PARALLEL LAGRANGE-NEWTON-KRYLOV-SCHUR ALGORITHMS FOR PDE-CONSTRAINED OPTIMIZATION PART II: THE LAGRANGE-NEWTON SOLVER AND ITS APPLICATION TO OPTIMAL CONTROL OF STEADY VISCOUS FLOWS*

GEORGE BIROS† AND OMAR GHATTAS‡

Abstract. In this paper we follow up our discussion on algorithms that are suitable for optimization of systems governed by steady-state partial differential equations. In the first part of of this paper we proposed a Lagrange-Newton-Krylov-Schur method (LNKS) that uses Krylov iterations to solve the Karush-Kuhn-Tucker system of optimality conditions, but invokes a preconditioner inspired by reduced space quasi-Newton algorithms. In the second part we focus our discussion to the outer iteration and we provide details on how to obtain a robust and globally convergent algorithm. Newton's step is known to lead to divergence for points far from the optimum. Furthermore for highly nonlinear problems the computation of a step by itself is very difficult (for both QN-RSQP and LNKS methods). As a remedy we employ line search methods, hybrid Newton/quasi-Newton algorithms, truncated nonlinear iterations and continuation. We test the globalized LNKS algorithm on a optimal flow control problem were the constraints are the steady incompressible Navier-Stokes equations. The objective function is the minimization of the dissipation functional. We report results from runs on up to 128 processors on a T3E-900 at the Pittsburgh Supercomputing Center. Our numerical experiments demonstrate the very good scalability of the new method. Moreover, LNKS is an order of magnitude faster than reduced quasi-Newton SQP, and we are able to solve previously intractable problems of up to 800,000 state and 5,000 decision variables—at 5 times the cost of a single PDE solution.

Key words. Sequential quadratic programming, Adjoint methods, PDE-constrained optimization, optimal control, Lagrange-Newton-Krylov-Schur methods, Navier-Stokes, finite elements, preconditioners, indefinite systems, nonlinear equations, parallel algorithms

AMS subject classifications. 49K20, 65F10, 65K05, 65K10, 65J22, 65N55, 65W10, 65Y05, 65Y20, 76D05, 76D07, 76D55, 90C52, 90C55, 90C90, 93C20

1. Introduction.

In the first part of the paper we proposed a Lagrange-Newton-Krylov method for PDE-constrained optimization. We concentrated our discussion on the inner iteration: the solution of the linear system associated to a Newton step for the KKT optimality conditions. The algorithm is based on a Krylov solver combined with a Schur-type preconditioner which is equivalent to an approximate quasi-Newton RSQP step. We termed the method "Lagrange-Newton-Krylov-Schur", (LNKS) for concatenation of "Lagrange-Newton" method which is used in the outer iteration and "Krylov-Schur" is used to converge the inner iteration. We also provided theoretical and numerical evidence that these preconditioners work very well.

In the second part we follow up with algorithmic details on the outer Lagrange-Newton solver. We also look at more stringent test problems that contain many features of the most challenging PDE-constrained optimization problems: three-dimensionality, multicomponent coupling, large scale, nonlinearity, and ill-conditioning. The problem is one of optimal control of a viscous incompressible fluid by boundary velocities, a problem of both theoretical and industrial interest.

*This work is a part of the Terascale Algorithms for Optimization of Simulations (TAOS) project at CMU, with support from NASA grant NAG-1-2090, NSF grant ECS-9732301 (under the NSF/Sandia Life Cycle Engineering Program), and the Pennsylvania Infrastructure Technology Alliance. Computing services on the Pittsburgh Supercomputing Center's Cray T3E were provided under PSC grant BCS-960001P.

†Courant Institute of Mathematical Sciences, Department of Computer Science, New York University, New York, NY 10012, USA (biros@cs.nyu.edu).

‡Mechanics, Algorithms, and Computing Laboratory, Departments of Biomedical Engineering and Civil & Environmental Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA (oghattas@cs.cmu.edu).

Following Part I, we refer to the unknown PDE field quantities as the *state variables*; the PDE constraints as the *state equations*; solution of the PDE constraints as the *forward problem*; the inverse, design, or control variables as the *decision variables*; and the problem of determining the optimal values of the inverse, design, or control variables as the *optimization problem*.

This paper is organized as follows: in Section 2 we briefly review the problem formulation. We discuss algorithmic issues related to Lagrange-Newton methods and in particular globalization methodologies. We give details on three techniques: line search, mixing QN-RSQP steps with LNKS steps, and continuation. To enhance robustness, these methodologies are combined in the globalized LNKS algorithm. We also discuss inexact Newton methods and how they interact with a line search algorithm. In Section 3 we present the overall LNKS algorithm; in Section 4 we discuss the formulation of the optimal control problem for the Navier-Stokes equations; and in Section 5 we conclude with results from computations for a Poiseuille flow, a flow around a cylinder, and a flow around a Boeing-707 wing.

Notation: we use boldface characters to denote vector valued functions and vector valued function spaces. We use roman characters to denote discretized quantities and italics for their continuous counterparts. For example $\boldsymbol{u}$ will be the continuous velocity field and $\mathbf{u}$ will be its discretization. Greek letters are overloaded and whether we refer to the discretization or the continuous fields should be clear from context. We also use $(+)$ as a subscript or superscript to denote variable updates within an iterative algorithm.

2. The Newton solver. Let us reconsider the constrained optimization problem formulation,

$$\min_{\mathbf{x}\in\mathbb{R}^N} f(\mathbf{x}) \quad \text{subject to} \quad \mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{2.1}$$

where $\mathbf{x} \in \mathbb{R}^N$ are the optimization variables, $f : \mathbb{R}^N \to \mathbb{R}$ is the objective function and $\mathbf{c} : \mathbb{R}^N \to \mathbb{R}^n$ are the constraints. In our context these constraints are discretizations of the state equations. The Lagrangian,

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) := f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{c}(\mathbf{x}), \tag{2.2}$$

is used to convert the constrained optimization problem to a system of nonlinear equations. These equations are the first order optimality conditions:

$$\left\{ \begin{array}{c} \partial_x \mathcal{L} \\ \partial_\lambda \mathcal{L} \end{array} \right\} (\mathbf{x}, \boldsymbol{\lambda}) = \left\{ \begin{array}{c} \mathbf{g}(\mathbf{x}) + \mathbf{A}(\mathbf{x})^T \boldsymbol{\lambda} \\ \mathbf{c}(\mathbf{x}) \end{array} \right\} = \mathbf{0} \quad (\text{or } \mathbf{h}(\mathbf{q}) = \mathbf{0}). \tag{2.3}$$

We use $\mathbf{g}$ for the gradient of the objective function, $\mathbf{A}$ for the Jacobian of the constraints, and $\mathbf{W}$ for the Hessian of the Lagrangian. We use Newton's method to solve for $\mathbf{x}$ and λ. A Newton step on the optimality conditions is given by

$$\left[\begin{array}{cc} \mathbf{W} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{array} \right] \left\{ \begin{array}{c} \mathbf{p}_x \\ \mathbf{p}_\lambda \end{array} \right\} = - \left\{ \begin{array}{c} \mathbf{g} + \mathbf{A}^T \boldsymbol{\lambda} \\ \mathbf{c} \end{array} \right\} \quad (\text{ or } \mathbf{K}\mathbf{v} = -\mathbf{h}), \tag{2.4}$$

where $\mathbf{p}_x$ and $\mathbf{p}_\lambda$ are the updates of $\mathbf{x}$ and $\boldsymbol{\lambda}$ from current to next iterations. In Part I we reviewed the most popular algorithm for solving for a KKT point, the RSQP algorithm and its quasi-Newton variant (Algorithm 3 in the first paper). Although these algorithms are very efficient and robust, they do not scale very well with the number of decision variables. They avoid solving (2.4) directly, but require a large number of linearized forward solves and thus can be inefficient for large-scale PDE-constrained optimization. We argued that a better approach would be to stay in the full space and use a Krylov method to solve (2.4). For most problems however, the KKT matrix is notoriously ill-conditioned. In the first part of this paper we addressed this problem by proposing efficient preconditioning techniques. The key idea was to use an approximate QN-RSQP as a preconditioner. We showed that RSQP can be viewed as a block-LU factorization in which the reduced Hessian $\mathbf{W}_z$ is the Schur complement for the decision variables. Therefore, approximate versions of this factorization can be used as preconditioners. A sketch of the LNKS method is given by Algorithm 1.

Algorithm 1 Lagrange-Newton-Krylov-Schur

1: Choose $\mathbf{x}, \boldsymbol{\lambda}$
2: **loop**
3: Check for convergence
4: Compute $\mathbf{c}, \mathbf{g}, \mathbf{A}, \mathbf{W}$
5: Solve $\mathbf{P}^{-1}\mathbf{K}\mathbf{v} = \mathbf{P}^{-1}\mathbf{h}$ (Newton Step)
6: Update $\mathbf{x} = \mathbf{x} + \mathbf{p}_x$
7: Update $\boldsymbol{\lambda} = \boldsymbol{\lambda} + \mathbf{p}_\lambda$
8: **end loop**

Nevertheless, there are two questions that should be answered before we can claim a fast and robust general-purpose algorithm. The first question is whether LNKS algorithm is convergent for any initial guess $(\mathbf{x}_0, \boldsymbol{\lambda}_0)$ and the second one is whether we can utilize inexact[1] Newton methods to further accelerate LNKS. Within this framework we examine line search algorithms, mixing QN-RSQP and LNKS algorithms, continuation, and inexact Newton methods.

2.1. Line search methods. Algorithm 1 is only locally convergent. Popular methodologies to globalize Newton's method include line search and trust region algorithms. Details can be found in [19]. Recently, there has been an increased interest in trust region methodologies, especially in combination with RSQP and inexact Newton methods. These methods have been successfully applied to PDE-constrained optimization [12], [15], [17]. Global convergence proofs for these methods can be found in [3]. Trust region methods are based on the Steihaug modification [22] of the Conjugate Gradient (CG) algorithm. However, this approach works well only with positive definite systems. It is not obvious how to use trust-regions with an indefinite Krylov solver (which is required for the KKT system) and thus we have opted to use a line search algorithm.

The basic component of a line search algorithm is the choice of a merit function: a scalar function (on the optimization variables) that monitors the progress of the algorithm. In contrast with unconstrained optimization, the choice of a merit function is not straightforward since we are trying to balance minimization of the objective function with feasibility. Two common choices are the l_1-merit function,

$$\phi(\mathbf{x}) := f + \rho_\phi \|\mathbf{c}\|_1, \tag{2.5}$$

and the augmented Lagrangian,

$$\phi(\mathbf{x}, \boldsymbol{\lambda}) := f + \mathbf{c}^T\boldsymbol{\lambda} + \frac{\rho_\phi}{2}\mathbf{c}^T\mathbf{c}. \tag{2.6}$$

The scalar ρ_ϕ is the *penalty parameter*—a weight chosen to bring the right balance between minimizing the objective function and minimizing the residuals of the constraints. Both

[1]Some authors use the term "truncated" instead of "inexact".

functions are "exact", provided the penalty parameter is large enough. By *exact* we mean that if $(\mathbf{x}_*, \boldsymbol{\lambda}_*)$ is a minimizer for (2.1), then it is also an (unconstrained) minimizer for the merit function. A crucial property of a merit function is that it should accept unit step lengths close to a solution in order to allow full Newton steps and thus quadratic convergence. The l_1-merit function suffers from the "Maratos" effect, that is, sometimes it rejects good steps and slows down the algorithm. The augmented Lagrangian merit function does not exhibit such behavior but its drawback is that it requires accurate estimates of the Lagrange multipliers.

The outline of a general line search method is given in Algorithm 2. To simplify notation we use $\phi(\alpha)$ for $\phi(\mathbf{q}+\alpha\mathbf{v})$ and $\phi(0)$ for $\phi(\mathbf{q})$ (likewise for the derivative $\nabla\phi$). The algorithm

Algorithm 2 Line search

1: Choose $\mathbf{q}$, $\delta_A > 0$ and κ_1, κ_2 arbitrary constants (strictly positive)
2: **while** Not converged **do**
3: Compute search direction $\mathbf{v}$ so that
$\mathbf{v}^T\nabla\phi(0) < 0$
$|\ \mathbf{v}^T\nabla\phi(0)| \geq \kappa_1 \|\mathbf{v}\| \, \|\nabla\phi(0)\|$
$\|\mathbf{v}\| \geq \kappa_2 \|\nabla\phi(0)\|$
4: Compute α such that $\phi(\alpha) \leq \phi(0) + \alpha\delta_A\mathbf{v}^T\nabla\phi(0)$ Armijo condition
5: Set $\mathbf{q} = \mathbf{q} + \alpha\mathbf{v}$
6: **end while**

used to compute the search direction $\mathbf{v}$ is intentionally left unspecified. All that matters to ensure global convergence is the properties of the merit function and the properties of $\mathbf{v}$. Step 3 in Algorithm 2 lists three conditions on $\mathbf{v}$: descent direction, sufficient angle and sufficient step size [7]. The condition in Step 4 is often called the Armijo condition. If ϕ is bounded and has a minimum, and if $\mathbf{v}$ is bounded, Algorithm 2 is guaranteed to converge to a local minimum [18]. We use a simple backtracking line search, with a factor of 0.5. The search is bounded so that $\alpha_{min} \leq \alpha \leq 1$. As mentioned before, the choice of the penalty parameter has a great effect on the performance of the algorithm.

For a step computed by (quasi-Newton) Algorithm 3, in Part I the update for the l_1-merit function is relatively straightforward. The directional derivative for a search direction $\mathbf{p}_x$ is given by

$$\nabla\phi^T\mathbf{p}_x = \mathbf{g}^T\mathbf{p}_x - \rho_\phi\|\mathbf{c}\|_1. \tag{2.7}$$

If $\mathbf{W}_z$ is positive definite it can be shown that by setting

$$\rho_\phi = \|\boldsymbol{\lambda}\|_\infty + \delta, \quad \delta > 0, \tag{2.8}$$

we obtain a descent direction. In our numerical experiments we have used l_1 with QN-RSQP and augmented Lagrangian with LNKS. The l_1-merit function performed reasonably well. However, we did observe the Maratos effect. To overcome this obstacle we have implemented a second order correction, in which an extra normal step towards feasibility is taken ([19], p.570).

When an augmented Lagrangian merit function is used, the penalty parameter should be chosen differently. The directional derivative of the augmented Lagrangian merit function is given by

$$\nabla\phi^T\mathbf{v} = (\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda} + \rho_\phi\mathbf{A}^T\mathbf{c})^T\mathbf{p}_x + \mathbf{c}^T\mathbf{p}_\lambda. \tag{2.9}$$

Lagrange multipliers slightly complicate the algorithm since we have to compute $\mathbf{p}_\lambda$. Some researchers consider $\boldsymbol{\lambda}$ as a function of $\mathbf{x}$ ([2], [6]), others treat it as an independent variable

([3], [21]), or simply ignore it by setting $\mathbf{p}_\lambda = 0$ [23]. In LNKS we solve for $\boldsymbol{\lambda}$ simultaneously with $\mathbf{x}$ and it is natural to use the step $\mathbf{p}_\lambda$. On the other hand, RSQP uses $\boldsymbol{\lambda} = -\mathbf{A}_s^{-T}\mathbf{g}_s$ and it seems natural to consider $\boldsymbol{\lambda}$ a function of $\mathbf{x}$. In this case the last term in (2.9) is given by

$$\mathbf{c}^T\mathbf{p}_\lambda = \mathbf{c}^T(\partial_x\boldsymbol{\lambda})\mathbf{p}_x,$$

where

$$\partial_x\boldsymbol{\lambda} := -\mathbf{A}_s^{-T}\begin{bmatrix} \mathbf{W}_{ss} & \mathbf{W}_{sd} \end{bmatrix}.$$

(However, this formula can not be used with QN-RSQP methods since second derivatives are not available.) If we set

$$\rho_\phi = \frac{(\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda})\mathbf{p}_x + \mathbf{c}^T\mathbf{p}_\lambda + \delta}{\mathbf{c}^T\mathbf{A}\mathbf{p}_x}, \quad \delta > 0, \tag{2.10}$$

we obtain a descent direction.

2.2. Combining QN-RSQP with LNKS. For iterates far from the solution, relying solely on a line search algorithm will not work since the Newton step is likely to be of poor quality. Usually global convergence can be shown if the reduced Hessian ($\mathbf{W}_z$) is positive definite (and not the full Hessian $\mathbf{W}$). If $\mathbf{W}_z$ is positive definite (and assuming the system (2.4) is solved exactly), then the resulting step $\mathbf{v}$ satisfies the Armijo descent criterion. Far from the minimum, however, $\mathbf{W}_z$ can be singular or indefinite. On the other hand, certain quasi-Newton methods, like BFGS, are preferable for iterates far from the solution since they can guarantee positive definiteness. For this reason (and for preconditioning purposes) LNKS does maintain a BFGS approximation for $\mathbf{W}_z$: if a computed search direction fails to satisfy the Armijo criterion we discard it and we switch to a QN-RSQP step.

2.3. Continuation. One of the standard assumptions in global convergence proofs is the non singularity of the constraint Jacobian—for all iterates. For highly nonlinear PDEs, like the Navier-Stokes equations, this is a rather unrealistic assumption. Even if the Jacobian is nonsingular, severe ill-conditioning will cause both QN-RSQP and LNKS algorithms to stall. Indeed, in our numerical experiments the most difficult computation, for iterates far from the solution, was converging the $\mathbf{A}_s$-related linear solves. Krylov solvers reached their maximum iteration counts without a significant decrease the linear system residual. As a result, the iterates were of very poor quality and the algorithm stagnated as it was impossible to compute a search direction, be it from QN-RSQP or LNKS iteration.

A remedy to this problem is parameter continuation. This idea (in its simplest form) works when we can express the nonlinearity of the problem as a function of a single scalar parameter. Continuation is particularly suitable for PDE-constrained optimization because it is quite typical for a PDE to have a parameter that scales the nonlinearities. Examples of such parameters are the Reynolds and Mach numbers in fluid mechanics, the Peclet number in general convection diffusion equations, or the Hartman number in magnetohydrodynamics. In problems where such a parameter cannot be found an alternative is pseudo-transient continuation [16].

Continuation allows uphill steps (unlike monotone line search methods) to be taken and generates good initial guesses, not only for the optimization variables, but also for the penalty parameter in the merit function. The most important feature of the continuation algorithm is that it globalizes trivially (for certain problems[2]). If the continuation step brings the next

[2]This is true only when the initial problem is a well posed quadratic programming problem (like Stokes) and all iterates on the continuation path are far from turning and bifurcation points.

iterate outside the attraction basin of the Newton method the we simply reduced the step size. In principle, the method globalizes LNKS without the need to use line search or some other globalization strategy. Nevertheless, taking a large number of continuation steps significantly slows down the algorithm. Experience from our numerical experiments suggests that the best strategy is a combination of line searching, quasi-Newton steps, and continuation.

2.4. Inexact Newton's method. Before we discuss inexact Newton's method in the context of LNKS, we briefly summarize a few results for a general nonlinear system of equations. Assume we want to solve $\mathbf{h}(\mathbf{q}) = \mathbf{0}$. Further assume the following: (1) $\mathbf{h}$ and $\mathbf{K} := \partial_q \mathbf{h}$ are sufficiently smooth in a neighborhood of a solution $\mathbf{q}_*$; (2) at each iteration an inexact Newton method computes a step $\mathbf{v}$ that satisfies

$$\|\mathbf{K}\mathbf{v} + \mathbf{h}\| \leq \eta_N \|\mathbf{h}\|, \tag{2.11}$$

where η_N is often called the *forcing term*. It can be shown that if $\eta_N < 1$ then $\mathbf{q} \to \mathbf{q}^*$ linearly; if $\eta_N \to 0$ then $\mathbf{q} \to \mathbf{q}^*$ superlinearly; and if $\eta_N = \mathcal{O}(\|\mathbf{h}\|)$ then we recover the quadratic convergence rates of a Newton method. The forcing term is usually given by

$$\eta_N = \frac{\|\mathbf{h}_{(+)} - \mathbf{h} - \mathbf{K}\mathbf{v}\|}{\|\mathbf{h}\|}. \tag{2.12}$$

Other alternatives exist (for details see [5]).

The extension of inexact methods to optimization is relatively easy, especially for unconstrained problems. In [12] global convergence proofs are given for a trust region RSQP-based algorithm. Close to a KKT point the theory for Newton's method applies and one can use the analysis presented in [4] to show that the inexact version of the LNKS algorithm converges. However, the line search we are using is not based on the residual of the KKT equations but instead on the merit function discussed in the previous session. That means that an inexact step that simply reduces $\mathbf{h}$ may not satisfy the merit function criteria. *We will show that for points which are close enough to the solution, inexactness does not interfere with the line search*. Our analysis is based on the augmented Lagrangian merit function[3]. We assume that, locally, $\mathbf{A}$ and $\mathbf{K}$ are non-singular and uniformly bounded. We define $\kappa_1 := \max\|\mathbf{K}^{-1}(\mathbf{q})\|$ for $\mathbf{q}$ in the neighborhood of the solution $\mathbf{q}_*$. We also define $\mathbf{v}$ as the exact solution of the (linearized) KKT system so that

$$\mathbf{K}\mathbf{v} + \mathbf{h} = \mathbf{0}, \tag{2.13}$$

and $\tilde{\mathbf{v}}$ the approximate solution so that

$$\mathbf{K}\tilde{\mathbf{v}} + \mathbf{h} = \mathbf{r}. \tag{2.14}$$

We also have $\|\mathbf{r}\| = \eta\|\mathbf{h}\|$, $0 < \eta \leq \eta_N$, from the inexact Newton stopping criterion (2.11). By (2.3) we get that $\|\mathbf{h}\|^2 = \|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\|^2 + \|\mathbf{c}\|^2$ and since $\mathbf{A}$ is bounded, there is constant κ_2 such that:

$$\|\mathbf{A}^T\mathbf{c}\| \leq \kappa_2\|\mathbf{h}\|. \tag{2.15}$$

We assume the following: (1) ρ_ϕ is sufficiently large so that the merit function is exact and $\|\nabla\phi\| \geq \kappa_3\|\mathbf{h}\|$ for some constant κ_3; (2) $\mathbf{v}$ satisfies the gradient and length conditions,

[3]For brevity we drop the subscript from ϕ_L and we just use the symbol ϕ.

as well as the Armijo condition. From the latter it is immediate that $\mathbf{v}$ satisfies the Cauchy fraction condition[4]:

$$|\nabla\phi^T\mathbf{v}| \geq 2\kappa_4\|\nabla\phi\|^2. \tag{2.16}$$

We will show that if η is small enough then the approximate step $\tilde{\mathbf{v}}$ satisfies the Cauchy fraction, the gradient, and length conditions. Then, we will use a theorem from [18] to conclude that the Armijo condition is satisfied with unit step lengths. Therefore if $\eta = \mathcal{O}(\|\mathbf{h}\|)$ quadratic convergence is preserved.

From (2.13) and (2.14) we have

$$\nabla\phi^T\tilde{\mathbf{v}} = \nabla\phi^T\mathbf{v} + \nabla\phi^T\mathbf{K}^{-1}\mathbf{r},$$

and thus by (2.16) we get

$$|\nabla\phi^T\tilde{\mathbf{v}}| \geq 2\kappa_4\|\nabla\phi\|^2 - |\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}|.$$

Therefore to satisfy the Cauchy fraction condition

$$|\nabla\phi^T\tilde{\mathbf{v}}| \geq \kappa_4\|\nabla\phi\|^2, \tag{2.17}$$

we need to show that

$$|\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}| \leq \kappa_4\|\nabla\phi\|^2. \tag{2.18}$$

The gradient of the merit function is given by

$$\nabla\phi = \mathbf{h} + \rho_\phi \left\{ \begin{array}{c} \mathbf{A}^T\mathbf{c} \\ \mathbf{0} \end{array} \right\},$$

and thus

$$\begin{aligned}
|\nabla\phi^T\mathbf{K}^{-1}\mathbf{r}| &= |\mathbf{h}^T\mathbf{K}^{-1}\mathbf{r} + \rho_\phi \left\{ \begin{array}{c} \mathbf{A}^T\mathbf{c} \\ \mathbf{0} \end{array} \right\}^T \mathbf{K}^{-1}\mathbf{r}| \\
&\leq \kappa_1 \left(\|\mathbf{h}\|\,\|\mathbf{r}\| + \rho_\phi\|\mathbf{A}^T\mathbf{c}\|\,\|\mathbf{r}\|\right) \\
&\leq \kappa_1\eta \left(\|\mathbf{h}\|^2 + \rho_\phi\|\mathbf{A}^T\mathbf{c}\|\,\|\mathbf{h}\|\right) \\
&\leq \kappa_1\eta(1+\rho_\phi\kappa_2)\|\mathbf{h}\|^2 \\
&\leq \kappa_1\eta(1+\rho_\phi\kappa_2)\frac{\|\nabla\phi\|^2}{\kappa_3^2}.
\end{aligned}$$

If

$$\eta \leq \frac{\kappa_4\ \kappa_3^2}{\kappa_1(1+\rho_\phi\kappa_2)} \tag{2.19}$$

then (2.18) holds. If we choose a superlinearly convergent inexact Newton variant then

$$\eta \leq \eta_N \to 0,$$

[4]The Cauchy step is a steepest descent step for the merit function.

and therefore close to the solution (2.19) holds. We also have that

$$\begin{aligned}
\tilde{\mathbf{v}} &= \mathbf{K}^{-1}(\mathbf{r} - \mathbf{h}) \\
\|\tilde{\mathbf{v}}\| &\leq \kappa_1 (1+\eta)\|\mathbf{h}\| \\
\|\tilde{\mathbf{v}}\| &\leq \kappa_1 (1+\eta)\frac{\|\nabla\phi\|}{\kappa_3} \\
\|\tilde{\mathbf{v}}\| &\leq \kappa_5 \|\nabla\phi\|.
\end{aligned} \tag{2.20}$$

By combining (2.20) and (2.17) we get

$$|\nabla\phi^T \tilde{\mathbf{v}}| \geq \kappa_4 \|\nabla\phi\|^2 \geq \kappa_4\kappa_5 \|\nabla\phi\| \, \|\tilde{\mathbf{v}}\|,$$

and

$$\begin{gathered}
\|\nabla\phi\| \, \|\tilde{\mathbf{v}}\| \geq \kappa_6 |\nabla\phi^T \tilde{\mathbf{v}}| \geq \kappa_6\kappa_4 \|\nabla\phi\|^2 \;\Rightarrow \\
\|\tilde{\mathbf{v}}\| \geq \kappa_7 \|\nabla\phi\|.
\end{gathered}$$

That is, the gradient and angle conditions are satisfied. It can be shown ([18], Theorem 10.6) that there is α, bounded bellow, so that Armijo condition holds true. Thus by choosing δ_A small enough, the Armijo condition is satisfied with unit steplength. Hence the quadratic convergence rate associated with Newton's method is observed, i.e. the inexactness does not interfere with the merit function. In addition it can be shown that the augmented Lagrangian merit function allows unit steplength near the solution (see [6], [21] and the references therein). Finally, notice that convergence does not require that $\eta_N \to 0$; it only requires that η_N is small enough. This is in contrast with inexact reduced space methods which require the tolerances to become tighter as the iterates approach the solution [12].

2.5. The globalized LNKS algorithm. In the previous sections we discussed the various features of our globalization strategies. In this section we summarize by giving a high-level description of implementation details and heuristics we are using in the globalized LNKS. The basic steps of our method are given in Algorithm 3. The algorithm uses a three-level iteration. In the outer iteration the continuation parameter is gradually increased until the target number is reached.The middle iterations correspond to Lagrange-Newton linearizations of the optimality system for a fixed continuation number. Finally, the inner iteration consists of two core branches: the computation of an LNKS search direction and the computation of the search direction with QN-RSQP. The default branch is the LNKS step. If this step fails to satisfy the line search algorithm conditions then we switch to QN-RSQP. If QN-RSQP fails too, then we reduce the continuation parameter Re and we return to the outer loop.

Here we summarize the basic steps of the algorithm. We also mention several heuristics we have used to improve the performance of the method.

- Linear solves at steps 8, 16 and 17 are performed inexactly. We follow [5] in choosing the forcing term. In steps 16, and 17 the forcing term formula uses $\|\mathbf{c}\|$ and $\|\mathbf{g}_z\|$, whereas in step 8 it uses $\|\mathbf{h}\|$. In 7 we also experimented with $\|\nabla\phi\|$ as a way to control inexactness but we found no significant difference.
- In step 6 we use the adjoint variables to update the reduced gradient. This is equivalent to $\mathbf{g}_z = \mathbf{g}_d - \mathbf{A}_d^T \mathbf{A}_s^{-T} \mathbf{g}_s$, if $\boldsymbol{\lambda}$ is computed by solving exactly $\mathbf{A}_s^T \boldsymbol{\lambda} + \mathbf{g}_s = \mathbf{0}$. When $\boldsymbol{\lambda}$ is taken from LNKS, it includes second order terms (which reduce to zero as we approach the solution), and when $\boldsymbol{\lambda}$ is taken from QN-RSQP it also introduces extra error since we never solve the linear systems exactly.
- We use various heuristics to bound the penalty parameter and if possible reduce it. A new penalty parameter ρ_ϕ^+ is computed using the LNKS step and formula

Algorithm 3 Globalized LNKS

1:	Choose $\mathbf{x}_s,\ \mathbf{x}_d,\ \rho_\phi,\ t,\ \delta_A$, set $Re = Re_{start},\ tol = tol_0$	
2:	$\mathbf{A}_s^T\boldsymbol{\lambda} + \mathbf{g}_s \approx \mathbf{0}$	solve inexactly for $\boldsymbol{\lambda}$
3:	**while** $Re \neq Re_{target}$ **do**	
4:	**loop**	
5:	Evaluate $f,\ \mathbf{c},\ \mathbf{g},\ \mathbf{A},\ \mathbf{W}$	
6:	$\mathbf{g}_z = \mathbf{g}_d + \mathbf{A}_d^T\boldsymbol{\lambda}$	
7:	Check convergence: $\\|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\\| \leq tol$ and $\\|\mathbf{c}\\| \leq tol$	
8:	$\mathbf{P}^{-1}\mathbf{K}\mathbf{v} + \mathbf{P}^{-1}\mathbf{h} \approx \mathbf{0}$	solve inexactly for $\mathbf{v}$
9:	Compute ρ_ϕ such that $\nabla\phi^T(0)\mathbf{v} \leq 0$	
10:	Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{v})$	
11:	**if** Line search failed **then**	
12:	Compute α s.t. $\\|\mathbf{h}(\alpha)\\| < t\\|\mathbf{h}(0)\\|$	
13:	**end if**	
14:	**if** LNKS step failed **then**	
15:	$\mathbf{B}_z\mathbf{p}_d = -\mathbf{g}_z$	solve inexactly for $\mathbf{p}_d$
16:	$\mathbf{A}_s\mathbf{p}_s + \mathbf{A}_d\mathbf{p}_d + \mathbf{c} \approx \mathbf{0}$	solve inexactly for $\mathbf{p}_s$
17:	$\mathbf{A}_s^T\boldsymbol{\lambda}_+ + \mathbf{g}_s \approx \mathbf{0}$	solve inexactly for $\boldsymbol{\lambda}_+$
18:	Compute α s.t. $\phi(\alpha) \leq \phi(0) + \delta_A\alpha(\nabla\phi^T(0)\mathbf{v})$	
19:	**if** Line search on QN-RSQP step failed **then**	
20:	Reduce Re and go to step 5.	
21:	**end if**	
22:	**end if**	
23:	$\boldsymbol{\lambda}_+ = \boldsymbol{\lambda} + \mathbf{p}_\lambda$	(only for LNKS step)
24:	$\mathbf{x}_+ = \mathbf{x} + \mathbf{p}_x$	
25:	**end loop**	
26:	$Re = Re + \Delta Re$	
27:	Tighten tol	
28:	**end while**	

(2.10). If $\rho_\phi^+ > 4\rho_\phi$ we update the penalty parameter and we switch to QN-RSQP. If $\rho_\phi^+ < \rho_\phi/4$ we *reduce* the penalty parameter and set $\rho_\phi^+ = 0.5\rho_\phi$. We also reduce the penalty parameter if there is a successful search on the KKT residual (step 12).

- We allow for non-monotone line searches. If the LNKS step is rejected by the merit function line search we do not switch immediately to QN-RSQP. Instead we do a line search (step 12) on the KKT residual and if the step is accepted we use it to update the variables for the next iteration. However, we do store the iterate and the merit function gradient, and we insist that some step satisfies the conditions of the merit line search (evaluated at the failure point) after a fixed number of iterations. Otherwise, we switch to QN-RSQP. Typically, we permit two steps before we demand reduction of the merit function.
- A Lanzos algorithm can be used to (approximately) check the second-order optimality conditions. If the lowest eigenvalue of $\tilde{\mathbf{W}}_z$ is negative then a QN-RSQP step is taken without computing the full-space directions. The eigenvalues are frozen through a single continuation step, but if a negative direction is detected they are recomputed at each SQP iteration.

In the next section we study an optimal control problem of the steady incompressible Navier-Stokes equations. We cite results on the existence and uniqueness of solutions and make comparisons between the discrete and continuous forms of the optimality conditions.

3. Formulation of an Optimal Control Problem. In this section we turn our attention to the formulation and well-posedness of a specific optimization problem: the Dirichlet control of the steady incompressible Navier-Stokes equations. We present the continuous form of the Karush-Kuhn-Tucker optimality conditions and we cite convergence results for finite element approximations from [13] and [14]. A survey and articles on this topic can be found in [9]. More on the Navier-Stokes equations can be found in [8, 11]. We are studying problems in which we specify both Dirichlet and Neumann boundary conditions. The controls are restricted to be only of Dirichlet type but the theory is similar for distributed and Neumann controls [13].

We use the velocity-pressure $(\boldsymbol{u}, p)$ form of the incompressible steady state Navier-Stokes equations. We begin by writing the following strong form of the flow equations:

$$
\begin{aligned}
-\nu\nabla \cdot (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) + (\nabla \boldsymbol{u})\boldsymbol{u} + \nabla p = \boldsymbol{b} \quad &\text{in} \quad \Omega,\\
\nabla \cdot \boldsymbol{u} = 0 \quad &\text{in} \quad \Omega,\\
\boldsymbol{u} = \boldsymbol{u}_g \quad &\text{on} \quad \Gamma_u,\\
\boldsymbol{u} = \boldsymbol{u}_d \quad &\text{on} \quad \Gamma_d,\\
-p\boldsymbol{n} + \nu(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T)\boldsymbol{n} = \boldsymbol{0} \quad &\text{on} \quad \Gamma_N.
\end{aligned}
\tag{3.1}
$$

Here $\nu = 1/Re$ and the decision variables are the velocities $\mathbf{u}_d$ on Γ_d. For a forward solve we need not distinguish between Γ_d and Γ_u. In the optimization problem however, $\boldsymbol{u}_d$ is not known. We will present a mixed formulation that treats the tractions on the Dirichlet boundary Γ_d as additional unknown variables. The traction variables here play the role of Lagrange multipliers (not to be confused with the Lagrange multipliers or the optimal control problem) and are used to enforce the Dirichlet boundary conditions [1].

With $L^2(\Omega)$ we denote the space of scalar functions which are square-integrable in Ω, and with $\boldsymbol{H}^1(\Omega)$ we denote vector functions whose first derivatives are in $\boldsymbol{L}^2(\Omega)$. $\boldsymbol{H}^{1/2}(\Gamma)$ is the trace space (the restriction on Γ) of functions belonging to $\boldsymbol{H}^1(\Omega)$. Finally $\boldsymbol{H}^{-k}(D)$ is the set of bounded linear functionals on functions belonging to $\boldsymbol{H}^k(D)$, where D is some smooth domain in $\mathbb{R}^3$. We also define $\boldsymbol{V} := \left\{\boldsymbol{v} \in \boldsymbol{H}^1(\Omega) : \boldsymbol{v}|_{\Gamma_u} = \boldsymbol{0}\right\}$. We define the following bilinear and trilinear forms associated with the Navier-Stokes equations:

$$
\begin{aligned}
a(\boldsymbol{u}, \boldsymbol{v}) &:= \int_\Omega (\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T) \cdot (\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T)\, d\Omega \quad \forall\, \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{H}^1(\Omega),\\
c(\boldsymbol{w}, \boldsymbol{u}, \boldsymbol{v}) &:= \int_\Omega (\nabla \boldsymbol{u})\boldsymbol{w} \cdot \boldsymbol{v}\, d\Omega \quad \forall\, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in \boldsymbol{H}^1(\Omega),\\
b(q, \boldsymbol{v}) &:= \int_\Omega -q\nabla \cdot \boldsymbol{v}\, d\Omega \quad \forall\, q \in L,\ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega).
\end{aligned}
$$

We also use the notation $(\boldsymbol{x}, \boldsymbol{y})_D$ for $\int_D \boldsymbol{x} \cdot \boldsymbol{y}\, dD$.

In the weak formulation of (3.1) we seek $\boldsymbol{u} \in \boldsymbol{H}^1(\Omega),\ p \in L^2(\Omega)$ and $\boldsymbol{\sigma} \in H^{-1/2}(\Gamma_d)$ such that:

$$
\begin{aligned}
\nu a(\boldsymbol{u}, \boldsymbol{v}) + c(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) + b(p, \boldsymbol{v}) - (\boldsymbol{\sigma}, \boldsymbol{v})_{\Gamma_d} = (\boldsymbol{f}, \boldsymbol{v})_\Omega \qquad & \forall\, \boldsymbol{v} \in \boldsymbol{V},\\
b(q, \boldsymbol{u}) = 0 \qquad & \forall\, q \in L^2(\Omega),\\
-\,(\boldsymbol{t}, \boldsymbol{u})_{\Gamma_d} = -(\boldsymbol{t}, \boldsymbol{u}_d)_{\Gamma_d} \qquad & \forall\, \boldsymbol{t} \in \boldsymbol{H}^{-1/2}(\Gamma_d).
\end{aligned}
\tag{3.2}
$$

We also define $\boldsymbol{d}$ to be the decision field (so that $\boldsymbol{u}_d = \boldsymbol{d}$). Based on the above formulation we can proceed in defining the Lagrangian function for the optimization problem. The objective function is given by

$$\mathcal{J}(\boldsymbol{u},\boldsymbol{d}) := \frac{\nu}{2}a(\boldsymbol{u},\boldsymbol{u}) + \frac{\rho}{2}(\boldsymbol{d},\boldsymbol{d})_{\Gamma_d}, \tag{3.3}$$

and (the weak form of) the constraints are given by (3.2). We define the Lagrangian function as follows:

$$\begin{aligned}\mathcal{L}(\boldsymbol{u},p,\boldsymbol{d},\boldsymbol{\sigma},\boldsymbol{\lambda},\mu,\boldsymbol{\tau}) &:= \mathcal{J}(\boldsymbol{u},\boldsymbol{d})\\ &\quad + \nu a(\boldsymbol{u},\boldsymbol{\lambda}) + c(\boldsymbol{u},\boldsymbol{u},\boldsymbol{\lambda}) - (\boldsymbol{\sigma},\boldsymbol{\lambda})_{\Gamma_d} - (\boldsymbol{f},\boldsymbol{\lambda})_\Omega + b(p,\boldsymbol{\lambda})\\ &\quad + b(\mu,\boldsymbol{u}) - (\boldsymbol{\tau},\boldsymbol{u}-\boldsymbol{d})_{\Gamma_d},\\ &\qquad \forall\ \boldsymbol{u}\in\boldsymbol{H}^1(\Omega),\ p\in L^2(\Omega),\ \boldsymbol{\sigma}\in\boldsymbol{H}^{-1/2}(\Gamma_d),\ \boldsymbol{d}\in\boldsymbol{H}^{1/2}(\Gamma_d),\\ &\qquad \forall\ \boldsymbol{\lambda}\in\boldsymbol{V},\ \mu\in L^2(\Omega),\ \boldsymbol{\tau}\in\boldsymbol{H}^{-1/2}(\Gamma_d).\end{aligned} \tag{3.4}$$

Here $\boldsymbol{\lambda},\mu,\boldsymbol{\tau}$ are the Lagrange multipliers for the state variables $\boldsymbol{u},p,\boldsymbol{\sigma}$. By taking variations with respect to the Lagrange multipliers we obtain (3.2) augmented with $\boldsymbol{u}_d = \boldsymbol{d}$ on Γ_d. Taking variations with respect to the states $\boldsymbol{u},p,\boldsymbol{\sigma}$ we obtain the weak form of the adjoint equations:

$$\begin{aligned}\nu a(\boldsymbol{v},\boldsymbol{\lambda}) + c(\boldsymbol{v},\boldsymbol{u},\boldsymbol{\lambda}) + c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}) + b(\mu,\boldsymbol{v}) + (\boldsymbol{\tau},\boldsymbol{v})_{\Gamma_d} &= -\nu a(\boldsymbol{u},\boldsymbol{v}) \;\forall\, \boldsymbol{v}\in\boldsymbol{V},\\ b(q,\boldsymbol{\lambda}) &= 0 \qquad\qquad \forall\, q\in L^2(\Omega)\\ (\boldsymbol{t},\boldsymbol{\lambda})_{\Gamma_d} &= 0 \qquad\qquad \forall\, \boldsymbol{t}\in\boldsymbol{H}^{-1/2}(\Gamma_d).\end{aligned} \tag{3.5}$$

Finally, by taking variations with respect to $\boldsymbol{d}$ we obtain the decision equation

$$\rho(\boldsymbol{d},\boldsymbol{r})_{\Gamma_d} + (\boldsymbol{\tau},\boldsymbol{r})_{\Gamma_d} = 0 \qquad \forall\, \boldsymbol{r}\in\boldsymbol{H}^{1/2}(\Gamma_d). \tag{3.6}$$

Equations (3.2), (3.5), (3.6) are the weak form of the first order optimality conditions. In [13], [14] there is extensive discussion on the existence of a solution and the existence of the Lagrange multipliers. In [13] the existence of a local minimum for the optimization problem and the existence of Lagrange multipliers that satisfy the first order optimality conditions is asserted[5]. Furthermore, uniqueness is shown upon sufficiently small data. Note that in the absence of a Neumann condition ($\Gamma_N = \emptyset$) the the controls have to satisfy the incompressibility condition $(\boldsymbol{d}\cdot\boldsymbol{n})_{\Gamma_d} = 0$.

The strong form of the adjoint and decision equations can be obtained by using the following integration by parts formulas:

$$\begin{aligned}a(\boldsymbol{u},\boldsymbol{v}) &= -(\boldsymbol{v},\Delta\boldsymbol{u})_\Omega + ((\nabla\boldsymbol{u})\boldsymbol{n},\boldsymbol{v})_\Gamma,\\ c(\boldsymbol{u},\boldsymbol{v},\boldsymbol{\lambda}) &= -c(\boldsymbol{u},\boldsymbol{\lambda},\boldsymbol{v}) - ((\nabla\cdot\boldsymbol{u})\boldsymbol{\lambda},\boldsymbol{v})_\Omega + ((\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{\lambda},\boldsymbol{v})_\Gamma,\\ b(\mu,\boldsymbol{v}) &= (\nabla\mu,\boldsymbol{v})_\Omega - (\mu\boldsymbol{n},\boldsymbol{v})_\Gamma.\end{aligned}$$

Upon sufficient smoothness we arrive at the strong form of the optimality conditions. Equation (3.1) is the strong form of the constraints. The strong form of the adjoint equations is given by

$$\begin{aligned}-\nu\nabla\cdot(\nabla\boldsymbol{\lambda}+\nabla\boldsymbol{\lambda}^T) + (\nabla\boldsymbol{u})^T\boldsymbol{\lambda} - (\nabla\boldsymbol{\lambda})\boldsymbol{u} + \nabla\mu &= \nu\nabla\cdot(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T) \quad \text{in} \quad \Omega,\\ \nabla\cdot\boldsymbol{\lambda} &= 0 \quad \text{in} \quad \Omega,\\ \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{on} \quad \Gamma_u,\\ \boldsymbol{\lambda} &= \boldsymbol{0} \quad \text{on} \quad \Gamma_d,\\ -\mu\boldsymbol{n} + \nu(\nabla\boldsymbol{\lambda}+\nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + (\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{\lambda} &= -\nu(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T)\boldsymbol{n} \quad \text{on} \quad \Gamma_N,\end{aligned} \tag{3.7}$$

[5]The objective functional used in [13] is different than ours. An $\boldsymbol{L}^4$ functional is used for the matching problems and a $\boldsymbol{H}^1_{\Gamma_d}$ is used for the penalization of $\boldsymbol{u}_d$—resulting on a surface Laplacian equation for the decision variables.

and (equation for $\boldsymbol{\tau}$)

$$\nu(\nabla\boldsymbol{\lambda} + \nabla\boldsymbol{\lambda}^T)\boldsymbol{n} + \nu(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T)\boldsymbol{n} - \boldsymbol{\tau} = \mathbf{0} \quad \text{on} \quad \Gamma_d. \tag{3.8}$$

We may also determine that the strong form of the decision equation is given by

$$\boldsymbol{\tau} = \rho\boldsymbol{d} \quad \text{on} \quad \Gamma_d. \tag{3.9}$$

In [13] estimates are given on the convergence rates of the finite element approximations to the exact solutions for the optimal control of steady viscous flow. For the case of boundary velocity control, the basic result is that, if the exact solutions are smooth enough, then, provided the Taylor-Hood element is used (for both adjoints and states), the solution error satisfies the following estimates:

$$\begin{aligned} \|\boldsymbol{u} - \boldsymbol{u}_h\|_0 &\leq \mathcal{O}(h^3), \\ \|p - p_h\|_0 &\leq \mathcal{O}(h^2), \\ \|\boldsymbol{\lambda} - \boldsymbol{\lambda}_h\|_0 &\leq \mathcal{O}(h^3), \\ \|\mu - \mu_h\|_0 &\leq \mathcal{O}(h^2). \end{aligned} \tag{3.10}$$

Here h is the maximum element size, and $\|\cdot\|_0$ is the $L^2(\Omega)$ norm.

3.1. Discrete and discretized optimality conditions. In our implementation we have not discretized the continuous forms of the optimality conditions. Instead we have discretized the objective function and the Navier-Stokes equations and then we used this discretization to form the optimality conditions. In general, discretization and differentiation (to obtain optimality conditions) do not commute. That is, if $\boldsymbol{A}$ is the infinite dimensional (linearized) forward operator and $\boldsymbol{A}^*$ is its adjoint then in general

$$(\boldsymbol{A}^*)_h \neq (\boldsymbol{A}_h)^T,$$

where the subscript h indicates discretization. We will show that for Galerkin approximation of the steady incompressible Navier-Stokes optimal control problem, discretization and differentiation do commute.

For the discretized equations we use the following notation:

$$\begin{aligned} a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{U}(\mathbf{u})\mathbf{u}, \\ a(\boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{p}_h, \boldsymbol{u}_h, \boldsymbol{v}_h) + c(\boldsymbol{u}_h, \boldsymbol{p}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{V}(\mathbf{u})\mathbf{p}, \\ a(\boldsymbol{u}_h, \boldsymbol{v}_h) &\dashrightarrow \mathbf{Q}\mathbf{u}, \\ b(q_h, \boldsymbol{u}_h) &\dashrightarrow \mathbf{P}\mathbf{u}, \\ (\boldsymbol{t}_h, \boldsymbol{u}_h)_{\Gamma_d} &\dashrightarrow \mathbf{T}\mathbf{u}, \\ (d_h, r_h)_{\Gamma_d} &\dashrightarrow \mathbf{M}\mathbf{d}. \end{aligned}$$

then, the discrete form of the Navier-Stokes equations is given by

$$\begin{aligned} \mathbf{U}(\mathbf{u})\mathbf{u} + \mathbf{P}^T\mathbf{p} + \mathbf{T}^T\boldsymbol{\sigma} &= \mathbf{f}_1, \\ \mathbf{P}\mathbf{u} &= \mathbf{f}_2, \\ \mathbf{T}\mathbf{u} &= \mathbf{T}\mathbf{d}. \end{aligned} \tag{3.11}$$

The discrete Lagrangian function is given by

$$\begin{aligned} \frac{1}{2}\mathbf{u}^T\mathbf{Q}\mathbf{u} + \frac{\rho}{2}\mathbf{d}^T\mathbf{M}\mathbf{d} + \boldsymbol{\lambda}^T\left\{\mathbf{U}(\mathbf{u})\mathbf{u} + \mathbf{P}^T\mathbf{p} + \mathbf{T}^T\boldsymbol{\sigma} - \mathbf{f}_1\right\} \\ +\boldsymbol{\mu}^T\left\{\mathbf{P}\mathbf{u} - \mathbf{f}_2\right\} + \boldsymbol{\tau}^T\left\{\mathbf{T}\mathbf{u} + \mathbf{T}\mathbf{d}\right\} = \mathbf{0}. \end{aligned} \tag{3.12}$$

By taking derivatives with respect to the discrete Lagrange multiplier vectors $\boldsymbol{\lambda}, \boldsymbol{\mu}, \boldsymbol{\tau}$, we recover the state equations (3.11). By taking derivatives with respect to the discrete state variables $\mathbf{u}, \mathbf{p}, \boldsymbol{\sigma}$, we obtain the discrete adjoint equations:

$$\begin{aligned} \mathbf{V}^T(\mathbf{u})\boldsymbol{\lambda} + \mathbf{P}^T\boldsymbol{\mu} + \mathbf{T}^T\tau &= -\mathbf{Q}\mathbf{u}, \\ \mathbf{P}\boldsymbol{\lambda} &= \mathbf{0}, \\ \mathbf{T}\boldsymbol{\lambda} &= \mathbf{0}. \end{aligned} \tag{3.13}$$

These equations correspond to the discretization of equations (3.5) provided that $\mathbf{V}^T\boldsymbol{\lambda}$ is the discretization of $a(\boldsymbol{\lambda}, \boldsymbol{u}) + c(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{\lambda}) + c(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\lambda})$. The bilinear form $a(\cdot, \cdot)$ is symmetric so we omit it from our discussion. If ϕ denotes the basis function for $\boldsymbol{u}$, ψ_v for $\boldsymbol{v}$, and ψ_λ for $\boldsymbol{\lambda}$ then the (3×3)-block elements for the linearized state and the adjoint are:

$$\int_\Omega \mathbf{I}\,(\nabla\phi \cdot \boldsymbol{u})\psi_v + (\nabla\boldsymbol{u})\phi\psi_v \; d\Omega \qquad \text{state element matrix,}$$

$$\int_\Omega \mathbf{I}\,(\nabla\psi_v \cdot \boldsymbol{u})\psi_\lambda + (\nabla\boldsymbol{u})^T\psi_\lambda\psi_v \; d\Omega \qquad \text{adjoint element matrix.}$$

Therefore, the transpose of the discretized (linearized) state equations coincides with the discretization of the adjoint equations, i.e.

$$(\boldsymbol{A}^*)_h = (\boldsymbol{A}_h)^T.$$

One needs to be careful to use the weak form given by equation (3.5). If (3.7) were used without employing the reverse integration by parts on the term $c(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{\lambda})$, this would result in a discretization which is incompatible with the discrete optimization problem (which is what the optimizer sees). It would result in an unsymmetric KKT-matrix and possibly would prevent the optimizer from converging to a KKT point. A Petrov-Galerkin formulation would also be incompatible.

In our formulation we do not solve explicitly for $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. We approximate both tractions and velocity traces in $\boldsymbol{H}^1(\Gamma_d)$; in this case the stresses can be eliminated. For a discussion on choice of $H^1(\Gamma_d)$ for the stresses see [10]. The resulting equations are equivalent to the formulations described in this section. We use a standard Galerkin approximation scheme (no upwinding) with Taylor-Hood elements to approximate the velocities, the pressures, and their adjoints.

We conclude this section with a note on continuation. To solve a Navier-Stokes control problem with large Reynolds number, some kind of continuation scheme is usually needed. We first solve a Stokes-flow optimal control problem ($Re_0 = 0$) and then we progressively increase the Reynolds number by $Re_{(+)} = Re + \Delta Re$. One could set $\mathbf{u}_{(+)} = \mathbf{u} + (\partial_{Re})\mathbf{u}\Delta Re$, where $\partial_{Re}\mathbf{u}$ can be easily computed through a linearized forward solve. Since we consider only steady flows, we follow [8] and use fixed ΔRe, and simply set $\mathbf{u}_{(+)} = \mathbf{u}$, i.e. the initial guess at the new Reynolds number is the solution from the previous optimization step. Additionally, quasi-Newton information is carried forward to the next Reynolds number.

4. Numerical Results. In this section we use four numerical tests to investigate the accuracy and scalability of LNKS method. First we test the finite element approximation convergence rates with a problem that has an analytic solution. Then we revisit the Poiseuille flow problem—which is also a solution for the Navier-Stokes—and use it to study the effectiveness of the limited-BFGS method as a preconditioner for the reduced Hessian. In both cases we solve for the boundary conditions that reproduce the exact solution by minimizing a matching velocity functional.

We continue with the more challenging problem a the control of flow around a cylinder. The objective function to be minimized is the dissipation functional. We use this problem to test the LNKS line search algorithm, the effectiveness of the Krylov-Schur preconditioner for highly nonlinear problems, and the various heuristics we introduced in Section 2. The last problem is the optimal control of a flow around a wing.

4.1. Finite element approximation error. In this section we use a model problem to verify the convergence rate estimates given in the previous section. The velocity and pressure given by

$$\boldsymbol{u}^*(x,y,z) = \left\{1-(x^2+y^2)^2, x, -y\right\}^T, \; p^*(x,y,z) = x^2+y^2-z^2.$$

satisfy the Navier-Stokes equations. We restrict this solution in a cylindrical domain we choose some part of its boundary as the control domain Γ_d. We defined the velocity boundary conditions on the circumferential walls to be the decision variables. On Γ/Γ_d we set $\boldsymbol{u} = \boldsymbol{u}^*$. The objective function is given by

$$\mathcal{J}(\boldsymbol{u}, \boldsymbol{u}_d, p) = \frac{1}{2}\int_\Omega (\boldsymbol{u}^* - \boldsymbol{u})^2 \, d\Omega.$$

Since the boundary conditions for $\boldsymbol{u}, p$ on Γ/Γ_d are compatible with $(\boldsymbol{u}^*, p^*)$ the values for the objective function and the Lagrange multipliers at the minimum are zero.

In Table 4.1 we give convergence rates for the state variables and the Lagrange multipliers. The results are in good agreement with the theoretical predictions. The convergence rate

TABLE 4.1

In this table the convergence rate of the finite element approximation for a matching velocity problem is given. Here **n** *is the number of elements; h is the cube root of the volume of the maximum inscribed sphere inside a tetrahedron of the finite element mesh. Near optimal convergent rates can be observed for the state and adjoint variables.*

n	h	$\|\boldsymbol{u}^* - \boldsymbol{u}_h\|_0$	$\|p_* - p_h\|_0$	$\|\boldsymbol{\lambda}_* - \boldsymbol{\lambda}_h\|_0$	$\|\mu_* - \mu_h\|_0$
124,639	0.80	1.34×10^{-4}	2.01×10^{-5}	3.88×10^{-4}	1.76×10^{-5}
298,305	0.53	4.40×10^{-5}	9.00×10^{-6}	1.19×10^{-4}	7.90×10^{-6}
586,133	0.40	1.70×10^{-5}	5.20×10^{-6}	5.00×10^{-5}	4.50×10^{-6}

for the velocities and their adjoints is approximately 2.92 (comparing errors between the first and second rows) and 2.96 (comparing errors between the second and third rows). For the pressures and their adjoints the convergence rate is 1.96 and 1.97, respectively.

4.2. Poiseuille flow. The Poiseuille flow is a stable solution of the Navier-Stokes equations for small Reynolds numbers. We use this example to study the effectiveness of BFGS as a preconditioner. Since the optimization problem is nonlinear, LNKS takes several iterations and quasi-Newton curvature information can be built up. Quasi-Newton theory predicts that $\mathbf{B}_z$ approaches $\mathbf{W}_z$ as the iterates get closer to the solution. Therefore, we expect the effectiveness of the preconditioner to improve as the optimization algorithm progresses. To approximate the reduced Hessian we invoke the limited-memory BFGS formula we described the first paper. In our test we used 30 vectors.

We look at a fixed-size/fixed-granularity problem. The target Reynolds number is 500. We start at Reynolds number 100 and we use a continuation step $\Delta Re = 200$. The continuation is *not* used to initialize the state and control variables, but only to carry BFGS information to the next Reynolds number.

TABLE 4.2

Work efficiency of the proposed preconditioners for a Poiseuille flow matching problem for fixed size and fixed granularity as a function of the Reynolds number. Recall that LNKS-I requires two linearized forward solves per iteration, whereas LINKS-II involves just application of the Schwarz approximation. **Re** *is the Reynolds number;* **N/QN** *denotes the number of outer iterations. The number of iterations for the KKT system is averaged across the optimization iterations. The problem has 21,000 state equations and 3,900 control variables; results are for 4 processors of the T3E-900. Wall-clock time is in hours.*

Re	**method**	**N/QN iter**	**KKT iter**	$\Vert \mathbf{g}_z \Vert$	**time**
100	QN-RSQP	262	—	1×10^{-4}	5.9
	LNK	3	186,000	9×10^{-6}	7.1
	LNKS-I	3	48	9×10^{-6}	3.2
	LNKS-II	3	4,200	9×10^{-6}	1.3
300	QN-RSQP	278	—	1×10^{-4}	6.4
	LNK	3	198,000	9×10^{-6}	7.6
	LNKS-I	3	40	9×10^{-6}	3.1
	LNKS-II	3	4,300	9×10^{-6}	1.4
500	QN-RSQP	289	—	1×10^{-4}	7.3
	LNK	3	213,000	9×10^{-6}	9.0
	LNKS-I	3	38	9×10^{-6}	3.0
	LNKS-II	3	4,410	9×10^{-6}	1.4

The forward problem preconditioner is given by Equation 4.5 (Part I). In QN-RSQP we use QMR for the linearized Navier-Stokes operator, preconditioned with an overlapping additive Schwarz method with ILU(1) in each subdomain[6]. Results for a problem with 21,000 state and 3,900 design variables on 4 processors and for a sequence of three Reynolds numbers are presented in Table 4.2. The number of KKT iterations in LNKS I reveals the effect of the BFGS preconditioner. They drop from an average 48 iterations to 38. The effect of BFGS in LNKS II is hidden since the KKT iterations are dominated from the ill-conditioning of the forward and adjoint operators (In LNKS I these solves are exact in each iteration). Overall, we can observe that LNKS reduces significantly the execution time relative to QN-RSQP.

The Newton solver performed well requiring only 3 iterations to converge. In these problems we did not use inexact Newton's method with the KKT solves, they were fully converged at each iteration. No line search was used in the LNKS variants; we used the l_1-merit function for the QN-RSQP.

It is rather surprising that the quasi-Newton works well as a preconditioner for the KKT system where as it stagnates within the QN-RSQP method. One explanation could be that the the QN-RSQP in these runs suffered the "Maratos" effect. In our subsequent tests we switched to a second-order correction method ([19], p.570).

4.3. Flow around a cylinder. All the problems examined so far were useful in verifying certain aspects of the LNKS method but they are linear or mildly nonlinear. In order to test LNKS further we study a highly nonlinear problem: that of flow around a cylinder with a dissipation-type objective function. The cylinder is anchored inside a rectangular duct, much like a wind tunnel. A quadratic velocity profile is used as an inflow Dirichlet condition and we prescribe a traction-free outflow. The decision variables are defined to be the velocities on the downstream portion of the cylinder surface. We have investigated flows in the laminar steady-state regime. For exterior problems the transition Reynolds number is 40 but for the duct problem we expect higher Reynolds numbers due to the dissipation from the duct walls.

[6] For definitions of ILU(0) and ILU(1) see [20].

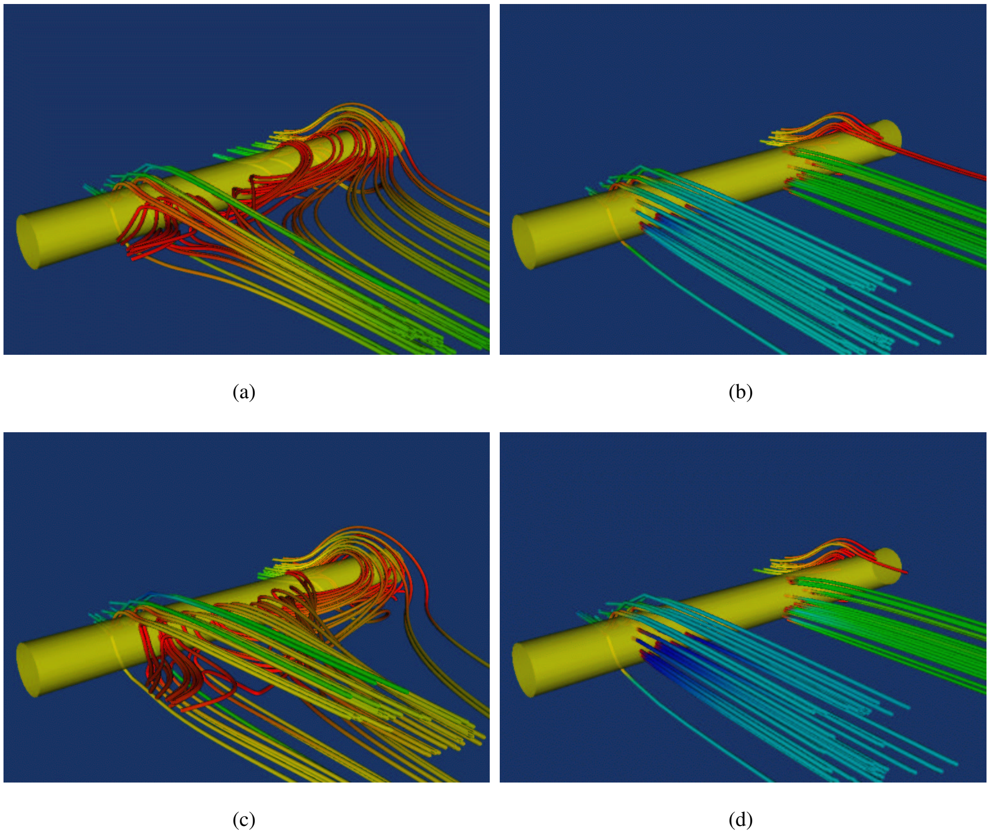


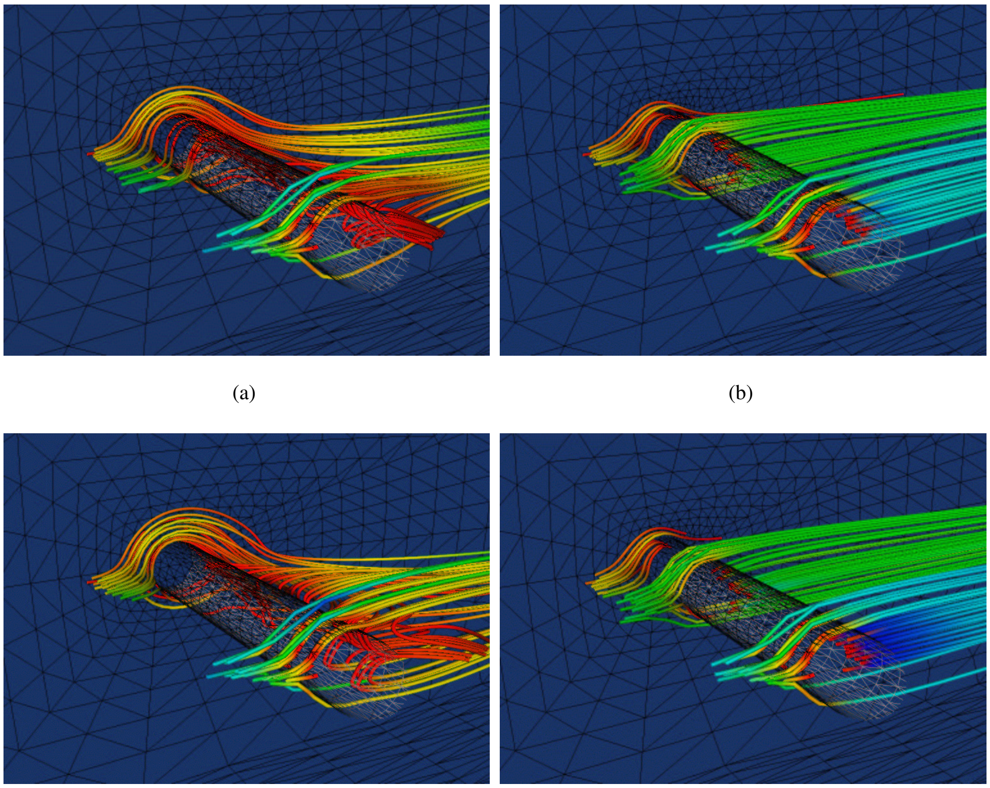
FIG. 4.1. *The top row depicts stream tubes of the flow for Reynolds number* 20 *and the bottom row for Reynolds number* 40*. The left column depicts the uncontrolled flow. The right column depicts the controlled flow. These pictures depict the flow pattern on the downstream side of the cylinder.*

Figures 4.1 and 4.2 illustrate the optimal results for different Reynolds numbers. LNKS eliminates the recirculation region in the downstream region of the cylinder. In order to avoid the excessive suction that we observed in the Stokes case, we imposed Dirichlet boundary conditions on the outflow of the domain. The incompressibility condition prevents the optimizer from driving the flow inside the cylinder[7].

Our experiments on the Stokes optimal control established the relation between the performance of the Krylov-Schur iteration and the forward problem preconditioner. Thus before we discuss results on the LNKS algorithm we give some representative results for the Navier-Stokes forward solver. We use an inexact Newton's method combined with the preconditioner we presented in Part I. A block-Jacobi ILU(0) preconditioner is used for the velocity block and as well as for the pressure mass matrix (scaled by $1/Re$); the latter is used to precondition the pressure Schur complement block. We would very much like to use an ILU(1), as we did for the Poiseuille flow case, but memory limitations[8] prevented us from doing so.

[7]When Dirichlet conditions are specified everywhere on Γ, then $\int_\Gamma \boldsymbol{u} \cdot \boldsymbol{n} \, d\Gamma$ should be zero. The constraint needs to either be imposed explicitly, or with implicitly by using a proper function space. In our implementation we use a penalty approach by modifying the objective function.

[8]In our Navier-Stokes implementation, we store the state operator, the Hessian of the constraints and the Hessian of the objective. PSC's T3E-900 (where the majority of our runs took place), has only 128MB of memory per



(a) (b)

(c) (d)

FIG. 4.2. *The top row depicts stream tubes of the flow for Reynolds number* 20 *and the bottom row for Reynolds number* 40*. The left column depicts the uncontrolled flow. The right column depicts the controlled flow. The decision variables are Dirichlet boundary conditions for the velocities on the downstream half of the cylinder surface. Here we see the flow from the upstream side of the cylinder. In (c) we can clearly identify the two standing vortices formed on the lower left corner of the image.*

TABLE 4.3

Forward solver efficiency in relation to problem size and the Reynolds number for a 3D flow around a cylinder. **PEs** *is processor number;* **n** *is the problem size; (Re) is the Reynolds number;* **qmr** *is the number of aggregate Krylov iterations required to satisfy* $\|\boldsymbol{r}\|/\|\boldsymbol{r}_0\| \leq 1 \times 10^{-7}$*;* **nw** *is the number of Newton steps to satisfy* $\|\mathbf{c}\|/\|\boldsymbol{c}_0\| \leq 1 \times 10^{-6}$*; and* **t** *is time in seconds. The runs were performed on a T3E-900.*

		$Re = 20$			$Re = 30$			$Re = 60$		
PEs	**n**	**qmr**	**nw**	**t**	**qmr**	**nw**	**t**	**qmr**	**nw**	**t**
32	117.048	2,905	5	612	3,467	7	732	2,850	6	621
64	389,440	4,845	5	1,938	5,423	7	2,101	5,501	7	2,310
128	615,981	6,284	5	2,612	8,036	8	3,214	7,847	7	3,136

Table 4.3 gives statistics for three different Reynolds numbers and for three different problem sizes. We report the (aggregate) number or Krylov iterations required to converge the Newton solver, the number of Newton iterations, and the total execution time. For these

processor.

runs we did not use continuation, but we did use an inexact Newton method. Comparing with Table 4.1 (Part I) we observe that the time for a forward solve has increased almost sixfold. However the time *per* iteration is roughly the same with that of the linear, Stokes case. For example, in the 128 processor problem and for Reynolds number 30, the average (Krylov) iteration count is 1005, whereas in the linear case it is 882. Similarly, the average time per Newton step is 401 seconds. The time for the Stokes solver is little higher, 421 seconds[9]. We can conclude that the forward preconditioner performs reasonably well.

Table 4.4 shows results for 32, 64, and 128 processors of a T3E-900 for increasing problem sizes. Results for two different preconditioning variants of LNKS are presented: the exact (LNKS-I) and inexact (LNKS-II) version of the Schur preconditioner. The globalized

TABLE 4.4

The table shows results for 32, 64, and 128 processors of a Cray T3E for a roughly doubling of problem size. Results for the QN-RSQP and LNKS algorithms are presented. **QN-RSQP** *is quasi-Newton reduced-space SQP;* **LNKS-I** *requires two exact solves per Krylov step combined with 2-step-stationary-BFGS preconditioner for the reduced Hessian; in* **LNKS-II** *the exact solves have been replaced by approximate solves;* **LNKS-II-TR** *uses a truncated Newton method and avoids fully converging the KKT system for iterates that are far from a solution.* **time** *is wall-clock time in hours on a T3E-900. Continuation was used only for $Re = 60$.*

$Re = 30$

states controls	**method**	**N or QN iter**	**KKT iter**	**time**
117,048	QN-RSQP	161	—	32.1
2,925	LNKS-I	5	18	22,8
(32 procs)	LNKS-II	6	1,367	5,7
	LNKS-II-TR	11	163	1.4
389,440	QN-RSQP	189	—	46.3
6,549	LNKS-I	6	19	27.4
(64 procs)	LNKS-II	6	2,153	15.7
	LNKS-II-TR	13	238	3.8
615,981	QN-RSQP	204	—	53.1
8,901	LNKS-I	7	20	33.8
(128 procs)	LNKS-II	6	3,583	16.8
	LNKS-II-TR	12	379	4.1

$Re = 60$

states controls	**preconditioning**	**Newton iter**	**average KKT iter**	**time (hours)**
117,048	QN-RSQP	168	—	33.4
2,925	LNKS-I	6	20	31,7
(32 procs)	LNKS-II	7	1,391	6,8
	LNKS-II-TR	11	169	1.5
389,440	QN-RSQP	194	—	49.1
6,549	LNKS-I	8	21	44.2
(64 procs)	LNKS-II	7	2,228	18.9
	LNKS-II-TR	15	256	4.8
615,981	QN-RSQP	211	—	57.3
8,901	LNKS-I	8	22	45.8
(128 procs)	LNKS-II	8	3,610	13.5
	LNKS-II-TR	16	383	5.1

LNKS algorithm is compared with QN-RSQP. In LNKS-II-TR we activate the inexact Newton method. In this example we have used continuation to warm start the $Re = 60$ problem.

[9]The reason for this is related to the scaling between the velocity and pressure block of the forward problem. Increasing the Reynolds number improves this scaling and thus improves the eigenvalue distribution. Of course this is true up to certain Reynolds number. For higher values the Jacobian becomes highly unsymmetric and ill-conditioned.

The reduced Hessian preconditioner is a combination of the BFGS and 2-step preconditioners (as we described in Part I, Section 3.2). In the line search we use the augmented Lagrangian merit function.

In this problem, QN-RSQP did converge, but only after 48 hours. LNKS-I, although faster, does not reduce the required time significantly. LNKS-II does better—4 to 5 times faster than QN-RSQP.

The most noticeable finding in Table 4.4 is the dramatic acceleration of the LNKS algorithm which is achieved by using LNKS-II-TR—the inexact version of the Newton method. The inexactness did not interfere at any point with the merit function and in all cases we observed quadratic convergence. Overall LNKS-II-TR runs more than 10 times faster than QN-RSQP. This is in agreement with the performance improvements we observed with the Stokes equations.

Undoubtedly the external cylinder flow problem is highly nonlinear. The augmented Lagrangian globalization performed robustly and we did not have problems converging the equations. Not once did the QN-RSQP safeguard get activated—triggered from a negative curvature direction. Finally, it is worth noting that the optimization solution is found at a cost of *5 to 6 flow simulations*—remarkable considering that there are thousands of control variables.

4.4. Flow around a Boeing 707 wing. For our last test we solved for control of a flow around a Boeing-707 wing. In this problem the control variables are the velocities (Dirichlet conditions) on the downstream half of the wing. The Reynolds number (based on the length of the wing root) was varied from 100 to 500 and the angle of attack was fixed at 12.5 degrees. The problem size in this example is 710,023 state variables and 4,984 control variables.

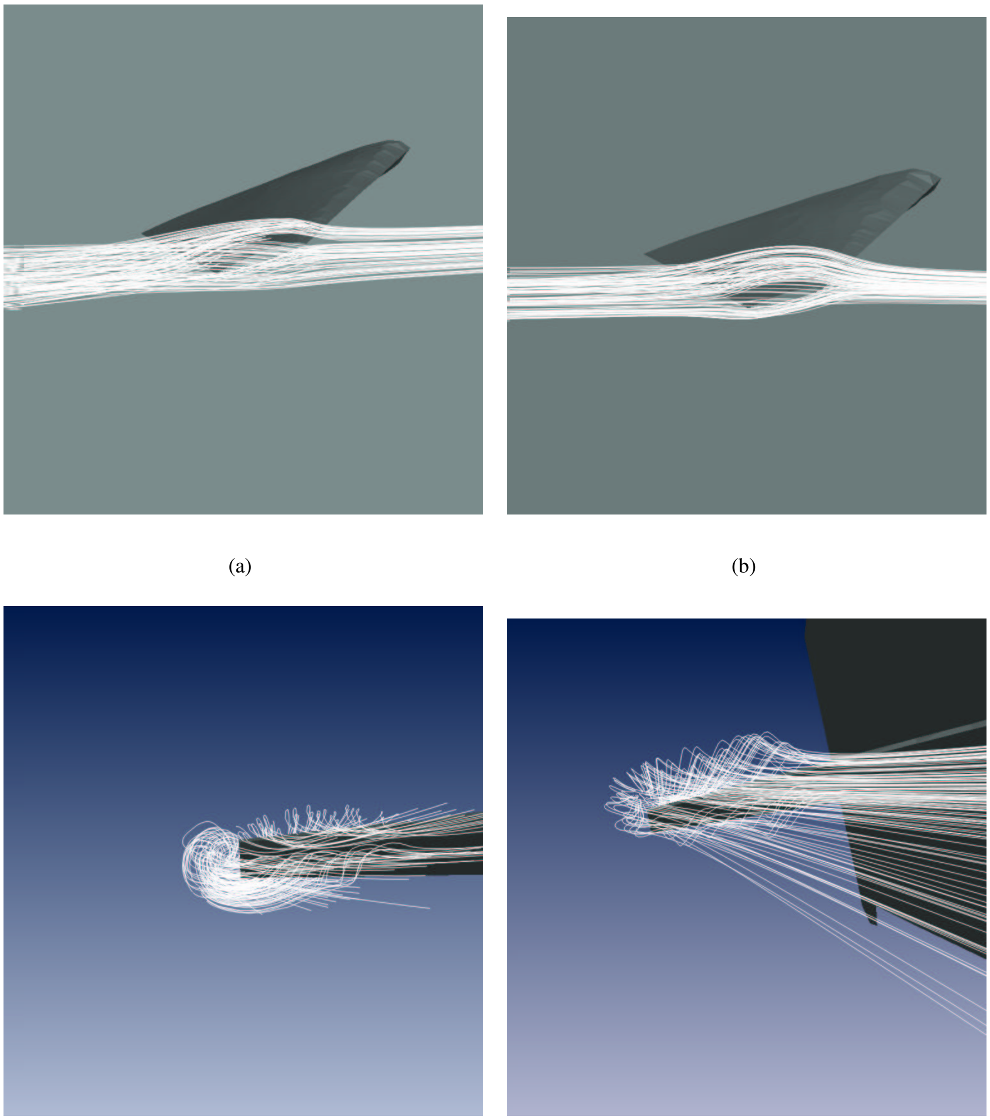
Table 4.5 summarizes the results from this set of experiments. The main purpose of this analysis is to compare continuation with the other globalization techniques. In addition we employ the double inexactness idea, that is, we solve inexactly in both the continuation loop and the Lagrange-Newton loop. It is apparent that in this problem continuation is crucial.

TABLE 4.5

In this table we present results for the wing flow test case. The size of this problem is 710,023 state and 4,984 decision variables. The runs were performed on 128 processors on a T3E-900. Here Re is the Reynolds number; **iter** *is the aggregate number of Lagrange-Newton iterations—the number in parenthesis is the number of iteration in the last step;* **time** *is the overall time in hours;* **qn** *is the number of QN-RSQP steps—the number in parenthesis gives how many times a negative curvature was detected;* **minc** *is the number of non-monotone line search iterations—in parenthesis is the number of times this heuristic failed. The globalized LNKS-II-TR algorithm is used. The Lagrange-Newton solver was stopped after 50 iterations. In the last column $Re \times \Delta f$ gives the reduction of the objective function (with the respect the uncontrolled flow). "no cont" means that continuation was not activated.*

Re	**iter**	**time**	**qn**	**minc**	$\|\mathbf{g} + \mathbf{A}^T\boldsymbol{\lambda}\|$	$\|\mathbf{c}\|$	$Re \times \Delta f$
100 no cont	19	4.06	2	4	9×10^{-6}	9×10^{-6}	4.065
cont							
200 no cont	39	7.8	6(1)	2	9×10^{-6}	9×10^{-6}	5.804
cont	20(10)	4.6	0	3	9×10^{-6}	9×10^{-6}	5.805
300 no cont	48	11.8	16(3)	0	9×10^{-6}	9×10^{-6}	6.012
cont	29(11)	6.4	0	2	9×10^{-6}	9×10^{-6}	6.016
400 no cont	50	13.6	40(3)	0	2×10^{-4}	3×10^{-3}	3.023
cont	33(11)	7.36	0	6(1)	9×10^{-6}	9×10^{-6}	8.345
500 no cont	50	16.7	42(5)	0	4×10^{-2}	9×10^{-2}	1.235
cont	39(14)	9.09	1	5(1)	9×10^{-6}	9×10^{-6}	10.234

For Reynolds numbers larger than 300, LNKS was forced to early termination (we set the Lagrange-Newton iteration bound to 50). In the last row ($Re = 500$) and when we did not



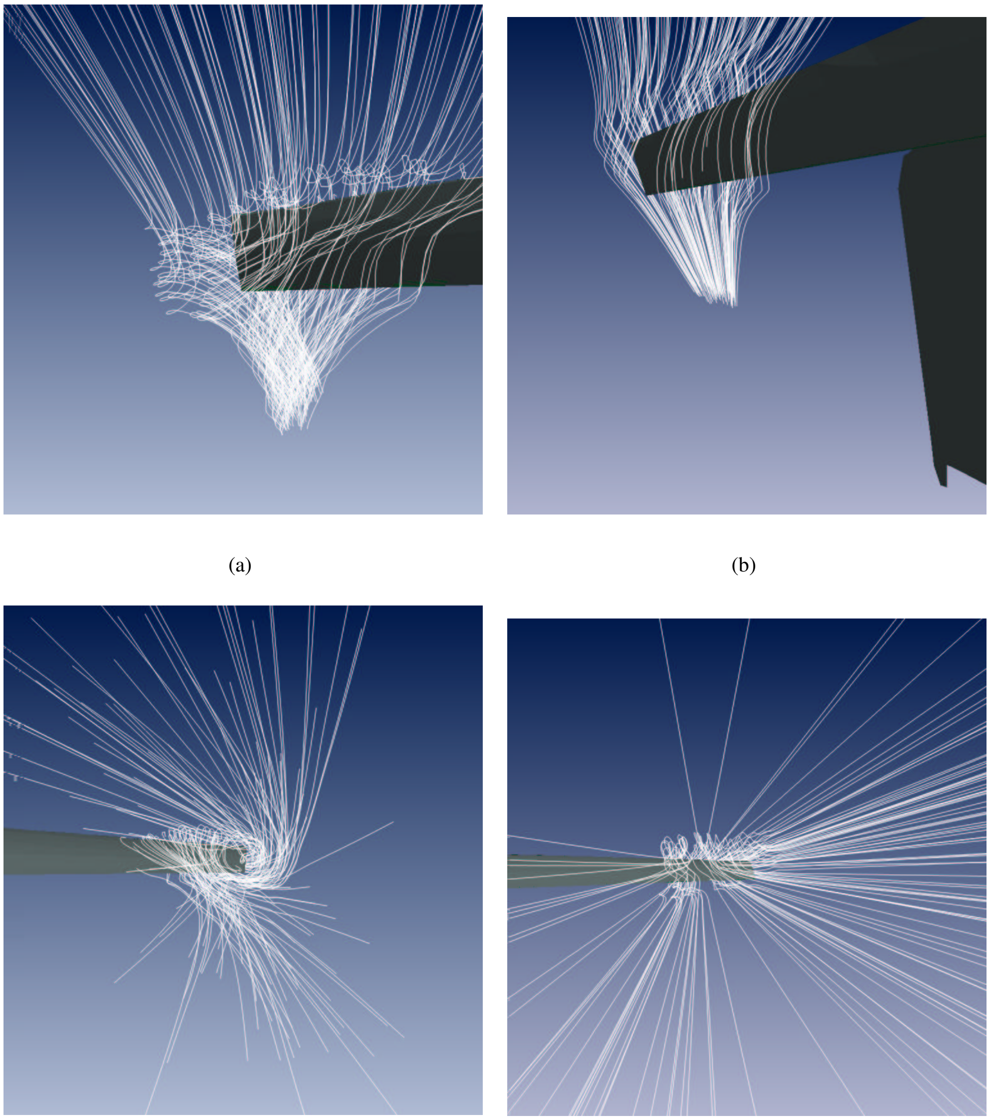
(a) (b) (c) (d)

FIG. 4.3. *The left column depicts streamlines of the uncontrolled flow. The right column depicts streamlines of the controlled flow. Top row gives a side snapshot of the flow; bottom row gives a front view. The Reynolds number (based on the length the root of the wing) is 500.*

use continuation, LNKS ends up switching to a QN-RSQP step 42 times out of a total of 50 iterations; 5 times a negative curvature direction was detected.

As a result LNKS was terminated without reaching the convergence criteria. Furthermore, the small reduction in the objective function and the residuals (last three columns) indicate small progress at each optimization step. Notice that in these examples we did not activate backtracking in the continuation parameter.

(It could be argued that a reason the algorithm stagnated was the early termination of



(a) (b)

(c) (d)

FIG. 4.4. *The left column depicts streamlines of the uncontrolled flow. The right column depicts streamlines of the controlled flow. Top row gives a snapshot of the flow from below; bottom row gives a back view. Reynolds number is 500. We can clearly identify the wing tip vortices on the left, with non-slip boundary conditions on the wing. These vortices are directly associated to the lift. The images on the right column depict the flow with the wing boundary conditions modified by the optimizer; the vorticity is eliminated. So is the lift.*

the Krylov-Schur solver—because of the inexactness. We did not conduct exhaustive experiments to confirm or reject this claim. However, our experience on numerous problems suggests that it is the ill-conditioning and nonlinearity of these problems that leads to stagnation and not the inexactness. In our tests (systematic or during debugging and development) it was never the case that a run with exact solves converged in reasonable time, and the inexact

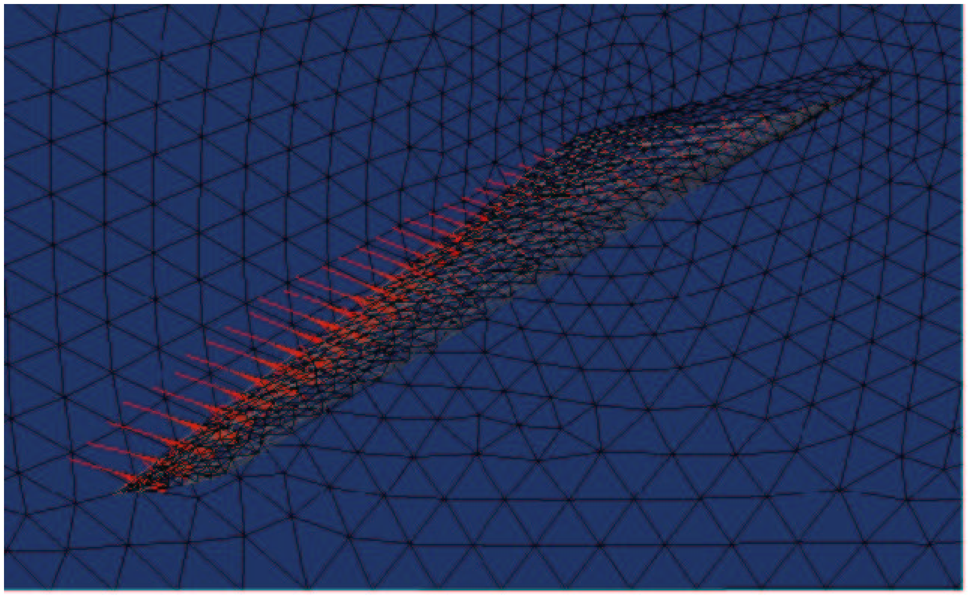


FIG. 4.5. *Snapshot of the (Dirichlet control) velocity field on the wing.*

version did not. On the contrary, inexactness significantly reduced execution times.)

On the other hand, when we used continuation (fairly large steps on the Reynolds number), the algorithm successfully converged after 39 Lagrange-Newton iterations. Switching to QN-RSQP was required just once. In the **minc** column of Table 4.5 we monitor the non-monotone line search criterion. Recall that if the merit function line search on the LNKS step fails, we perform a line search with a different merit—the KKT residual (i.e. the first order optimality conditions). If the step gets accepted, via backtracking, we use it as an update direction. Eventually, we insist that the (augmented Lagrangian) merit gets reduced. This strategy was very successful 20 times and it failed only twice[10].

Finally we conclude with some comments on the physics of this problem. Figures 4.3 and 4.4 depict snapshots of the uncontrolled and controlled flow for Reynolds number 500. The wing-tip vortices are eliminated by the optimizer. But at what cost? Figure 4.5 shows a snapshot of the (scaled) control variables—the velocity boundary conditions. It is obvious that the optimizer designed a perforated wing, which means a significant reduction in lift. This plane will never leave the ground! (Additional inequality constraints on the lift can be treated within the framework of LNKS with interior point methods.)

5. Conclusions. In the second part of the paper we presented the algorithmic components of the outer (Newton) solver and we studied the application of the LNKS method to a set of different optimal flow control problems. Our tests demonstrate the effectiveness and scalability of the LNKS method in PDE-constrained optimization. The Krylov-Schur preconditioner maintained its effectiveness; the Lagrange-Newton method exhibited the well known mesh-independence convergence properties. Inexact Newton steps dramatically accelerated the algorithm and continuation ensured global convergence.

The results reveal at least an order of magnitude improvement in time over popular quasi-Newton methods, rendering tractable some problems that were out of reach previously. In-

[10]In general, using the residual of the KKT conditions to test a step can compromise robustness since the optimizer could get trapped to a saddle point or a local maximum.

deed, the optimum is often found in a *small* multiple of the cost of a single simulation.

The LNKS algorithm is more suitable for steady PDE constraints. Although the method is in principle applicable to time-dependent problems it is not a recommended approach for 3D problems. The adjoint problem is a final value problem that requires the velocity history, and this requires large amount of memory. We are investigating various ways to circumvent this problem. Another important extension of LNKS is the treatment of inequality constraints via interior point methods.

We believe that the LNKS method is a very powerful tool. For this reason we decided to direct effort at developing a code that will be usable by the scientific community. In a forthcoming paper we will discuss *Veltisto*, a PETSc-based library for large-scale, PDE-constrained optimization on parallel computers.

Acknowledgments. We thank the PETSc development group at Argonne National Lab for making this work possible. We also thank Jonathan Shewchuk of UC Berkeley for providing the meshing and partitioning routines Pyramid and Slice, and David Marcum of Mississippi State for providing the meshing module AFLR. Finally, we thank David Keyes of ICASE/Old Dominion University, David Young of Boeing, and the other members of the TAOS project—Roscoe Bartlett, Larry Biegler, Greg Itle, Ivan Malčević, and Andreas Wächter—for their useful comments.

REFERENCES

[1] I. Babuška, *The finite element method with lagrangian multipliers*, Numerische Mathematik, 20 (1973), pp. 179–192.

[2] P. T. Boggs and J. W. Tolle, *A strategy for global convergence in a sequential quadratic programming algorithm*, SIAM Journal on Numerical Analysis, 26 (1989), pp. 600–623.

[3] J. Dennis E., Jr., M. El-Alem, and M. C. Magiel, *A global convergence theory for general trust-region-based algorithms for equality constrained optimization*, SIAM Journal on Optimization, 7 (1997), pp. 177–207.

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

[5] ———, *Choosing the forcing terms in an inexact Newton method*, SIAM Journal on Scientific Computing, 17 (1996), pp. 16–32.

[6] R. Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, second ed., 1987.

[7] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, Academic Press, 1981.

[8] M. D. Gunzburger, *Finite Element for Viscous Incompressible Flows*, Academic Press, 1989.

[9] M. D. Gunzburger, ed., *Flow Control*, vol. 68 of IMA Math. Appl., Springer-Verlag, New York, 1995.

[10] M. D. Gunzburger and S. L. Hou, *Treating inhomogeneous essential boundary conditions in finite element methods and the calculation of boundary stresses*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 390–424.

[11] M. D. Gunzburger and R. A. Nicolaides, eds., *Incompressible Computational Fluid Dynamics*, Cambridge University Press, 1993.

[12] M. Heinkenschloss and L. N. Vicente, *Analysis of inexact trust-region SQP algorithms*, Tech. Rep. TR99-18, Rice University, Department of Computational and Applied Mathematics, 1999.

[13] L. S. Hou, *Analysis and Finite Element Approximation of some Optimal Control Problems Associated with the Navier-Stokes Equations*, PhD thesis, Carnegie Mellon University, Department of Mathematical Sciences, Pittsburgh, August 1989.

[14] L. S. Hou and S. S. Ravindran, *Numerical approximation of optimal flow control problems by a penalty method: Error estimates and numerical results*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1753–1777.

[15] C. Kelley and E. W. Sachs, *Truncated Newton methods for optimization with inaccurate functions and gradients*, SIAM Journal on Optimization, 10 (1999), pp. 43–55.

[16] C. T. Kelley and D. E. Keyes, *Convergence analysis of pseudo-transient continuation*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 508–523.

[17] F. Leibritz and E. W. Sachs, *Inexact SQP interior point methods and large scale optimal control problems*, SIAM Journal on Control and Optimization, 38 (1999), pp. 272–293.

[18] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, 1996.

[19] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, Springer, 1999.

[20] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, 1996.

[21] K. SCHITTKOWSKI, *The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function*, Numerische Mathematik, 38 (1981), pp. 83–114.

[22] T. STEIHAUG, *The conjugate gradient method and trust regions in large scale optimization*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 626–637.

[23] H. YAMASHITA, *A globally convergent constrained quasi-Newton method with an augmented Lagrangian type penalty function*, Mathematical Programming, 23 (1982), pp. 75–86.