Navier-Stokes 유체의 최적제어를 위한 SQP 기법의 개발

Large-scale SQP Methods for Optimal Control of steady Incompressible Navier-Stokes Flows

박 재 형† 홍 순 조*
Bark, Jaihyeong Hong, soonjo
(논문접수일 : 2002년 5월 20일 ; 심사종료일 : 2002년 11월 15일)

요 지

본 연구의 목적은 Navier-Stokes 유체와 같은 대용량 문제를 위한 최적화 기법의 개발에 있다. 이를 위해 본 연구에서는 reduced Hessian sequential quadratic programming을 개발하였다. 첫째, 유체의 해석을 위한 평형 방정식을 최적화 과정에서 제거하여 변수를 줄였고, 또한 평형방정식과 최적화 과정에서 연속기법을 사용하여 최적해를 보장하면서 더욱 해에 쉽게 접근하도록 하였다. 그리고 각 단계에서는 테일러 시리즈를 이용한 근사치를 이용하여 각 단계에서 대단히 좋은 초기치 값을 제공하여 최적해에 더욱 빠르게 접근하게 하고 아울러 유체의 평형방정식을 풀 때에도 해에 더욱 빠르고 쉽게 접근하도록 하였다. 이 기법을 항력을 줄이기 위한 유체의 최적 제어를 위한 문제에 적용하였다. 유체의 흐름을 제어하기 위하여 물체의 경계면에서 유체의 흡입(suction)과 방출(injection)이라는 기법을 사용하여 경계면에서 속도를 제어하였고, 목적함수로써 항력을 표현하기 위하여 에너지 소실의 변화율을 사용하였다. 예제를 통해 본 연구에서 개발한 최적화 기법의 효용성을 입증하였다.

핵심용어 : 최적제어, Navier-Stokes 유체, GRG 기법, SQP 기법, 리듀스트 헤이시언 기법, 뉴턴법, 유사 뉴턴법, 흡입, 방출

Abstract

The forcus of this work is on the development of large-scale numerical optimization methods for optimal control of steady incompressible Navier-Stokes flows. The control is affected by the suction or injection of fluid on portions of the boundary, and the objective function of fluid on portions of the boundary, and the objective function represents the rate at which energy is dissipated in the fluid. We develop reduced Hessian sequential quadratic programming. Both quasi-Newton and Newton variants are developed and compared to the approach of eliminating the flow equations and variables, which is effectively the generalized reduced gradient method. Optimal control problems are solved for two-dimensional flow around a cylinder. The examples demonstrate at least an order-of-magnitude reduction in time taken, allowing the optimal solution of flow control problems in as little as half an hour on a desktop workstation.

Keywords : *optimal control, navier-stokes flow, GRG method, SQP method, reduced hessian method, newton method, quasi-newton method, suction, injection*

† 책임저자, 정회원 · 목원대학교 건축도시공학부 조교수
전화: 042-829-7617; Fax: 042-822-5260
E-mail: jhbark@mokwon.ac.kr
* 정회원 · 우석대학교 건축공학과 조교수

• 이 논문에 대한 토론을 2003년 3월 31일까지 본 학회에 보내주시면 2003년 6월호에 그 결과를 게재하겠습니다.

1. Introduction

Flow control has had a long history since Prandtl's early experiments demonstrated the feasibility of preventing flow separation by sucking fluid away from the boundary layer in a diverging channel.[1] Since then, much experimental work has been devoted to establishing the technological basis for flow control, and certain flow control problems have become amenable to analytical investigation, albeit often with simplifying assumptions; see the review by Gad-el-Hak.[2]

Recently, interest has increased in optimal flow control of viscous fluids, that is, the determination of optimal values of controls based on the governing partial differential equations of the fluid, i.e., the Navier-Stokes equation.[3] These problems are among the most challenging optimization problems in computational science and engineering. They owe their complexity to their being constrained by numerical approximations of the Navier-Stokes equations. These constraints are highly nonlinear and can number in the millions. Conventional optimization approaches prove inadequate for such large-scale optimization problems.

The development of numerical optimization methods for optimal flow control is built on a mathematical foundation that continues to be enlarged. A number of basic results concerning existence and regularity of solutions to the continuous problem, as well as error estimates for its numerical approximation, have been established mostly over the last decade; see the article by Gunzburger *et al.* for a good overview.[4]

This rich mathematical basis, the increasing power of computers, and the maturation of numerical methods for the flow simulation itself motivate the desire to develop numerical optimization methods for solution of optimal flow control problems. The latter forms the subject of this article. Here, we focus on a prototype problem of optimal control of fluids governed by the steady incompressible Navier-Stoke equations. The control is affected by the suction or injection of fluid on portions of the boundary, and the objective function represents the rate at which energy is dissipated in the fluid. We define the mathematical model in Section 2, and in Section 3. Actually we have proposed a method to solve this problem.[5] That method is superior to the steepest descent method, because it uses (approximate) curvature information in solving the optimization problem. However that method still entails much work. We perform analysis to get the value of state variables, which are used for the objective function and the sensitivity analysis. Usually the number of state variables is much larger than that of control variables. Therefore most of cost will be spent on getting the values of the state variables. One analysis step itself requires a lot of work; furthermore we need to perform several analysis steps at each optimization iteration. Therefore solving analysis with only continuation at each control iteration requires a great deal of work especially for higher Reynolds number problem.

In this study we propose a number of methods to overcome the computational complexity of the method proposed before.

These include integrating the optimizer with the solver, extending the use of continuation to the optimizer, incorporating sensitivity information to generate good initial guesses, and developing optimization methods that use exact curvature information.

We develop reduced Hessian sequential quadratic programming(SQP) methods that exploit the structure of the optimality conditions and avoid converging the flow equations at each optimization iteration.

Both quasi-Newton and Newton variants are developed. The SQP methods are compared in Section 4 to the approach of eliminating the

flow equations and variables, which is effectively the generalized reduced gradient(GRG) method. The examples demonstrates at least an order-of-magnitude reduction in time taken, allowing the solution of some two-dimensional optimal flow control problems in around a half hour.

2. A Problem in Optimal Control of Navier-Stokes Flows

Consider a steady, uniform, external, viscous, incompressible flow of a Newtonian fluid around a body with bounding surface Γ. We distinguish two possibly disjoint regions of the boundary: Γ_0, on which the velocity is specified to zero (i.e., the no-slip condition is specified), and Γ_c, on which velocity controls are applied. Thus $\Gamma = \Gamma_0 \cup \Gamma_c$. To approximate the farfield velocity condition, we truncate the domain of the problem with an inflow boundary Γ_1 on which is enforced the freestream velocity $\mathbf{u}_\infty$, and an outflow boundary Γ_2 on which a zero-traction condition is maintained. The flow domain is denoted as Ω. Let us represent the velocity vector, pressure, stress tensor, density, and viscosity of the fluid by, respectively, $\mathbf{u}$, p, σ, ρ, and μ.

The optimal control problem is to find the velocity control function $\mathbf{u}_c$, acting on Γ_c and the resulting fluid field variables that minimize the rate of energy dissipation, subject to the Navier-Stokes equations. Mathematically, the problem is to minimize

$$\frac{\mu}{2}\int_\Omega(\nabla \mathbf{u}+\nabla \mathbf{u}^T):(\nabla \mathbf{u}+\nabla \mathbf{u}^T)d\Omega \qquad (1)$$

subject to

$$\rho(\mathbf{u}\cdot\nabla)\,\mathbf{u}-\nabla\cdot\boldsymbol{\sigma}=\mathbf{0} \text{ in } \Omega \qquad (2)$$

$$\boldsymbol{\sigma}=-\boldsymbol{I}p+\frac{\mu}{2}(\nabla \mathbf{u}+\nabla \mathbf{u}^T) \text{ in } \Omega \qquad (3)$$

$$\nabla\cdot\mathbf{u}=0 \text{ in } \Omega \qquad (4)$$

$$\mathbf{u}=\mathbf{0} \quad \text{on } \Gamma_0 \qquad (5)$$

$$\mathbf{u}=\mathbf{u}_c \quad \text{on } \Gamma_c \qquad (6)$$

$$\mathbf{u}=\mathbf{u}_\infty \quad \text{on } \Gamma_1 \qquad (7)$$

$$\boldsymbol{\sigma}\cdot\mathbf{n}=\mathbf{0} \quad \text{on } \Gamma_2 \qquad (8)$$

where $(\nabla\mathbf{u})_{ij}=\partial u_j/\partial x_i$, and the symbol ":" represents the scalar product of two tensors, so that

$$\nabla \mathbf{u}:\nabla \mathbf{v}=\sum_{i,j}\frac{\partial u_i}{\partial x_j}\frac{\partial v_i}{\partial x_j}$$

Here, equation (1) is the dissipation function, (2) is the conservation of linear momentum equation (3) is the constitutive law, (4) is the conservation of mass(actually volume) equation, and equations (5)~(8) are boundary conditions. We eliminate the constitutive law and stress by substituting (3) into (2), and we further reduce the size of the problem by employing a penalty method: let us relax (4) by replacing it with

$$\nabla\cdot\mathbf{u}=-\varepsilon p \quad \text{in } \Omega \qquad (9)$$

Clearly as $\varepsilon\to 0$, we recover the original equation; in fact, the error in the derivative of $\mathbf{u}$ is of order ε.[12] By introducing the pressure in the mass equation, we can eliminate it from the problem by solving for p in (9) and substituting the resulting expression into (3).

In general it is not possible to solve infinite dimensional optimization problems such as equations (1)~(8) in closed form. Thus, we seek numerical approximations. Here, we use a Galerkin finite element method, Let the Sobolev subspace U^h be the space of all C^0 continuous piecewise polynomials that vanish on Γ_1 and Γ_0, and define the Sobolev subspace V^h similarly, with the added requirement that the functions also vanish on Γ_c. By restricting the velocity and control vectors to U^h, the infinite-dimensional

optimization problem (1)~(8) becomes finite-dimensional. We "triangulate" the computational domain to obtain N_s nodes in Ω and on Γ_2, N_c nodes on Γ_c, and N_1 nodes on Γ_1. Corresponding to a node i with coordinates $\mathbf{x}_i$, we have the compactly supported finite element basis function $\phi_i(\mathbf{x})$. Let N represent the total number of unknown nodal velocity vectors in the optimal control problem, i.e., $N=N_s+N_c$. Thus,

$$U^h = \text{span}\{\phi_1, \cdots\cdots\cdots, \phi_N\}$$
$$V^h = \text{span}\{\phi_1, \cdots\cdots\cdots, \phi_{Ns}\}$$

Furthermore, we associate with each node in Ω, on Γ_2, and on Γ_c the nodal velocity vector $\mathbf{u}_i$. The unknown velocities, and hence optimization variables, consist of the nodal state velocities $\mathbf{u}_i$, $i=1, \ldots, N_s$ and the nodal control velocities $\mathbf{u}_i$, $i=N_s+1, \ldots, N$. Let $\mathbf{u}_\infty^h$ and $\mathbf{u}_c^h$ i.e.,be the finite element interpolations of $\mathbf{u}_\infty$ and $\mathbf{u}_c$, i.e.,

$$\mathbf{u}_c^h(\mathbf{x}) = \sum_{i=N_s+1}^{N} \mathbf{u}_c(\mathbf{x}_i)\,\phi_i(\mathbf{x})$$

and

$$\mathbf{u}_\infty^h(\mathbf{x}) = \sum_{i=N+1}^{N+N_1} \mathbf{u}_\infty(\mathbf{x}_i)\,\phi_i(\mathbf{x})$$

Finally, we can represent $\mathbf{u}^h$, the finite element approximation to the velocity field, as

$$\mathbf{u}^h(\mathbf{x}) = \mathbf{u}_\infty^h(\mathbf{x}) + \sum_{i=1}^{N} \mathbf{u}_i\,\phi_i(\mathbf{x})$$

where, because of the property of finite element basis function, $\mathbf{u}^h(\mathbf{x}_i)=\mathbf{u}_i$. We note that the control function has been rendered finite dimensional: the control variables are simply the nodal values of the control function:

$$\mathbf{u}_i=\mathbf{u}_c(\mathbf{x}_i), \quad i=N_s+1, \ldots, N.$$

The finite element approximation of the optimal control problem (1)~(8) can now be stated as finding $\mathbf{u}^h \in U^h$ so that

$$\frac{\mu}{2}\int_\Omega (\nabla \mathbf{u}^h + (\nabla \mathbf{u}^h)^T) : (\nabla \mathbf{u}^h + (\nabla \mathbf{u}^h)^T)\, d\Omega \tag{10}$$

is minimized, subject to

$$\frac{\mu}{2}\int_\Omega (\nabla \mathbf{u}^h + (\nabla \mathbf{u}^h)^T) : (\nabla \mathbf{v}^h + (\nabla \mathbf{v}^h)^T)\, d\Omega + \frac{1}{\epsilon}\int_\Omega (\nabla \cdot \mathbf{u}^h)(\nabla \cdot \mathbf{v}^h)\, d\Omega + \int_\Omega \mathbf{v} \cdot \rho(\mathbf{u}^h \cdot \nabla)\,\mathbf{u}^h\, d\Omega = 0 \quad \text{for all } \mathbf{v}^h \in V^h \tag{11}$$

The objective function (10) is the discrete form of the dissipation function; the constraints (11) are a discretion of linear momentum equation.

Let us define $n=dN$ as the total number of unknown velocity components, where d is the physical dimension of the flow, i.e., $d=2$ or 3. Let $\mathbf{u}\in R^n$ denote the vector of unknown nodal velocity components. The vector $\mathbf{u}$ includes both state and control variables and can be symbolically partitioned as

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_s \\ \mathbf{u}_c \end{pmatrix}$$

where $\mathbf{u}_s\in R^{n'}$ represents the nodal velocities components associated with the state variables, and $\mathbf{u}_c\in R^{n'}$ the nodal velocities corresponding to the controls. Similarly, we can partition $\mathbf{h}(\mathbf{u})$: $R^n \rightarrow R^n$, the discrete form of the nonlinear convective term in equation (2), into $\mathbf{h}_s(\mathbf{u})$: $R^n \rightarrow R^{n'}$, a component associated with momentum equations written at nodes belonging to state variables, and $\mathbf{h}_c(\mathbf{u})$: $R^n \rightarrow R^{n'}$, a control component, associated with control nodes:

$$\mathbf{h}(\mathbf{u}) = \begin{pmatrix} \mathbf{h}_s(\mathbf{u}) \\ \mathbf{h}_c(\mathbf{u}) \end{pmatrix}$$

As can be seen from (11), $\mathbf{h}(\mathbf{u})$ is quadratic in $\mathbf{u}$. Finally, we define the matrix $\mathbf{K}^{\mu} \in R^{n \times n}$ arising from the discrete form of the viscous term, and $\mathbf{K}^{\varepsilon} \in R^{n \times n}$ corresponding to the discrete "pressure" term, in the momentum equation. Both $\mathbf{K}^{\mu}$ and $\mathbf{K}^{\varepsilon}$ are symmetric positive definite. These matrices can be partitioned into blocks corresponding to state and control variables,

$$\mathbf{K}^{\mu} = \begin{bmatrix} \mathbf{K}^{\mu}_{ss} & \mathbf{K}^{\mu}_{sc} \\ \mathbf{K}^{\mu}_{cs} & \mathbf{K}^{\mu}_{cc} \end{bmatrix}$$

$$\mathbf{K}^{\varepsilon} = \begin{bmatrix} \mathbf{K}^{\varepsilon}_{ss} & \mathbf{K}^{\varepsilon}_{sc} \\ \mathbf{K}^{\varepsilon}_{cs} & \mathbf{K}^{\varepsilon}_{cc} \end{bmatrix}$$

The sparsity of $\mathbf{K}^{\mu}$, $\mathbf{K}^{\varepsilon}$, and the Jacobian of $\mathbf{h}(\mathbf{u})$ is dictated by the sparsity of the graph underlying the finite element mesh. For quasi-uniform meshes, a node has a bounded number of neighbors, so that the number of nonzeroes per row is independent of problem size. Thus, these matrices have $O(n)$ nonzeroes. The constant is determined by d, by the order of the basis functions $\phi(\mathbf{x})$, and by the structure of the mesh, but is usually small.

The optimization problem (10)~(11) can be rewritten as follows: find $\mathbf{u}_s \in R^{n_s}$ and $\mathbf{u}_c \in R^{n_c}$ so that

$$\frac{1}{2}\mathbf{u}_s^T \mathbf{K}^{\mu}_{ss} \mathbf{u}_s + \frac{1}{2}\mathbf{u}_c^T \mathbf{K}^{\mu}_{cc} \mathbf{u}_c + \mathbf{u}_s^T \mathbf{K}^{\mu}_{sc} \mathbf{u}_c \qquad (12)$$

is minimized, subject to

$$(\mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss})\mathbf{u}_s + (\mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc})\mathbf{u}_c + \mathbf{h}(\mathbf{u}) = \mathbf{0} \qquad (13)$$

The problem (12)~(13) is characterized by $n = n_s + n_c$ variables and n_s nonlinear equality constraints. Both the objective function and constraints are quadratic in the optimization variables. For practical problem, $n_s \gg n_c$, since the control is affected at a small number of boundary points, while the state variables are associated with a triangulation of the flow domain. The number of degrees of freedom of the optimization problem, i.e., the number of variables n less the number of independent constraints n_s, is equal to the number of control variables and is thus small relative to the size of the problem. For problems of industrial interest, values as high as $n_s = O(10^6)$, $n_c = O(10^3)$ are desirable, although such problems are currently beyond the range of existing computers. We are thus in the realm of extremely large-scale, sparsely constrained nonlinear optimization. In the next section we develop SQP methods for this problem.

3. SQP METHODS

The most popular approach to solving optimization problem that are constrained by discretized partial differential equations(PDEs) is to eliminate the constraints by solving them for the state variables, given values of the control variables. For example, solve for $\mathbf{u}_s$ in equation (13) as a function of $\mathbf{u}_c$, and substitute this into the objective (12). The state variables are thus eliminated from the problem, and we now have an unconstrained optimization problem in the n_c variables $\mathbf{u}_c$. The gradient of the objective can be obtained through the implicit function theorem. This method is essentially the GRG method, since we satisfying the constraints at each iteration.[6]

Advantages of GRG for PDE-constrained optimization problems include: (i) a large reduction in size of the problem, especially when $n_c \ll n_s$, as is common in optimal control or optimal design; (ii) avoiding the large, sparse Hessians matrices inherent in the formulation as a constrained

problem of the form (12)~(13), in favor of a small, dense Hessian, thus enabling the use of standard dense nonlinear optimization software; (iii) the ability to use existing PDE solution algorithms in eliminating the state equations given values of control variables(i.e., *the forward problem*). This last advantage should not be taken lightly-over the past decade, many specialized and sophisticated algorithms(e.g., multilevel methods and preconditioned Krylov subspace methods) have been developed for solving various classes of PDEs and incorporated into robust software. If we retain the discrete PDEs as constraints, the optimizer becomes responsible for converging the state equations. Thus it is not immediately obvious how to extend sophisticated PDE solution techniques, such as domain decomposition methods and multilevel preconditioners, to the optimization problem, as solution of (12)~(13) appears to require. For these reasons, state equation elimination methods are almost universal for PDE-constrained problems. For examples of this approach in the context of flow control, see the papers contained in reference [21].

Nevertheless, the approach outlined in the last tow paragraphs possesses a distinct disadvantage: it requires exact solution of the state equation at each iteration. This can be an onerous requirement, especially for highly nonlinear problems such as those governed by Navier-Stokes equations. Instead, we pursue here *bona fide* SQP methods for the problem (12)~(13). SQP requires satisfaction of only a linear approximation of the constraints at each iteration, thereby avoiding the need to converge them fully.[7] Thus, state equations are satisfied simultaneously as control variables are converged to their optimal values. Furthermore, we consider a special reduced Hessian SQP method that retains the three advantages attributed to GRG in the paragraph above. In order to retain the last advantage, i.e., that existing(GRG-ready) PDE solvers can be used, several conditions must be met. First, the PDE solver must be Newton-based. Second, the sensitivity analysis capability of the PDE solver must be based on an adjoint approach. Finally, the solver must be modular enough so that the inner Newton linear step can be isolated from the code.

We begin with some definitions. Let $\mathbf{c}_s(\mathbf{u})$: $R^n \rightarrow R^n$, represent the residual or the discrete Navier-Stokes equations,

$$\mathbf{c}_s \equiv (\mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss})\,\mathbf{u}_s + (\mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc})\,\mathbf{u}_c + \mathbf{h}_s(\mathbf{u})$$

and let $\mathbf{J}(\mathbf{u}) \in R^{n \times n}$ denote the Jacobian of the nonlinear convective term $\mathbf{h}(\mathbf{u})$ with respect to the velocities $\mathbf{u}$. The matrix $\mathbf{J}(\mathbf{u})$ is nonsymmetric and indefinite and can be partitioned into state and control blocks, in the same manner as the sparsity structure of $\mathbf{J}(\mathbf{u})$ is identical to that of $\mathbf{K}^{\mu}$. Let the matrix $\mathbf{A}_s(\mathbf{u}) \in R^{n_s \times n}$ represent the Jacobian of $\mathbf{c}_s(\mathbf{u})$ with respect to $\mathbf{u}$. We can partition $\mathbf{A}_s(\mathbf{u})$ into $\mathbf{A}_{ss}(\mathbf{u}) \in R^{n_s \times n_s}$, $\mathbf{K}^{n_s \times n_s}$, the Jacobian of $\mathbf{c}_s(\mathbf{u})$ with respect to $\mathbf{u}_c$. Explicitly,

$$\mathbf{A}_s(\mathbf{u}) = [\,\mathbf{A}_{ss}(\mathbf{u}),\ \mathbf{A}_{sc}(\mathbf{u})]$$

with

$$\mathbf{A}_{ss}(\mathbf{u}) \equiv \mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss} + \mathbf{J}_{ss}(\mathbf{u}),$$

$$\mathbf{A}_{sc}(\mathbf{u}) \equiv \mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc} + \mathbf{J}_{sc}(\mathbf{u}).$$

Thus $\mathbf{A}_{ss}(\mathbf{u})$ and $\mathbf{A}_{sc}(\mathbf{u})$ have the same sparsity structure as $\mathbf{K}^{\mu}_{ss}$ and $\mathbf{K}^{\mu}_{sc}$, respectively.

We define the Lagrangian function, $L(\mathbf{u}, \boldsymbol{\lambda}_s)$, for the optimization problem (12)~(13), as

$$L(\mathbf{u}, \boldsymbol{\lambda}_s) \equiv \frac{1}{2}\mathbf{u}_s^T \mathbf{K}^{\mu}_{ss}\mathbf{u}_s + \frac{1}{2}\mathbf{u}_c^T \mathbf{K}^{\mu}_{cc}\mathbf{u}_{c+} \mathbf{u}_s^T \mathbf{K}^{\mu}_{sc}\mathbf{u}_c + \boldsymbol{\lambda}_s^T(\mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss})\mathbf{u}_s + \boldsymbol{\lambda}_s^T(\mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc})\mathbf{u}_c + \boldsymbol{\lambda}_s^T \mathbf{h}_s(\mathbf{u}) \quad (14)$$

where $\lambda_s \in R^{n_s}$ is the vector of Lagrange multipliers corresponding to the state equations. Let $\mathbf{g}(\mathbf{u}) \equiv \mathbf{K}^{\mu}\mathbf{u}$ represent the gradient of the objective function (12). We denote by $\mathbf{H}_k \in R^{n\times n}$ the symmetric, indefinite Hessian matrix of the kth element of $\mathbf{h}(\mathbf{u})$, and by $\mathbf{W}(\lambda_s) \in R^{n\times n}$ the symmetric, indefinite Hessian matrix of the Lagrangian function $L(\mathbf{u}, \lambda)$ (both Hessians are with respect to $\mathbf{u}$). Because $\mathbf{h}(\mathbf{u})$ is quadratic in $\mathbf{u}$, $\mathbf{H}_k$ is a constant matrix. Note that $(\mathbf{H}_k)_{ij}$ is nonzero only when nodes i, j, and k all belong to the same element; it follows that the (i, j) entry of the matrix

$$\mathbf{H}(\lambda) \equiv \sum_{k=1}^{n_s} \lambda_k \mathbf{H}_k$$

is nonzero only when nodes i and j belong to the same element. Therefore, $\mathbf{H}(\lambda)$ has the same mesh-based sparsity structure as the other finite element matrices ($\mathbf{K}^{\mu}$, $\mathbf{K}^{\varepsilon}$, and $\mathbf{J}$), as dose $\mathbf{W}(\lambda)$, since $\mathbf{W}(\lambda) = \mathbf{K}^{\mu} + \mathbf{H}(\lambda)$.

The necessary conditions for an optimal solution of the problem (12)~(13) reflect the vanishing of the gradient of the Lagrangian (14) and can be expressed by the set of nonlinear equations

$$\begin{bmatrix} \mathbf{K}^{\mu}_{ss} & \mathbf{K}^{\mu}_{sc} & \mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss} \\ \mathbf{K}^{\mu}_{cs} & \mathbf{K}^{\mu}_{cc} & \mathbf{K}^{\mu}_{cs} + \mathbf{K}^{\varepsilon}_{cs} \\ \mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss} & \mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{u}_s \\ \mathbf{u}_c \\ \lambda_s \end{Bmatrix} + \begin{Bmatrix} \mathbf{J}^T_{ss}\lambda_s \\ \mathbf{J}^T_{sc}\lambda_s \\ \mathbf{h}_s(\mathbf{u}) \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}. \quad (15)$$

SQP can be derived as a Newton method for solving the optimality conditions.[6] One step of Newton's method for (15) is given by solving the linear system

$$\begin{bmatrix} \mathbf{K}^{\mu}_{ss} + \mathbf{H}^k_{ss} & \mathbf{K}^{\mu}_{sc} + \mathbf{H}^k_{sc} & \mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss} + (\mathbf{J}^k_{ss})^T \\ \mathbf{K}^{\mu}_{cs} + \mathbf{H}^k_{cs} & \mathbf{K}^{\mu}_{cc} + \mathbf{H}^k_{cc} & \mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc} + (\mathbf{J}^k_{sc})^T \\ \mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss} + \mathbf{J}^k_{ss} & \mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc} + \mathbf{J}^k_{sc} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}^k_s \\ \mathbf{p}^k_c \\ \lambda^{k+1}_s \end{Bmatrix} = - \begin{Bmatrix} \mathbf{K}^{\mu}_{ss}\mathbf{u}^k_s + \mathbf{K}^{\mu}_{sc}\mathbf{u}^k_c \\ \mathbf{K}^{\mu}_{cs}\mathbf{u}^k_s + \mathbf{K}^{\mu}_{cc}\mathbf{u}^k_c \\ (\mathbf{K}^{\mu}_{ss} + \mathbf{K}^{\varepsilon}_{ss})\mathbf{u}^k_s + (\mathbf{K}^{\mu}_{sc} + \mathbf{K}^{\varepsilon}_{sc})\mathbf{u}^k_c + \mathbf{h}^k_s = \mathbf{0} \end{Bmatrix} \quad (16)$$

for the increments in the state and control velocities, $\mathbf{p}_s{}^{\kappa}$, and $\mathbf{p}_c{}^{\kappa}$, and for the new multiplier estimate $\lambda_s{}^{\kappa+1}$, where the superscript k indicates evaluation of a quantity at $\mathbf{u}^{\kappa}$ or λ^{κ}. The state and control variables are then updated by

$$\mathbf{u}_s{}^{\kappa+1} = \mathbf{u}_s{}^{\kappa} + \mathbf{p}_s{}^{\kappa}, \quad \mathbf{u}_c{}^{\kappa+1} = \mathbf{u}_c{}^{\kappa} + \mathbf{p}_c{}^{\kappa}$$

Let us rewrite the Newton equations (16) in terms of the symbols defined at the beginning of this section as

$$\begin{bmatrix} \mathbf{W}^k_{ss} & \mathbf{W}^k_{sc} & (\mathbf{A}^k_{ss})^T \\ \mathbf{W}^k_{cs} & \mathbf{W}^k_{cc} & (\mathbf{A}^k_{sc})^T \\ \mathbf{A}^k_{ss} & \mathbf{A}^k_{sc} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{p}^k_s \\ \mathbf{p}^k_c \\ \lambda^{k+1}_s \end{Bmatrix} = \begin{Bmatrix} -\mathbf{g}^k_s \\ -\mathbf{g}^k_c \\ -\mathbf{c}^k_s \end{Bmatrix}. \quad (17)$$

The major difficulty in solving this linear system is its extremely larger, sparse coefficient matrix, the Karush-Kuhn-Tucker(KKT) matrix , which is of order $(n+n_s)\times(n+n_s)$. For industrial flow problems on unstructured meshes, the forward (i.e., flow simulation) problem is memory-bound, and even on large supercomputers one can barely afford memory for $\mathbf{A}_{ss}$, let alone the entire matrix. So sparse factorization of the KKT matrix is not viable. One possibility to solve equation (17) using a Krylov method such as the minimum residual method. However it is not immediately obvious how to precondition the coefficient matrix.

Instead, consider the following block elimination. In the remainder of this section, we drop the superscript κ; it is understood that all quan-

tities depending on $\mathbf{u}$ or λ_s are evaluated at $\mathbf{u}^\kappa$ or λ_s^κ. First, solve for $\mathbf{p}_s$ from the last block of equations to obtain

$$\mathbf{p}_s = -\mathbf{A}_{ss}^{-1}(\mathbf{c}_s + \mathbf{A}_{sc}\mathbf{p}_c) \tag{18}$$

where the invertibility of $\mathbf{A}_{ss}$ is guaranteed by the wellposedness of the boundary value problem (provided of course that we are away from limit points). Then, substitute this expression into the first block of equations. Finally, premultiply the first block of equations by $-\mathbf{A}_{sc}^T\mathbf{A}_{ss}^{-T}$ and add it to the second block of equations, thereby eliminating λ_s. The result is a linear system that can be solved for $\mathbf{p}_c$, namely

$$\begin{aligned}\mathbf{W}_z\ \mathbf{p}_c &= (\mathbf{A}_{sc}^T\mathbf{A}_{ss}^{-T}\mathbf{W}_{ss} - \mathbf{W}_{cs})\mathbf{A}_{ss}^{-1}\mathbf{c}_s + \mathbf{A}_{sc}^T\mathbf{A}_{ss}^{-T}\mathbf{g}_s - \mathbf{g}_c \\ \mathbf{W}_z &\equiv (\mathbf{A}_{sc}^T\mathbf{A}_{ss}^{-T}\mathbf{W}_{ss}\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc} - \mathbf{A}_{sc}^T\mathbf{A}_{ss}^{-T}\mathbf{W}_{sc} \\ &\quad - \mathbf{W}_{cs}\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc} + \mathbf{W}_{cc})\end{aligned} \tag{19}$$

Finally, the new estimate of the Lagrange multiplier vector is recovered from

$$\lambda_s^{\kappa+1} = -\mathbf{A}_{ss}^{-T}(\mathbf{W}_{ss}\mathbf{p}_s + \mathbf{W}_{sc}\mathbf{p}_c + \mathbf{g}_s) \tag{20}$$

which is a second-order multiplier estimate. This leads to the following algorithm.

[*Algorithm-1 : Newton SQP*]
$\kappa = 0$; $\mathbf{u}_s^0 = \mathbf{u}_c^0 = \lambda_s^0 = \mathbf{0}$
while $\| (\mathbf{A}_s^\kappa)^T \lambda_s^\kappa - \mathbf{g}^\kappa \| \neq 0$ and $\| \mathbf{c}_s^\kappa \| \neq 0$
$\kappa = \kappa + 1$
Solve equation (19) for $\mathbf{p}_c^\kappa$
$\mathbf{u}_c^{\kappa+1} = \mathbf{u}_c^\kappa + \mathbf{p}_c^\kappa$
Find $\mathbf{p}_s^\kappa$ from equation (18)
$\mathbf{u}_s^{\kappa+1} = \mathbf{u}_s^\kappa + \mathbf{p}_s^\kappa$
Find $\lambda_s^{\kappa+1}$ from equation (20)
end

Algorithm-1 displays a quadratic convergence rate provided that (i) at the optimal solution we are away from a limit or bifurcation point in the forward problem(i.e. ,$\mathbf{A}_{ss}$ is nonsingular); (ii) $\mathbf{W}_z$ is positive definite at the optimal solution; and (iii) ($\mathbf{u}^0$, λ_s^0)is sufficiently close to the optimal solution.

The justification for the block elimination (18)~(20) and the resulting Algorithm-1 is that there result only two types of linear system to be solved: those involving $\mathbf{A}_{ss}$ or its transpose as the coefficient matrix, and the system that determines $\mathbf{p}_c$, equation (19), with coefficient matrix $\mathbf{W}_z$. In the former case, these systems are "easy" to solve, since they have the same coefficient matrix as that of a Newton step for the state equations. Thus any(Newton-based) Navier-Stokes solver can be enlisted for this task, enabling the exploitation of many of the advances in solving the forward problem developed over the last decade(including domain decomposition and multilevel methods). In the latter case, solution of equation (19) is also easy since $\mathbf{W}_z$ is of order of the number of control variables, which under the assumption that $n_s \gg n_c$, is very small. standard dense factorization is therefore appropriate.

When implementing *Algorithm-1*, one of course dose not invert $\mathbf{A}_{ss}$; one instead forms the matrix $\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}$ by solving, with coefficient matrix $\mathbf{A}_{ss}$, for the n_c right-hand sides composed of the columns of $\mathbf{A}_{sc}$. An additional solve with the same coefficient matrix for the right-hand side $\mathbf{c}_s$ is necessary. Finally, equation (20) implies an additional right-hand side solve, but with transpose of $\mathbf{A}_{ss}$ as coefficient matrix. So each iteration of *Algorithm-1* requires solving a linear system with coefficient matrix $\mathbf{A}_{ss}$(the state equation Jacobian matrix) and having n_c+1 right-hand sides, as well as a linear system with $\mathbf{A}_{ss}^T$ as coefficient matrix and one right-hand side. If sparse factorization of $\mathbf{A}_{ss}$ is viable, for example for two-dimensional flows or low Reyolds number three-dimensional flows, then one iteration of *Algorithm-1* entails one factorization and n_c+2

pairs of triangular solve(compare this with full solution of the flow equations, as in GRG). If quasi-uniform meshes and nested dissection orderings are used, and if pivoting is not required, $\mathbf{A}_{ss}$ can be factored with $O(n_s^2)$ work in 3D and $O(n_s^{1.5})$ work in 2D.[7] In any case, the cost of one iteration of *Algorithm-1* is a fraction of the cost of the forward problem. On the other hand, if sparse factorization is not practical, and an iterative method must be used, one is faced with n_c+2 solves. When n_c is large, it becomes imperative to use iterative methods tailored to multiple right-hand sides; it also pays to invest in a good preconditioner, since its construction can be amortized over the right-hand sides. In particular, domain decomposition methods tailored to multiple right-hand sides appear to be attractive.[8]

Once the matrix $\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}$ is created, forming its products with submatrices of $\mathbf{W}$ presents no difficulty. Recall that $\mathbf{W}$ has $O(n)$ nonzeroes and sparsity structure dictated by the underlying finite element mesh. In particular it is stored using the same(sparse compressed row) data structure that all the other finite element matrices use. Therefore forming products with submatrices of $\mathbf{W}$ requires work proportional to the row dimension of the submatrix. Thus, it is easy to see that, beyond forming the matrix $\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}$, the only other major effort in *Algorithm-1* is $O(n_s n_c^2)$ work associated with forming $\mathbf{W}_z$, and $O(n_c^3)$ in factoring $\mathbf{W}_z$.

Let us examine the connection with other SQP methods. In fact, the block elimination (18)~(20) is identical to a reduced Hessian SQP method with a particular choice of null and range bases. This can be seen by decomposing the search direction $\mathbf{p}$ into to components,

$$\mathbf{p}=\mathbf{Z}\mathbf{p}_z+\mathbf{Y}\mathbf{p}_y \tag{21}$$

in which $\mathbf{Z}\in R^{n\times n_c}$ is a matrix whose columns form a basis for the null space of $\mathbf{A}_s$, and $\mathbf{Y}\in R^{n\times n_s}$ is chosen so that the matrix

$$\mathbf{Q}=[\mathbf{Z}\mathbf{Y}]$$

is nonsingular, and hence $\mathbf{Z}$ and $\mathbf{Y}$ form a basis for R^n. We refer to P_y as the range space component, even though strictly speaking, the columns of $\mathbf{Y}$ need not span the range space of $\mathbf{A}_s^T$.

The range space step is completely determined by substituting equation (21) into the last block of equation (17), resulting in the $n_s\times n_s$ systems

$$\mathbf{A}_s\mathbf{Y}\mathbf{p}_y=-\mathbf{c}_s \tag{22}$$

The null space move is found by substituting equation (21) into the first two blocks of equation (17), and premultiplying by $\mathbf{Z}^T$, to obtain the equations for $\mathbf{p}_z$,

$$\mathbf{Z}^T\mathbf{W}\mathbf{Z}\mathbf{p}_z=-\mathbf{Z}^t(\mathbf{g}+\mathbf{W}\mathbf{Y}\mathbf{p}_y) \tag{23}$$

The $n_c\times n_c$ matrix $\mathbf{Z}^T\mathbf{W}\mathbf{Z}$ is known as the reduced Hessian matrix. If one chooses the nonorthogonal bases

$$\mathbf{Z}=\begin{bmatrix} -\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc} \\ \mathbf{I} \end{bmatrix} \tag{24}$$

and

$$\mathbf{Y}=\begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix} \tag{25}$$

then one sees that the null space step (23) is identical to equation (19), the equation for determining the move in the control variables. Indeed, the coefficient matrix of (19), $\mathbf{W}_z$, is exactly the reduced Hessian $\mathbf{Z}^T\mathbf{W}\mathbf{Z}$ and is therefore at least positive semidefinite in the vicinity of a minimum. The state variable update (18) is comprised of the state equation Newton step, i.e., equation (22) using the range basis

(25), as well the null space contribution $-\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}\mathbf{p}_z$, where $\mathbf{p}_z \equiv \mathbf{p}_c$. The choice of bases (25) is known as a "coordinate basis" and as been applied to optimization problems in inverse heat conduction[9], structural design[10),11)], and compressible flow[12),13)]. SQP methods using these bases have been analyzed in references [14, 22, 23], among others.

As mentioned earlier, one of the difficulties with *Algorithm-1*, i.e., the *bona fide* Newton method, arises when iterative solution of systems involving $\mathbf{A}_{ss}$ is necessary; in this case the benefit from solving n_c+2 systems with the same coefficient matrix but different right-hand side is not as extensive as with sparse LU factorization. Might it be possible to give up the(local) quadratic convergence guarantee in exchange for the need to solve fewer systems involving $\mathbf{A}_{ss}$ at each iteration?

The answer turns out to be affirmative if we consider a quasi-Newton, rather than a true Newton, method. Consider equation (19), the control variable equation. Its coefficient matrix is the reduced Hessian $\mathbf{W}_z$. This matrix is positive definite at a strict local minimum, as well as being small($n_c \times n_c$) and dense. It makes sense to recur a quasi-Newton approximation to it; thus we can avoid the construction of the matrix $\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}$. By replacing $\mathbf{W}_z$ with its quasi-Newton approximation, $\mathbf{B}_z$, it is easy to see that equations (18)~(20) now entail only five solutions of systems with $\mathbf{A}_{ss}$ or its transpose as coefficient matrix. This is a big reduction, especially for large n_c. However, we can do even better. At the expense of a reduction from one-step to two-step superlinear convergence,[14] we ignore the second-order terms(those involving submatrices of $\mathbf{W}$) on the right-hand side of equation (19). Furthermore, we reduce equation (20) to a first-order Lagrange multiplier estimate by dropping terms involving blocks of $\mathbf{W}$. This results in the following algorithm, in which the BFGS formula is used to update the quasi-Newton approximation of the reduced Hessian.

[*Algorithm-2 ; Quasi-Newton SQP*]

$\kappa = 0$; $\mathbf{u}_s^0 = \mathbf{u}_c^0 = \boldsymbol{\lambda}_s^0 = \mathbf{0}$; $\mathbf{B}_z^0 = \mathbf{I}$

$\boldsymbol{\lambda}_s^0 = (\mathbf{A}_{ss}^0)^{-T}\mathbf{g}_s^0$

$\mathbf{g}_z^0 = -(\mathbf{A}_{sc}^0)^T\boldsymbol{\lambda}_s^0 + \mathbf{g}_c^0$

while $\| \mathbf{g}_z^k \| \neq 0$ and $\| \mathbf{c}_s^\kappa \| \neq 0$

$\mathbf{p}_c^k = -(\mathbf{B}_z^k)^{-1}\mathbf{g}_z$

$\mathbf{u}_c^{\kappa+1} = \mathbf{u}_c^\kappa + \mathbf{p}_c^\kappa$

$\mathbf{p}_s^k = -(\mathbf{A}_z^k)^{-1}(\mathbf{c}_s^\kappa + \mathbf{A}_{sc}^\kappa \mathbf{p}_c^k)$

$\mathbf{u}_s^{\kappa+1} = \mathbf{u}_s^\kappa + \mathbf{p}_s^\kappa$

$\boldsymbol{\lambda}_s^{k+1} = (\mathbf{A}_{ss}^{k+1})^{-T}\mathbf{g}_s^{\kappa+1}$

$\mathbf{g}_z^{k+1} = -(\mathbf{A}_{sc}^{k+1})^T\boldsymbol{\lambda}_s^{k+1} + \mathbf{g}_c^{\kappa+1}$

$\mathbf{y}_z^\kappa = \mathbf{g}_z^{\kappa+1} - \mathbf{g}_z^\kappa$

$\mathbf{B}_z^{k+1} = \mathbf{B}_z^k + \{(\mathbf{g}_z^\kappa)^T\mathbf{p}_c^\kappa\}^{-1}\mathbf{g}_z^\kappa(\mathbf{g}_z^\kappa)^T$

$+ \{(\mathbf{y}_z^\kappa)^T\mathbf{p}_c^\kappa\}^{-1}\mathbf{y}_z^\kappa(\mathbf{g}_z^\kappa)^T$

$\mathbf{g}_z^\kappa = \mathbf{g}_z^{\kappa+1}$

$\kappa = \kappa + 1$

end

Algorithm-2 requires only two solves involving $\mathbf{A}_{ss}$ per iteration as compared with n_c+2 in *Algorithm-1*. This represents a substantial reduction in effort when iterative solves are used and when n_c is significant. Of course, *Algorithm-2* will generally require more iteration to converge than *Algorithm-1*, since it does not compute exact curvature information. The first of the two linear solves has $\mathbf{A}_{ss}$ as its coefficient matrix, and we term it the *state variable update*. It comprises two components: a Newton step on the states equations ($-\mathbf{A}_{ss}^{-1}\mathbf{c}_s$), and a first order change in the states due to a change in the control variables ($-\mathbf{A}_{ss}^{-1}\mathbf{A}_{sc}\mathbf{p}_c$). The second linear system to be solved at each iteration has $\mathbf{A}^T_{ss}$ as its coefficient matrix, and is termed the adjoint step, because of parallels with adjoint methods for sensitivity analysis.[15),16)] The steps of this algorithm are almost identical to a quasi-Newton GRG methed; the major difference is that in GRG the state equations only a single

Newton step is carried out.

Algorithm-1 and Algorithm-2 as presented above are not sufficient to guarantee convergence to a stationary point from arbitrary initial points. It is well known that for the forward problem, i.e., solving the discrete Navier-Stokes equations, Newton's method is only locally convergent. The diameter of the ball of convergence is of the order of the inverse of the Reynolds number that characterizes the flow[17]; better initial guesses are thus required as the Reynolds number increases. The optimal control problem should be no easier to converge than the forward problem, give that the flow equations form part of the first-order optimality condition. This suggests continuation methods, which are popular techniques for globalizing the forward problem.[17] Here we use a simple continuation on Reynolds number. That is, suppose we want to solve an optimal control problem with a Reynolds number of Re^*, for which it is difficult to find a starting point from which Newton's method will converge. Instead, we solve a sequence of optimization problem characterized by increasing Reynolds number, beginning with Re=0, and incrementing by $\triangle$Re. Optimization problem i, with Reynolds number $i\triangle$Re/Re, is solved by either *Algorithm-1* or *Algorithm-2*, to generate a good starting point for problem $i+1$. *Algorithm-1* is initialized with the optimal Lagrange multipliers and state and good control variables from optimization problem i-1. *Algorithm-2* includes the same initialization, but in addition takes the initial BFGS approximation at the Lagrangian Hessian matrix to be the Hessian approximation at the solution of problem i-1. Note that when Re= 0, the nonlinear terms drop from the flow equations, and thus optimization problem (12)~(13) is an equality constrained quadratic programming program, solvable in one step. For subsequent problem, there exists a sufficiently small $\triangle$Re such that *Algorithm-1* and *Algorithm-2* converge to the solution of optimization problem $i+1$ using initial data from problem i (provided we are away from bifurcation or singular points).

In the next section we use continuation variants of *Algorithm-1* and *Algorithm-2* to solve some more problems in boundary control of viscous incompressible flow, and compare their performance to the GRG methods.

4. Numerical Examples

In this section we compare *Algorithm-1*(Newton-SQP or N-SQP) and *Algorithm-2* (quasi-Newton-SQP or QN-SQP) of the previous section with both a quasi-Newton-GRG method (QN-GRG) and a steepest descent-GRG method (SD-GRG). With the QN-GRG method we converge the flow equations fully at each optimization iteration using a Newton solver, and we employ a BFGS formula to approximate the Hessian of the objective function. SD-GRG refers to a similar method, except that a search direction is taken in a direction opposite to the gradient of the objective function. This method is chosen because of its popularity in the optimal control literature and since it is easy to implement. In both GRG cases, the sensitivity equations are used to compute the objective function gradient exactly using a direct(as opposed to adjoint) method (see, e.g., reference [15] or [16]).

Continuation is applied to N-SQP and QN-SQP as described at the end of Section 3. We apply continuation to both GRG methods at the level of the forward problem : since the GRG methods entail satisfaction of the flow equations at each iteration, the forward problem is completely solved at each optimization iteration using continuation on Reynolds number, i.e., starting with Re=0 and incrementing by $\triangle$Re until Re is reached. There is another alternative that is intermediate between the extremes of GRG(completely solving the flow equations at

each iteration, using continuation on Reynolds number) and SQP(solving a linearized approximation only). This is to use the continuation at the level of the optimization problem, as described at the end of Section 3, in conjunction with full solution of the flow equations *for the current value of* Re. We refer to this method as continuation QN-GRG, or CQN-GRG. It used the converged flow solution of the previous optimization iteration as an initial guess to the velocity field of the current iteration. We expect its efficiency to be between SQP and GRG.

Finite element approximation of the continuous problem is archived with isoparametric biquadratic rectangles in 2D. These elements produce errors in the derivatives of $\mathbf{u}^h$ of $O(h^2+\varepsilon)$.[17] All integrals are evaluated with Gauss-Legendre numerical integration using 3×3 scheme, with the exception of the penalized terms, which are "underintegrated" with a 2×2 scheme to avoid "locking".[18] The value of the penalty parameter ε is taken to be 10^{-7}. The flow solver has been verified against a standard benchmark, the driven cavity problem. We have chosen a value of 10^{-7} in the Euclidean norm of the first order optimality condition (15) to terminate optimization iteration. The Reynold number step size for continuation, $\triangle$Re, is in all cases 50.

Solution of systems involving $\mathbf{A}_{ss}$ and its transpose is at the heart of all five methods. In GRG, these systems characterize a Newton step on the state equations, as well as the sensitivity equations. In SQP their presence reflects the choice of a block elimination(or a null space basis) that favors "inverting" $\mathbf{A}_{ss}$. The cost of solving these systems asymptotically dominates an optimization iteration, whether in SQP or GRG guise, since it is the only step that is superlinear in n_s(all others are linear at worst). Cleary one would like to perform these linear solves as cheaply as possible. Our initial desire was to use a Krylov subspace method, specifically the quasi-minimum residual (QMR) method, to solve the systems involving $\mathbf{A}_{ss}$ and its transpose, since methods of this type are representative of large-scale CFD solvers. However, after trying QMR on the discrete penalty-based Navier-Stokes equations, we concludes that the equations were too ill-conditioned for iterative solution to be competitive. Even incomplete LU preconditioning was ineffective in allowing convergence in reasonable time. This no doubt stems from the penalty formulation, and we expect that a different conclusion would have been reached had a mixed formulation (one that included both velocity and pressure) been chosen.

Instead, we have chosen the multifrontal sparse LU factorization code UMFPACK[19),20)] for solving the systems involving $\mathbf{A}_{ss}$ and its transpose. UMFPACK provides a routine for computing the LU factors of a given sparse matrix. Once this has been computed, UMFPACK provides further routines for finding the solutions to systems involving the triangular factors of a matrix as well as their transposes. Thus, using UMFPACK, only a single factorization of $\mathbf{A}_{ss}$ is required at each iteration of the Newton-SQP and quasi-Newton SQP methods; the primary difference between the two methods therefore lies in the number of triangular solves each performs(in addition to the computation of second derivatives). Using a sparse direct method of course ultimately limits the maximum size of problems we can solve, relative to no-fill method such as ILU-QMP. Even though we have found UMFPACK to be very effective at reducing fill, a significant amount of fill is unavoidable for three-dimensional, higher-order, vector finite element problems, due to the high average degree of nodes in the finite element graph.

To compare the five different optimal control methods, we choose a model problem of two-dimensional flow around an infinite cylinder.

Here, we define the Reynolds number as

$$\mathrm{Re}=\frac{\rho D|\ \mathbf{u}_\infty|}{\mu}$$

where D is the cylinder diameter. Without boundary control, the behavior of the velocity field with increasing Reynolds number is depicted in, for example, reference [24]. Flow separation is evident for Reynolds numbers as low as 10. The flowfield remains stationary and exhibits two symmetric standing eddies up to around Re=50. Beyond this range, the wake becomes increasingly unstable and oscillatory, and a vortex street forms in the wake and persists down-stream. Beyond a Reynolds number of about 60, the flow is neither symmetric about the cylinder centerline, nor steady, as assumed by our model.

In this section we solve optimal control problem for flows around a cylinder with Reynolds numbers as high as 500. However, we use the steady form of the governing flow equations as constraints; furthermore, our flow model assumes symmetry of the velocity field about the centerline of the cylinder. In principle, the optimal control problem for flow around a cylinder at Re=500 should be constrained by the time-dependent version of the Navier-Stokes equations and should not assume symmetric of the velocity field. However, if one makes two *ad hoc* assumptions-that the optimal velocity field for the time-dependent problem is both steady and symmetric, even if the uncontrolled flow is neither-then one ought to be able to use the steady Navier-Stokes equations as constraints in the formulation of the optimal problem and cut the computational domain in half to exploit symmetry. Computational complexity is greatly reduced under these two assumptions. It is clear that a sufficiently close initial guess to this steady, symmetric optimal control problem would converge to the optimal defined by the unsteady, unsymmetric control problem.[5]

Thus, we consider the computational domain and boundary conditions depicted in Fig. 1 and associated mesh of Fig. 2. The mesh uses 680 isoparametric biquadratic elements, resulting in N=2829 nodes and n=5658 unknown velocity components.

Boundary control of the velocity is applied at five equally spaced points on the backside of the cylinder. Since each has two velocity components, we have a total of n_c=10 control variables. Streamlines for the case of no control and Re=500 are shown in Fig. 3. The streamlines are seen to detach near the top of the cylnder, and there is a large recirculation zone behind the cylinder. After solving the optimization problem (12)~(13), the streamlines shown in Fig. 4 are obtained. The resulting flow resembles a potential flow, and separation is greatly reduced.

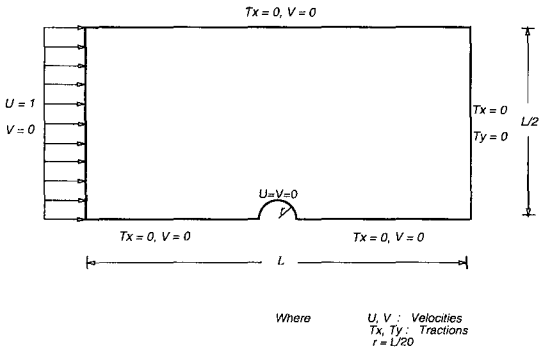

Figure 1 Computational domain and boundary conditions, flow around infinite cylinder

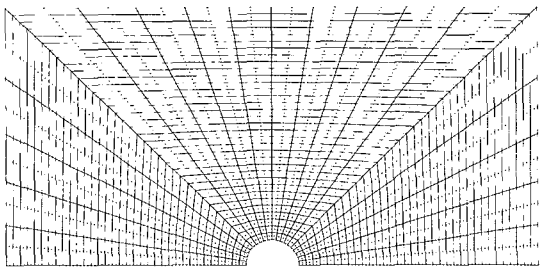

Figure 2 Mesh of 680 biquadratic elements, 2829 nodes

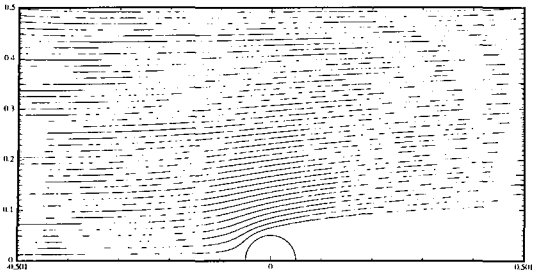

Figure 3 Streamlines, without control, Re=500

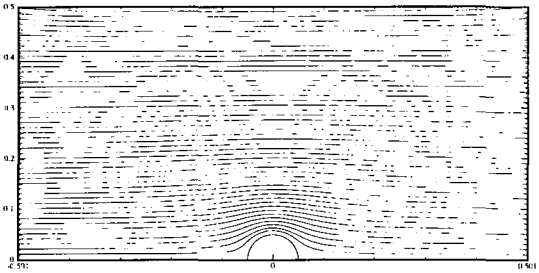

Figure 4 Optimal streamlines

As a comparison of the methods, we solve a sequence of five optimization problems, corresponding to Re=100, 200, 300, 400, and 500, using the five optimization methods described above. Table 1 compares the number of optimization iterations taken by the five methods. For the continuation methods, i.e., CQN-GRG, QN-SQP, and N-SQP, the numbers reported in the table are the sum of iterations across all optimization problems(each corresponding to a value of Re).

As expected, SD-GRG takes by far the largest number of iterations. The symbol "×" means that the method failed to converge to a stationary point, which occurred with SD-GRG for Re=400

Table 1 Number of Optimization Iteration

Re	SD-GRG	QN-GRG	CQN-GRG	QN-SQP	N-SQP
100	643	23	34	29	13
200	265	30	54	30	14
300	291	35	70	37	15
400	×	45	89	45	18
500	×	×	102	52	20

Table 2 Timings in minutes on a DEC 3000/700

Re	SD-GRG	QN-GRG	CQN-GRG	QN-SQP	N-SQP
100	3766.75	74.10	41.17	18.37	16.68
200	2922.52	168.47	66.18	19.32	17.95
300	4744.93	278.98	86.77	25.02	22.10
400	×	462.65	110.52	30.72	26.27
500	×	×	126.98	35.15	30.10

and 500, and QN-GRG for Re=500. QN-GRG takes an order of magnitude fewer iterations than SD-GRG, due to its ability to approximate curvature of the control variable space. However, CQN-GRG takes almost twice as many iterations as QN-GRG. The reason is that the sequence of Reynolds number steps is "packed" into a single optimization iteration with QN-GRG, while CQN-GRG "promotes" the continuation on Re to the level of the optimization problem; thus, these steps contribute to the number reported in Table 1. On the other hand, the cost per iteration of CQN-GRG should be significantly lower than QN-GRG, since each flow solution need only be converged for the current Reynolds number, and the solver benefits from an initial guess taken from the previous converged value. QN-SQP reduces the number of iterations by almost 50% over CQN-GRG, since it liberates the optimizer from having to follow a path dictated by satisfaction of the flow equations. The result is that the number of iterations taken by QN-GRG and QN-SQP are similar. Of course the cost per iteration of QN-GRG(and of CQN-GRG) will be much higher than QN-SQP, which will be reflected in CPU time. N-SQP offers the best performance from the point of view of iterations taken, providing on average 2.5 times fewer iterations than the QN methods. Of course, this reduction in steps taken must be balanced with increased work per iteration associated with N-SQP relative to QN-SQP.

Table 2 shows timings of each method for the sequence of Reynolds numbers solved on a

DEC 3000/700 with 225MHz Alpha Processor and 512Mb Memory. Note that these timings are in minutes, so the SD method requires several days to find an optimal solution, which is unacceptable. The QN-GRG method offers over an order of magnitude reduction, but the measured times are still on the order of hours. Partially integrating flow solution with optimization, as in CQN-GRG, further reduces CPU time by a factor of about three. A further factor of three reduction is achieved by fully integrating flow solution with optimization, through the use of QN-SQP. This result from its requiring only two linear solves per iteration, against CQN-GRG's fully converging the flow equations. On the other hand, CPU time decreases only marginally when the N-SQP method is used, typically between 10 and 15%, even though the number of iterations is significantly lower. This results from the additional work N-SQP must do at each iteration, chiefly through the additional right-hand side solves and construction of the exact reduced Hessian $\mathbf{W}_z$ in equation (19). While the cost of constructing $\mathbf{W}_z$ is linear in n_s, the constant is large, since it involves element generation- and assembly- like finite element computations. The conclusion is that, while, the *asymptotic* costs per iteration of QN-GRG and N-SQP are the same, it turns out that, for the value of n_s we are considering, the lower order terms contribute meaningfully, and conspire to make N-SQP roughly twice as expensive per iteration as QN-SQP. Still, the Newton method does take less time; whether this is worth the additional effort of implementing second derivatives will depend on the particular application.

5. Final Remarks

Based on the comparison of the previous section, we conclude that the reduced Hessian SQP methods are overwhelmingly superior to the GRG methods that are popular for optimization problems involving PDEs as constraints, offering over an order of magnitude improvement in time required for the optimal flow control problems considered. In particular, the methods are so efficient that the optimal control for a two-dimensional flow around a cylinder at Reynolds number 500 is found in about a half hour on a desktop workstation. Even though the Newton-SQP method takes significantly fewer iterations, its need to construct exact Hessians and to solve linear systems that number on the order of the number of control variables makes it only marginally more efficient than its quasi-Newton counter-part, based on our two-dimensional results.

The GRG methods descried here represent a strict interpretation of the GRG idea-at each optimization iteration, the flow equations are converged to a tolerance of 10^{-7}. However, in the spirit of SQP, one might choose a greater value of the tolerance at early iterations, making sure to reduce the tolerance to its target value as the optimization iterations converge. Indeed, in the limit of one Newton step on the flow equations per optimization iteration, we essentially recover the reduced SQP method. A related idea is to pose the early optimum is approached; this can be very effective, as advanced in references [25] and [26]. Finally, in reference [27], a number of possibilities are defined that are intermediate between the extremes of GRG(full flow convergence per optimization iteration) and reduced SQP(one state equation Newton step per optimization iteration).

We imagine that for larger problems, the cost of factoring the state equation Jacobian matrix will begin to display its asymptotic behavior and dominate the lower order terms, leading to increasing efficiency of the Newton method relative to quasi-Newton. However, problem size cannot continue to grow indefinitely and allow sparse LU factorization.

For the largest problem, parallel computing will become essential. Of course, there is a long history of parallel algorithms and implementations for the forward problem, i.e., Navier-Stokes flow simulation. *Algorithm-1* and *Algorithm-2* are well suited to parallel machines, since the majority of their work involves solution of linear systems having state equation Jacobians as their coefficient matrix; this is just a step of the forward problem, the parallelization of which is well-understood.

References

1. Prandtl, L. and Tietjens,, O. G., "Applied Hydro-and Aeromechanics", Dover, New York, 1934, p.81
2. Gad-el-Hak, M., "Flow control", *Appl. Mech. Rev.* Vol. 42, No. 261, 1989
3. Moin, P. and Bewley, T., "Feedback control of turbulence", *Appl. Mech. Rev.* Vol. 47, 1994
4. Gunzburger, M. D., Hou, L. S. and Svobodny, T.P., *Optimal control and optimization of viscous, incompressible flows, in Incompressible Computational Fluid Dynamics*, edited by M. D. Gunzburger and R. A, Nicolaides, Cambridge Univ. Press, Cambridge, UK, 1993
5. 박재형, 홍순조, "Navier-Stokes 유체의 최적 제어", 한국전산구조공학회, 제15권, 제4호, 2002, pp.661~674
6. Gill, P. E., Murray, W. and Wright, M. H., *Practical Optimization*, Academic Press, New York, 1981
7. Khaira, M. S., Miller, G. L. and Sheffler, T. J., "Nested Dissection : A Survey and Comparison of Various Nested Dissection Algorithms", Tech. Rep. CMU-CS-92-106R, Carnegie Mellon University, 1992
8. Farthat, C. and Chen, P. S., Tailoring domain decomposition methods for efficient parallel coarse grid solution and for system with many right hand sides, in Domain Decomposition Methods in Science and Engineering, Contemporary Mathematics, *American Mathematical Society*, Providence, RI, 1994
9. Kupfer, F. S. and Sachs, E. W., "A prospective look at SQP methods for semilinear parabolic control problem, in Optimal control of partial Differential Equations", edited by K Hoffman and W. Krabs, Springer-Verlag, Berlin/New York, 1991, p.145
10. Ringerz, U., "Optimal Design of Nonlinear Shell structure", Tech. Rep, FFA TN 91-18, The Aeronautical Research Institute of Sweden, 1991
11. Ringerz, U., "An algorithm for optimization of nonlinear shell structure", *Int. F. Numer. Methods Eng.* Vol. 38, No. 299, 1995.
12. Orozoc, C. E. and Ghattas, O., "Massively parallel aerodynamic shape optimization", *Comput. Syst. Eng.*, 1992
13. Orozco, C. E. and Ghattas, O., "Optimal design of systems governed by nonlinear partial differential equation", in Forth AIAA/USAF/NASA/OAI Symposium on Multidisciplinary Analysis and Optimization, AIAA, p.1126, 1992
14. Biegler, L. T., Nocedal, J. and Schmid, C., "A reduced Hessian method for large-scale constrained optimization", *SIAM. J.* Optim. Vol. 5, No. 314, 1995
15. Haftka, R. T., Gurdal, Z. and Kamat, M. P., "Elements of Structural Optimization", Kluwer Academic, Dordercht/Norwall, MA, 1990
16. Haug, E. J. and Arora, J. S., *Applied Optimal Design*, Wiley-Interscience, New York, 1979
17. Gunzburger, M. D., *Finite Element Methods for Viscous Incompressible Flow*, Academic Press, San Diego, 1989
18. Hugher, T. J. R., Liu, W. K., and Brooks, A., "Finite element analysis of incompressible viscous flow by the penalty function formulation", *J. Comput. Phys.* Vol. 30, No. 1, 1979

19. Davis, T. A., "Users' Guide for the Unsymmetric Pattern Multifrontal Package (UMFPACK)", Tech. Rep. TR-93-020, CIS Dept., University of Florida, Gainesville, FL, 1993
20. Davis, T. A., and Duff, I. S., "An Unsymmetic Pattern Multifrontal Method for Sparse LU Factorization", Tech. Rep. TR-93-018, CIS Dep., University of Florida, Gainesville, FL, 1993
21. Gunzburger, M. D., "Flow Control", IMA Volumes in Mathematics and Its Applications, Vol. 68, Springer-Verlag, Berlin/New York, 1995
22. Gabay, D., "Reduced Quasi-Newton methods with feasibility improvement for nonlinearly constrained optimization", *Math, Programming Study* Vol. 16, No. 18, 1982
23. Xie, Y., "Reduced Hessian Algorithms for solving Large-Scale Equality Constrained Optimization Problem", Ph. D. thesis, University of Cololado, Boulder, Department of Computer Science, 1991
24. Batchelor, G. K., *An introduction to Fluid Dynamics*, Cambridge Univ. Press, Cambridge, UK, 1967
25. Huffman, W. P., Melvin, R. G., Young, D. P., Johson, F. T., Bussoletti, J. E., Bieterman, M. B. and Hilmes, C. L., *Practical design and optimization in computation fluid dynamics*, in Proceedings of 24th AIAA Fluid Dynamics Conference, Orlando, Florda, 1993
26. Young, D. P., Huffman, W. P., Melvin, R. G., Bieterman, M. B., Hilmes, C. L. and Jhonson, F. T., "Inexactness and global convergence in design optimization", in proceedings of the 5th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Panama City, Florida, 1994
27. Cramer, E. J., Dennis, J. E., Frank, P. D., Lewis, R. M. and Shubin, G. R., "Problem formulation for multidisciplinary optimization", *SIAM J.* Optim. Vol. 4, No. 754, 1994