \mathcal{F}(\mathbf{s}, \mathbf{u}) = \frac{1}{2} \int_\Omega \omega^2\, d\Omega + \frac{c}{2} \int_{\Gamma_u} \|\mathbf{u}\|_2^2\, d\Gamma \tag{4.4}
$$

is achieved subject to the constraints

$$
\left\{
\begin{array}{lcll}
-\Delta v_1 - \dfrac{\partial \omega}{\partial x_2} & = & 0 & \text{in } \Omega, \\
-\Delta v_2 + \dfrac{\partial \omega}{\partial x_1} & = & 0 & \text{in } \Omega, \\
-\Delta \omega + Re\ v_1 \dfrac{\partial \omega}{\partial x_1} + Re\ v_2 \dfrac{\partial \omega}{\partial x_2} - Re\ \text{curl}\ \mathbf{f} & = & 0 & \text{in } \Omega, \\
v_i - v_{D,i} & = & 0 & \text{on } \Gamma_{D,i}, \quad i = 1,2, \\
\dfrac{\partial v_i}{\partial \boldsymbol{\nu}} - v_{N,i} & = & 0 & \text{on } \Gamma_{N,i}, \quad i = 1,2, \\
\mathbf{v} - \mathbf{u} & = & \mathbf{0} & \text{on } \Gamma_u, \\
\omega + \dfrac{\partial v_1}{\partial x_2} - \dfrac{\partial v_2}{\partial x_1} & = & 0 & \text{on } \Gamma, \\
\displaystyle\int_\Gamma \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma & = & 0, &
\end{array}
\right. \tag{4.5}
$$

where, for $i = 1, 2$, $\Gamma = \Gamma_{D,i} \cup \Gamma_{N,i} \cup \Gamma_u$, $\Gamma_{D,i}$ ($\Gamma_{N,i}$) is the part of the boundary where the v_i velocity component is specified through a Dirichlet (Neumann) condition with a prescribed velocity $v_{D,i}$, and Γ_u is the part of the boundary where the velocity is specified through a Dirichlet condition with a control velocity $\mathbf{u}$. The positive constant parameter c is used to adjust the relative importance of the control norms in achieving the minimization, thus indirectly constraining the size of those norms. The physical objective behind problem (4.4)–(4.5) is the minimization of turbulence [1, 7, 30]. The last constraint, given by

$$
\int_\Gamma \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma = 0, \tag{4.6}
$$

is necessary for the consistency with the physical law of mass conservation. Therefore, the control $\mathbf{u} = (u_1, u_2)$ cannot be any control: it must belong to the space of functions satisfying (4.6). We denote problems like (4.4)–(4.5), where controls are allowed to assume nonzero normal values at the boundary, as *full* boundary control problems (BCPs) [30]. In these kinds of problems assumption (3.1) is not valid, due to the extra constraint (4.6). This fact also complicates the parallel finite differences approximation of Jacobian matrices, since one does not have only PDEs (i.e., equations with local behavior) anymore. The integral condition (3.1) couples nonneighboring mesh points. An alternative formulation, which compromises between the physical law of mass conservation and the complexity of parallel Jacobian approximations, eliminates the explicit constraint (4.6), thus making (3.1) valid, but adds to the objective function the term $\tilde{c} \left[\int_\Gamma \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma\right]^2 / 2$, with $\tilde{c} \gg 1$ [9].

One can also deal with *tangential* BCPs, where the control is allowed to be just tangential to the boundary (i.e., $\mathbf{u} \cdot \boldsymbol{\nu} = 0$ on Γ_u) and the velocity $\mathbf{v}$ is assumed to satisfy $\int_{\Gamma \setminus \Gamma_u} \mathbf{v} \cdot \boldsymbol{\nu}\ d\Gamma = 0$. Thus, (3.1) is valid. Since tangential BCPs restrict even more the space where the control $\mathbf{u} = (u_1, u_2)$ can exist, one naturally expects better objective function values with full BCPs. In this paper we study only tangential boundary control problems.

5. Numerical experiments. In our numerical experiments we deal with two-dimensional rectangular domains $\Omega = (0, L_1) \times (0, L_2)$, $L_2 \leqslant L_1$. All notation related to the geometry of the computational domain is depicted in Figure 5.1. In particular,

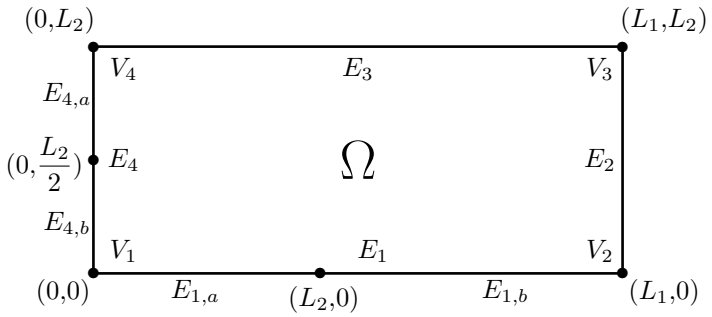

FIG. 5.1. *General rectangular domain* $\Omega = (0, L_1) \times (0, L_2)$, *with* $L_2 \leqslant L_1$, *involved in our numerical experiments.*

we define $E_{1,a} = \{(x_1, x_2) \in E_1 : 0 < x_1 \leqslant L_2\}$, $E_{1,b} = E_1 \setminus E_{1,a}$, $E_{4,a} = \{(x_1, x_2) \in E_4 : \frac{L_2}{2} \leqslant x_2 < L_2\}$, and $E_{4,b} = E_4 \setminus E_{4,a}$.

5.1. Flow problems. In order to simplify the implementation, we rewrite (4.4)–(4.5), defining the control $\mathbf{u}$ everywhere on Γ, but not necessarily allowing it to vary everywhere. We consider two flow problems: cavity flow and backward-facing step flow problems. In each case we describe a simulation problem and a tangential BCP. Also, we describe only the boundary conditions on the velocity and the constraints on the boundary controls, since all problems have in common the three differential equations in Ω and the boundary condition on ω that appear in (4.1). All control problems seek the minimization of the objective function (4.4) with $c = 10^{-2}$.

Cavity flow problems. In this case, $L_1 = L_2 = 1$ and (see Figure 5.2(a))

$$\mathbf{f} = \left(-\sin^2(\pi x_1)\cos(\pi x_2)\sin^2(\pi x_2),\ \sin^2(\pi x_2)\cos(\pi x_1)\sin^2(\pi x_1)\right).$$

The simulation problem assumes slip rigid walls on Γ:

$$\left\{ \begin{array}{rcll} \mathbf{v} \cdot \boldsymbol{\nu} & = & 0 & \text{on } \Gamma, \\ \dfrac{\partial \mathbf{v}}{\partial \boldsymbol{\nu}} & = & \mathbf{0} & \text{on } \Gamma. \end{array} \right. \tag{5.1}$$

The tangential BCP has $\Gamma_u = \Gamma$ and velocity and control constraints given by

$$\left\{ \begin{array}{rcll} \mathbf{v} - \mathbf{u} & = & \mathbf{0} & \text{on } \Gamma_u, \\ \mathbf{u} \cdot \boldsymbol{\nu} & = & 0 & \text{on } \Gamma_u. \end{array} \right. \tag{5.2}$$

Backward-facing step problems. In this case, $L_1 = 6$, $L_2 = 1$, and $\mathbf{f} = \mathbf{0}$. We define $v_{\text{in}}(x_1, x_2) = 8(1 - x_2)(x_2 - \frac{1}{2})$ and $v_{\text{out}}(x_1, x_2) = x_2(1 - x_2)$. See Figures 5.3 and 5.6. The simulation problem has velocity boundary conditions given by

$$\left\{ \begin{array}{rcll} v_1 & = & 0 & \text{on } \{V_1\} \cup E_{1,b} \cup \overline{E}_3 \cup E_{4,b}, \\ v_1 & = & v_{\text{in}} & \text{on } E_{4,a}, \\ v_1 & = & v_{\text{out}} & \text{on } E_2, \\ \dfrac{\partial v_1}{\partial \boldsymbol{\nu}} & = & 0 & \text{on } E_{1,a}, \\ v_2 & = & 0 & \text{on } \Gamma \setminus E_{4,b}, \\ \dfrac{\partial v_2}{\partial \boldsymbol{\nu}} & = & 0 & \text{on } E_{4,b}. \end{array} \right. \tag{5.3}$$

The tangential BCP has $\Gamma_u = E_{4,b} \cup \{V_1\} \cup E_{1,a}$ and velocity and control constraints given by

$$\left\{ \begin{array}{lcll} \mathbf{v} - \mathbf{u} & = & \mathbf{0} & \text{on } \Gamma_u, \\ \mathbf{u} \cdot \boldsymbol{\nu} & = & 0 & \text{on } \Gamma_u, \\ v_1 & = & 0 & \text{on } E_{1,b} \cup \overline{E}_3, \\ v_1 & = & v_{\text{in}} & \text{on } E_{4,a}, \\ v_1 & = & v_{\text{out}} & \text{on } E_2, \\ v_2 & = & 0 & \text{on } \Gamma \setminus \Gamma_u. \end{array} \right. \tag{5.4}$$

We run flow problems for $Re = 200$ and $Re = 300$. In this range of Reynolds numbers the flow is known to be steady and two-dimensional [30].

5.2. Details of numerical approaches. To discretize the flow problems, we use a five-point second order finite difference method on a uniform mesh.[1] All derivative terms of interior PDEs are discretized with a second order central difference scheme. The boundary condition $\omega + \partial v_1/\partial x_2 - \partial v_2/\partial x_1 = 0$ on Γ is also discretized with a second order approximation, using, however, only mesh points adjacent to the boundary, by applying (4.2) at these points also. This combination of second order approximation with only mesh points adjacent to the boundary is good for parallel performance reasons. We show how to achieve this combination for the boundary mesh points on edge E_1 and at corner V_1. Equations for the other boundary regions are similarly deduced. Let h denote the uniform mesh spacing. Given a mesh point, let subindices r, l, a, and b denote mesh points located, respectively, to the right of, to the left of, above, and below this given mesh point. The meanings for subindices ar, br, bl, al, rr, ll, aa, bb, etc. are straightforward. For mesh points located on edge E_1 we know that

$$w = -\frac{-3v_1 + 4v_{1,a} - v_{1,aa}}{2h} + \frac{v_{2,r} - v_{2,l}}{2h} + O(h^2),$$

$$w_a = -\frac{v_{1,aa} - v_1}{2h} + \frac{v_{2,ar} - v_{2,al}}{2h} + O(h^2).$$

Thus, taking the sum of the expressions, $v_{1,aa}$ is cancelled:

$$w + w_a + 2\frac{v_{1,a} - v_1}{h} - \frac{v_{2,ar} - v_{2,al}}{2h} - \frac{v_{2,r} - v_{2,l}}{2h} = O(h^2).$$

Using a similar technique, we obtain, for corner V_1,

$$w + w_a + w_r + w_{ar} + 2\frac{v_{1,a} - v_1}{h} + 2\frac{v_{1,ar} - v_{1,r}}{h} - 2\frac{v_{2,r} - v_2}{h} - 2\frac{v_{2,ar} - v_{2,a}}{h} = O(h^2).$$

To form an algebraic system of nonlinear equations from the finite difference 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 required by all SQP methods related to the matrix structure (3.2). At each mesh point the unknowns are ordered in the order of $v_1, v_2, \omega, u_1, u_2, \lambda_1, \lambda_2$, and λ_3. The mesh points are ordered subdomain by subdomain, for the purpose of

[1] Since our research focus in this paper is the iterative method LNKSz, we use a simple nonstaggered mesh. Another approach is to use staggered meshes in order to numerically better preserve the physical conditions (4.2) and (4.3) [18, 25, 37].

parallel processing. The ordering of the subdomains is not important since we use an additive method whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting during the sparse LU method (used in our experiments), the corresponding functions are ordered as $\nabla_{\lambda_1}\mathcal{L}$,$\nabla_{\lambda_2}\mathcal{L}$,$\nabla_{\lambda_3}\mathcal{L}$, $\nabla_{u_1}\mathcal{L}$,$\nabla_{u_2}\mathcal{L}$, $\nabla_{v_1}\mathcal{L}$,$\nabla_{v_2}\mathcal{L}$, and $\nabla_{\omega}\mathcal{L}$. Because the orderings for the unknowns and for the function components are different, the Jacobian matrix becomes nonsymmetric and so we use GMRES.

Some of the numerical results obtained using LNKSz for tangential boundary flow control problems are compared with the corresponding simulation results obtained using NKSz [12]. Although NKSz has been well studied for flow simulation problems, little is known about its applicability and performance for solving KKT systems arising from flow control problems. The NKSz algorithm can be briefly described as follows:

$$
\left\{
\begin{array}{ll}
1. & \text{Initialize the iteration } k = 0 \text{ and } \mathbf{X}^{(0)} = \mathbf{0};\\
2. & \text{Decide whether to stop based on } \|\mathbf{F}(\mathbf{X}^{(k)})\|_2;\\
3. & \text{Approximately solve } \left[\boldsymbol{\nabla}\mathbf{F}(\mathbf{X}^{(k)})\right]\mathbf{p}^{(k)} = -\mathbf{F}(\mathbf{X}^{(k)}),\\
 & \text{using GMRES with a left Schwarz preconditioner;}\\
4. & \text{Perform line search } \mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \alpha^{(k)}\mathbf{p}^{(k)}\\
 & \text{using the standard merit function } \|\mathbf{F}\|_2^2/2;\\
5. & \text{Set } k \leftarrow k+1; \text{ go to step 2.}
\end{array}
\right. \tag{5.5}
$$

LNKSz is very similar to NKSz, except that it uses an augmented Lagrangian merit function in step 4 of (5.5). Note that Reynolds continuation is not used in any of the algorithms.

In all experiments, the Jacobian matrix is constructed approximately using a multicolored central finite difference method with step size 10^{-5} [17]. In problem (4.4)–(4.5), since there is no variable with power greater than two, central finite difference approximations of the Jacobian are exact up to roundoff errors. To solve the Jacobian systems we use restarted GMRES with an absolute tolerance equal to 10^{-6}, a relative tolerance equal to 10^{-4}, a restart parameter equal to 90, and a maximum number of iterations equal to 5,000. The GMRES tolerances are checked over preconditioned residuals. 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 sparse LU. The line search with the merit function defined in section 2 is performed with cubic backtracking, with $c_1 = 10^{-4}$ in (2.5) and a minimum allowed step length $\alpha^{(k)}$ equal to 10^{-6}. For augmented Lagrangian merit functions we follow the strategy explained in section 2 with $\rho^{(-1)}$=10. For Newton iterations we use an absolute stopping tolerance equal to 10^{-6} and a relative tolerance equal to 10^{-10} times the initial residual. The maximum allowed number of Newton iterations is 100.

5.3. Results of numerical experiments. All tests were performed on a cluster of Linux PCs, and our parallel object-oriented software was developed using the C++ programming language and the Portable, Extensible Toolkit for Scientific Computing (PETSc) library [2] from Argonne National Laboratory. Our main concern is the scalability of the algorithms in terms of the linear and nonlinear iteration numbers. Computing times are also reported, but they should not be taken as a reliable measure of the scalability of the algorithms because our network is relatively slow and is shared by other processes.

Results are grouped into tables according to a unique combination of problem type (simulation or control), Reynolds number *Re*, ASM type (standard or restricted), overlap δ, and merit function (standard or augmented Lagrangian). Each table presents

results for nine situations, related to three different sizes of meshes (parameter h) and three different numbers of processors (parameter H). For each case we report the following:

- the total number of Newton iterations: n;
- the average number of GMRES iterations per Newton iteration: $\bar{l}$;
- the total computing time in seconds spent on all Newton iterations: t_n;
- the average computing time, in seconds, per Newton iteration, spent on solving for the Newton steps: $\bar{t}_l$.

For each table we compare the behavior of $\bar{l}$ against (3.6)–(3.8). Predictions (3.6) and (3.7) can be checked by observing the values of $\bar{l}$ in a column (fixed h and δ) and in a row (fixed H and δ), respectively. Prediction (3.8) can be checked by observing the values of $\bar{l}$ at the same situation (fixed problem type, Re, ASM type, merit function, H, and h) in different tables (different δ). We also compare approximate values of

$$\|\omega\|_h^2 = \int_{\Omega_h} \omega^2 \, d\Omega_h. \tag{5.6}$$

We show some *parallel efficiency* results, defined as follows: the base computing time is obtained on 16 processors, and the parallel efficiencies E_n (related to t_n) and $\overline{E}_l$ (related to $\bar{t}_l$) are given by the expression

$$\frac{(\text{computing time } (t_n \text{ or } \bar{t}_l) \text{ using 16 processors}) \times 16}{(\text{computing time } (t_n \text{ or } \bar{t}_l) \text{ using } N_p \text{ processors}) \times N_p}. \tag{5.7}$$

5.3.1. Cavity flow problems. Table 5.1 presents results for the simulation problem (5.1) with $Re = 200$. The preconditioner is ASM with $\delta = 1/64$ and the standard merit function is used in the line search. The total number of Newton iterations does not change with the mesh size or the number of processors. The average number of Krylov iterations per Newton iteration changes as expected in predictions (3.6) and (3.7).

TABLE 5.1

Results for the cavity flow simulation problem with $Re = 200$, standard ASM with overlap $\delta = 1/64$ and standard merit function $\|\mathbf{F}\|_2^2/2$. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 198,147 and $\|\omega\|_h^2 \approx 55.4$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 5$	$t_n \approx 0.83$	$n = 5$	$t_n \approx 3.6$	$n = 5$	$t_n \approx 18.3$
	$\bar{l} \approx 46$	$\bar{t}_l \approx 0.13$	$\bar{l} \approx 47$	$\bar{t}_l \approx 0.62$	$\bar{l} \approx 47$	$\bar{t}_l \approx 3.29$
32	$n = 5$	$t_n \approx 0.71$	$n = 5$	$t_n \approx 2.7$	$n = 5$	$t_n \approx 12.1$
	$\bar{l} \approx 60$	$\bar{t}_l \approx 0.091$	$\bar{l} \approx 63$	$\bar{t}_l \approx 0.46$	$\bar{l} \approx 63$	$\bar{t}_l \approx 2.20$
64	$n = 5$	$t_n \approx 0.70$	$n = 5$	$t_n \approx 1.99$	$n = 5$	$t_n \approx 7.50$
	$\bar{l} \approx 69$	$\bar{t}_l \approx 0.077$	$\bar{l} \approx 80$	$\bar{t}_l \approx 0.32$	$\bar{l} \approx 79$	$\bar{t}_l \approx 1.35$

The next three tables present results for the tangential BCP (5.2) with $Re = 200$. An augmented Lagrangian merit function is used in the line search. Several different overlap values are used in the ASM preconditioner, and the results are summarized as follows: Table 5.2 for $\delta = 1/64$, Table 5.3 for $\delta = 1/32$, and Table 5.4 for $\delta = 1/16$. Changes in the total number of Newton iterations w.r.t. the mesh size and the number

TABLE 5.2

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/64$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.92$	$n = 8$	$t_n \approx 52.1$	$n = 6$	$t_n \approx 238$
	$\bar{l} \approx 92$	$\bar{t}_l \approx 1.21$	$\bar{l} \approx 85$	$\bar{t}_l \approx 5.78$	$\bar{l} \approx 100$	$\bar{t}_l \approx 36.4$
32	$n = 7$	$t_n \approx 10.3$	$n = 8$	$t_n \approx 53.3$	$n = 6$	$t_n \approx 264$
	$\bar{l} \approx 208$	$\bar{t}_l \approx 1.34$	$\bar{l} \approx 204$	$\bar{t}_l \approx 6.26$	$\bar{l} \approx 272$	$\bar{t}_l \approx 42.5$
64	$n = 7$	$t_n \approx 5.39$	$n = 8$	$t_n \approx 25.6$	$n = 6$	$t_n \approx 108$
	$\bar{l} \approx 187$	$\bar{t}_l \approx 0.67$	$\bar{l} \approx 182$	$\bar{t}_l \approx 2.96$	$\bar{l} \approx 216$	$\bar{t}_l \approx 17.1$

TABLE 5.3

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/32$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6). Case "(∗)" is discussed in subsection 5.3.1.

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 8.22$	$n = 8$	$t_n \approx 49.1$	$n = 7$	$t_n \approx 254$
	$\bar{l} \approx 58$	$\bar{t}_l \approx 0.96$	$\bar{l} \approx 59$	$\bar{t}_l \approx 5.38$	$\bar{l} \approx 62$	$\bar{t}_l \approx 33.1$
32	$n = 7$	$t_n \approx 8.32$	$n = 8$	$t_n \approx 46.6$	$n = 6$	$t_n \approx 199$
	$\bar{l} \approx 119$	$\bar{t}_l \approx 1.05$	$\bar{l} \approx 130$	$\bar{t}_l \approx 5.41$	$\bar{l} \approx 140$	$\bar{t}_l \approx 31.6$
64	$n = 7$ (*)	$t_n \approx 6.21$	$n = 8$	$t_n \approx 27.6$	$n = 6$	$t_n \approx 110$
	$\bar{l} \approx 155$	$\bar{t}_l \approx 0.78$	$\bar{l} \approx 132$	$\bar{t}_l \approx 3.18$	$\bar{l} \approx 143$	$\bar{t}_l \approx 17.4$

TABLE 5.4

Results for the cavity tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.65$	$n = 8$	$t_n \approx 59.2$	$n = 7$	$t_n \approx 744$
	$\bar{l} \approx 47$	$\bar{t}_l \approx 1.16$	$\bar{l} \approx 44$	$\bar{t}_l \approx 6.62$	$\bar{l} \approx 50$	$\bar{t}_l \approx 103$
32	$n = 7$	$t_n \approx 10.8$	$n = 8$	$t_n \approx 75.1$	$n = 7$	$t_n \approx 616$
	$\bar{l} \approx 100$	$\bar{t}_l \approx 1.39$	$\bar{l} \approx 108$	$\bar{t}_l \approx 8.96$	$\bar{l} \approx 149$	$\bar{t}_l \approx 86.3$
64	$n = 6$	$t_n \approx 6.36$	$n = 8$	$t_n \approx 56.5$	$n = 7$	$t_n \approx 331$
	$\bar{l} \approx 104$	$\bar{t}_l \approx 0.94$	$\bar{l} \approx 115$	$\bar{t}_l \approx 6.79$	$\bar{l} \approx 128$	$\bar{t}_l \approx 46.4$

of processors are not pronounced. We observe that $\bar{l}$ follows (3.8). With the same δ used on the simulation problem, we can see in Table 5.2 that the average number of GMRES iterations is now more sensitive to both h and, especially, H. Table 5.3 is

the one where $\bar{l}$ best follows both (3.6) and (3.7) and the computing times t_n and $\bar{t}_l$ for the finest mesh decrease with the increase in the number of processors. Table 5.4 shows that if δ gets too big, then the consequent decrease on $\bar{l}$ might not compensate for the increased time taken by sparse LU on the larger extended subdomains; that is, $\bar{t}_l$ increases. Comparing values of $\bar{t}_l$ in Tables 5.2 and 5.3 with the values in Table 5.1, we see that the average time spent on computing $\mathbf{p}^{(k)}$ can be more than 10 times longer in control problems than in simulation problems on the same mesh, instead of being around $8/3 \approx 3$ times longer, in accordance to the ratio between the number of variables per mesh point on control problems and on simulation problems.

As reported in subsection 5.2, we use a GMRES relative tolerance of 1.0×10^{-4}, a GMRES absolute tolerance of 1.0×10^{-6}, and a Newton absolute tolerance of 1.0×10^{-6}. Case "(*)" in Table 5.3, however, gives results for a Newton absolute tolerance of 1.2×10^{-6}. When a Newton absolute tolerance of 1.0×10^{-6} is used, although full steps are accepted in some iterations, the line search stalls once $||F||_2 \approx 1.14 \times 10^{-6}$. If we change the GMRES relative tolerance to 1.0×10^{-6} and the GMRES absolute tolerance to 1.0×10^{-13} (in order to obtain a more accurate Newton step), then we achieve $||F||_2 < 1.00 \times 10^{-6}$ with $n = 6$, $\bar{l} \approx 272$, $t_n \approx 9.38$, and $\bar{t}_l \approx 1.42$. Although we performed our tests with fixed GMRES tolerances, this experiment suggests that for more demanding Newton tolerances one might need to use decreasing GMRES tolerances as the outer loop proceeds, as expected by the theory for superlinear convergence of the inexact Newton method [19, 36].

In the next two tables, we change the preconditioner to RAS and everything else stays the same; i.e., these results are for the tangential BCP (5.2) with $Re = 200$ and we use an augmented Lagrangian merit function in the line search. We increase the overlap size in the RAS preconditioner as follows: Table 5.5 for $\delta = 1/32$ and Table 5.6 for $\delta = 1/16$. The average number of GMRES iterations continues to follow (3.8) but now it better follows (3.6) and (3.7). The computing times t_n and $\bar{t}_l$ for the finest mesh in Table 5.6 decrease with the increase in the number of processors. The average number of GMRES iterations is larger in Table 5.5 than in Table 5.3, but the saving in communications of RAS compensates for this increase so that $\bar{t}_l$ does not increase proportionally. By comparing finest mesh results on Tables 5.6 and 5.4 we see that RAS performs better than the standard ASM in terms of both $\bar{l}$ and $\bar{t}_l$, resulting in a smaller t_n.

Table 5.7 presents results for the tangential BCP (5.2) with $Re = 250$, restricted ASM with $\delta = 1/16$, and augmented Lagrangian merit function. We can see that, with a Reynolds number greater than that in the previous table, both nonlinear (n) and average linear ($\bar{l}$) complexities increase.

In order to determine if tighter fixed GMRES tolerances would improve the number of Newton iterations and thus, eventually, improve the total computing time, we rerun the tests of Table 5.7 with the same parameters, except for a GMRES relative tolerance of 1.0×10^{-6}, instead of 1.0×10^{-4}, and a GMRES absolute tolerance of 1.0×10^{-13}, instead of 1.0×10^{-6}. When compared to the results in Table 5.7, at least one Newton step is saved in all nine tests, but the increase in $\bar{l}$ (and consequently $\bar{t}_l$) makes t_n increase in all cases with 32 and 64 processors. In fact, t_n decreases just for the case of 256×256 mesh and 16 processors, from 1,090 seconds to 1,040 seconds. So, although there was some improvement in the number of Newton steps, it was usually not enough to compensate for the expected increase in linear complexity, resulting in a longer overall computing time t_n.

Next, in Table 5.8 we present some parallel efficiency results derived from data presented in previous tables.

TABLE 5.5

Results for the cavity tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/32$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 8.33$	$n = 9$	$t_n \approx 53.1$	$n = 7$	$t_n \approx 253$
	$\bar{l} \approx 59$	$\bar{t}_l \approx 0.98$	$\bar{l} \approx 57$	$\bar{t}_l \approx 5.15$	$\bar{l} \approx 62$	$\bar{t}_l \approx 32.9$
32	$n = 7$	$t_n \approx 8.47$	$n = 8$	$t_n \approx 46.9$	$n = 6$	$t_n \approx 211$
	$\bar{l} \approx 131$	$\bar{t}_l \approx 1.07$	$\bar{l} \approx 134$	$\bar{t}_l \approx 5.45$	$\bar{l} \approx 154$	$\bar{t}_l \approx 33.6$
64	$n = 7$	$t_n \approx 6.38$	$n = 8$	$t_n \approx 30.1$	$n = 6$	$t_n \approx 132$
	$\bar{l} \approx 175$	$\bar{t}_l \approx 0.81$	$\bar{l} \approx 162$	$\bar{t}_l \approx 3.50$	$\bar{l} \approx 184$	$\bar{t}_l \approx 21.1$

TABLE 5.6

Results for the cavity tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 32.5$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 7$	$t_n \approx 9.66$	$n = 8$	$t_n \approx 60.7$	$n = 7$	$t_n \approx 743$
	$\bar{l} \approx 47$	$\bar{t}_l \approx 1.15$	$\bar{l} \approx 46$	$\bar{t}_l \approx 6.80$	$\bar{l} \approx 51$	$\bar{t}_l \approx 103$
32	$n = 7$	$t_n \approx 9.49$	$n = 8$	$t_n \approx 65.5$	$n = 6$	$t_n \approx 436$
	$\bar{l} \approx 90$	$\bar{t}_l \approx 1.21$	$\bar{l} \approx 86$	$\bar{t}_l \approx 7.7$	$\bar{l} \approx 98$	$\bar{t}_l \approx 71.0$
64	$n = 7$	$t_n \approx 7.18$	$n = 9$	$t_n \approx 56.9$	$n = 7$	$t_n \approx 309$
	$\bar{l} \approx 105$	$\bar{t}_l \approx 0.91$	$\bar{l} \approx 97$	$\bar{t}_l \approx 6.1$	$\bar{l} \approx 114$	$\bar{t}_l \approx 43.3$

TABLE 5.7

Results for the cavity tangential boundary flow control problem with $Re = 250$, restricted ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 256×256, the number of variables is 528,392 and $\|\omega\|_h^2 \approx 50.2$. See (5.6).

# Procs.	Mesh					
	64×64		128×128		256×256	
16	$n = 11$	$t_n \approx 15.8$	$n = 13$	$t_n \approx 110$	$n = 10$	$t_n \approx 1090$
	$\bar{l} \approx 51$	$\bar{t}_l \approx 1.22$	$\bar{l} \approx 55$	$\bar{t}_l \approx 7.67$	$\bar{l} \approx 55$	$\bar{t}_l \approx 106$
32	$n = 11$	$t_n \approx 17.2$	$n = 13$	$t_n \approx 126$	$n = 9$	$t_n \approx 726$
	$\bar{l} \approx 107$	$\bar{t}_l \approx 1.42$	$\bar{l} \approx 112$	$\bar{t}_l \approx 9.23$	$\bar{l} \approx 123$	$\bar{t}_l \approx 79.0$
64	$n = 10$	$t_n \approx 12.6$	$n = 13$	$t_n \approx 102$	$n = 9$	$t_n \approx 504$
	$\bar{l} \approx 135$	$\bar{t}_l \approx 1.16$	$\bar{l} \approx 139$	$\bar{t}_l \approx 7.62$	$\bar{l} \approx 180$	$\bar{t}_l \approx 55.1$

Finally, we present a comparison of computed velocity fields for $Re = 200$. Figure 5.2(b) refers to the simulation problem (5.1), and Figures 5.2(c) and 5.2(d) refer to the tangential BCP (5.2). Recall that the objective of the tangential boundary control is to minimize the intensity of the velocity field in the vortex shown in Figure 5.2(b). Figure 5.2(d) shows that the tangential boundary control acts in the opposite direction

TABLE 5.8

Efficiency analysis of NKSz and LNKSz, w.r.t. the number of processors for cavity flow problems with a 256×256 *mesh and different values of Reynolds number Re and overlap* δ. E_n *is the efficiency related to* t_n, *the total time in seconds spent on all Newton iterations.* $\overline{E}_l$ *is the efficiency related to* $\bar{t}_l$, *the average time in seconds, per Newton iteration, spent on solving for Newton steps. See* (5.7). *Results are based on values of* t_n *and* $\bar{t}_l$ *from Tables* 5.1, 5.6, *and* 5.7.

# Procs.	Simulation		Boundary control			
	$Re = 200,\ \delta = 1/64$ Standard ASM		$Re = 200,\ \delta = 1/16$ Restricted ASM		$Re = 250,\ \delta = 1/16$ Restricted ASM	
	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$
16	1.00	1.00	1.00	1.00	1.00	1.00
32	0.76	0.75	0.85	0.73	0.75	0.67
64	0.61	0.61	0.60	0.59	0.54	0.48

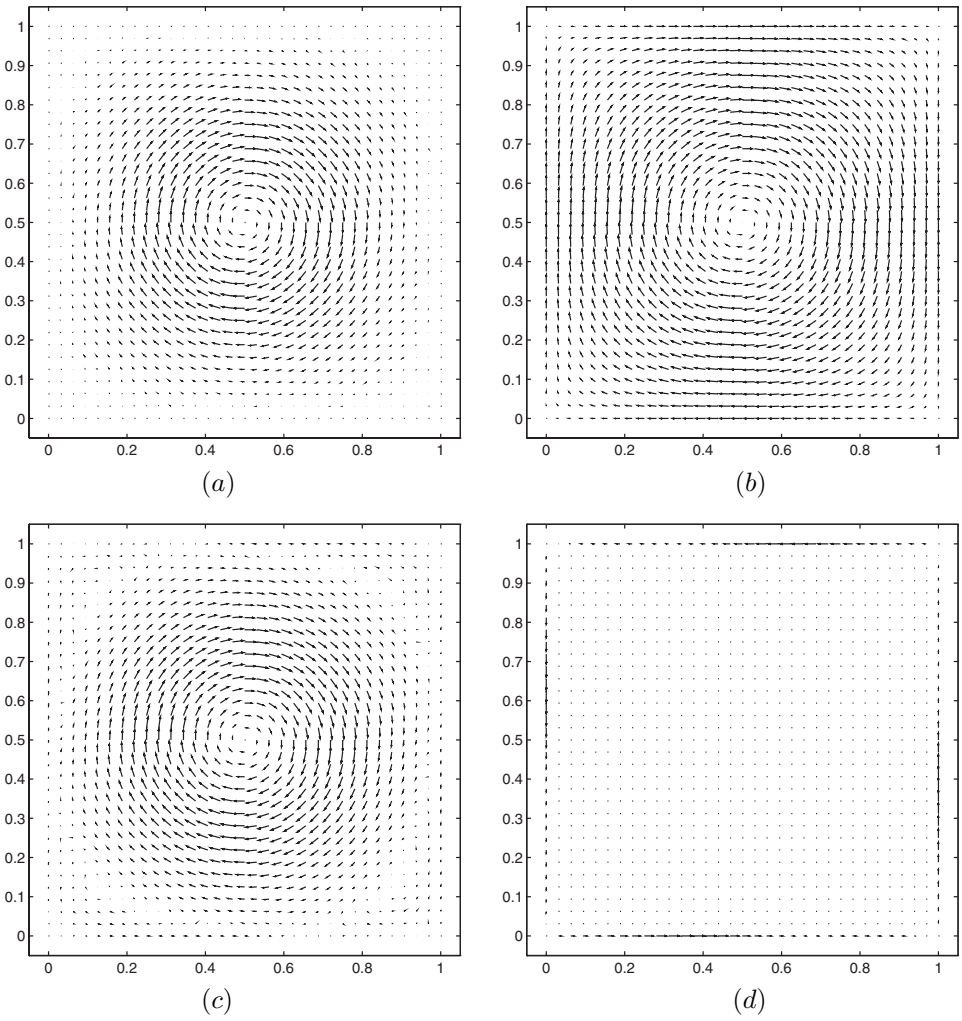

FIG. 5.2. *Information on cavity flow problems:* (a) *External force.* (b) *Computed velocity field for the simulation problem* (5.1) *for* $Re = 200$. (c) *Computed velocity field for the tangential BCP* (5.2) *for* $Re = 200$. (d) *Highlight of the boundary velocity in* (c).

of the interior flow. The achieved minimization can be observed in two ways: by comparing Figure 5.2(c) against Figure 5.2(b) and by comparing the value $\|\omega\|_h^2 \approx 32.5$ in Tables 5.2–5.6 to the value $\|\omega\|_h^2 \approx 55.4$ in Table 5.1.

5.3.2. Backward-facing step flow problems. In this subsection we present results for the backward-facing step problem. Table 5.9 contains results for the simulation problem (5.3) with $Re = 200$. The standard merit function is used in the line search, and an overlap $\delta = 1/32$ is used in the ASM preconditioner. The total number of Newton iterations does not change with the mesh size or the number of processors. The average number of GMRES iterations does not seem to follow (3.6) and (3.7) as closely as in the case of cavity flow simulation problems. But $\bar{l}$ behaves better when RAS is employed.

TABLE 5.9

Results for the backward-facing step flow simulation problem with $Re = 200$, standard ASM with overlap $\delta = 1/32$, and standard merit function $\|\mathbf{F}\|_2^2/2$. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 297,603 and $\|\omega\|_h^2 \approx 3.35$. See (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 5$	$t_n \approx 1.35$	$n = 5$	$t_n \approx 6.49$	$n = 5$	$t_n \approx 37.2$
	$\bar{l} \approx 54$	$\bar{t}_l \approx 0.22$	$\bar{l} \approx 59$	$\bar{t}_l \approx 1.15$	$\bar{l} \approx 64$	$\bar{t}_l \approx 6.85$
32	$n = 5$	$t_n \approx 1.37$	$n = 5$	$t_n \approx 5.96$	$n = 5$	$t_n \approx 30.7$
	$\bar{l} \approx 92$	$\bar{t}_l \approx 0.22$	$\bar{l} \approx 109$	$\bar{t}_l \approx 1.09$	$\bar{l} \approx 121$	$\bar{t}_l \approx 5.82$
64	$n = 5$	$t_n \approx 0.96$	$n = 5$	$t_n \approx 3.35$	$n = 5$	$t_n \approx 13.6$
	$\bar{l} \approx 89$	$\bar{t}_l \approx 0.13$	$\bar{l} \approx 102$	$\bar{t}_l \approx 0.58$	$\bar{l} \approx 102$	$\bar{t}_l \approx 2.50$

Next, we consider the tangential BCP (5.4) with $Re = 200$ and an augmented Lagrangian merit function in the line search. In this case the standard ASM with $\delta = 1/32$ does not result in a good enough preconditioner for GMRES; i.e., the maximum allowed number (5,000) of Krylov iterations is eventually achieved without any tolerance being satisfied. Table 5.10 presents results for the standard ASM with $\delta = 1/16$. Interestingly, the total number of Newton iterations decreases as the mesh is refined, although the overall effort to find a solution increases. Comparing these results with Table 5.9, we can see that the average number of GMRES iterations is now much more sensitive to both h and H.

With $\delta > 1/16$ we expect a better convergence of LNKSz. But instead of applying the standard ASM with $\delta = 1/8$, we apply, in Table 5.11, RAS with $\delta = 1/8$, trying to balance more sparse LU time spent on bigger extended subdomains with less communication time spent on RAS. The number of Newton iterations continues to decrease as the mesh is refined. With an overlap δ four times as large as the one used on the simulation problem, the average number of GMRES iterations now follows (3.6) and (3.7) more closely. Comparing values of $\bar{t}_l$ in Tables 5.11 and 5.9, we see that the average time spent on computing $\mathbf{p}^{(k)}$ can be more than 20 times greater in control problems than in simulation problems with the same mesh, instead of being around $8/3 \approx 3$ times greater, in accordance to the ratio between the number of variables per mesh point on control problems and on simulation problems.

We now increase the Reynolds number to $Re = 300$, and the performance data for the tangential BCP (5.4) is provided in Table 5.12. We use the augmented Lagrangian merit function in the line search and $\delta = 1/8$ in RAS. With the increase in Re, both

TABLE 5.10

Results for the backward-facing step tangential boundary flow control problem with $Re = 200$, standard ASM with overlap $\delta = 1/16$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 *and* $\|\omega\|_h^2 \approx 3.28$. *See* (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 40$	$t_n \approx 89.1$	$n = 31$	$t_n \approx 392$	$n = 23$	$t_n \approx 1610$
	$\bar{l} \approx 81$	$\bar{t}_l \approx 1.92$	$\bar{l} \approx 85$	$\bar{t}_l \approx 11.5$	$\bar{l} \approx 90$	$\bar{t}_l \approx 64.8$
32	$n = 41$	$t_n \approx 322$	$n = 31$	$t_n \approx 2100$	$n = 23$	$t_n \approx 13400$
	$\bar{l} \approx 605$	$\bar{t}_l \approx 7.65$	$\bar{l} \approx 1110$	$\bar{t}_l \approx 67.0$	$\bar{l} \approx 1800$	$\bar{t}_l \approx 581$
64	$n = 41$	$t_n \approx 209$	$n = 31$	$t_n \approx 1460$	$n = 23$	$t_n \approx 8520$
	$\bar{l} \approx 701$	$\bar{t}_l \approx 4.97$	$\bar{l} \approx 1410$	$\bar{t}_l \approx 46.6$	$\bar{l} \approx 2200$	$\bar{t}_l \approx 369$

TABLE 5.11

Results for the backward-facing step tangential boundary flow control problem with $Re = 200$, restricted ASM with overlap $\delta = 1/8$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 *and* $\|\omega\|_h^2 \approx 3.28$. *See* (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 40$	$t_n \approx 111$	$n = 31$	$t_n \approx 648$	$n = 23$	$t_n \approx 4990$
	$\bar{l} \approx 62$	$\bar{t}_l \approx 2.44$	$\bar{l} \approx 61$	$\bar{t}_l \approx 19.6$	$\bar{l} \approx 62$	$\bar{t}_l \approx 211$
32	$n = 41$	$t_n \approx 87.9$	$n = 32$	$t_n \approx 367$	$n = 23$	$t_n \approx 2250$
	$\bar{l} \approx 78$	$\bar{t}_l \approx 1.93$	$\bar{l} \approx 80$	$\bar{t}_l \approx 10.8$	$\bar{l} \approx 83$	$\bar{t}_l \approx 94.9$
64	$n = 40$	$t_n \approx 50.0$	$n = 31$	$t_n \approx 246$	$n = 23$	$t_n \approx 1560$
	$\bar{l} \approx 80$	$\bar{t}_l \approx 1.12$	$\bar{l} \approx 83$	$\bar{t}_l \approx 7.55$	$\bar{l} \approx 86$	$\bar{t}_l \approx 66.4$

TABLE 5.12

Results for the backward-facing step tangential boundary flow control problem with $Re = 300$, restricted ASM with overlap $\delta = 1/8$, and augmented Lagrangian merit function. n is the total number of Newton iterations, $\bar{l}$ is the average number of Krylov iterations per Newton iteration, t_n is the total time in seconds spent on all Newton iterations, and $\bar{t}_l$ is the average time in seconds, per Newton iteration, spent on solving for Newton steps. For the case of finest mesh, 768×128, the number of variables is 793,608 *and* $\|\omega\|_h^2 \approx 3.65$. *See* (5.6).

# Procs.	Mesh					
	192×32		384×64		768×128	
16	$n = 53$	$t_n \approx 148$	$n = 41$	$t_n \approx 875$	$n = 30$	$t_n \approx 6600$
	$\bar{l} \approx 61$	$\bar{t}_l \approx 2.47$	$\bar{l} \approx 62$	$\bar{t}_l \approx 20.1$	$\bar{l} \approx 63$	$\bar{t}_l \approx 214$
32	$n = 55$	$t_n \approx 118$	$n = 41$	$t_n \approx 498$	$n = 31$	$t_n \approx 3090$
	$\bar{l} \approx 81$	$\bar{t}_l \approx 1.97$	$\bar{l} \approx 86$	$\bar{t}_l \approx 11.5$	$\bar{l} \approx 88$	$\bar{t}_l \approx 96.6$
64	$n = 53$	$t_n \approx 67.1$	$n = 41$	$t_n \approx 333$	$n = 30$	$t_n \approx 2050$
	$\bar{l} \approx 82$	$\bar{t}_l \approx 1.13$	$\bar{l} \approx 86$	$\bar{t}_l \approx 7.75$	$\bar{l} \approx 89$	$\bar{t}_l \approx 66.8$

nonlinear (n) and average linear ($\bar{l}$) complexities increase, although the increase in the total number of Newton iterations is more pronounced.

Again, as done in the case of cavity flow control problems, in order to determine if smaller fixed GMRES tolerances would improve the number of Newton iterations

TABLE 5.13

Efficiency analysis of NKSz and LNKSz, w.r.t. the number of processors, for backward-facing step flow problems with a 768×128 *mesh and different values of Reynolds number Re and overlap* δ. E_n *is the efficiency related to* t_n, *the total time in seconds spent on all Newton iterations.* $\overline{E}_l$ *is the efficiency related to* $\bar{t}_l$, *the average time in seconds, per Newton iteration, spent on solving for Newton steps. See* (5.7). *Results are based on values of* t_n *and* $\bar{t}_l$ *from Tables* 5.9, 5.11, *and* 5.12.

# Procs.	Simulation $Re = 200,\ \delta = 1/16$ Standard ASM		Boundary control $Re = 200,\ \delta = 1/8$ Restricted ASM		$Re = 300,\ \delta = 1/8$ Restricted ASM	
	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$	E_n	$\overline{E}_l$
16	1.00	1.00	1.00	1.00	1.00	1.00
32	0.61	0.59	1.11	1.11	1.07	1.11
64	0.68	0.69	0.80	0.79	0.80	0.80

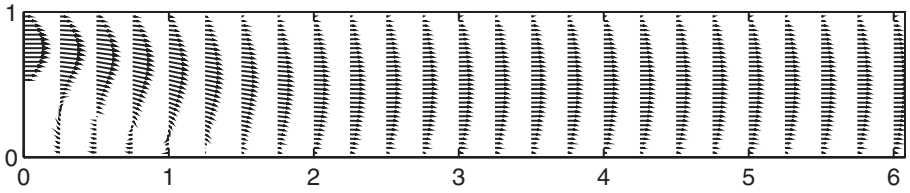

FIG. 5.3. *Computed velocity for the backward-facing step problem* (5.3) *with* $Re = 200$.

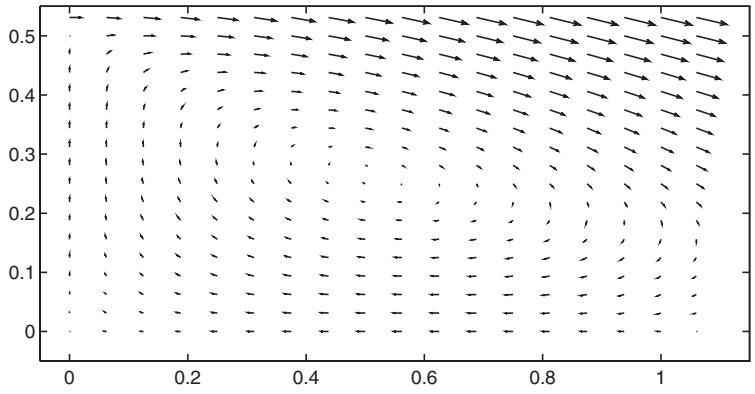

FIG. 5.4. *Highlight of the velocity field in the left bottom region of Figure* 5.3.

and thus, eventually, improve the total computing time, we rerun the tests of Table 5.12 with the same parameters, except for a GMRES relative tolerance of 1.0×10^{-6} instead of 1.0×10^{-4}, and a GMRES absolute tolerance of 1.0×10^{-13} instead of 1.0×10^{-6}. When compared to the results in Table 5.12, one or two Newton steps are saved in just two of the nine tests, while the increase in $\bar{l}$ (and $\bar{t}_l$, consequently) makes t_n increase in all nine cases.

In Table 5.13 some parallel efficiency results derived from data presented in previous tables are given.

Finally, we look at the computed velocity fields for $Re = 200$. Figures 5.3–5.5 are for the simulation problem (5.3), and Figures 5.6–5.8 are for the tangential BCP

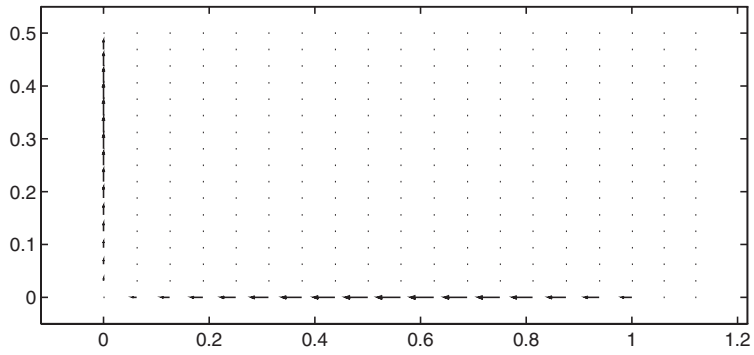

FIG. 5.5. *Highlight of the boundary velocity in the left bottom region of Figure* 5.3.

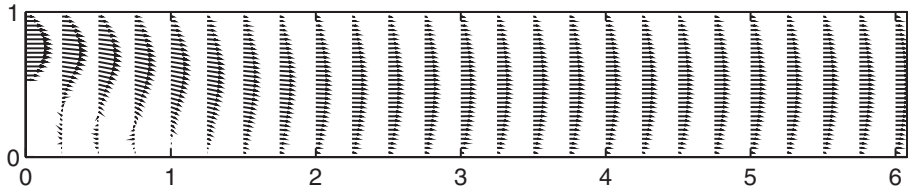

FIG. 5.6. *Computed velocity for the backward-facing step tangential BCP* (5.4) *with* $Re = 200$.

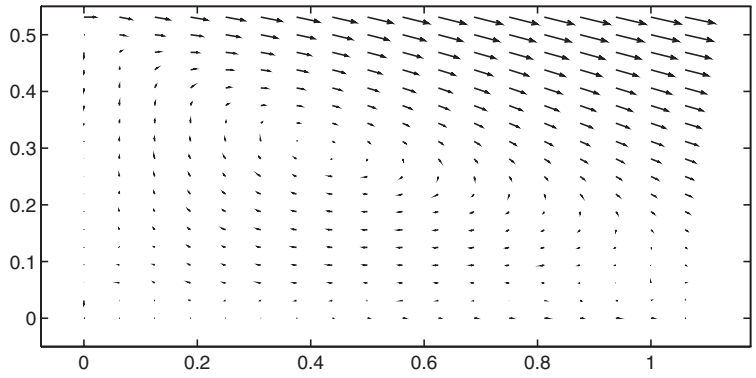

FIG. 5.7. *Highlight of the velocity field in the left bottom region of Figure* 5.6.

(5.4). The results are very similar to those of the cavity flow problem; more precisely speaking, Figures 5.4 and 5.8 show that the tangential boundary control acts in the opposite direction of the interior flow. The achieved minimization values can be measured either by comparing Figure 5.7 with Figure 5.4, or by comparing the value $\|\omega\|_h^2 \approx 3.28$ in Tables 5.10–5.11 with the value $\|\omega\|_h^2 \approx 3.35$ in Table 5.9.

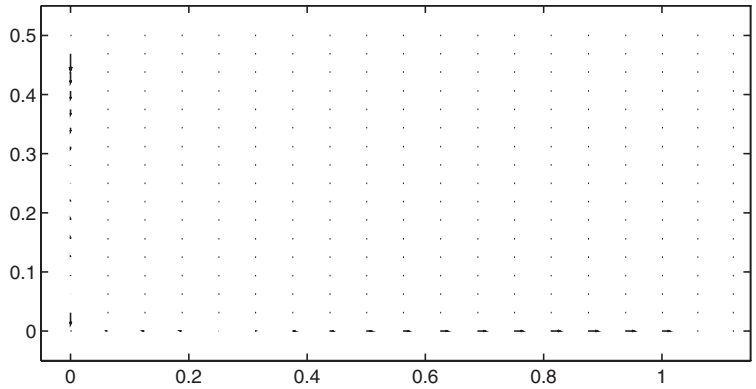

FIG. 5.8. *Highlight of the boundary velocity in the left bottom region of Figure* 5.6.

6. Conclusions. We have developed a general LNKSz algorithm for PDE constrained optimization problems and applied it to some tangential BCPs involving two-dimensional incompressible Navier–Stokes equations. In our numerical experiments the LNKSz algorithm, together with an augmented Lagrangian merit function, provides a fully parallel and robust solution method. The one-level additive Schwarz preconditioned GMRES, with a proper overlap, works well for the indefinite linearized KKT systems. A proper overlap for a control problem seems to be greater than a proper overlap for a simulation problem. More precisely, in our experiments the proper overlaps are two to four times greater in the control problems. For larger overlaps the restricted version of ASM seems to perform better than the standard ASM as a preconditioner for linearized KKT systems.

Our experiments also show that control problems are computationally more demanding, in terms of the total number of nonlinear iterations, the average number of linear iterations per Newton iteration, and the total computing time, than the corresponding simulation problems. The overall computational effort grows as Re increases, in terms of the number of both nonlinear and average linear iterations.

Acknowledgments. We would like to thank the PETSc team of Argonne National Laboratory for their help on using the PETSc library. We would also like to thank Professors G. Biros, O. Ghattas, S. Hou, and S. Ravindran, as well as two anonymous referees, for their useful comments.

REFERENCES

[1] F. ABERGEL AND R. TEMAM, *On some control problems in fluid mechanics*, Theoret. Comput. Fluid Dyn., 1 (1990), pp. 303–325.

[2] S. BALAY, K. BUSCHELMAN, W. D. GROPP, D. KAUSHIK, M. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *Portable, Extensible Toolkit for Scientific Computation (PETSc)*, http://www.mcs.anl.gov/petsc (2004).

[3] H. T. BANKS AND K. KUNISCH, *Estimation Techniques for Distributed Parameter Systems*, Systems Control Found. Appl. 1, Birkhäuser Boston, Boston, 1989.

[4] A. BATTERMANN AND M. HEINKENSCHLOSS, *Preconditioners for Karush-Kuhn-Tucker matrices arising in the optimal control of distributed systems*, in Control and Estimation of Distributed Parameter Systems, Internat. Ser. Numer. Math. 126, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser-Verlag, Basel, Switzerland, 1998, pp. 15–32.

[5] M. Bergounioux and K. Kunish, *Augmented Lagrangian algorithms for state constrained optimal control problems*, in Control and Estimation of Distributed Parameter Systems, Internat. Ser. Numer. Math. 126, W. Desch, F. Kappel, and K. Kunisch, eds., Birkhäuser-Verlag, Basel, Switzerland, 1998, pp. 33–48.

[6] M. Bergounioux and K. Kunish, *On the structure of the Lagrange multiplier for state-constrained optimal control problems*, Systems Control Lett., 48 (2002), pp. 16–176.

[7] T. R. Bewley, *Flow control: New challenges for a new renaissance*, Progr. Aerospace Sci., 37 (2001), pp. 21–58.

[8] G. Biros and O. Ghattas, *Parallel Lagrange–Newton–Krylov–Schur methods for PDE-constrained optimization. Part* I: *The Krylov–Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687–713.

[9] G. Biros and O. Ghattas, *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 J. Sci. Comput., 27 (2005), pp. 714–739.

[10] A. Borzi and K. Kunish, *A multigrid method for optimal control of time-dependent reaction diffusion processes*, in Fast Solution of Discretized Optimization Problems, Internat. Ser. Numer. Math. 138, K.-H. Hoffman, R. H. W. Hoppe, and V. Schulz, eds., Birkhäuser-Verlag, Basel, Switzerland, 2000, pp. 50–57.

[11] X.-C. Cai, M. Dryja, and M. Sarkis, *Restricted additive Schwarz preconditioners with harmonic overlap for symmetric positive definite linear systems*, SIAM J. Numer. Anal., 41 (2003), pp. 1209–1231.

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

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

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

[15] X.-C. Cai and M. Sarkis, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792–797.

[16] H. Choi, M. Hinze, and K. Kunish, *Instantaneous control of backward-facing step flows*, Appl. Numer. Math., 31 (1999), pp. 133–158.

[17] T. F. Coleman and J. J. Moré, *Estimation of sparse Jacobian matrices and graph coloring problems*, SIAM J. Numer. Anal., 20 (1983), pp. 187–209.

[18] O. Daube, *Resolution of the* 2*D Navier-Stokes equations in velocity-vorticity form by means of an influence matrix technique*, J. Comput. Phys., 103 (1992), pp. 402–414.

[19] J. E. Dennis, Jr., and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics Appl. Math. 16, SIAM, Philadelphia, 1996.

[20] M. Dryja and O. B. Widlund, *Domain decomposition algorithms with small overlap*, SIAM J. Sci. Comp., 15 (1994), pp. 604–620.

[21] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton methods*, SIAM J. Optim., 4 (1994), pp. 393–422.

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

[23] A. V. Fursikov, *Optimal Control of Distributed Systems. Theory and Applications*, Trans. Math. Monogr. 187, AMS, Providence, RI, 2000.

[24] A. Greenbaum, *Iterative Methods for Solving Linear Systems*, Frontiers Appl. Math. 17, SIAM, Philadelphia, 1997.

[25] G. Guj and F. Stella, *Numerical solutions of high-Re recirculating flows in vorticity-velocity form*, Internat. J. Numer. Methods Fluids, 8 (1988), pp. 405–416.

[26] M. D. Gunzburger, *Perspectives in Flow Control and Optimization*, Adv. Des. Control 5, SIAM, Philadelphia, 2002.

[27] M. D. Gunzburger and L. S. Hou, *Finite-dimensional approximation of a class of constrained nonlinear optimal control problems*, SIAM J. Control Optim., 34 (1996), pp. 1001–1043.

[28] M. Heinkenschloss, *Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet boundary control of Navier-Stokes flow*, in Optimal Control: Theory, Algorithms and Applications, Appl. Optim. 15, W. W. Hager and P. M. Pardalos, eds., Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998, pp. 178–203.

[29] M. Hinze and K. Kunish, *Second order methods for optimal control of time-dependent fluid flow*, SIAM J. Control Optim., 40 (2001), pp. 925–946.

[30] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results*, SIAM J. Sci. Comput., 20 (1999), pp. 1753–1777.
[31] F.-N. Hwang and X.-C. Cai, *A parallel nonlinear additive Schwarz preconditioned inexact Newton algorithm for incompressible Navier-Stokes equations*, J. Comput. Phys., 204 (2005), pp. 666–691.
[32] A. D. Ioffe and V. M. Tihomirov, *Theory of Extremal Problems*, Stud. Math. Appl. 6, North–Holland, Amsterdam, New York, 1979.
[33] K. Ito and K. Kunish, *The augmented Lagrangian method for equality and inequality constrained problems in Hilbert spaces*, Math. Programming, 46 (1990), pp. 341–360.
[34] K. Ito and K. Kunish, *The augmented Lagrangian method for parameter estimation in elliptic systems*, SIAM J. Control Optim., 28 (1990), pp. 113–136.
[35] B. Mohammadi and O. Pironneau, *Applied Shape Optimization for Fluids*, Numer. Math. Sci. Comput., The Clarendon Press, Oxford University Press, New York, 2001.
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Ser. Oper. Res., Springer-Verlag, New York, 1999.
[37] L. Quartapelle, *Numerical Solution of the Incompressible Navier-Stokes Equations*, Internat. Ser. Numer. Math. 113, Birkhäuser-Verlag, Basel, 1993.
[38] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
[39] B. Smith, P. Bjørstad, and W. Gropp, *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, Cambridge, UK, 1996.
[40] S. S. Sritharan, *Optimal Control of Viscous Flow*, SIAM, Philadelphia, 1998.
[41] V. M. Tihomirov, *Fundamental Principles of the Theory of Extremal Problems*, John Wiley & Sons, New York, 1986.
[42] A. Toselli and O. Widlund, *Domain Decomposition Methods—Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.
[43] M. Ulbrich, *Semismooth Newton methods for operator equations in function spaces*, SIAM J. Optim., 13 (2003), pp. 805–841.
[44] M. Ulbrich, S. Ulbrich, and M. Heinkenschloss, *Global convergence of trust-region interior-point algorithms for infinite-dimensional nonconvex minimization subject to pointwise bounds*, SIAM J. Control Optim., 37 (1999), pp. 731–764.
[45] C. R. Vogel, *Computational Methods for Inverse Problems*, Frontiers Appl. Math., SIAM, Philadelphia, 2002.