COMPUTATIONAL STRATEGIES FOR SHAPE OPTIMIZATION OF TIME-DEPENDENT NAVIER-STOKES FLOWS*

BEICHANG HE[†], OMAR GHATTAS[‡], AND JAMES F. ANTAKI[§]

Abstract. We consider the problem of shape optimization of two-dimensional flows governed by the time-dependent Navier-Stokes equations. For this problem we propose computational strategies with respect to optimization method, sensitivity method, and unstructured meshing scheme. We argue that, despite their superiority for steady Navier-Stokes flow optimization, reduced sequential quadratic programming (RSQP) methods are too memory-intensive for the time-dependent problem. Instead, we advocate a combination of generalized reduced gradients (for the flow equation constraints) and SQP (for the remaining inequality constraints). With respect to sensitivity method, we favor discrete sensitivities, which can be implemented with little additional storage or work beyond that required for solution of the flow equations, and thus possess a distinct advantage over discretized continuous sensitivities, which require knowledge of the entire time history of the flow variables. Finally, we take a two-phase approach to unstructured meshing and grid sensitivities. Far from the optimum, we remesh each new shape completely using an unstructured mesh generator to accommodate the large shape changes that are anticipated in this phase, while an inconsistent but easily computed form of grid sensitivities is employed. Close to the optimum, where differentiability of the mesh movement scheme and consistency of grid sensitivities are desirable, we use elastic mesh movement to generate meshes corresponding to new shapes. Elastic mesh movement is valid only for small shape changes but is differentiable and permits computation of exact grid sensitivities in a straightforward manner. Two examples characterized by a viscous dissipation objective function illustrate the approach.

1. Introduction. The problem of finding the optimal design of a system governed by the incompressible Navier-Stokes equations arises in many design problems in aerospace, automotive, hydraulic, ocean, structural, and wind engineering. Example applications include aerodynamic design of automotive vehicles, trains, low speed aircraft, sails, and flexible structures, and hydrodynamic design of ship hulls, propellers, turbomachinery, and offshore structures. In many cases, the flow equations do not admit steady-state solutions, and the optimization model must incorporate the time-dependent form of the Navier-Stokes equations.

Most of the theoretical and numerical work on optimization of Navier-Stokes flows has been done in the context of both *optimal control* and *steady* flows (see [17] for an overview). A few studies have considered optimal control of *time-dependent* flows, where the control is in the form of boundary velocities [23] [41]. Our concern in this article is on *shape optimization* of time-dependent Navier-Stokes flows, and in particular on devising efficient numerical strategies for solution of two-dimensional (2D) realizations of such problems. Our ultimate goal is to solve three-dimensional (3D) shape optimization problems arising in the design of artificial heart devices [2] [8].

*Technical Report CMU–CML–97–102, June, 1997. This work was supported in part by the Engineering Design Research Center, an Engineering Research Center of the National Science Foundation, under Grant No. EEC-8943164, and by Algor, Inc. Computations were performed on computers purchased with funds provided in part by NSF equipment grant BCS-9212819.

[†]Engineering Mechanics Laboratory, General Electric Company, Niskayuna, NY 12309 (heb@crd.ge.com).

[‡]Computational Mechanics Laboratory, Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, 15213, USA (oghattas@cs.cmu.edu).

[§]Artificial Heart Program, Department of Surgery, University of Pittsburgh Medical Center, Pittsburgh, PA 15213, USA (antaki@pittsurg.nb.upmc.edu).

One of the major difficulties in numerically solving optimization problems governed by time-dependent Navier-Stokes flows is the large number of equality constraints that arise upon space and time discretization of the flow equations. The size of this constraint set is the number of spatial flow variables multiplied by the number of time steps, and can be of the order of millions for typical 2D problems, such as those solved in this paper; several orders of magnitude larger can be expected in 3D. This puts the time-dependent Navier-Stokes optimization problem in the category of extremely large scale, nonlinearly-constrained optimization. In general, problems with numbers of constraints as large as these pose difficulties for such modern nonlinear optimization methods as the sequential quadratic programming (SQP) method.

One approach common in optimal design for accommodating large constraints sets arising from discretized partial differential equations (PDEs) is to eliminate the state equations and state variables at each optimization iteration. This is done by solving the state equations for the states variables given values of the design variables. The state variables are then used to evaluate the objective function and remaining constraints, and the implicit function theorem is invoked for derivative computations. See, for example, [18] [19]. This method therefore treats the equality constraints in a generalized reduced gradient (GRG) fashion, since it satisfies them exactly at each optimization iteration. The remaining constraints are then treated using one's favorite nonlinear optimizer. The GRG idea greatly reduces the size of the optimization problem, since it is now of dimension of the design variables, and relieves the optimizer from its role as PDE solver. On the other hand, the disadvantage of this method is that the state equations must be completely solved for a given set of design variables; this can be quite onerous when the state equations are highly nonlinear.

A different approach that is particularly effective for optimization problems governed by nonlinear boundary value problems has emerged in recent years. The state equations are retained as constraints, and the optimization problem is solved by a *reduced* SQP (RSQP) method. Two key ideas are: unlike GRG (but like full-space SQP), only a *linear approximation* of the state equations is satisfied at each iteration; and unlike full space SQP (but like GRG), curvature information is required only in a subspace defined by the null space of the Jacobian of the state equations, which is of the dimension of the design variable vector, and is thus typically small relative to the number of state variables. When second derivatives are difficult to compute, as is often the case in optimal design, this reduced Hessian can be approximated cheaply by a quasi-Newton update. In a sense, RSQP combines the best of GRG and full-space SQP. Indeed, there is an intimate connection between GRG and RSQP for optimization problems governed by state equations. The two can be seen roughly as extremes of a continuum, the endpoints of which are: completely converging the state equations at each optimization iteration versus performing a single Newton step on them (see [28]). The efficacy of RSQP for PDE-constrained optimization problems has been demonstrated in a number of applications including structural optimization [35] [39] [40], heat equation boundary control [26], compressible flow airfoil design [33] [34] [43], boundary control of viscous incompressible flows [14] [21], and inverse parameter estimation problems [11] [24]. Reduced SQP methods have been analyzed for the finite dimensional case in [5] [9] [32], and for infinite dimensions in [22] [25].

In light of its vastly superior performance over GRG (for steady optimal boundary velocity control [14]), our initial idea was to use RSQP for solution of the time-dependent Navier-Stokes shape optimization problem. However, the large size of state constraints in the form of the spatio-temporally discretized Navier-Stokes equations forced us to rethink that desire. In exchange for not completely solving the state constraints at each time step—instead satisfying the flow equations asymptotically as the optimum is approached—RSQP requires storage of the entire time history of flow variables. This is a serious drawback. Indeed, the very advantage of SQP has turned out to result in a disadvantage: simultaneous storage of flow variables at all time steps is needed precisely because we are permitted to update them at each optimization iteration—with the goal of eventually converging the nonlinear algebraic equations characterizing a time step. GRG does not suffer from this storage problem: because the time dimension represents the "initial" part of the initial-boundary value problem, the states evolve through time, and a numerical algorithm for time-stepping simply replaces one time step's flow variables with the next. The advantage of RSQP in this context is the reduction in computational effort per optimization iteration, by avoiding solution of the flow equations in favor of a single Newton step that updates them. Here, however, the flow equations are not terribly nonlinear; since time steps are chosen for time-accuracy, it is usually the case that the flow equations for a given time step can be solved for the flow variables at the next time step, given the current flow variables, in just two or three Newton iterations (provided the flow is away from turning or bifurcation points). So time-accurate integration of the spatially-discretized flow equations—as necessitated by the time-dependent nature of the objective and inequality constraints—has rendered the fully-discretized state equations only weakly nonlinear, and RSQP's contribution is limited to reducing those two or three Newton steps to just one. Thus, the benefit of RSQP relative to GRG for time-dependent flow optimization is not as great as the steady case, in which over an order of magnitude improvement has been observed [14].

Instead, we pursue here a combination of SQP and GRG. At each optimization iteration, we solve the equality state constraints (i.e. the spatio-temporally discretized Navier-Stokes equations) completely, by stepping sequentially through time, obtaining the flow variables at the next time step given those of the current. Sensitivity derivatives are found discretely also at each time step, again obviating the need to store the entire time history of flow variables, as would be required with discretized continuous sensitivities. Objective and constraint functions defined as integrals over time are simply accumulated step-by-step using a quadrature rule, as are their derivatives. In exchange for a big reduction in storage, we do some more work. This is the GRG component of the method. The treatment of the remaining, inequality, constraints is in the fashion of SQP; only a linear approximation of the active constraints is satisfied at any optimization iteration. The storage thus required can be as little as the current and previous flow field (if a matrix-free method is used to solve the state equations). We stress that this approach to state equation-constrained optimization problems is not novel: indeed this is the usual way design optimization problems are solved, and is what is usually referred to in the engineering literature as an "SQP method."

Several caveats are in order here. First, if the nonlinear iteration used to solve the flow equations at each time step is not a true Newton method (i.e. it doesn't compute the exact Jacobian), it may require many more iterations at each time step, and so the reduction to a single linear solve of the state equations offered by RSQP may result in a substantial reduction in computational effort. Second, this situation is an example of the classic time–memory tradeoff. For us, storage wins out, since unstructured mesh methods are already very memory intensive. However, when memory is not a problem, RSQP's offer of a reduction to just one Newton step on the state equations per optimization iteration is certainly worthwhile. For 1D initial-boundary problems (i.e., when the PDEs are posed in one spatial dimension, and time represents the other), the storage associated with discretizing the time-space continuum is not onerous, and there is no reason not to use a RSQP method; see [26] for an application to boundary control involving nonlinear heat conduction. Another example of memory being less important than computational work is when the state equations are naturally discrete in "space," and are posed as (ordinary) differential–algebraic equations in time. In this case, RSQP is also a good idea. An example application is optimal control of robots, the behavior of which is governed by rigid multibody dynamics. [43].

In the remainder of this article, we will define the continuous optimization problem, discuss it spatial and temporal discretization, develop sensitivity expressions, consider grid sensitivities, and apply the method to two time-dependent Navier-Stokes shape optimization model problems. More extensive discussion can be found in [20].

2. Mathematical model. In this section we state the continuous form of the mathematical model of the shape optimization problem addressed in this paper. Let Ω denote an open, bounded, and possibly multiply connected domain in $\Re^2$ or $\Re^3$. Let Γ denote the boundary of Ω. The laminar time-dependent flow of a homogeneous, incompressible, Newtonian fluid inside Ω is governed by the Navier-Stokes equations, i.e. the conservation of linear momentum equation

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} - \nabla\cdot\boldsymbol{\sigma} - \mathbf{f} = \mathbf{0} \qquad \text{in } \Omega\times[0,T]\,, \tag{2.1}$$

the constitutive law

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mu\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) \qquad \text{in } \Omega\times[0,T]\,, \tag{2.2}$$

and the conservation of mass equation

$$\nabla\cdot\mathbf{v} = 0 \qquad \text{in } \Omega\times[0,T]\,, \tag{2.3}$$

where ∇ is the spatial gradient vector, $\mathbf{v}$ the fluid velocity, $\boldsymbol{\sigma}$ the stress tensor, p the pressure, $\mathbf{f}$ the body force, ρ the fluid density, and μ the dynamic viscosity.

The initial-boundary value problem requires the imposition of boundary conditions, which may consist of the Dirichlet condition

$$\mathbf{v}(\mathbf{x},t) = \mathbf{v}_d(t) \qquad \text{in } \Gamma_d\times[0,T]\,, \tag{2.4}$$

and the Neumann condition

$$\boldsymbol{\sigma}(\mathbf{x},t)\cdot\mathbf{n}(\mathbf{x}) = \mathbf{t}_n(t) \qquad \text{in } \Gamma_n\times[0,T]\,, \tag{2.5}$$

as well as an initial condition

$$\mathbf{v}(\mathbf{x},0) = \mathbf{v}_0(\mathbf{x}) \qquad \text{in } \Omega\,, \tag{2.6}$$

where $\mathbf{x}$ is the spatial coordinate, $\mathbf{n}(\mathbf{x})$ the outward normal unit vector at Γ_n, and

$$\Gamma_d\cup\Gamma_n = \Gamma\,. \tag{2.7}$$

We consider the following *local* (defined at a point in space and an instant in time) and *global* (integrated over the problem domain and time interval of interest) functions as objective (or constraint) functions in the optimization problem: the rate of energy dissipation due to viscosity,

$$\Phi = \frac{1}{2}\mu \left[\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right] : \left[\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right] \,, \tag{2.8}$$

the maximum shear stress, e.g. in two dimensions,

$$\tau = \sqrt{\left(\frac{\sigma_{xx} - \sigma_{yy}}{2}\right)^2 + \sigma_{xy}^2} \,, \tag{2.9}$$

the magnitude of vorticity,

$$\varpi = |\ \nabla \times \mathbf{v}\ | \,, \tag{2.10}$$

the spatially and temporally averaged dissipation function, both unscaled and scaled by area or volume of the domain,

$$\widehat{\Phi}_u = \int_{t_L}^{t_M} \int_\Omega \Phi \; d\Omega dt \,, \qquad \widehat{\Phi}_s = \frac{\widehat{\Phi}_u}{S} \,, \tag{2.11}$$

the spatially and temporally averaged maximum shear stress, both unscaled and scaled,

$$\widehat{\tau}_u = \int_{t_L}^{t_M} \int_\Omega \tau^2 \; d\Omega dt \,, \qquad \widehat{\tau}_s = \frac{\widehat{\tau}_u}{S} \,, \tag{2.12}$$

and the spatially and temporally averaged vorticity magnitude, both unscaled and scaled,

$$\widehat{\varpi}_u = \int_{t_L}^{t_M} \int_\Omega \varpi^2 \; d\Omega dt \,, \qquad \widehat{\varpi}_s = \frac{\widehat{\varpi}_u}{S} \,, \tag{2.13}$$

where

$$S = \int_\Omega d\Omega$$

is the area or volume of the flow domain Ω, and $[t_L, t_M]$ is a characteristic time interval. Definitions of t_L and t_M are problem dependent. For instance, if the optimization problem of interest involves flow with initial-condition-generated transients that are damped out, the characteristic time interval may be chosen to start at the initial time, and end after steady state is reached. On the other hand, if the fluid motion is periodic and the influence of the initial condition is irrelevant to the design problem, we may set t_L to be large enough so that viscous damping filters out the contribution of the initial condition, and $[t_L, t_M]$ should span one or several periods of flow oscillation. These functions are inspired directly or indirectly by the desire to avoid blood damage by thrombosis and hemolysis in artificial heart devices [2].

Later in this article we describe a code we have developed that implements any convex combination of the six global functions in Equations (2.11)–(2.13) as the objective function. In addition, limits on allowable values of the local functions (2.8)–(2.10) may be introduced as inequality constraints. Conditions that ensure a valid shape and limit the size and position of the shape have to be introduced as geometric constraints. We will discuss such geometric constraints in Sections 5 and 6 in the context of some specific geometries. An example shape optimization problem that might arise in artificial heart design is to minimize the global dissipation $\widehat{\Phi}$ in (2.11), subject to geometry and physics constraints, including equalities in the form of the Navier-Stokes equations (2.1)—(2.3), and inequalities induced by demanding that the local maximum shear stress (2.9) not exceed an allowable value throughout the domain. Since the aspect of the design that is under our control is the domain boundary (or part of it), this problem is one of shape optimization.

Since it is in general impossible to solve the infinite dimensional shape optimization problem in closed form, we resort to numerical approximation in both space and time, as well as representing the shape by a finite number of design parameters. These issues will be addressed in subsequent sections. The resulting optimization problem can then be stated as finding, from the space of shapes spanned by the design variables, the one that minimizes a numerical approximation of the objective while satisfying the discrete time-dependent Navier-Stokes equations, as well as discretized geometric and flow-related inequality constraints. The problem can thus be transformed into a smooth, large-scale, nonlinear programming problem (NLP). In the next section we discuss spatial and temporal discretization of the flow equations and functions of interest.

3. Spatial and temporal discretization. We use finite elements in space and finite differences in time to numerically approximate the flow equations. For spatial approximation, we choose the Galerkin finite element method, which begins with a weak form of the equations. Let $H^1(\Omega \times [0,T])$ denote the Sobolev space of all functions whose first derivatives are square integrable over $\Omega \times [0,T]$ and let $L^2(\Omega \times [0,T])$ denote the space of functions that are square integrable over $\Omega \times [0,T]$. We restrict ourselves to the following subspaces,

$$\mathcal{V} = \{\mathbf{v} \mid \mathbf{v} \in [H^1(\Omega \times [0,T])]^{dim} \text{ and } \mathbf{v} \text{ satisfies (2.4)}\},$$

$$\mathcal{W} = \{\mathbf{w} \mid \mathbf{w} \in [H^1(\Omega \times [0,T])]^{dim} \text{ and } \mathbf{w} = \mathbf{0} \text{ on } \Gamma_d \times [0,T]\},$$

and

$$\mathcal{P} = \{p \mid p \in L^2(\Omega \times [0,T])\},$$

where $dim = 2$ and 3 for two and three dimensional problems, respectively.

Stress variables are eliminated from the flow equations by substituting (2.2) into (2.1). The weak form of (2.1) and (2.3) can then by stated as: Find $\mathbf{v} \in \mathcal{V}$ and $p \in \mathcal{P}$ satisfying the initial conditions (2.6), such that

$$h(\frac{\partial \mathbf{v}}{\partial t}, \mathbf{w}) + a(\mathbf{v}, \mathbf{w}) + b(p, \mathbf{w}) + c(\mathbf{v}, \mathbf{v}, \mathbf{w}) + g(\mathbf{w}) = \mathbf{0} \quad \text{for all } \mathbf{w} \in \mathcal{W}(\Omega), \tag{3.1}$$

$$b(q, \mathbf{v}) = 0 \qquad \text{for all } q \in \mathcal{P}, \tag{3.2}$$

where

$$h\left(\frac{\partial \mathbf{v}}{\partial t}, \mathbf{w}\right) = \int_\Omega \rho \, \frac{\partial \mathbf{v}}{\partial t} \cdot \mathbf{w} \, d\Omega, \tag{3.3}$$

$$a(\mathbf{v},\mathbf{w}) = \int_\Omega \frac{\mu}{2}\left(\boldsymbol{\nabla}\mathbf{v} + (\boldsymbol{\nabla}\mathbf{v})^T\right) : \left(\boldsymbol{\nabla}\mathbf{w} + (\boldsymbol{\nabla}\mathbf{w})^T\right)\, d\Omega\,, \tag{3.4}$$

$$b(p,\mathbf{w}) = -\int_\Omega p\boldsymbol{\nabla}\cdot\mathbf{w}\; d\Omega\,, \tag{3.5}$$

$$c(\mathbf{v},\mathbf{v},\mathbf{w}) = \int_\Omega \rho\,\mathbf{w}\cdot(\mathbf{v}\cdot\boldsymbol{\nabla})\,\mathbf{v}\; d\Omega\,, \tag{3.6}$$

and

$$g(\mathbf{w}) = -\int_\Omega \mathbf{f}\cdot\mathbf{w}\; d\Omega - \int_{\Gamma_n} \mathbf{t}_n\cdot\mathbf{w}\; d\Gamma_n\,. \tag{3.7}$$

Since the weak formulation explicitly involves traction on the boundary, (2.5) is easily enforced.

Galerkin finite element approximation begins by meshing the flow domain Ω with nodes and elements. Finite element spaces $\mathcal{V}_h$, $\mathcal{W}_h$, and $\mathcal{P}_h$, which are subspaces of $\mathcal{V}$, $\mathcal{W}$, and $\mathcal{P}$, respectively, can then be established as

$$\mathcal{V}_h = \{\mathbf{v}_h \mid \mathbf{v}_h = \sum_{i=1}^{N^v} \phi_i\,\mathbf{v}_i\,,\ \text{and } \mathbf{v}_h \text{ satisfies (2.4)}\}\,, \tag{3.8}$$

$$\mathcal{W}_h = \{\mathbf{w}_h \mid \mathbf{w}_h = \sum_{i=1}^{N^v} \phi_i\,\mathbf{w}_i\,,\ \text{and } \mathbf{w}_h = \mathbf{0} \text{ on } \Gamma_d\times[0,T]\}\,, \tag{3.9}$$

and

$$\mathcal{P} = \{p_h \mid p_h = \sum_{j=1}^{N^p} \psi_j\, p_j\}\,, \tag{3.10}$$

where N^v and N^p are the number of nodes for, respectively, velocity and pressure unknowns; the basis functions $\phi_1,\ldots,\phi_{N^v}$ and $\psi_1,\ldots,\psi_{N^p}$ are continuous piecewise polynomials; and $\mathbf{v}_i \in \Re^{dim}$, $\mathbf{w}_i \in \Re^{dim}$, and $p_j \in \Re$. Since the ϕ_i and ψ_j are finite element basis functions, $\mathbf{v}_i$ and p_j are just velocity and pressure values at nodes i and j, respectively.

In the case of isoparametric finite elements, the transformation between the spatial coordinate $\mathbf{x}$ and the curvilinear coordinate $\boldsymbol{\xi}$ is given by

$$\mathbf{x} = \sum_{k=1}^{N^v} \phi_k(\boldsymbol{\xi})\,\mathbf{x}_k\,, \tag{3.11}$$

where $\mathbf{x}_k$ is the spatial coordinate at node k. The Jacobian matrix and determinant of the transformation are

$$\mathbf{J} = \left(\frac{\partial\mathbf{x}}{\partial\boldsymbol{\xi}}\right)^T \in \Re^{dim\times dim}\,, \tag{3.12}$$

$$J = det(\mathbf{J}) \in \Re\,. \tag{3.13}$$

Note that the curvilinear coordinate $\boldsymbol{\xi}$ is defined locally with respect to each element, so that each element has its own coordinate transformation. When we refer to $\boldsymbol{\xi}$ globally, what we have in mind is the local curvilinear coordinate system corresponding to the particular element that is referenced.

Making use of (3.8)–(3.13) for spatial discretization of (3.1)–(3.2), we obtain a system of ordinary differential-algebraic equations (DAEs) in time t,

$$\sum_{j=1}^{N^v} \mathbf{M}_{ij} \frac{\partial \mathbf{v}_j}{\partial t} + \sum_{j=1}^{N^v} \mathbf{A}_{ij} \mathbf{v}_j + \sum_{k=1}^{N^p} \mathbf{B}_{ik}^T p_k + \mathbf{f}_i = \mathbf{0} \qquad i = 1, \ldots, N^v, \tag{3.14}$$

$$\sum_{j=1}^{N^v} \mathbf{B}_{kj} \mathbf{v}_j = 0 \qquad k = 1, \ldots, N^p, \tag{3.15}$$

where

$$\begin{aligned}
\mathbf{M}_{ij} &= \int_{\Omega^\xi} \rho \phi_i \phi_j \, \mathbf{I} \, J \, d\Omega^\xi \,, \\
\mathbf{A}_{ij} &= \int_{\Omega^\xi} \mu \left[\mathbf{J}^{-T} \, (\nabla^\xi \phi_j)(\nabla^\xi \phi_i)^T \, \mathbf{J}^{-1} + (\nabla^\xi \phi_i)^T \, \mathbf{J}^{-1} \, \mathbf{J}^{-T} \, (\nabla^\xi \phi_j) \, \mathbf{I} \right] J \, d\Omega^\xi \\
&\quad + \int_{\Omega^\xi} \rho \phi_i \, (\nabla^\xi \phi_j)^T \, \mathbf{J}^{-1} \, \mathbf{v}_h \, \mathbf{I} \, J \, d\Omega^\xi \,, \\
\mathbf{B}_{kj} &= - \int_{\Omega^\xi} \psi_k \, (\nabla^\xi \phi_j)^T \, \mathbf{J}^{-1} \, J \, d\Omega^\xi \,, \\
\mathbf{f}_i &= - \int_{\Omega^\xi} \mathbf{f} \, J \, d\Omega^\xi - \int_{\Gamma_n^\xi} \phi_i \, \mathbf{t}_n \, \mathbf{n}^T \, \mathbf{J}^{-T} \, \mathbf{n}^\xi \, J \, d\Gamma^\xi \,.
\end{aligned}$$

Here, ∇^ξ is the gradient vector in the curvilinear coordinate space; Ω^ξ and Γ^ξ are the images in curvilinear coordinate space of, respectively, Ω and Γ in physical space; and $\mathbf{n}^\xi$ and $\mathbf{n}$ are the outward normal unit vectors at the boundaries in curvilinear and physical coordinate spaces, respectively. All of these symbols are defined element-by-element.

Let $0 = t_0 < \ldots < t_L < \ldots < t_M = T$ be a uniform partition in the time dimension. Equations (3.14) and (3.15) are transformed into a system of algebraic equations by finite differencing in time. Consider a family of single-step methods,

$$\begin{aligned}
&\sum_{j=1}^{N^v} \mathbf{M}_{ij} \, \frac{\mathbf{v}_j^{m+1} - \mathbf{v}_j^m}{\Delta t} + \theta \left(\sum_{j=1}^{N^v} \mathbf{A}_{ij}^{m+1} \mathbf{v}_j^{m+1} + \sum_{k=1}^{N^p} \mathbf{B}_{ik}^T p_k^{m+1} + \mathbf{f}_i^{m+1} \right) \\
&+ (1-\theta) \left(\sum_{j=1}^{N^v} \mathbf{A}_{ij}^m \mathbf{v}_j^m + \sum_{k=1}^{N^v} \mathbf{B}_{ik}^T p_k^m + \mathbf{f}_i^m \right) \; = \; \mathbf{0} \qquad i = 1, \ldots, N^v \,,
\end{aligned} \tag{3.16}$$

$$\sum_{j=1}^{N^v} \mathbf{B}_{kj} \left[\theta \mathbf{v}_j^{m+1} + (1-\theta) \mathbf{v}_j^m \right] = 0 \qquad k = 1, \ldots, N^p \,, \tag{3.17}$$

where $\Delta t = t_{m+1} - t_m$; $\mathbf{v}_j^m$, p_k^m, $\mathbf{f}_i^m$, and $\mathbf{A}_{ij}^m$ are values of $\mathbf{v}_j$, p_k, $\mathbf{f}_i$, and $\mathbf{A}_{ij}$ at time t_m, respectively; and $0 \leq \theta \leq 1$ is an algorithm parameter. The Crank-Nicholson scheme, which is second-order accurate in time, is obtained by setting $\theta = 0.5$. First-order accurate Euler forward and backward schemes are obtained with $\theta = 0$ and 1, respectively. The scheme is implicit whenever $\theta \neq 0$. Equations (3.16) and (3.17) constitute a set of coupled nonlinear ($\theta \neq 0$) algebraic equations with unknowns $\mathbf{v}_i^{m+1}$ and p_j^{m+1}, which can be solved given velocity field $\mathbf{v}_i^m$ and pressure field p_j^m at $t = t_m$. Upon completion of this step we march forward in time, solving for flow variables at time t_{m+2}, and so on.

The spatio-temporally discretized Navier-Stokes equations (3.16)–(3.17) can be solved at each time step by Newton's method, which yields the linear system

$$\sum_{j=1}^{N^v} \left(\frac{1}{\Delta t} \mathbf{M}_{ij} + \theta\, \mathbf{A}_{ij}^{m+1} + \theta\, \mathbf{C}_{ij}^{m+1} \right) \Delta \mathbf{v}_j^{m+1} + \theta \sum_{k=1}^{N^p} \mathbf{B}_{ik}^T\, \Delta p_k^{m+1} = -\mathbf{h}_i^v \qquad i = 1, \ldots, N^v, \tag{3.18}$$

$$\theta \sum_{j=1}^{N^v} \mathbf{B}_{kj}\, \Delta \mathbf{v}_j^{m+1} = -h_k^p \qquad k = 1, \ldots, N^p, \tag{3.19}$$

where the "momentum residual" $\mathbf{h}_i^v$ is the lefthand side of (3.16), the "mass residual" h_k^p is the lefthand side of (3.17), and

$$\mathbf{C}_{ij} = \int_{\Omega^\xi} \rho\, \phi_i \phi_j \sum_{k=1}^{N^v} \mathbf{v}_k\, \left(\nabla^\xi \phi_k\right)^T\, \mathbf{J}^{-1}\, J\, d\Omega^\xi\,.$$

The Newton step (3.18)–(3.19) is a set of nonsymmetric linear equations that, given current values of the the pair $(\mathbf{v}_j^{m+1}, p_k^{m+1})$, can be solved for the increment in velocity $\Delta \mathbf{v}_i^{m+1}$ and the increment in pressure Δp_j^{m+1}. In our 2D implementation, we use a sparse direct method to solve this linear system; in 3D one would probably want to use a preconditioned Krylov subspace method. Once computed, the increments are used to update the estimate of velocity and pressure at time t_{m+1} according to

$$\mathbf{v}_i^{m+1} \leftarrow \mathbf{v}_i^{m+1} + \Delta \mathbf{v}_i^{m+1} \qquad i = 1, \ldots, N^v,$$

$$p_j^{m+1} \leftarrow p_j^{m+1} + \Delta p_j^{m+1} \qquad j = 1, \ldots, N^p\,.$$

The coefficient matrix and righthand side of the Newton step (3.18)–(3.19) are then reevaluated with the updated values of $\mathbf{v}_i^{m+1}$ and p_j^{m+1}, and the process is iterated until the nonlinear equations (3.16)–(3.17) are satisfied to within a tolerance.

Flow functions (2.8)–(2.13) are also spatially discretized using the finite element spaces $\mathcal{V}_h$ and $\mathcal{P}_h$, and time integration is performed using the trapezoidal rule, which is second-order accurate. For instance, the viscous energy dissipation in (2.8), as a function of space and time, is approximated by

$$\Phi = \frac{1}{2}\mu \sum_{i=1}^{N^v} \sum_{j=1}^{N^v} \left[\mathbf{J}^{-T}(\nabla^\xi \phi_i)\mathbf{v}_i^T + \mathbf{v}_i(\nabla^\xi \phi_i)^T \mathbf{J}^{-1}\right] : \left[\mathbf{J}^{-T}(\nabla^\xi \phi_j)\mathbf{v}_j^T + \mathbf{v}_j(\nabla^\xi \phi_j)^T \mathbf{J}^{-1}\right], \tag{3.20}$$

and the spatially and temporally integrated dissipation functions in (2.11) are discretized as

$$\widehat{\Phi}_u = \frac{1}{2}\, \Delta t \sum_{m=L}^{M-1} \left(\int_{\Omega^\xi} \Phi_u^m J d\Omega^\xi + \int_{\Omega^\xi} \Phi_u^{m+1} J d\Omega^\xi \right), \quad \widehat{\Phi}_s = \frac{\widehat{\Phi}_u}{\int_{\Omega^\xi} J d\Omega^\xi}. \tag{3.21}$$

The global flow functions in (3.21) depend on a sequence of velocity fields $\mathbf{v}_i^L, \ldots, \mathbf{v}_i^M$. It appears necessary to store all flow variables within the time interval $[t_L, t_M]$. However, we can avoid storing the flow history due to the fact that we are using a one-step method to integrate the DAEs and a one-step integration rule to evaluate flow functions. In this case the flow functions in (3.21) are accumulated at each time step rather than after the complete flow solution; hence only the current and previous velocity fields are needed.

Spatial and temporal discretization now renders the flow constraints and variables finite dimensional. As will be illustrated in Sections 5 and 6, the shape of the flow domain will be parameterized with a finite number of design variables. The shape optimization problem may then be represented as follows:

$$\text{minimize} \quad \widehat{\Psi}\left(\mathbf{u}^L, \ldots, \mathbf{u}^M, \boldsymbol{\pi}\right) , \tag{3.22}$$

$$\begin{aligned} \text{subject to} \quad \mathbf{h}\left(\mathbf{u}^m, \mathbf{u}^{m+1}, \boldsymbol{\pi}\right) &= \mathbf{0} \qquad m = 0, \ldots, L, \ldots, M-1 \,, \\ \mathbf{r}\left(\mathbf{u}^m, \boldsymbol{\pi}\right) &\geq \mathbf{0} \qquad m = L, \ldots, M \,, \\ \mathbf{q}\left(\boldsymbol{\pi}\right) &\geq \mathbf{0} \end{aligned}$$

where $\mathbf{u}^m$ denotes the vector of nodal velocities and pressures at the m-th time step, $\boldsymbol{\pi}$ the vector of design variables, $\widehat{\Psi}$ any convex combination of the global flow functions in (2.11)–(2.13), $\mathbf{h}$ the system of nonlinear algebraic equations representing the spatio-temporally discretized Navier-Stokes equations (3.16)–(3.17), $\mathbf{r}$ the vector representing the difference between the allowable and actual values of the local flow functions in (2.8)–(2.10) at time m, and $\mathbf{q}$ the vector representing the geometry constraints, i.e. those depending solely on the design, and not flow, variables.

4. Sensitivity analysis. As discussed in the introduction, we use a GRG treatment of the flow equations $\mathbf{h}{=}\mathbf{0}$, and an SQP treatment of the remaining inequality constraints to solve the NLP problem (3.22). That is, the flow equations and variables are eliminated from the optimization problem by sequentially stepping through time and solving the nonlinear algebraic equations (3.16)–(3.17) that arise at each time step. The values of the flow variables computed at each time step are used to both evaluate the (active) inequality constraints $\mathbf{r}^m$ as well as to contribute to the evaluation of the objective function. The gradient of the objective and the Jacobian of the inequality constraints can then be found through the implicit function theorem. This generates a small, dense constraint Jacobian matrix (and a small, dense Hessian matrix of the Lagrangian function). Thus, standard dense SQP methods are appropriate. In fact, in this work we use the *IMSL* implementation of *NLPQL*, Schittkowski's dense BFGS-SQP code [42].

In this section we discuss sensitivity analysis, i.e. how we compute derivatives of flow quantities of interest with respect to design variables. Since the BFGS method is used to update a quasi-Newton approximation of the Hessian of the Lagrangian function, only first derivatives of the objective and active inequality constraint functions with respect to design variables are needed. Broadly speaking, one has two choices for computing sensitivities: differentiate first then discretize, or discretize first then differentiate. The latter choice is often referred to as "discrete sensitivities" and the former "discretized continuous sensitivities," or "continuous sensitivities" for short. Generally, the two operations need not commute, and the two are equivalent only in the limit of infinitesimal mesh size (in certain special cases, equivalence is obtained independent of mesh size). See the discussion in [1] for inviscid and viscous flows, and

[27] for a more general setting. For time-dependent Navier-Stokes equations, finding continuous sensitivities requires solving a final–boundary value problem in the adjoint variables [23]. This PDE has time-dependent coefficients that depend on the velocities, which are found by solving the forward problem. Thus storage of the state variable history for all time is required. In contrast, by taking a discrete sensitivity approach, the necessity for this storage is eliminated, as we shall see below.

The (unknown) state variables at time t_{m+1}, $\mathbf{u}^{m+1}$, are related to the design variables $\boldsymbol{\pi}$ and to the (known) state variables at time t_m, $\mathbf{u}^m$, through the spatio-temporally discretized Navier-Stokes equations (3.16)–(3.17), which can be represented symbolically as

$$\mathbf{h}\left(\mathbf{u}^m(\boldsymbol{\pi}), \mathbf{u}^{m+1}(\boldsymbol{\pi}), \boldsymbol{\pi},\right) = \mathbf{0}\,, \quad m = 0, \ldots, L, \ldots, M-1. \tag{4.1}$$

First-order sensitivity equations are obtained through the implicit function theorem, resulting in the linear system

$$\frac{\partial \mathbf{u}^{m+1}}{\partial \boldsymbol{\pi}} \frac{\partial \mathbf{h}}{\partial \mathbf{u}^{m+1}} = -\frac{\partial \mathbf{u}^m}{\partial \boldsymbol{\pi}} \frac{\partial \mathbf{h}}{\partial \mathbf{u}^m} - \frac{\partial \mathbf{h}}{\partial \boldsymbol{\pi}}, \quad m = 0, \ldots, L, \ldots, M-1, \tag{4.2}$$

which can be solved for the state variable sensitivities $\partial \mathbf{u}^{m+1}/\partial \boldsymbol{\pi}$ at time t_{m+1}.

The sensitivity equations, at each time step, are linear and have coefficient matrix $(\partial \mathbf{h}/\partial \mathbf{u})^T$ that is just the asymptotic Jacobian matrix of the discretized Navier-Stokes equations (3.16)–(3.17) i.e. the coefficient matrix of the Newton step (3.18)–(3.19), evaluated at t_{m+1}. It is thus natural to use the same linear solver to solve both the time dependent equations and the sensitivity equations. Furthermore, the sensitivity equations are initialized with the state variable sensitivity at time 0, $\partial \mathbf{u}^0/\partial gbf\pi$, and are thus an initial value problem. Therefore, they can lag the (nonlinear) state equation solve at each time step, and storage of the entire state history is not required. Solving the sensitivity equations (4.2 involves a single factorization of the Jacobian evaluated at the current design variables and converged flow variables, followed by pairs of triangular solves for each of the right hand side vectors, whose number is equal to the number of design variables N^π. The need to obtain accurate sensitivities at each time step is of course in marked contrast to (non-history-dependent) steady problems, in which one must solve the sensitivity equations only at the asymptotic solution, and not at each "pseudo time-step."

The method (4.2 is a discrete *direct* sensitivity method, in the sense that the derivatives of the state variables with respect to the design variables are found. A discrete *adjoint* method is also possible, which would have the same low storage as the discrete direct method, but would involve as many pairs of triangular solves as there are active flow constraints, plus one. Since we use a direct method to solve the sensitivity system, the cost is asymptotically dominated by the LU factorization, so there is no advantage to this discrete adjoint method, even if the active flow constraints number less than the design variables.

The remaining unexplained term in (4.2) is the derivative of the residual with respect to the design variables, $\partial \mathbf{h}/\partial \boldsymbol{\pi}$, which is given for the momentum equations by

$$\frac{\partial \mathbf{h}_i^v}{\partial \boldsymbol{\pi}} = \sum_{j=1}^{N^v} \frac{\partial \mathbf{M}_{ij}}{\partial \boldsymbol{\pi}} \frac{\mathbf{v}_j^{m+1} - \mathbf{v}_j^m}{\Delta t} + \theta \left(\sum_{j=1}^{N^v} \frac{\partial \mathbf{A}_{ij}^{m+1}}{\partial \boldsymbol{\pi}} \mathbf{v}_j^{m+1} + \sum_{k=1}^{N^p} \frac{\partial \mathbf{B}_{ik}^T}{\partial \boldsymbol{\pi}} p_k^{m+1} + \frac{\partial \mathbf{f}_i^{m+1}}{\partial \boldsymbol{\pi}} \right)$$

$$+(1-\theta)\left(\sum_{j=1}^{N^v}\frac{\partial \mathbf{A}_{ij}^m}{\partial \boldsymbol{\pi}}\mathbf{v}_j^m+\sum_{k=1}^{N^v}\frac{\partial \mathbf{B}_{ik}^T}{\partial \boldsymbol{\pi}}p_k^m+\frac{\partial \mathbf{f}_i^m}{\partial \boldsymbol{\pi}}\right) \qquad i=1,\ldots,N^v\,, \tag{4.3}$$

and for the mass equations by

$$\frac{\partial h_k^p}{\partial \boldsymbol{\pi}}=\sum_{j=1}^{N^v}\frac{\partial \mathbf{B}_{kj}}{\partial \boldsymbol{\pi}}\left[\theta \mathbf{v}^{m+1}+(1-\theta)\mathbf{v}^m\right] \qquad k=1,\ldots,N^p\,. \tag{4.4}$$

In these expressions,

$$\frac{\partial \mathbf{M}_{ij}}{\partial \boldsymbol{\pi}}=\int_{\Omega^\xi}\rho\phi_i\phi_j\,\mathbf{I}\,\frac{\partial J}{\partial \boldsymbol{\pi}}\,d\Omega^\xi\,, \tag{4.5}$$

$$\begin{aligned}\frac{\partial \mathbf{A}_{ij}}{\partial \boldsymbol{\pi}}=&\int_{\Omega^\xi}\mu\left[\mathbf{J}^{-T}\left(\nabla^\xi\phi_j\right)\left(\nabla^\xi\phi_i\right)^T\mathbf{J}^{-1}+\left(\nabla^\xi\phi_i\right)^T\mathbf{J}^{-1}\,\mathbf{J}^{-T}\,\left(\nabla^\xi\phi_j\right)\mathbf{I}\right]\frac{\partial J}{\partial \boldsymbol{\pi}}d\Omega^\xi\\&+\int_{\Omega^\xi}\mu\left[\frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}}\left(\nabla^\xi\phi_j\right)\left(\nabla^\xi\phi_i\right)^T\mathbf{J}^{-1}+\mathbf{J}^{-T}\left(\nabla^\xi\phi_j\right)\left(\nabla^\xi\phi_i\right)^T\frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\pi}}\right]Jd\Omega^\xi\\&+\int_{\Omega^\xi}\mu\left[\left(\nabla^\xi\phi_i\right)^T\frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\pi}}\mathbf{J}^{-T}\left(\nabla^\xi\phi_j\right)+\left(\nabla^\xi\phi_i\right)^T\mathbf{J}^{-1}\frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}}\left(\nabla^\xi\phi_j\right)\right]\mathbf{I}Jd\Omega^\xi\\&+\int_{\Omega^\xi}\rho\phi_i\left(\nabla^\xi\phi_j\right)^T\left(\frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\pi}}\mathbf{v}_hJ+\mathbf{J}^{-1}\mathbf{v}_h\frac{\partial J}{\partial \boldsymbol{\pi}}\right)\mathbf{I}\,d\Omega^\xi\,,\end{aligned} \tag{4.6}$$

$$\frac{\partial \mathbf{B}_{kj}}{\partial \boldsymbol{\pi}}=-\int_{\Omega^\xi}\psi_k\,\left(\nabla^\xi\phi_j\right)^T\,\left(\frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\pi}}\,J+\mathbf{J}^{-1}\,\frac{\partial J}{\partial \boldsymbol{\pi}}\right)\,d\Omega^\xi\,, \tag{4.7}$$

$$\begin{aligned}\frac{\partial \mathbf{f}_i}{\partial \boldsymbol{\pi}}=&-\int_{\Omega^\xi}\mathbf{f}\,\frac{\partial J}{\partial \boldsymbol{\pi}}\,d\Omega^\xi-\int_{\Gamma_n^\xi}\phi_i\,\mathbf{t}_u\,\mathbf{n}^T\,\mathbf{J}^{-T}\mathbf{n}^\xi\,\frac{\partial J}{\partial \boldsymbol{\pi}}d\Gamma^\xi-\int_{\Gamma_p^\xi}\phi_i\,\mathbf{t}\,\mathbf{n}^T\,\mathbf{J}^{-T}\mathbf{n}^\xi\,\frac{\partial J}{\partial \boldsymbol{\pi}}d\Gamma^\xi\\&-\int_{\Gamma_n^\xi}\phi_i\,\left[\frac{\partial \mathbf{t}_n}{\partial \boldsymbol{\pi}}\,\mathbf{n}^T\,\mathbf{J}^{-T}+\mathbf{t}_n\,\frac{\partial \mathbf{n}^T}{\partial \boldsymbol{\pi}}\,\mathbf{J}^{-T}+\mathbf{t}_n\,\mathbf{n}^T\,\frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}}\right]\mathbf{n}^\xi\,J\,d\Gamma^\xi\\&-\int_{\Gamma_p^\xi}\phi_i\,\left[\frac{\partial \mathbf{t}}{\partial \boldsymbol{\pi}}\,\mathbf{n}^T\,\mathbf{J}^{-T}+\mathbf{t}\,\frac{\partial \mathbf{n}^T}{\partial \boldsymbol{\pi}}\,\mathbf{J}^{-T}+\mathbf{t}\,\mathbf{n}^T\,\frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}}\right]\mathbf{n}^\xi\,J\,d\Gamma^\xi\,.\end{aligned} \tag{4.8}$$

Since the domain Ω^ξ, the boundary Γ^ξ, and the finite element basis functions ϕ_j and ψ_k have been defined with respect to curvilinear coordinates, they are independent of the design variables $\boldsymbol{\pi}$. Hence, we may treat them as constants in the above differentiation operations. This is one of the advantages of working with isoparametric finite elements: the sensitivity of the shape change can be limited to terms involving the Jacobian of the mapping, $\mathbf{J}$ (e.g. [7]). The derivative of $\mathbf{J}$ with respect to the design variables is given by

$$\frac{\partial \mathbf{J}}{\partial \boldsymbol{\pi}}=\sum_{j=1}^{N^v}\frac{\partial \mathbf{x}_j}{\partial \boldsymbol{\pi}}\left(\nabla^\xi\phi_j\right)^T\,,$$

and of its determinant, J, by

$$\frac{\partial J}{\partial \boldsymbol{\pi}}=J\sum_{i=1}^{N^v}\frac{\partial \mathbf{x}_i}{\partial \boldsymbol{\pi}}\cdot\left(\nabla^\xi\phi_i\right)\,.$$

The critical quantity, then, is $\partial \mathbf{x}/\partial \boldsymbol{\pi}$, often called the *grid sensitivity* or *design velocity field*. It reflects how the locations of the nodes change as the design variables change, and can be symbolically represented as

$$\frac{\partial \mathbf{x}}{\partial \boldsymbol{\pi}} = \frac{\partial \mathbf{x}}{\partial \mathbf{s}} \frac{\partial \mathbf{s}}{\partial \boldsymbol{\pi}},$$

where $\mathbf{s}$ are the coordinates of the surface nodes. The use of the chain rule above exposes the dependence of the grid sensitivities on the derivative of the "solid modeler" (i.e. how the surface nodes move as the design variables change), as well as the derivative of the "mesh generator" (i.e. how the interior nodes move as the surface nodes move). So in general, grid sensitivities depend on the particular form of mesh generation and solid modeling used (which in some cases may not even be differentiable!). We return to this subject in the next sections, in which two examples are solved utilizing different approaches to computing these expressions. The remaining terms in (4.5)–(4.8) are the derivatives of the traction vector, $\partial \mathbf{t}_n/\partial \boldsymbol{\pi}$, which are known from the boundary condition on Γ_n, and the expression for $\partial \mathbf{n}/\partial \boldsymbol{\pi}$ on Γ_n and Γ_p, which can be found straightforwardly from the particular relationship between surface representation and design variables.

In any case, once the grid sensitivities are known, expressions (4.5)–(4.8) can be computed, and the residual derivatives (4.3)–(4.4) are readily evaluated. Thus, all quantities needed for solution of the sensitivity equations (4.2) are known, and this system can be solved for the state variable sensitivities $\partial \mathbf{u}/\partial \boldsymbol{\pi}$ at time t_{m+1}. Now we turn our attention to computing design gradients of the local and global derived quantities (2.8)–(2.13).

Let $\Psi^{m+1} = \Psi(\mathbf{u}^{m+1}(\boldsymbol{\pi}), \boldsymbol{\pi})$ denote any of the local (in space and time) flow functions in (2.8)–(2.10) at time step t_{m+1}, $m = 0, \ldots, L, \ldots, M-1$. Then $D\Psi^{t+1}/D\boldsymbol{\pi}$, i.e. the gradient of the generic local flow function Ψ with respect to the design variables at time t_{m+1}, can be found from

$$\frac{D\Psi^{m+1}}{D\boldsymbol{\pi}} = \frac{\partial \Psi^{m+1}}{\partial \boldsymbol{\pi}} + \frac{\partial \mathbf{u}^{m+1}}{\partial \boldsymbol{\pi}} \frac{\partial \Psi^{m+1}}{\partial \mathbf{u}^{m+1}}, \qquad m = 0, \ldots, L, \ldots, M-1. \tag{4.9}$$

The state variable sensitivities $\partial \mathbf{u}/\partial \boldsymbol{\pi}$ enter into the expression for $D\Psi^{t+1}/D\boldsymbol{\pi}$, as do the partial derivatives of Ψ, with respect to both the state variables as well as the design variables. The former expressions are straightforward; the latter expressions are given by, for the rate of energy dissipation due to viscosity,

$$\begin{aligned} \frac{\partial \Phi}{\partial \boldsymbol{\pi}} = \mu \sum_{i=1}^{N^v} \sum_{j=1}^{N^v} & \left[\frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}} (\nabla^\xi \phi_i) \mathbf{v}_i^T + \mathbf{v}_i (\nabla^\xi \phi_i)^T \frac{\partial \mathbf{J}^{-1}}{\partial \boldsymbol{\pi}} \right] \\ & : \left[\mathbf{J}^{-T} (\nabla^\xi \phi_j) \mathbf{v}_j^T + \mathbf{v}_j (\nabla^\xi \phi_j)^T \mathbf{J}^{-1} \right], \end{aligned} \tag{4.10}$$

for the maximum shear stress, e.g. in two dimensions,

$$\begin{aligned} \frac{\partial \tau}{\partial \boldsymbol{\pi}} = \frac{\mu^2}{\tau} & \left[\sum_{i=1}^{N^v} \sum_{j=1}^{N^v} \left(v_{ix} \frac{\partial}{\partial \boldsymbol{\pi}} \frac{\partial \phi_i}{\partial x} - v_{iy} \frac{\partial}{\partial \boldsymbol{\pi}} \frac{\partial \phi_i}{\partial y} \right) \left(v_{jx} \frac{\partial \phi_j}{\partial x} - v_{jy} \frac{\partial \phi_j}{\partial y} \right) \right. \\ & \left. + \sum_{i=1}^{N^v} \sum_{j=1}^{N^v} \left(v_{ix} \frac{\partial}{\partial \boldsymbol{\pi}} \frac{\partial \phi_i}{\partial y} + v_{iy} \frac{\partial}{\partial \boldsymbol{\pi}} \frac{\partial \phi_i}{\partial x} \right) \left(v_{jx} \frac{\partial \phi_i}{\partial y} + v_{jy} \frac{\partial \phi_i}{\partial x} \right) \right], \end{aligned} \tag{4.11}$$

where

$$\frac{\partial}{\partial \boldsymbol{\pi}} \nabla \phi_i = \frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}} \nabla^{\xi} \phi_i \,,$$

and for the magnitude of vorticity,

$$\frac{\partial \varpi}{\partial \boldsymbol{\pi}} = \frac{1}{\varpi} \left[\sum_{i=1}^{N^v} \frac{\partial \mathbf{J}^{-T}}{\partial \boldsymbol{\pi}} \left(\nabla^{\xi} \phi_i \right) \times \mathbf{v}_i \right]^T \left[\sum_{i=1}^{N^v} \mathbf{J}^{-T} \left(\nabla^{\xi} \phi_i \right) \times \mathbf{v}_i \right] . \tag{4.12}$$

Finally, we show how to compute sensitivities of the global flow functions, once local function sensitivities have been found. Let $\widehat{\Psi}_u$ and $\widehat{\Psi}_s$ represent any of the unscaled and scaled global flow functions in (2.11)–(2.13), respectively. Thus,

$$\widehat{\Psi}_u = \frac{1}{2} \Delta t \sum_{m=L}^{M-1} \int_{\Omega^{\xi}} \left[\Psi \left(\mathbf{u}^m, \boldsymbol{\pi} \right) + \Psi \left(\mathbf{u}^{m+1}, \boldsymbol{\pi} \right) \right] \, J d\Omega^{\xi} \,, \qquad \widehat{\Psi}_s = \frac{\widehat{\Psi}_u}{\int_{\Omega^{\xi}} J d\Omega^{\xi}} \,, \tag{4.13}$$

where the time integration is carried out according to the trapezoidal rule. (Although the unscaled global flow functions involving shear stress and vorticity are, in fact, defined as

$$\widehat{\Psi}_u = \int_{\Omega^{\xi}} \Psi^2 \left(\mathbf{u}, \boldsymbol{\pi} \right) \, J d\Omega^{\xi} \,,$$

the following discussion still applies with slight modification.) Again, since the domain Ω^{ξ} is defined in the curvilinear coordinate space and is independent of design variables, the differentiation operator with respect to $\boldsymbol{\pi}$ acts directly on the integrand in (4.13). Thus, we have

$$\frac{D\widehat{\Psi}_u}{D\boldsymbol{\pi}} = \frac{1}{2} \Delta t \sum_{m=L}^{M-1} \int_{\Omega^{\xi}} \left(\frac{D\Psi^m}{D\boldsymbol{\pi}} J + \Psi^m \frac{\partial J}{\partial \boldsymbol{\pi}} + \frac{D\Psi^{m+1}}{D\boldsymbol{\pi}} J + \Psi^{m+1} \frac{\partial J}{\partial \boldsymbol{\pi}} \right) d\Omega^{\xi} \,, \tag{4.14}$$

$$\frac{D\widehat{\Psi}_s}{D\boldsymbol{\pi}} = \frac{D\widehat{\Psi}_u}{D\boldsymbol{\pi}} \frac{1}{S} - \frac{\widehat{\Psi}_u}{S^2} \int_{\Omega^{\xi}} \frac{\partial J}{\partial \boldsymbol{\pi}} d\Omega^{\xi} \,. \tag{4.15}$$

Thus, the gradients of global flow functions entering into the objective function, as well as gradients of local (in time and space) flow functions involved in inequality constraints, can be computed in much the same way as the flow function themselves are, i.e. time step by time step.

In summary, at each time step,

- the discrete Navier-Stokes equations (3.16)–(3.17) are advanced a step in time;
- the velocities and pressures at the current time step are computed by the (nonlinear) Newton iteration (3.18)–(3.19);
- the local flow functions are computed at various points in space and at the current time step, given current velocities and pressures, e.g. for viscous dissipation from (3.20);
- the values of the global flow functions, accumulated up to the current time step, are incremented using current local function quantities, e.g. for viscous dissipation by (3.21);

- the (linear) sensitivity equations (4.2) are solved for the velocity and pressure sensitivities corresponding to each design variable at the current time step, given the current and previous velocities and pressures and the previous velocity and pressure sensitivities;
- the sensitivities of the local flow functions are computed at various points in space and at the current time step from (4.9), given current velocities and pressures and their sensitivities; and
- the values of the sensitivities of global flow functions, accumulated up to the current time step, are incremented using current local function sensitivities by (4.14) or (4.15).

Thus, sensitivity computations add little additional work and storage beyond that required for the flow solution, since they are linear and share the same coefficient matrix with the converged linearized discrete Navier-Stokes equations at each tie step. This is especially true when direct LU factorization is used to solve the sensitivity system, since sensitivity requires just one extra factorization beyond that required by the flow solver, and the storage associated with $\partial \mathbf{h}/\partial \boldsymbol{\pi}$ is small if the number of design variables $\boldsymbol{\pi}$ is small.

We have developed a code that implements the algorithms and methods described in Sections 3 and 4. Spatial discretization is by triangular Taylor-Hood finite elements, which use piecewise quadratic velocity and piecewise linear pressure basis functions. The Taylor-Hood element is known to be stable in the sense of Ladyzhenskaya–Babuska–Brezzi, and produces errors of order h^3 for velocity and h^2 for pressure [16]. Time-integration is by the Crank-Nicolson method. The multifrontal sparse LU factorization code *UMFPACK* [10] is used to solve the linear systems that arise at each step of Newton's method. The optimizer used is *IMSL*'s implementation of the dense BFGS-SQP code *NLPQL* [42]. Analytical gradients of the objective and inequality constraints are found as described in this section. In the next two sections, we present two time-dependent shape optimization model problems solved using our code. The first involves finding the best shape of a 90° tube subject to an internal flow, and uses a logically-rectangular structured mesh. The second problem concerns finding the best shape of a obstacle subjected to a uniform external flow, and is discretized by an unstructured mesh. Approaches to computing the grid sensitivities $\partial \mathbf{x}/\partial \boldsymbol{\pi}$ depend on whether the mesh is structured or unstructured, and are described below for each case. All computations are performed on one processor of a 4-CPU Digital AlphaServer 8400 5/300 equipped with 4 Gbytes of memory.

5. Shape optimization of flow in a two-dimensional tube. Our first example is a two-dimensional tube, shown in Figure 5.1a, in which fluid enters horizontally from the left, undergoes a 90° bend, and exits vertically downward. This is a simplified model for a blood flow cannula of an artificial heart device. It is assumed that the tube has a uniform section with a width or diameter d. The problem we consider is to design the shape of the middle axis of the tube, such that the viscous dissipation (2.11) is minimized subject to certain geometric constraints. The shape of the middle axis is taken as a function $r(\theta)$ in the polar coordinate system, and it can be represented by a cosine series

$$r(\theta) = \sum_{i=0}^{P-1} r_i \cos(2i\theta), \qquad \theta \in [0°, 90°].$$

The coefficients r_i, $i = 0, \ldots, P-1$ are thus the design variables. We generate a structured mesh similar to the one shown in Figure 5.1b, and compute the grid

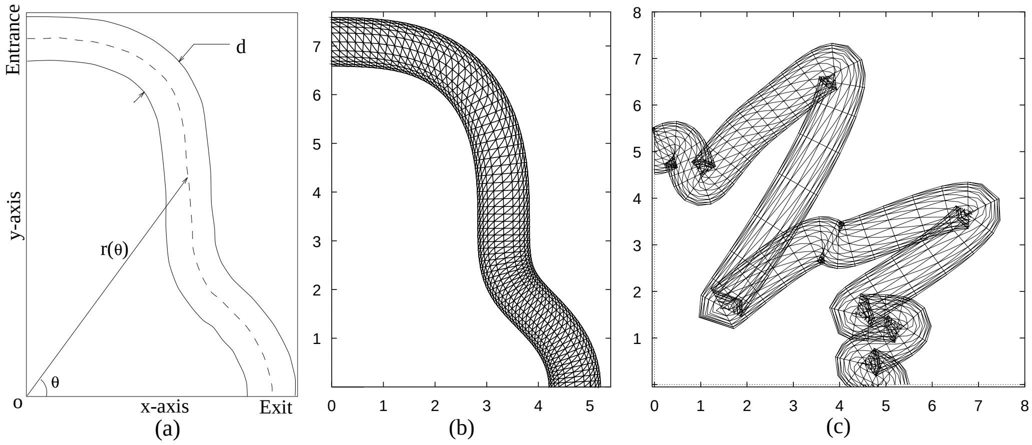

FIG. 5.1. *2D tube: (a) geometry, (b) computational mesh, (c) invalid shape.*

sensitivities analytically. This is straightforwardly done, since the mesh is structured, and hence closed-form, smooth expressions are available for the grid sensitivities, $\partial \mathbf{x}/\partial \boldsymbol{\pi}$.

The middle axis $r(\theta)$ is subject to one of the following end constraints:

- both positions of entrance and exit are fixed, specifically, $r(90°) = \gamma_1$ and $r(0°) = \gamma_2$;
- the entrance is fixed while the exit position is allowed to vary within a given range, i.e., $r(90°) = \gamma_1$ and $\gamma_2^l \leq r(0°) \leq \gamma_2^u$;
- the entrance position is allowed to vary within a given range while the exit is fixed, i.e., $\gamma_1^l \leq r(90°) \leq \gamma_1^u$ and $r(0°) = \gamma_2$;
- both positions of entrance and exit are allowed to vary within given ranges, namely, $\gamma_1^l \leq r(90°) \leq \gamma_1^u$ and $\gamma_2^l \leq r(0°) \leq \gamma_2^u$;

where γ_1, γ_2, γ_1^l, γ_1^u, γ_2^l, and γ_2^u are all prescribed positive constants. These conditions impose linear constraints on the design variables, since

$$r(0°) \;=\; \sum_{i=0}^{P-1} r_i\,, \qquad r(90°) \;=\; \sum_{i=0}^{P-1} (-1)^i\, r_i\,.$$

In addition, design variables have to satisfy certain conditions to ensure a valid shape; otherwise, invalid shapes such as the one shown in Figure 5.1c may arise (as actually happened). Since we deal only with a discretized geometric model of the tube, shape validity constraints need only be imposed at K distinct points $(r(\theta_i), \theta_i)$, $i = 1, \ldots, K$, uniformly spaced along the middle axis of the tube. First, $r(\theta)$ must be non-negative,

$$r(\theta_i) \geq 0, \quad i = 1, \ldots, K.$$

Second, the shape cannot penetrate itself,

$$(\bar{x} - x_{0,i})^2 + (\bar{y} - y_{0,i})^2 \geq \frac{1}{4} d^2,$$

$$(\bar{x} - x_{0,i+1})^2 + (\bar{y} - y_{0,i+1})^2 \geq \frac{1}{4} d^2 ,$$
$$i = 1, \ldots, K .$$

Here, $(x_{0,i}, y_{0,i})$ is the Cartesian coordinate of the i-th node along the middle axis of the tube; $(\bar{x}, \bar{y})$ is the coordinate of the intersection point of the two lines perpendicular to each other and passing through $(x_{0,i}, y_{0,i})$ and $(x_{0,i+1}, y_{0,i+1})$. The values of $(\bar{x}, \bar{y})$ and $(x_{0,i}, y_{0,i})$ are computed with the following formulae:

$$\begin{aligned}
\bar{x} &= \Delta_x / \Delta , \qquad \bar{y} = \Delta_y / \Delta , \\
\Delta_x &= l_{y,i} l_{y,i+1} (y_{0,i} - y_{0,i+1}) + x_{0,i} l_{x,i} l_{y,i+1} - x_{0,i+1} l_{y,i} l_{x,i+1} , \\
\Delta_y &= l_{x,i} l_{x,i+1} (x_{0,i+1} - x_{0,i}) + y_{0,i+1} l_{x,i} l_{y,i+1} - y_{0,i} l_{y,i} l_{x,i+1} , \\
\Delta &= l_{x,i} l_{y,i+1} - l_{y,i} l_{x,i+1} , \\
x_{0,i} &= r(\theta_i) \cos\theta_i , \qquad y_{0,i} = r(\theta_i) \sin\theta_i , \\
l_{x,i} &= \frac{r'(\theta_i)\cos\theta_i - r(\theta_i)\sin\theta_i}{[r'(\theta_i)^2 + r(\theta_i)^2]^{1/2}} , \qquad l_{y,i} = \frac{r'(\theta_i)\sin\theta_i + r(\theta_i)\cos\theta_i}{[r'(\theta_i)^2 + r(\theta_i)^2]^{1/2}} , \\
r'(\theta_i) &= \frac{\partial r(\theta_i)}{\partial \theta} .
\end{aligned}$$

Boundary conditions for solution of the Navier-Stokes equations include Neumann conditions at the outlet and Dirichlet conditions on the rest of the boundary. Specifically, no-slip boundary conditions are proposed at the walls of the tube, the outflow is assumed to be traction free, and the inflow is taken to have a parabolic velocity profile,

$$v_x = \frac{6q}{d^3}\left(y - y_0 + \frac{d}{2}\right)\left(y_0 - y + \frac{d}{2}\right) , \qquad v_y = 0 ,$$

where q is the flow rate, and d and y_0 are, respectively, the diameter and y-coordinate of the tube centroid at the inlet.

For sensitivity analysis, velocity design gradients vanish on the side walls; the outflow is kept traction free, which implies sensitivities of the traction at the outlet are zero; and the inflow velocity preserves its parabolic profile with the same flow rate, which leads to the following inlet conditions,

$$\frac{\partial v_x}{\partial \pi} = 0 , \qquad \frac{\partial v_y}{\partial \pi} = 0 .$$

The flow rate is assumed to be

$$q = \begin{cases} 0.1\,t & t \in [0,\ 5) \\ 0.5 & t \in [5,\ 8) \\ 0.5 + 0.1 \sin[2\pi(t-8)] & t \in [8,\ \infty) \end{cases} .$$

Thus, after $t > 8$, the inflow oscillates with a period of 1 and an amplitude of 0.1. The diameter of the tube d is kept at 1. Thus, the Reynolds number based on the average inflow velocity and tube diameter is $Re = 500$. Let us fix the tube at both the exit and entrance with $r = 5.1$, and let its shape $r(\theta)$ be approximated with the first 14 Fourier cosine coefficients. We optimize the tube shape such that the unscaled global dissipation function $\widehat{\Phi}_u$ in (2.11) is a minimum. The mesh has 31 nodes through the cross section and 175 nodes along the middle axis, resulting in 2610 elements, 5425 nodes, and 11,471 velocity and pressure unknowns.

The initial shape of the tube is described in Table 5.1 under the heading "initial". Time histories of the dissipation function $\int_\Omega \Phi d\Omega$ are computed with time steps $\Delta t = 0.125$ and 0.03125. They are plotted in the left graph of Figure 5.2. A Fourier transform is performed on these data, and the results are summarized in the right

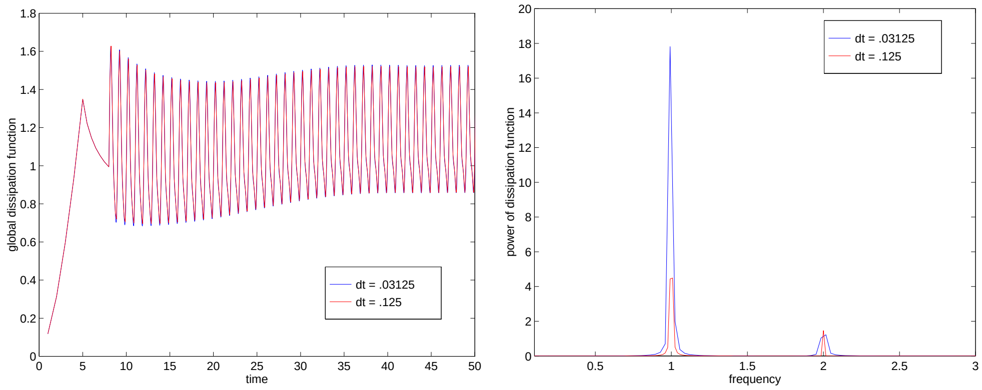

FIG. 5.2. *Time history of the global dissipation function $\overline{\Phi}_u$ (left) and its Fourier transforms (right) corresponding to two different time steps.*

graph of Figure 5.2. Several observations can be made based on these plots:

- a time step of $\Delta t = 0.125$ yields a sufficiently accurate flow solution;
- the initial part of the time-dependent solution exhibits transient behavior and should be ignored for optimization purposes; and
- the rest of the solution oscillates primarily with the same frequency as the inflow, which is 1.

Therefore, the characteristic time interval for objective function computation is taken as a multiple of the inflow time period. For this problem, the objective function $\widehat{\Phi}_u$ equals $\int_\Omega \Phi d\Omega$ integrated over

$$[t_L, t_M] = \begin{cases} [30,\ 33] & \text{Case I} \\ [30,\ 36] & \text{Case II} \end{cases},$$

which correspond to 3 and 6 inflow periods, respectively. For the remainder of the computation, we use $\Delta t = 0.125$. Figure 5.3 gives several snapshots in time of velocity streamlines corresponding to the initial shape. The development and oscillation of a separated and recirculating flow region is evidenced in the figure, and is induced by the abrupt change in geometry.

Optimal shape coefficients are tabulated in Table 5.1 under "Case I" and "Case II". The optimal shape for a steady inflow (of $Re = 500$) coupled with a steady Navier-Stokes flow model is also included under the heading "Steady" for comparison. The middle axes defined in Table 5.1 are plotted in Figure 5.4. There is no substantial difference among the three optimal shapes, although it takes about 42 and 51 hours of CPU time to reach the optimum for Cases I and II, respectively, while only about an hour is needed in the steady case. Thus, we see a factor of 40 to 50 increase in CPU time when the flow model is time-dependent. The large increases in CPU time over the steady solution are a result of (i) the need for time-accurate integration, (ii) the necessity of integrating through initial transients to reach periodic solutions, and (iii) the need for solving sensitivity equations at each time step. For this particular

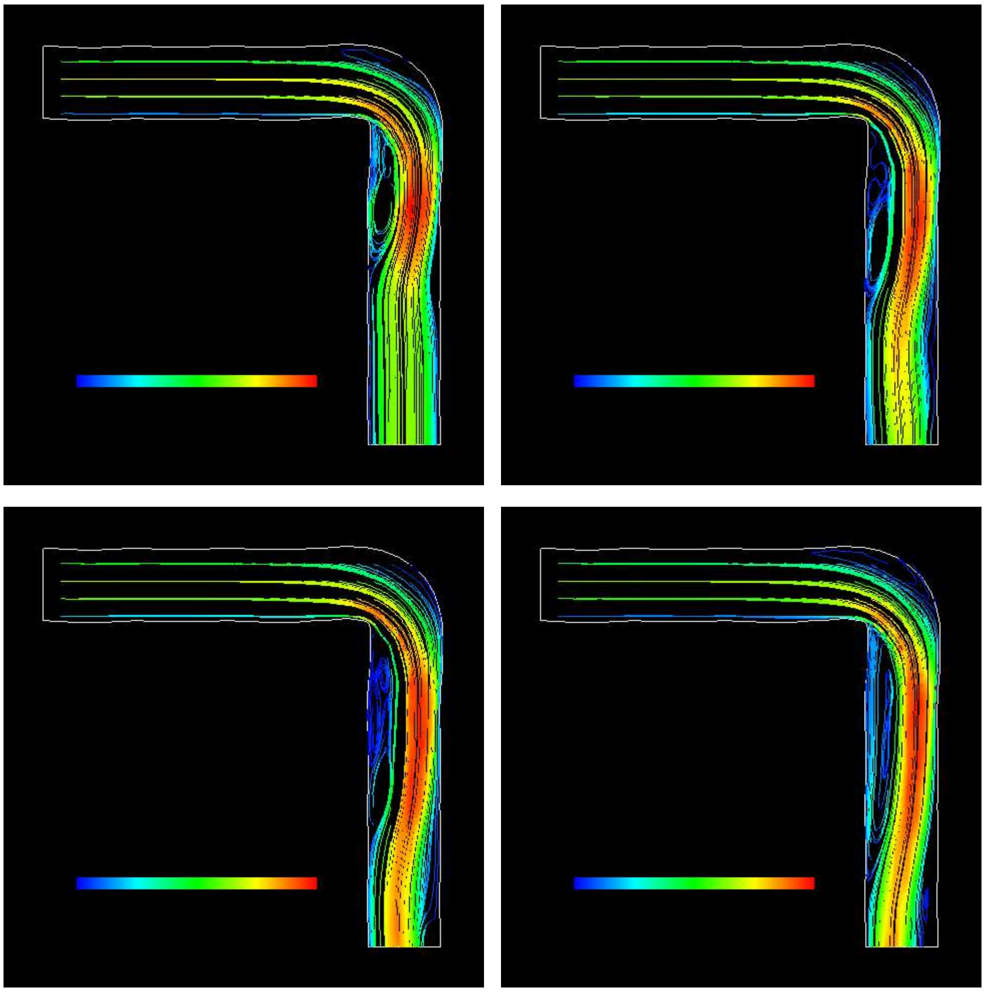

FIG. 5.3. *Streamlines taken at* $t = 12.875$ *(upper left),* 19.125 *(upper right),* 25.375 *(bottom left), and* 31.625 *(bottom right) for the initial shape; Re = 500. Colors represent velocity magnitude, with blue and red corresponding to low and high values, respectively.*

problem, it seems to be a waste of time to perform optimization with the time-dependent Navier-Stokes equations. More generally, however, there is no guarantee that optimizing with a steady flow model will yield the same optimum shape as with a time-dependent flow model. This is almost certainly the case when the optimum is not characterized by steady flow.

The number of complete flow solutions for Cases I and II is, respectively, 31 and 29; the number of optimization iterations is respectively, 24 and 25. (In most optimization iterations, the initial step length (of 1) is accepted by the optimizer.) Optimization histories are plotted in Figure 5.5. The left graph shows the value of the objective function $\widehat{\Phi}_u$ as a function of the number of optimization iterations. The right graph plots the Euclidean norm of the Kuhn-Tucker optimality condition (the sum of the Euclidean norms of the reduced gradient and active constraints) against the optimization iteration number. The reduction in viscous energy dissipation in both

TABLE 5.1
Design variables for 2D tube problem for steady and time-dependent flow.

Variable	Initial	Steady	Case I	Case II
r_0	5.6109985	5.1174910	5.0673756	5.0758872
r_1	0.0000000	0.0502573	0.0432338	0.0431676
r_2	-0.7800000	0.0216108	0.0644936	0.0566008
r_3	0.0000000	-0.0233347	-0.0234054	-0.0229796
r_4	0.2400000	-0.0131780	-0.0114753	-0.0113118
r_5	0.0000000	-0.0093722	-0.0090339	-0.0087382
r_6	-0.1100000	-0.0084052	-0.0068513	-0.0071648
r_7	0.0000000	-0.0056902	-0.0039357	-0.0041412
r_8	0.0600000	-0.0064563	-0.0049552	-0.0051214
r_9	0.0000000	-0.0043257	-0.0025236	-0.0026561
r_{10}	-0.0300000	-0.0056489	-0.0042339	-0.0043963
r_{11}	0.0000000	-0.0037718	-0.0019915	-0.0021727
r_{12}	0.0200000	-0.0054134	-0.0043535	-0.0044938
r_{13}	0.0000000	-0.0037627	-0.0023436	-0.0024798

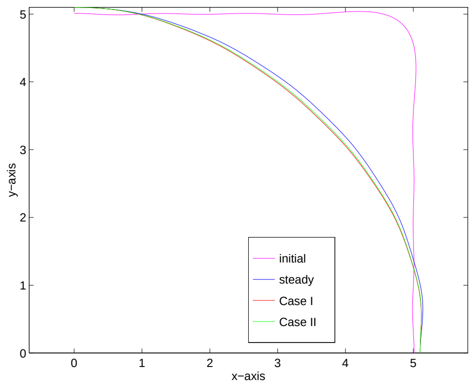

FIG. 5.4. *Initial and optimal tube centerlines.*

cases is 50%. The convergence behavior of the two cases is similar, which indicates no advantage is gained by using an unnecessarily long characteristic period. The optimal shape, at several instants in time, is shown in Figure 5.6 for the Case I objective (Case II is indistinguishable). Gone are the abrupt geometry change and the separated flow region. Furthermore, the optimum flow field is for the most part time-independent. Of course, one does not need a sophisticated optimization algorithm to know that a smoothly curved tube will produce a flow field that dissipates less viscous energy than one with a sharp corner; however, a problem with such an intuitively-expected optimal solution is valuable for helping to validate the method and code used.

6. Shape optimization of flow around a two-dimensional obstacle. Placing a body into a free stream flow generates a disturbance to the fluid motion. If the body is bluff, separation may result, leading to wake formation and extension downstream. Depending on the shape of the body, the flow may be unsteady for Reynolds number as low as 40, despite the fact that the free stream is steady. As Reynolds number increases, a vortex street forms and discrete vortices are shed at regular intervals. Therefore, shape optimization of such a system should be performed using a time-dependent flow model. In this section, we optimize the shape of an obstacle by

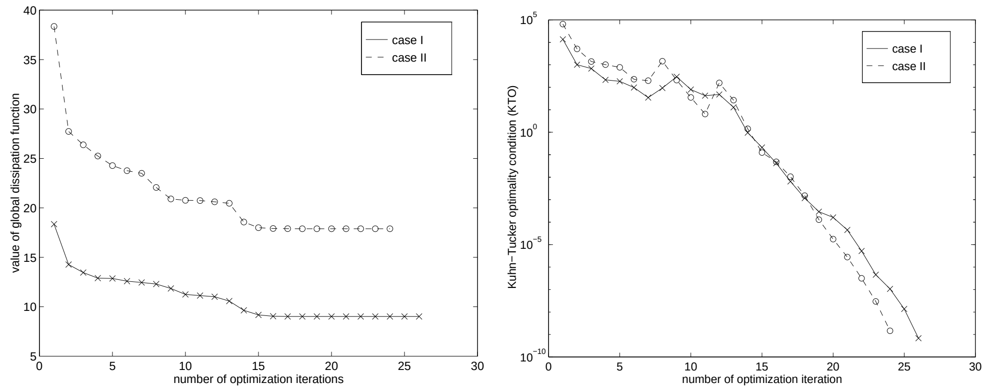

FIG. 5.5. *Objective function (left) and Euclidean norm of Kuhn-Tucker optimality condition (right) as a function of number of optimization iterations for 2D tube problem.*

minimizing the scaled global viscous dissipation function $\widehat{\Phi}_s$ in (2.11). The characteristic time interval for the objective function evaluation is taken as a multiple of the vortex shedding period. This can be found from the relationship

$$f_s = \frac{S\,U}{D}\,,$$

where f_s is the predominant vortex shedding frequency, U is the speed of the free stream flow, D is the cylinder diameter, and S is a dimensionless Strouhal number. The dependence of S on Reynolds number Re has been established experimentally for $Re = 40$ to 10^7 [6].

In the finite element method, it is well known that poor element quality can have a deleterious effect on the solution. Angles near 0 can lead to ill-conditioning of the discrete operator [13], while angles near π can lead to large approximation errors in the H_1 norm [4]. When the geometry undergoes large changes during optimization, it becomes very difficult, if not impossible, to construct a structured mesh scheme that maintains element quality for a wide range of shapes. An unstructured mesh is capable of resolving complicated, deforming geometries with guaranteed aspect ratio elements (certainly in 2D, less assuredly in 3D), and thus is more appropriate for general shape optimization problems. Therefore, in anticipation of large shape changes for the obstacle problem solved in this section, we discretize the computational domain with unstructured triangular meshes. The central difficulty with the use of unstructured meshes in shape optimization is the lack of differentiability of typical mesh generators. That is, the resulting triangulations they produce do not vary smoothly with the input boundary shapes. For example, we use a state-of-the-art 2D triangular mesh generator, *Triangle*, which is a constrained Delaunay code that incorporates adaptive arithmetic for stability and robustness [44]. *Triangle* makes a number of binary decisions that render its output a discontinuous function of its input, including whether edges should be flipped, boundary edges split, triangles refined, etc. As a consequence, two meshes of related shape may not share the same topology nor the same number of nodes or elements. Note, however, that differences in functions such as (2.11)–(2.13) defined as integrals over the two meshes will tend to be small between such a pair of meshes.

We are lead to adopt the following two-phase strategy for accommodating unstructured meshes in shape optimization. In the early optimization iterations, shape

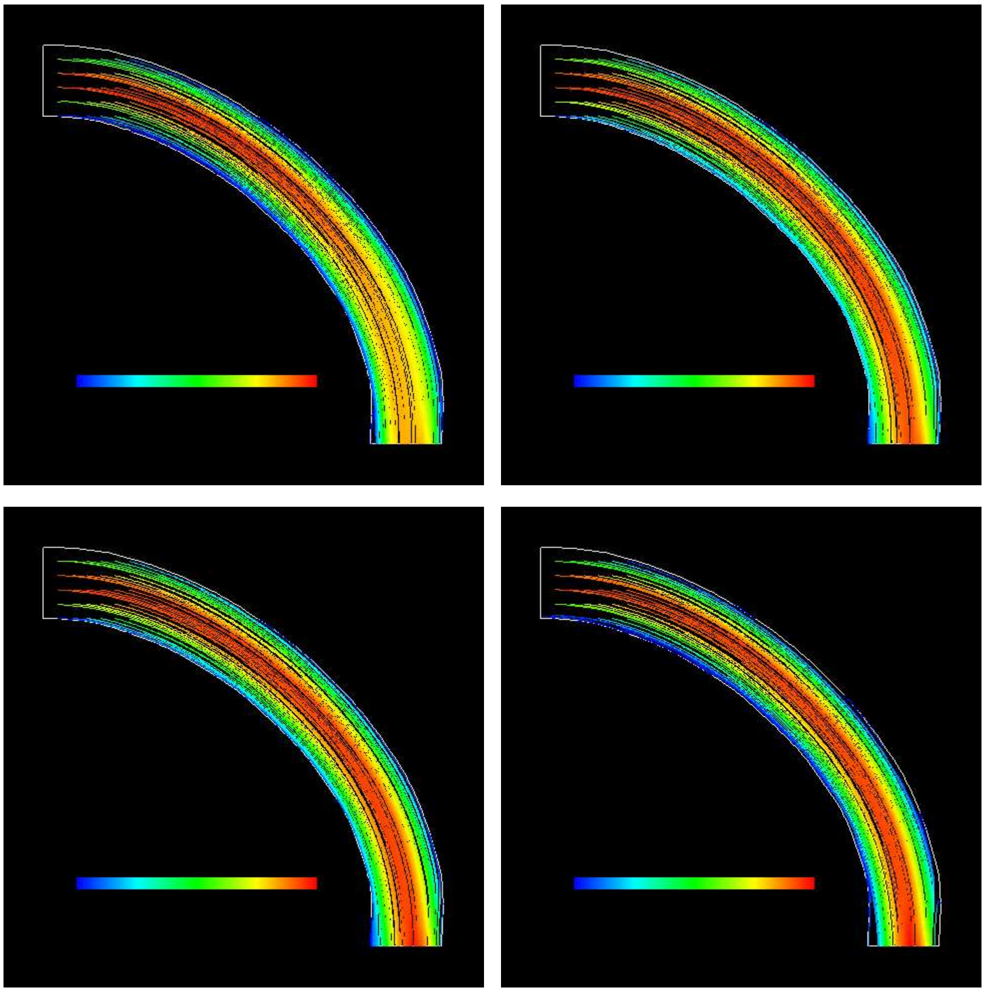

FIG. 5.6. *Flow streamlines at* $t = 12.875$ *(upper left),* 19.125 *(upper right),* 25.375 *(bottom left), and* 31.625 *(bottom right) seconds for the optimal shape. The optimum represents a 50% reduction in viscous energy dissipation over the initial shape.*

change is expected to be very large, which necessitates topological changes to the mesh to maintain element quality. Thus, in Stage I, the unstructured mesh generator *Triangle* is used to completely remesh the flow domain at each iteration. In light of the discontinuities associated with Delaunay-based meshing algorithms, the question is: how should $\partial \mathbf{x}/\partial \boldsymbol{\pi}$ be computed? The rows of this matrix corresponding to boundary nodes are easy: they are simply derivatives of the expressions for representing the shape in terms of the design variables (i.e., derivatives of the "solid modeler"). For example, a specific parameterization for the obstacle problem is given later in this section. For rows of $\partial \mathbf{x}/\partial \boldsymbol{\pi}$ corresponding to interior nodes, we discard the idea of attempting to differentiate *Triangle*; instead, in Stage I, we simply set these rows, i.e., the interior grid sensitivities, to zero. The motivation for this is a result from sensitivity analysis that states that the *continuous* shape sensitivity of a flow function is independent of the grid sensitivity of interior points, at least for certain objective functions expressed in terms of domain integrals [37]. Therefore, for the sake of speed and simplicity, we might as well take grid sensitivities to be zero. Because the problem will be solved on a finite mesh, there generally will be a small discrepancy between the computed gradients and the change in the objective function "seen" by the optimizer. Far from the optimum, the optimizer takes large steps, so provided a sufficient decrease in the merit function is observed, this discrepancy and the lack of smoothness are not a problem. Discussions of the influence of inexact sensitivities on design optimization can be found in [45].

Close to the optimum, however, we sometimes find that the inconsistency between the way the mesh is actually moved and the way it is *assumed* to move results in premature termination of the optimization iterations. Typically, a vicinity of a stationary point is reached, but the line search breaks down due to insufficient decrease in the merit function—a sign that gradients are not consistent with actual changes in the objective function. To make further progress in these cases, consistent derivatives and differentiable mesh movement are needed. Both of these conditions can be readily satisfied by using what we refer to as *elastic mesh movement*. The algorithm starts with a mesh of acceptable quality. The change in the location of (a portion of) the boundary of the computational domain is treated as an imposed displacement—an inhomogeneous Dirichlet boundary condition—and a linear elasticity boundary value problem (discretized by finite elements and posed on the same mesh) is solved. The solution of the elasticity problem, i.e., the displacement field, is taken to be the change in location of the interior nodes of the mesh. Because the discrete displacement field depends smoothly on the value of inhomogeneous Dirichlet boundary conditions, this mesh movement algorithm guarantees that coordinates of interior nodes are smooth functions of the locations of boundary nodes. Furthermore, since the relationship between displacement boundary conditions and the resulting solution of the elasticity problem is explicit in form, interior grid sensitivities are easily found through direct differentiation. Grid sensitivities are thus exact (modulo rounding errors). Finding grid sensitivity essentially involves solving the discrete elasticity equations for a number of right hand sides equal to the number of design variables N^{π}. Using a direct linear solver, this implies a single factorization coupled with N^{π} pairs of triangular solves. The cost of mesh movement and of obtaining grid sensitivities are inconsequential compared to solution of the time-dependent Navier-Stokes equations at each optimization iteration. Note that elastic mesh movement maintains mesh topology, and therefore may lead to severely stretched or even invalid meshes for large shape changes. Thus, it is best left for small shape changes contemplated in Stage II. Similar physically-based mesh moving schemes are popular for moving boundary problems, and have been in use as early as in [3]. Related mesh moving schemes using networks of springs have been used in various flow optimization settings in [1] [12] [30] [31].

So in summary, mesh movement is by unstructured mesh generation in Stage I and elastic mesh movement in Stage II; internal mesh sensitivities are simply taken as zero in Stage I and computed by taking the derivative of the elastic mesh movement algorithm in Stage II. Table 6.1 summarizes what each phase of optimization needs and can tolerate, with respect to meshing and grid sensitivity. Thus, the strategies of Stages I and II are well-suited to the requirements of Table 6.1. Finally, note that we have not automated switching between these two strategies; we simply switch to Stage II if no further progress is possible in Stage I and the current point is not a stationary point.

We turn now to the shape parameterization. The shape of the obstacle is repre-

TABLE 6.1
Meshing and grid sensitivity requirements of different phases of optimization.

Phase	Meshing	Grid sensitivity
far from optimum	*need*: accommodate large shape change *tolerate*: discontinuity or nonsmoothness	*need*: speed *tolerate*: inaccuracy
near optimum	*need*: differentiability *tolerate*: restriction to small shape change	*need*: accuracy *tolerate*: cost

sented by two Bezier curves,

$$\overline{x}^t(s) = \sum_{j=0}^{K} p_j^t \, B_{K,j}(s) , \qquad \overline{y}^t(s) = \sum_{j=0}^{K} q_j^t \, B_{K,j}(s)$$

for the upper boundary; and

$$\overline{x}^b(s) = \sum_{j=0}^{K} p_j^b \, B_{K,j}(s) , \qquad \overline{y}^b(s) = \sum_{j=0}^{K} q_j^b \, B_{K,j}(s)$$

for the lower boundary. Here, $(p_j^t,\, q_j^t)$ and $(p_j^b,\, q_j^b)$ are coordinates of Bezier control points,

$$B_{K,j}(s) = \frac{K!}{j!\,(K-j)!} s^j (1-s)^{K-j}$$

is a Bernstein polynomial, K is the order of the Bernstein polynomial, and $s \in [0,1]$ is a parameter. As s sweeps over $[0,1]$, $(\overline{x}^t(s),\, \overline{y}^t(s))$ and $(\overline{x}^b(s),\, \overline{y}^b(s))$ depict the upper and lower surfaces of the obstacle, respectively. Bezier parameterization defines a smooth curve that falls into a polygon determined by Bezier points and passes the two Bezier end points, i.e., (p_0, q_0) and (p_K, q_K). The shape of a Bezier curve is anticipated by that of the containing polygon; thus, Bezier parameterization is controllable. Shape design variables are the $2(K+1)$ Bezier points $(p_0^t, q_0^t), \ldots, (p_K^t, q_K^t)$ and $(p_0^b, q_0^b), \ldots, (p_K^b, q_K^b)$.

Similarly to the previous section, we propose the following end conditions to limit the size and position of the obstacle:

- the position of both leading and trailing edges of the obstacle are fixed, i.e., p_0^t, q_0^t, p_K^t, and q_K^t are prescribed;
- the leading edge is fixed while the trailing edge is free to move within a range, i.e., p_0^t and q_0^t are specified while p_K^t and q_K^t may change within certain intervals;
- the position of the leading edge is allowed to change within a range while the trailing edge is fixed, i.e., p_0^t and q_0^t may vary within specified intervals while p_K^t and q_K^t are equal to assigned values;
- both edges are free to move within specified ranges, i.e., p_0^t, q_0^t, p_K^t, and q_K^t may all vary within given intervals.

To prevent separation of the two Bezier curves at the leading and trailing edges, we must enforce

$$p_0^t = p_0^b \qquad \text{and} \qquad q_K^t = q_K^b \, .$$

We require that the obstacle area S exceeds a certain minimum value, for example to insure structural integrity of the obstacle. Thus,

$$S \geq S_{\min} \, .$$

The upper and lower surfaces of the obstacle may overlap during optimization iterations. To inhibit such an invalid shape, we choose N nodes on the top surface and N corresponding nodes on the bottom of the obstacle, which correspond to

$$s = s_m = \frac{m}{N-1} \qquad m = 0, \dots, N-1 .$$

Then, we triangulate the area contained within these nodes, as shown (coarsely) in Figure 6.1. Shape constraints are that all triangles within the obstacle must have

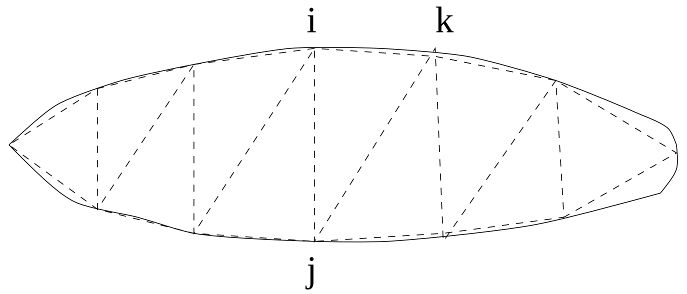

FIG. 6.1. *Triangulation of the obstacle for exclusion of invalid shapes.*

non-negative areas. For instance, the area of triangle ijk must be nonnegative. We obtain $2(N-2)$ such conditions in total. Gradients of these geometric constraints can be readily evaluated once grid sensitivity fields become available.

Finally, we consider boundary conditions for solution of the Navier-Stokes equations:

- the inflow velocity is uniform and the attack angle is α to the horizontal, which leads to

$$v_x = U \cos\alpha , \qquad v_y = U \sin\alpha ,$$

  where U is the magnitude of the inflow velocity;
- a no-slip condition is assumed on the obstacle boundary, i.e.,

$$v_x = v_y = 0 ;$$

- the outflow is traction-free, namely,

$$t_x = t_y = 0 .$$

These conditions are invariant to modification of the obstacle shape. Thus, as in the previous example, discrete design sensitivities of these boundary conditions vanish. The line integrals in (3.7) and the corresponding sensitivity equations vanish under this set of boundary conditions. The fluid is assumed to be still initially.

In the specific problem solved below, the obstacle surfaces are represented by two fourth-order Bezier curves. The initial shape is a circle, for which the 20 Bezier coefficients are tabulated in Table 6.2 under the heading "initial." Figure 6.2 shows the initial computational domain and mesh. The mesh has 7,373 nodes, 3,539 quadratic triangular elements, and 15,581 velocity and pressure unknowns. The diameter of the cylinder is $D = 1$.

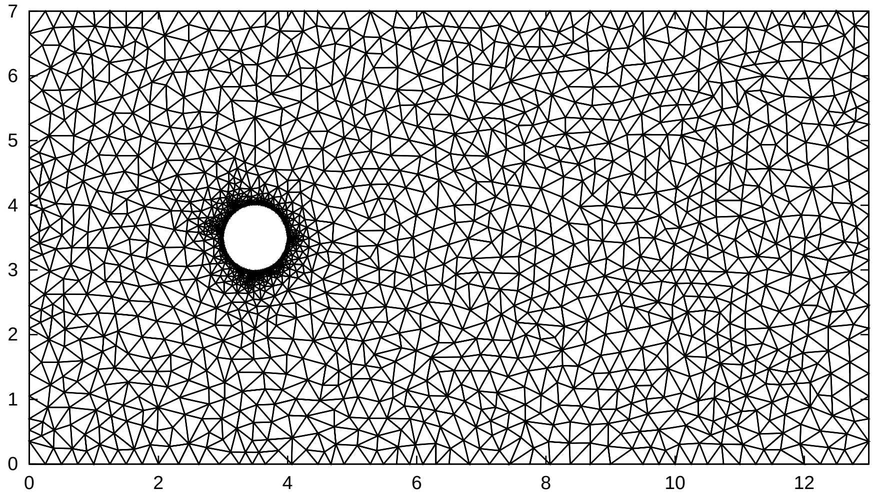

FIG. 6.2. *Computational domain and mesh for flow around obstacle.*

TABLE 6.2
Bezier coefficients for initial and optimal shapes for flow around a 2D obstacle.

Variable	Initial	Stage I	Stage II
p_0^t	3.00	3.00000000	3.00000000
p_1^t	3.03	3.02013099	3.12910217
p_2^t	3.50	3.69794100	3.70840703
p_3^t	3.97	4.34598915	4.36940633
p_4^t	4.00	4.51956184	4.96051309
q_0^t	3.50	3.50000000	3.50000000
q_1^t	4.00	3.69582998	3.59317261
q_2^t	4.20	3.53482103	3.61930290
q_3^t	4.00	3.71255744	3.58691209
q_4^t	3.50	3.46783775	3.48024608
p_0^b	3.00	3.00000000	3.00000000
p_1^b	3.03	2.96428727	2.97515449
p_2^b	3.50	3.53084206	3.52474404
p_3^b	3.97	3.98254581	3.96751124
p_4^b	4.00	4.51956184	4.96051309
q_0^b	3.50	3.50000000	3.50000000
q_1^b	3.00	3.31686601	3.42029354
q_2^b	2.80	3.37770498	3.34979508
q_3^b	3.00	3.35356769	3.31674445
q_4^b	3.50	3.46783775	3.48024608

We first compare vortex shedding frequencies computed with our code and those obtained experimentally. This step serves two purposes: validation of our program and identification of an appropriate time step. Several flow problems with different Reynolds numbers and time steps are solved:

- $Re = 1000$ and $\Delta t = 0.125$,
- $Re = 500$ and $\Delta t = 0.5$, and
- $Re = 200$ and $\Delta t = 0.125$.

All Reynolds numbers in this section are based on the freestream velocity and the diameter of the cylinder, that is the initial shape. Scaled viscous dissipation functions

$\int_\Omega \Phi d\Omega / \int_\Omega d\Omega$ are plotted in Figure 6.3. Clearly, the functions begin to oscillate

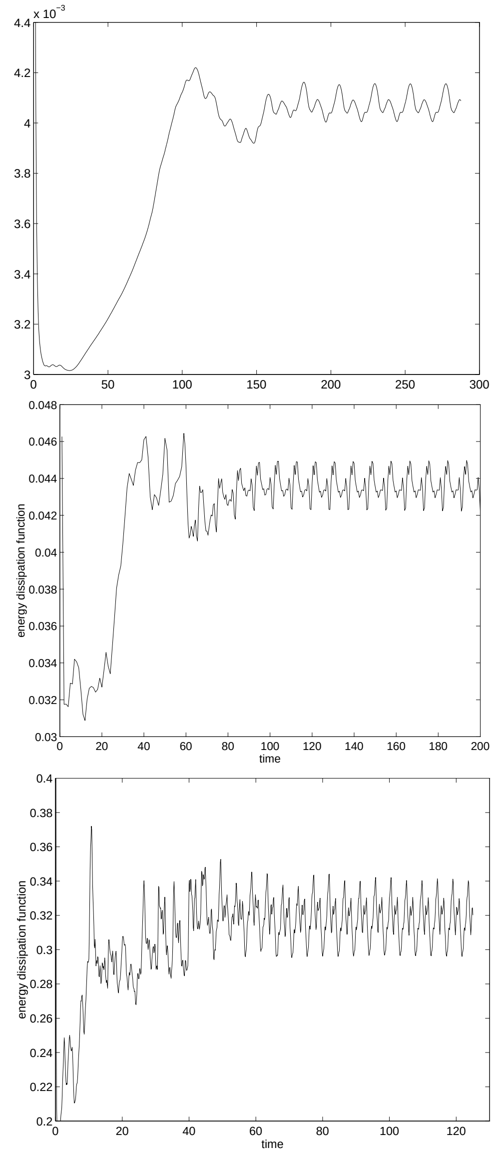

FIG. 6.3. *Time histories of the scaled dissipation function* Φ_s *for* $Re = 200$ *(top),* 500 *(middle), and* 1000 *(bottom).*

after some time. The oscillation frequencies are equal to those of vortex shedding. Table 6.3 compares frequencies obtained numerically and experimentally, and we see very good agreement between the two. Strouhal numbers in Table 6.3 are from [6]. In all calculations, we have used the upper bounds of the Strouhal number, which correspond to a cylinder with a smooth surface.

At Reynolds number 500, $\int_\Omega \Phi d\Omega / \int_\Omega d\Omega$ oscillates with a period of about 9. Thus, it is reasonable, with a second-order method such as Crank-Nicolson, to use a time step $\Delta t = 0.5$ for the flow simulation. If the characteristic time for objective

TABLE 6.3
Experimental and numerical vortex shedding frequencies for flow around a cylinder.

Reynolds number	Strouhal number	Experimental	Numerical
200	0.20	0.039	0.041
500	0.22	0.110	0.114
1000	0.21	0.210	0.219

function evaluation is taken as one vortex shedding period, then we may integrate $\int_\Omega \Phi d\Omega / \int_\Omega d\Omega$ over $[t_L, t_M]$ to get $\widehat{\Phi}_s$, where $t_L = 90$ and $t_M = 99$. Several snapshots of flow streamlines are shown in Figure 6.4. Gradual formation of the von Kármán

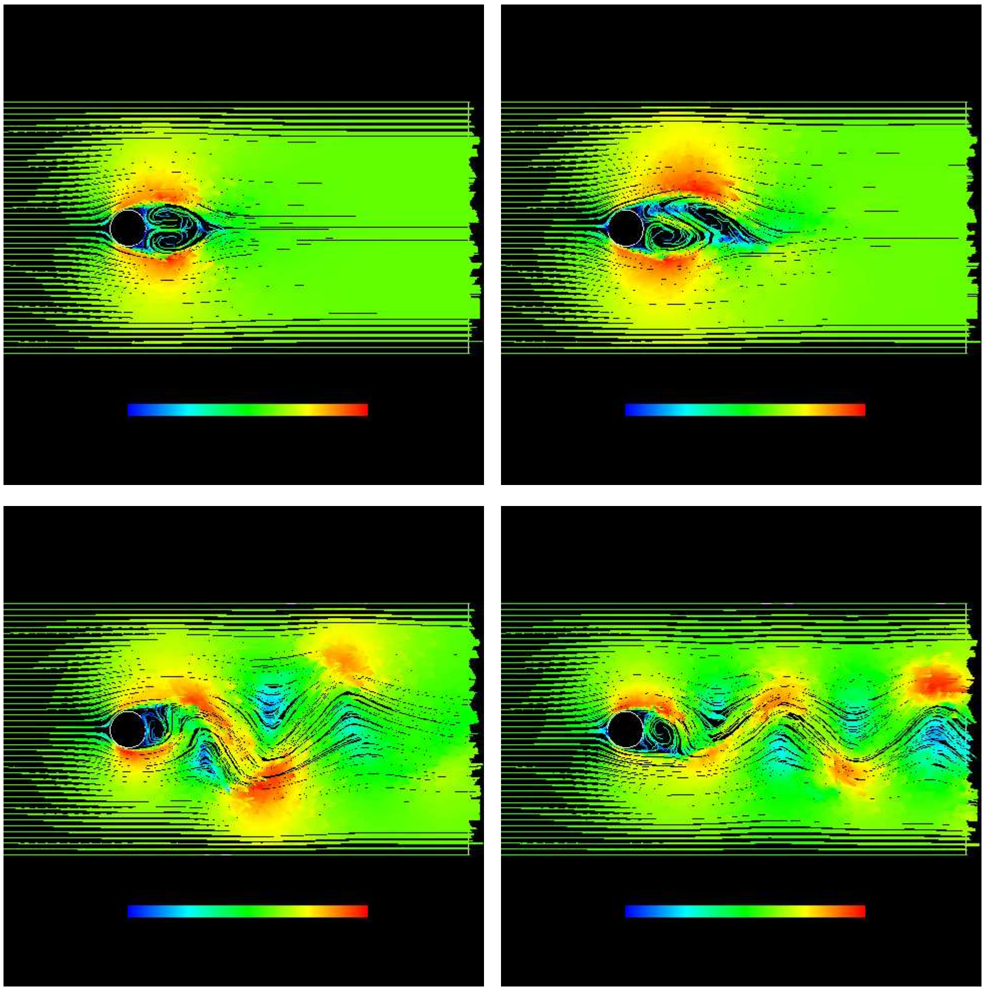

FIG. 6.4. *Streamlines for 2D obstacle problem at $t = 10$ (upper left), 20 (upper right), 30 (bottom left), and 100 (bottom right) for the initial shape. $Re = 500$.*

vortex street is evident from the figure.

Optimization constraints include the minimum area requirement $S \geq 0.3$, and constraints on the position of leading and trailing edges:

$$p_0^t = 3.0, \quad q_0^t = 3.5, \quad 3.5 \leq p_4^t \leq 5.0, \quad \text{and} \quad 1.0 \leq q_4^t \leq 2.0\,.$$

The objective is to minimize $\widehat{\Phi}_s$ at $Re = 500$. The optimization process is divided into two stages: in Stage I, the mesh is updated by a complete remeshing using *Triangle*, and internal grid sensitivities are assigned zero values; in Stage II, we begin with the optimal shape from Stage I, but mesh updating and sensitivity analysis are both carried out by the elastic mesh movement algorithm. Table 6.2 gives the optimal Bezier coefficients after each stage of optimization under "Stage I" and "Stage II." Figure 6.5 shows the evolution of shapes—from the initial to the optimum of Stage I,

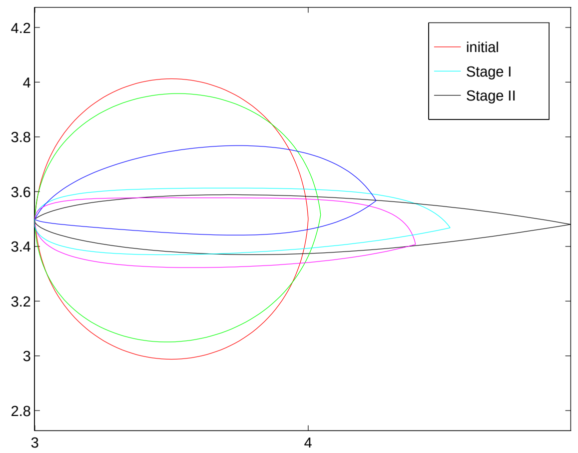

FIG. 6.5. *Evolution of shapes for 2D obstacle problem, Re = 500.*

and then to the optimum of Stage II. Shape modification is very large in Stage I, and thus remeshing is the better strategy for updating the mesh. But this results in (small) discontinuities in the objective function and optimization finally terminates due to an unsuccessful line search. The shape obtained is a not quite symmetric airfoil. Better shapes are possible; a smooth mesh movement scheme as well as consistent derivatives are required if the optimizer is to find them. Shape change is relatively small in Stage II, so it is appropriate to enlist the elastic mesh movement scheme for both mesh updating and sensitivity analysis. Still, we have to remesh three times in this stage of optimization due to invalid meshes produced by the elastic mesh movement algorithm. The final optimal shape of Stage II encounters the area bound of 0.3, and the horizontal coordinate of the trailing edge nearly reaches its upper bound of 5. The resulting slender and almost symmetric profile is what we might expect for the shape of a zero angle of attack body that minimizes dissipation while meeting constraints on area and chord length.

The optimization takes about 100 CPU hours to complete. The objective function value and Euclidean norm of the Kuhn-Tucker optimality conditions (KTO) for the initial, Stage I optimal, and Stage II optimal shapes are given in Table 6.4. In addition, the table lists the number of optimization iterations and number of complete flow solutions taken to reach the optimal solution for each stage. The optimum represents an 82% reduction in the viscous energy dissipation relative to the initial circular shape. Several snapshots of flow streamlines corresponding to the Stage II optimal shape are shown in Figure 6.6. The figure shows that the flowfield is largely time-independent, vortex shedding is eliminated, and the wake region is greatly reduced

TABLE 6.4
Solution parameters, shape optimization of flow around a 2D obstacle.

Stage	Number of iterations	Number of flow solutions	Objective function	KTO
initial	–	–	0.42022025	0.3051
Stage I	16	23	0.14259630	1.375e-3
Stage II	19	27	0.07740351	9.250e-6

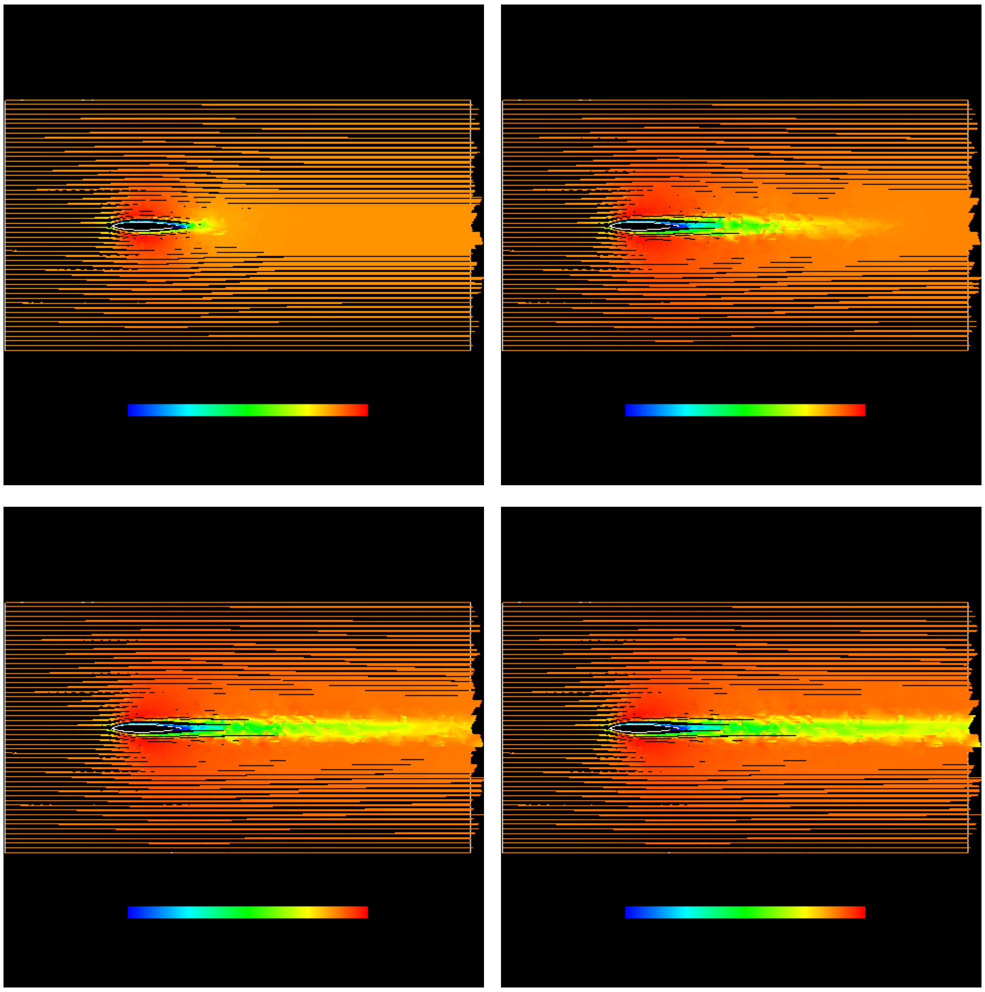

FIG. 6.6. *Streamlines at $t = 2.5$ (upper left), 12.5 (upper right), 22.5 (bottom left), and 32.5 (bottom right) for the optimal shape of the 2D obstacle problem. $Re = 500$. The optimum represents a 82% reduction in viscous energy dissipation over the initial shape.*

in size. Note that the optimizer has found a symmetric optimum shape, despite the fact that symmetry has not been imposed as a constraint in this problem, and the unstructured mesh produced by *Triangle* is not symmetric.

7. Final remarks. We have considered the problem of shape optimization of two-dimensional flows governed by the time-dependent Navier-Stokes equations. For this problem we have proposed computational strategies with respect to optimization method, sensitivity method, and unstructured meshing scheme. We have argued that, despite their superiority for steady Navier-Stokes flow optimization, reduced SQP methods are too memory-intensive for the time-dependent problem. Instead, we have advocated a combination of GRG (for the flow equation constraints) and SQP (for the remaining inequality constraints). With respect to sensitivity method, we have favored discrete sensitivities, which can be implemented with little additional storage or work beyond that required for solution of the flow equations, and thus possess a distinct advantage over discretized continuous sensitivities, which require knowledge of the entire time history of the flow variables. Finally, we have taken a two-phase approach to unstructured meshing and grid sensitivities. Far from the optimum, we have remeshed completely using an unstructured mesh generator to accommodate the large shape changes that are anticipated in this phase, while an inconsistent but easily computed form of grid sensitivities has been employed. Close to the optimum, where differentiability of the mesh movement scheme and consistency of grid sensitivities are desirable, we have used elastic mesh movement to generate meshes corresponding to new shapes. Elastic mesh movement is valid only for small shape changes but is differentiable and permits computation of exact grid sensitivities in a straightforward manner.

Two examples characterized by viscous dissipation objective functions illustrated the approach. The latter example involved finding the geometrically-constrained optimum shape of an initially circular cylinder placed in a uniform flow at $Re = 500$ and described by 20 Bezier shape variables and over 15,000 spatial flow variables. The optimum shape, a symmetric airfoil, was found in a total of 35 iterations, 50 flow solutions, and 100 CPU hours (on a DEC Alpha system). The CPU time taken represents over an order of magnitude increase over a similar, but steady, problem—this is the price paid for optimizing with a time-dependent model of the flow.

What can be done to reduce computational work (without significantly increasing storage)? First, in order to reduce work per optimization iteration, one might try to strike a compromise between GRG (no state variable time history stored) and SQP (entire time history stored) by storing an approximation of the state variables at *a portion* (depending on available storage) of the time steps; iterating on the nonlinear equations that update the states can then be integrated with updates of the design variables, à la SQP. For example, one might use a multiple shooting method for solving the system of DAEs (as was done in [36] for a time-dependent heat equation), or a very high order method in time. Second, with the goal of reducing the number of optimization iterations, one might pursue incorporating second order derivatives for use in a Newton method. Our experience (for a problem in optimal boundary control of Navier-Stokes flows [14]) is that use of exact Hessians cuts the number of iterations significantly (in half or third); however, the reduction in CPU time is not as substantial, due to the additional work involved in computing second derivatives. Finally, CPU time can certainly be reduced through parallel computing. In earlier work, we devised and implemented parallel RSQP algorithms for full potential flow optimal design problems [33], [15]; recently, parallel implementations of GRG-based optimal design methods for Euler flows have appeared [38]. We have recently implemented a parallel SQP method for general 3D unstructured mesh shape optimization problems on a Cray T3E [29], and are currently tailoring the code to incompressible Navier-Stokes flows. Speedups induced by domain-based parallelism seem to offer the best hope for significant reductions in computing time necessary for practical shape optimization of time-dependent Navier-Stokes flows.

Acknowledgments. We thank Greg Burgreen for sharing his experiences in shape optimization and geometry modeling with us.

REFERENCES

[1] W. K. Anderson and V. Venkatakrishnan, *Aerodynamic design optimization on unstructured grids with a continuous adjoint formulation*, Tech. Rep. 97-9, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, Jan. 1997.

[2] J. Antaki, O. Ghattas, G. Burgreen, and B. He, *Computational flow optimization of rotory blood pump components*, Artifical Organs, 19 (1995), pp. 608–615.

[3] J. Argyris, J. Doltsinis, H. Fischer, and H. Wüstenberg, *$T\alpha$ $\pi\alpha\nu\tau\alpha$ $\rho\varepsilon\iota$*, Computer Methods in Applied Mechanics and Engineering, 51 (1985), pp. 289–362.

[4] I. Babuška and A. Aziz, *On the angle condition in the finite element method*, SIAM Journal on Numerical Analysis, 13 (1976), pp. 214–227.

[5] L. T. Biegler, J. Nocedal, and C. Schmid, *A reduced Hessian method for large-scale constrained optimization*, SIAM Journal on Optimization, 5 (1995), pp. 314–347.

[6] R. D. Blevins, *Flow-Induced Vibration*, Van Nostrand Reinhold, 1990.

[7] R. A. Brockman, *Geometric sensitivity analysis with isoparametric finite element*, Communications in Applied Numerical Methods, 3 (1987), pp. 495–499.

[8] G. W. Burgreen and J. F. Antaki, *CFD-based design optimization of a three-dimensional rotary blood pump*, in Proceedings of the Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, September 1996.

[9] R. H. Byrd and J. Nocedal, *An analysis of reduced Hessian methods for constrained optimization*, Mathematical Programming, 49 (1991), pp. 285–323.

[10] T. Davis, *Users' guide for the unsymmetric pattern multifrontal package (UMFPACK)*, Tech. Rep. TR-93-020, University of Florida, Gainesville, FL, 1993.

[11] J. E. Dennis and R. M. Lewis, *A comparison of nonlinear programming approaches to an elliptic inverse problem and a new domain decomposition approach*, Tech. Rep. TR94-33, Department of Computational and Applied Mathematics, Rice University, 1994.

[12] J. Elliott and J. Peraire, *Practical 3D aerodynamic design and optimization using unstructured meshes*, in Proceedings of the Sixth AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, September 1996.

[13] I. Fried, *Condition of finite element matrices generated from nonuniform meshes*, AIAA Journal, 10 (1972), pp. 219–221.

[14] O. Ghattas and J. Bark, *Optimal control of two- and three-dimensional Navier-Stokes flows*, Journal of Computational Physics, (1997). In press. Available from http://www.cs.cmu.edu/~oghattas.

[15] O. Ghattas and C. E. Orozco, *A parallel reduced Hessian SQP method for shape optimization*, in Multidisciplinary Design Optimization: State-of-the-Art, N. Alexandrov and M. Hussaini, eds., SIAM, 1997, pp. 133–152.

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

[17] M. D. Gunzburger, L. S. Hou, and T. P. Svobodny, *Optimal control and optimization of viscous, incompressible flows*, in Incompressible Computational Fluid Dynamics, M. D. Gunzburger and R. A. Nicolaides, eds., Cambridge, 1993, ch. 5, pp. 109–150.

[18] R. T. Haftka and Z. Gürdal, *Elements of Structural Optimization*, Kluwer Academic, third ed., 1991.

[19] E. Haug and J. Arora, *Applied Optimal Design*, Wiley Interscience, 1979.

[20] B. He, *Shape Optimization of Navier-Stokes Flows, with Application to Design of Artificial Heart Devices*, PhD thesis, Carnegie Mellon University, 1996.

[21] M. Heinkenschloss, *Numerical solution of optimal control problems governed by the Navier-Stokes equations using sequential quadratic programming.* In preparation.

[22] ———, *Formulation and analysis of a sequential quadratic programming method for the optimal Dirichlet boundary control of Navier-Stokes flow*, Tech. Rep. TR97-14, Department of Computational and Applied Mathematics, Rice University, May 1997.

[23] R. D. Joslin, M. D. Gunzburger, R. A. Nicolaides, G. Erlbacher, and M. Y. Hussaini, *Self-contained automated methodology for optimal flow control*, AIAA Journal, 35 (1997), pp. 816–824.

[24] K. Kunisch and E. W. Sachs, *Reduced SQP methods for parameter identification problems*, SIAM Journal on Numerical Analysis, 29 (1992), pp. 1793–1820.

[25] F.-S. Kupfer, *An infinite-dimensional convergence theory for reduced SQP methods in Hilbert space*, SIAM Journal on Optimization, 6 (1996), pp. 126–163.

[26] F.-S. Kupfer and E. W. Sachs, *Numerical solution of a nonlinear parabolic control problem by a reduced SQP method*, Computational Optimization and Applications, 1 (1992), pp. 113–135.

[27] R. M. Lewis, *A nonlinear programming perspective on sensitivity calculations for systems governed by state equations*, Tech. Rep. 97-12, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, Feb. 1997.

[28] R. M. Lewis, *Practical aspects of variable reduction formulations and reduced basis algorithms in multidisciplinary design optimization*, in Multidisciplinary Design Optimization: State-of-the-Art, N. Alexandrov and M. Hussaini, eds., SIAM, 1997, pp. 172–188.

[29] I. Malcevic, *Large-scale unstructured mesh shape optimization on parallel computers*, Master's thesis, Carnegie Mellon University, 1997.

[30] J. Newman and A. Taylor, *Three-dimensional aerodynamic shape sensitivity analysis and design optimization using the Euler equations on unstructured grids*, 1996. AIAA Paper 96–2464.

[31] J. Newman, A. Taylor, and G. Burgreen, *An unstructured grid approach to sensitivity analysis and shape optimization using the Euler equations*, 1995. AIAA Paper 95–1646.

[32] J. Nocedal and M. Overton, *Projected Hessian updating algorithms for nonlinearly constrained optimization*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 821–850.

[33] C. E. Orozco and O. Ghattas, *Massively parallel aerodynamic shape optimization*, Computing Systems in Engineering, 1–4 (1992), pp. 311–320.

[34] ———, *Infeasible path optimal design methods, with application to aerodynamic shape optimization*, AIAA Journal, 34 (1996), pp. 217–224.

[35] ———, *A reduced SAND method for optimal design of nonlinear structures*, International Journal for Numerical Methods in Engineering, (1997). To appear.

[36] L. Petzold, J. B. Rosen, P. E. Gill, L. O. Jay, and K. Park, *Numerical optimal control of parabolic PDEs using DASOPT*, Tech. Rep. NA 96-3, Department of Mathematics, University of California, San Diego, 1996.

[37] O. Pironneau, *On optimum design in fluid mechanics*, Journal of Fluid Mechanics, 64 (1974), pp. 97–110.

[38] J. Reuther, J. J. Alonso, M. J. Rimlinger, and A. Jameson, *Aerodynamic shape optimization of supersonic aircraft configurations via an adjoint formulation on parallel computers*, Tech. Rep. 96.17, RIACS, NASA Ames Research Center, Sept. 1996.

[39] U. Ringertz, *Optimal design of nonlinear shell structures*, Tech. Rep. FFA TN 91-18, The Aeronautical Research Institute of Sweden, 1991.

[40] ———, *An algorithm for optimization of nonlinear shell structures*, International Journal for Numerical Methods in Engineering, 38 (1995), pp. 299–314.

[41] A. Sahrapour, N. U. Ahmed, and S. Tavoularis, *Boundary control of the Navier-Stokes equations with potential applications to artificial hearts*, in Proceedings of the 2nd Conference of the CFD Society of Canada, J. Gottlieb and C. Ethier, eds., Toronto, 1994, pp. 387–394.

[42] K. Schittkowski, *NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems*, Annals of Operations Research, 5 (1985/86), pp. 485–500.

[43] V. H. Schulz and H. G. Bock, *Partially reduced SQP methods for large-scale nonlinear optimization problems*, in Proceedings of the Second World Congress of Nonlinear Analysis, Elsevier Verlag, 1997.

[44] J. R. Shewchuk, *Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator*, in First Workshop on Applied Computational Geometry, Association for Computing Machinery, May 1996, pp. 124–133.

[45] D. P. Young, W. P. Huffman, R. G. Melvin, M. B. Bieterman, C. L. Hilmes, and F. T. Johnson, *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, September 1994.