PARALLEL ONE-SHOT LAGRANGE-NEWTON-KRYLOV-SCHWARZ ALGORITHMS FOR SHAPE OPTIMIZATION OF STEADY INCOMPRESSIBLE FLOWS*

RONGLIANG CHEN† AND XIAO-CHUAN CAI‡

Abstract. We propose and study a new parallel one-shot Lagrange-Newton-Krylov-Schwarz (LNKSz) algorithm for shape optimization problems constrained by steady incompressible Navier-Stokes equations discretized by finite element methods on unstructured moving meshes. Most existing algorithms for shape optimization problems solve iteratively the three components of the optimality system: the state equations for the constraints, the adjoint equations for the Lagrange multipliers, and the design equations for the shape parameters. Such approaches are relatively easy to implement, but generally not easy to converge as they are basically nonlinear Gauss-Seidel algorithms with three large blocks. In this paper, we introduce a fully coupled, or the so-called one-shot, approach which solves the three components simultaneously. First, we introduce a moving mesh finite element method for the shape optimization problems in which the mesh equations are implicitly coupled with the optimization problems. Second, we introduce a LNKSz framework based on an overlapping domain decomposition method for solving the fully coupled problem. Such an approach doesn't involve any sequential steps that are necessary for the Gauss-Seidel type reduced space methods. The main challenges in full space approaches are that the corresponding nonlinear system is much harder to solve because it is two to three times larger and its indefinite Jacobian problems are also much more ill-conditioned. Effective preconditioning becomes the most important component of the method. Numerically, we show that LNKSz deals with these challenges quite well. As an application, we consider the shape optimization of an artery bypass problem in 2D. Numerical experiments show that our algorithms perform well on supercomputers with hundreds of processors.

Key words. Shape optimization, one-level additive Schwarz preconditioner, parallel computing, one-shot method, finite element, inexact Newton.

AMS subject classifications. 49K20, 49Q10, 65F08, 65F10, 65N30, 65N55, 65Y05, 76D05, 90C06, 90C30, 93C20

1. Introduction.

Shape optimization, or optimal shape design, aims to optimize an objective function by changing the shape of the computational domain. In the past few years, shape optimization problems have received considerable attentions. On the theoretical side there are several publications dealing with the existence of solution and the sensitivity analysis to the problems; see e.g., [19, 20, 21, 27, 39, 46] and references therein. On the practical side, optimal shape design has played an important role in many industrial applications, for example, aerodynamic shape design [25, 33, 34, 35, 44], artery bypass design [2, 3, 6, 7, 40, 43], microfluidic biochip design [4, 24, 38] and so on. Most of these optimization problems have constraints imposed by partial differential equations (PDE) or other physical and geometrical conditions. They are considerably more difficult and expensive to solve than the corresponding simulation problems, and often require large scale parallel computers for their memory capacity and processing speed. In this paper, we propose a general framework for the parallel solution of shape optimization problems, and study it in detail for the optimization of an artery bypass problem. The key element of the proposed framework is the domain decomposition preconditioner which ensures the convergence and also

*The research was supported in part by NSF under DMS-0913089 and DOE under DE-SC0001774. The first author was supported in part by NSF of China under No. 10971058.

†Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, P. R. China (rl.chen@siat.ac.cn)

‡Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309, USA (cai@cs.colorado.edu)

the parallel scalability of the method.

Traditional approaches for solving shape optimization problems view the state variables and the design variables as dependent variables and only regard the design variables as the optimization variables; i.e., the state variables are considered as functions of the design variables. These methods split the optimality system into three components: the state equations for the constraints, the adjoint equations for the Lagrange multipliers, and the design equations for the shape design parameters. A typical algorithm starts from an initial given shape and then goes through three steps:

(1) Solve the state equations.
(2) Solve the adjoint equations for the sensitivity analysis.
(3) Solve the design equations to update the shape; return to (1) unless a stopping condition is met.

The three steps have to be performed sequentially [2, 3, 6, 7, 26, 32, 34, 35, 43]. These algorithms are often called *nested analysis and design* (NAND) and they involve an iterative algorithm and need to solve the state equations repeatedly, which make this computing extremely time consuming and in some cases not practical. The attractiveness of NAND is that one can apply well developed PDE solvers and optimization solvers directly and it requires less memory. Sometimes convergence is not easy to achieve. Preconditioning can be applied to the subsystems, such as the state equations, but can not be applied to the whole system. An alternative to these approaches are the *simultaneous analysis and design* (SAND), or the so-called one-shot, approaches. In the one-shot approaches the state variables and design variables are viewed as independent variables in the optimization problem and the numerical solution of the state equations is an integral part of the optimization routine; i.e., retain the state equations as equality constraints in the optimization problem and the state variables are regarded as optimization variables [4, 5, 8, 22, 25, 38, 44]. The one-shot methods update the shape and the state simultaneously and they only need to solve the state equations one time, which saves the computational work and time considerably. The main challenges in the one-shot approaches are that the nonlinear system is two to three times larger, and the corresponding indefinite Jacobian system is a lot more ill-conditioned and also much larger. A review of NAND and SAND can be found in [9].

As computers become more powerful in processing speed and memory capacity, one-shot methods become more attractive due to their higher degree of parallelism and better scalability. To answer the challenges facing in the one-shot methods, one needs to design a preconditioner that can substantially reduce the condition number of the large fully coupled system and, at the same time, provides the scalability for parallel computing. There are several recent publications on one-shot methods for PDE constrained optimization problems. In [22], a parallel reduced Hessian sequential quadratic programming (RSQP) method was introduced for an aerodynamic design problem, which uses a sequential quadratic programming (SQP) method (which can be viewed as Lagrange-Newton method) to solve the discretized shape optimization problem and employs a reduced space method (which can be viewed as a Schur complement method) to solve the linearized Jacobian system in the SQP steps. And later in [44], RSQP was used as a preconditioner for a defect correction method to solve a similar problem. In [12, 13], a parallel *full space method* was introduced for boundary control problems, where a Newton-Krylov method is used together with Schur complement type preconditioners which split the Jacobian system into three sub-systems that are solved one after another. In [41, 42], an overlapping Schwarz based Lagrange-

Newton-Krylov approach was investigated for some boundary control problems. In these papers, the one-shot approach was shown to be very successful. A review of these methods can be found in [1]. As far as we know no one has studied shape optimization problems using Lagrange-Newton-Krylov-Schwarz (LNKSz) method, which has the potential to solve very large problems on machines with a large number of processors. The previous work in [41, 42] doesn't consider the change of the computational domain which makes the study much more difficult and interesting.

We restrict ourselves to the discretize-then-optimize approach which first discretizes the continuous shape optimization problem with a finite element method on a moving mesh to obtain a finite dimensional equality constrained optimization problem and then use an optimization method to solve this finite dimensional optimization problem. In the proposed new approach, the moving mesh equations are viewed as constraints of the optimization problem and the moving mesh variables are viewed as optimization variables which are independent of the state and design variables. This makes our algorithms very simple, and it doesn't require a sensitivity analysis as in the traditional algorithms [46]. For solving the finite dimensional equality constrained optimization problem we use a Lagrange multiplier method to reduce the optimization problem to a nonlinear equations problem, and then investigate a Newton-Krylov-Schwarz method [15] to solve the fully coupled nonlinear Karush-Kuhn-Tucker (KKT) system. In fact the linearized KKT system at each Newton step can be viewed as the KKT system of a related quadratic programming problem, so this method can be considered as a SQP method which is one of the most successful method for solving constrained nonlinear optimization problems [14]. We focus on the parallel additive Schwarz preconditioners which are quite efficient for the rather difficult problem considered in this paper.

In medical practice, artery bypass surgery aims to create new routes around narrowed and blocked arteries that allow the blood to flow smoothly along the artery. Artery bypass design is to find the best shape of the bypass. In large arteries, blood flows can be described by the incompressible Navier-Stokes equations and an optimal design can be obtained by solving a shape optimization problem. Several publications have studied this problem. In [6, 7, 43], a Newtonian flow modeled by the Stokes equations is used to study the design of the incoming branch of the bypass (the toe) into the coronary. A non-Newtonian flow described by the Navier-Stokes equations is used to study the design of the entire bypass in [2, 3, 40]. In our numerical experiments, a Newtonian flow governed by the Navier-Stokes equations is used to study the design of the entire bypass.

The rest of the paper is organized as follows. In Section 2, we describe the mathematical formulation of the problem and introduce a moving finite element discretization of the problem. Then, in Section 3, we give a description of the parallel one-shot Lagrange-Newton-Krylov-Schwarz algorithms. Some numerical experiments are given in Section 4 and some concluding remarks are given in Section 5. In the Appendix, we provide some details about the computation of the nonlinear KKT system and the construction of the analytic Jacobian matrix.

2. Shape optimization on a moving mesh. We consider a class of shape optimization problems governed by the stationary incompressible Navier-Stokes equations defined in a two dimensional domain Ω_α. Here the subscript α is a shape function which determines the shape of the computational domain. First we recall some definitions: For a scalar function ϕ, a vector-valued function $\mathbf{u} = (u, v)$ and matrices

$\mathbf{A} = (a_{ij})_{n\times n}$, $\mathbf{B} = (b_{ij})_{n\times n}$, we denote

$$\nabla\phi := \left(\frac{\partial\phi}{\partial x}, \frac{\partial\phi}{\partial y}\right), \quad \nabla\cdot\mathbf{u} := \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}, \quad \text{and} \quad \mathbf{A}:\mathbf{B} := \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}b_{ij}.$$

The velocity-pressure formulation of the stationary Navier-Stokes equations can be written as (**state equations**):

$$\left\{\begin{array}{rcll} -\mu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &=& \mathbf{f} & \text{in} \quad \Omega_\alpha, \\ \nabla\cdot\mathbf{u} &=& 0 & \text{in} \quad \Omega_\alpha, \\ \mathbf{u} &=& \mathbf{g} & \text{on} \quad \Gamma_{inlet}, \\ \mathbf{u} &=& \mathbf{0} & \text{on} \quad \Gamma_{wall}, \\ \mu\dfrac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\cdot\mathbf{n} &=& \mathbf{0} & \text{on} \quad \Gamma_{outlet}, \end{array}\right. \tag{2.1}$$

where $\mathbf{u} = (u, v)$ and p represent the velocity and pressure, $\mathbf{n}$ is the outward unit normal vector on the domain boundary $\partial\Omega_\alpha$ and μ is the kinematic viscosity. Γ_{inlet}, Γ_{outlet} and Γ_{wall} represent the inlet, outlet and wall boundaries, respectively and $\Gamma_{optimized}$, which needs to be designed, is also a wall boundary for the velocity $\mathbf{u}$; see Fig. 2.1. $\mathbf{f}$ is the given body force and $\mathbf{g}$ is the given velocity at the inlet Γ_{inlet}.

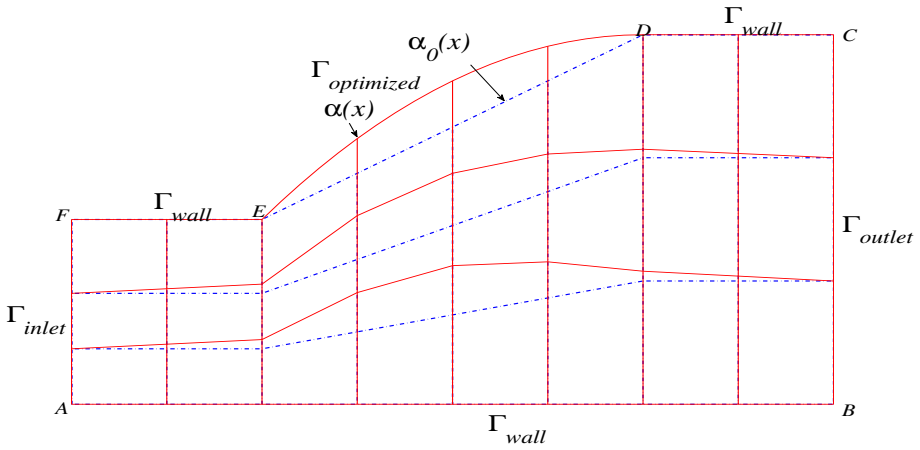

FIG. 2.1. *The initial flow domain Ω_{α_0} (blue dashed line) and deformed flow domain Ω_α (red solid line) over a simple mesh. The boundary $\Gamma_{optimized}$ (ED) denotes the part of the boundary whose shape is to be determined by the optimization process.*

Our goal is to computationally find the optimal shape for part of the boundary $\partial\Omega_\alpha$ such that a given objective function J_o is optimized. We represent the part of the boundary by a smooth function $\alpha(x)$ determined by a set of parameters $\mathbf{a} = (a_1, a_2, ..., a_p)$. By changing the shape of the computational domain defined by $\alpha(x)$, one can optimize certain properties of the flow according to some objective functions; defined as, for example, the total drag [34], the total energy dissipation [19] or the total vorticity in the whole or part of the flow domain [43]. In this paper, we focus on the minimization of the energy dissipation in the whole flow field and use the integral of the squared energy deformation as the objective function [19] (**objective function**):

$$J_o(\mathbf{u},\alpha) = 2\mu\int_{\Omega_\alpha}\epsilon(\mathbf{u}):\epsilon(\mathbf{u})dxdy + \frac{\beta}{2}\int_I(\alpha'')^2dx, \tag{2.2}$$

where $\epsilon(\mathbf{u}) = \frac{1}{2}(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)$ is the deformation tensor for the flow velocity $\mathbf{u}$ and β is a nonnegative constant. $I = [a, b]$ is an interval in which the shape function $\alpha(x)$ is defined. The last term of (2.2) is a regularization term which provides the regularity of the boundary of the domain Ω_α. In some approaches [2, 3], some restrictions of the geometry are included in the constraints, e.g., certain thickness or volume, instead of a regularization term in the objective function. In this paper, we use a regularization term as in [19].

The continuous shape optimization problem can be described as follows.

$$\begin{aligned}
&\min_{\mathbf{u},\alpha} \quad J_o(\mathbf{u},\alpha) = 2\mu\int_{\Omega_\alpha}\epsilon(\mathbf{u}):\epsilon(\mathbf{u})dxdy + \frac{\beta}{2}\int_I(\alpha'')^2dx\\
&\text{subject to}\\
&\begin{cases}
-\mu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &= \mathbf{f} \quad \text{in} \quad \Omega_\alpha,\\
\nabla\cdot\mathbf{u} &= 0 \quad \text{in} \quad \Omega_\alpha,\\
\mathbf{u} &= \mathbf{g} \quad \text{on} \quad \Gamma_{inlet},\\
\mathbf{u} &= \mathbf{0} \quad \text{on} \quad \Gamma_{wall},\\
\mu\dfrac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\cdot\mathbf{n} &= \mathbf{0} \quad \text{on} \quad \Gamma_{outlet},\\
\alpha(a) &= z_1, \qquad \alpha(b) = z_2.
\end{cases}
\end{aligned} \tag{2.3}$$

The last two equations indicate that the optimized boundary should be connected to the rest of the boundary and z_1 and z_2 are two given constants [21].

For PDE constrained optimization problems, there are two basic approaches: *optimize-then-discretize* (OTD) and *discretize-then-optimize* (DTO). As the differentiation and discretization steps do not commute, the two approaches are not the same and the general believe is that no approach has a clear advantage over the other in terms of the accuracy of the solution [19, 41]. In this paper, we use the DTO approach which turns out to be easier to formulate since the boundary conditions for the adjoint equations is rather difficult to derive when the shape derivatives are involved. In the DTO approach, such boundary conditions are not necessary since the adjoint problem is obtained algebraically. We next introduce a finite element discretization of the shape optimization problem (2.3). To obtain the weak form of the state equations in Ω_α, we define the function spaces

$$\begin{aligned}
\mathcal{U} &= \{\mathbf{u}|\mathbf{u}\in[H^1(\Omega_\alpha)]^2 \quad \mathbf{u} = \mathbf{g} \quad \text{on} \quad \Gamma_{inlet}\},\\
\mathcal{U}_0 &= \{\mathbf{u}|\mathbf{u}\in[H^1(\Omega_\alpha)]^2 \quad \mathbf{u} = \mathbf{0} \quad \text{on} \quad \Gamma_{inlet}\cup\Gamma_{wall}\}.
\end{aligned}$$

Then the weak form of the state equations can be defined as follows: Find $\mathbf{u}\in\mathcal{U}$, $p\in L^2(\Omega)$ such that

$$\begin{aligned}
\mu\int_{\Omega_\alpha}\nabla\mathbf{u}:\nabla\Phi dxdy + \int_{\Omega_\alpha}(\mathbf{u}\cdot\nabla)\mathbf{u}\cdot\Phi dxdy - \int_{\Omega_\alpha}p\nabla\cdot\Phi dxdy &= \int_{\Omega_\alpha}\mathbf{f}\cdot\Phi dxdy,\\
\int_{\Omega_\alpha}(\nabla\cdot\mathbf{u})\varphi dxdy &= 0,
\end{aligned} \tag{2.4}$$

hold for all $\Phi\in\mathcal{U}_0$ and $\varphi\in L^2(\Omega_\alpha)$.

Since the computational domain of the shape optimization problem changes during the optimization process, the mesh needs to be modified following the computational domain. Generally speaking, there are two strategies to modify the mesh. One is mesh reconstruction which often guarantees a good new mesh but is computationally expensive. The other strategy is moving mesh which changes the locations of the mesh points but keeps the number of mesh points and the connectivity unchanged. This approach is cheaper but the deformed mesh may become ill-conditioned when the boundary variation is large. In our test cases the boundary variations are not very large, so we use the latter strategy. The moving of the mesh is simply described by Laplace's equations ([28]) in this paper. Let α_0 be the initial shape of the boundary and Ω_{α_0} the initial computational domain, see Fig. 2.1. We define $\mathbf{x^0}$ as the coordinate of a point in Ω_{α_0}, $\mathbf{x}$ as the coordinate of the corresponding point in Ω_α and $\delta_{\mathbf{x}} := \mathbf{x} - \mathbf{x^0}$ as the displacement of this point. When the point $\mathbf{x^0}$ moves, we assume the displacement $\delta_{\mathbf{x}}$ satisfies the following equations (**moving mesh equations**):

$$\left\{ \begin{array}{rcll} -\Delta\delta_{\mathbf{x}} & = & \mathbf{0} & \text{in} \quad \Omega_{\alpha_0}, \\ \delta_{\mathbf{x}} & = & \mathbf{g}_\alpha & \text{on} \quad \partial\Omega_{\alpha_0}, \end{array} \right. \tag{2.5}$$

where $\mathbf{g}_\alpha = (g_\alpha^x, g_\alpha^y)$ is the displacement on the boundary and it is determined by the shape function $\alpha(x)$. Note that $\mathbf{g}_\alpha$ is not a given function, but a function obtained automatically during the iterative solution process. For example, in Fig. 2.1, $g_\alpha^x = 0$ and $g_\alpha^y = \alpha(x) - \alpha_0(x)$.

To obtain the weak form of the moving mesh equations (2.5) on domain Ω_{α_0}, we define the function spaces

$$\begin{aligned} \mathcal{X} &= \{\mathbf{x} | \mathbf{x} \in [H^1(\Omega_{\alpha_0})]^2 \quad \mathbf{x} = \mathbf{g}_\alpha \quad \text{on} \quad \partial\Omega_{\alpha_0}\}, \\ \mathcal{X}_0 &= \{\mathbf{x} | \mathbf{x} \in [H^1(\Omega_{\alpha_0})]^2 \quad \mathbf{x} = \mathbf{0} \quad \text{on} \quad \partial\Omega_{\alpha_0}\}. \end{aligned}$$

Then we arrive at the weak form of the moving mesh equations (2.5): Find $\delta_{\mathbf{x}} \in \mathcal{X}$ such that

$$\int_{\Omega_{\alpha_0}} \nabla\delta_{\mathbf{x}} : \nabla\Psi dxdy = 0, \tag{2.6}$$

holds for all $\Psi \in \mathcal{X}_0$. In our numerical experiments, this simple mesh movement scheme performs quite well (see Fig. 4.3). When the mesh deformation is large, sometimes one can use a model based on the linear or nonlinear elasticity equation with locally adjustable stiffness in stead of the Poisson's equation [48]. Changing the moving mesh equation in the algorithmic framework proposed in this paper is straightforward. For more moving mesh strategies see, e.g., [28, 34].

The shape optimization problem (2.3) is discretized with a LBB-stable $Q_2 - Q_1$ finite element method for the state equations (2.1) and a Q_2 finite element method for the moving mesh equations (2.5). We approximate the velocity $\mathbf{u}$, the pressure p and the grid displacement $\delta_{\mathbf{x}}$ in finite-dimensional space as follows

$$\mathbf{u} \approx \sum_i \phi_i \mathbf{u_i}, \quad p \approx \sum_i \varphi_i p_i, \quad \text{and} \quad \delta_{\mathbf{x}} \approx \sum_i \psi_i \delta_{\mathbf{x_i}},$$

where the finite element basis functions ϕ_i, φ_i and ψ_i are piecewise polynomials [23]. The basis functions ϕ_i and φ_i depend on the grid displacement $\delta_{\mathbf{x_i}}$, while ψ_i are independent of $\delta_{\mathbf{x_i}}$, because ϕ_i and φ_i are defined on the domain Ω_α and ψ_i are

defined on Ω_{α_0}. We denote n and m as the total number of nodes for velocity and pressure unknowns, respectively. We also use $\mathbf{u}$, p and $\delta_{\mathbf{x}}$ to denote, respectively, the vector consisting of the nodal values of $\mathbf{u}$, p and $\delta_{\mathbf{x}}$.

Using this approximation, the finite element discretization of the state equations (2.1) is given as follows

$$\left\{ \begin{array}{rcl} \mathbf{Ku}+\mathbf{B(u)u}-\mathbf{Q}p &=& \mathbf{F_f}+\mathbf{F_u}, \\ \mathbf{Q^T u} &=& 0, \end{array} \right. \tag{2.7}$$

where

$$\mathbf{K}=\begin{pmatrix}(k_{ij})_{n\times n} & \mathbf{0}\\ \mathbf{0} & (k_{ij})_{n\times n}\end{pmatrix},\ k_{ij}=\int_{\Omega_\alpha}\left(\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}+\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\right)dxdy,$$

$$\mathbf{Q}=\left((Q^u_{ij})_{m\times n},\ (Q^v_{ij})_{m\times n}\right)^T,\ Q^u_{ij}=\int_{\Omega_\alpha}\frac{\partial\phi_j}{\partial x}\varphi_i dxdy,\ Q^v_{ij}=\int_{\Omega_\alpha}\frac{\partial\phi_j}{\partial y}\varphi_i dxdy,$$

$$\mathbf{F_f}=\begin{pmatrix}(F^1_j)_{n\times 1}\\ (F^2_j)_{n\times 1}\end{pmatrix},\quad F^{\cdot}_j=\int_{\Omega_\alpha} f_{\cdot}\phi_j dxdy.$$

$\mathbf{F_u}$ is the Dirichlet boundary condition for $\mathbf{u}$ and

$$\mathbf{B(u)}=\begin{pmatrix}(B_{ij}(u))_{n\times n} & \mathbf{0}\\ \mathbf{0} & (B_{ij}(u))_{n\times n}\end{pmatrix},$$

$$B_{ij}(u)=\sum_{k=1}^{n}\int_{\Omega_\alpha}\left(\frac{\partial\phi_i}{\partial x}u_k\phi_k+\frac{\partial\phi_i}{\partial y}v_k\phi_k\right)\phi_j dxdy.$$

The discretization of the moving mesh equations reads:

$$\mathbf{D}\delta_{\mathbf{x}}=\mathbf{F_x}, \tag{2.8}$$

where

$$\mathbf{D}=\begin{pmatrix}(D_{ij})_{n\times n} & \mathbf{0}\\ \mathbf{0} & (D_{ij})_{n\times n}\end{pmatrix},\ D_{ij}=\int_{\Omega_{\alpha_0}}\left(\frac{\partial\psi_i}{\partial x}\frac{\partial\psi_j}{\partial x}+\frac{\partial\psi_i}{\partial y}\frac{\partial\psi_j}{\partial y}\right)dxdy,$$

and $\mathbf{F_x}$ is the Dirichlet boundary condition for $\delta_{\mathbf{x}}$. We discretize the objective function as:

$$J_o(\mathbf{u},\mathbf{a})=\mu\mathbf{u^T Ju}+\frac{\beta}{2}\mathbf{J}_\alpha, \tag{2.9}$$

where

$$\mathbf{J}=\begin{pmatrix}(J^{11}_{ij})_{n\times n} & (J^{12}_{ij})_{n\times n}\\ (J^{21}_{ij})_{n\times n} & (J^{22}_{ij})_{n\times n}\end{pmatrix},\ J^{11}_{ij}=\int_{\Omega_\alpha}\left(2\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}+\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\right)dxdy,$$

$$J^{12}_{ij}=J^{21}_{ji}=\int_{\Omega_\alpha}\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial x}dxdy,\ J^{22}_{ij}=\int_{\Omega_\alpha}\left(\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x}+2\frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y}\right)dxdy,$$

and

$$\mathbf{J}_\alpha=\int_I(\alpha'')^2dx.$$

Putting all the pieces together, the discretized shape optimization problem is given as follows

$$\begin{aligned}
&\min_{\mathbf{u},\mathbf{a},\delta_{\mathbf{x}}} \quad J_o(\mathbf{u},\mathbf{a},\delta_{\mathbf{x}}) = \mu \mathbf{u}^{\mathbf{T}}\mathbf{J}\mathbf{u} + \frac{\beta}{2}\mathbf{J}_\alpha \\
&\text{subject to} \\
&\left\{ \begin{array}{rcl} \mathbf{K}\mathbf{u} + \mathbf{B}(\mathbf{u})\mathbf{u} - \mathbf{Q}p &=& \mathbf{F_f} + \mathbf{F_u}, \\ \mathbf{Q^T u} &=& 0, \\ \mathbf{D}\delta_{\mathbf{x}} &=& \mathbf{F_x}, \\ \mathbf{A_a} &=& \mathbf{F_a}, \end{array} \right.
\end{aligned} \tag{2.10}$$

where

$$\mathbf{A_a} = \begin{pmatrix} \alpha(a) \\ \alpha(b) \end{pmatrix} \text{ and } \mathbf{F_a} = \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}.$$

Note that $\mathbf{K}$, $\mathbf{B}(\mathbf{u})$, $\mathbf{Q}$ and $\mathbf{J}$ are dependent on the grid displacement $\delta_{\mathbf{x}}$, while $\mathbf{D}$ is independent of $\delta_{\mathbf{x}}$. Here the mesh variable $\delta_{\mathbf{x}}$ is treated as an optimization variable and the moving mesh equations are viewed as constraints of the optimization problem which are solved simultaneously with the other equations. It should be pointed out here that the moving mesh equations appear only in the discretized shape optimization problem and not in the continuous shape optimization problem.

3. One-shot Lagrange-Newton-Krylov-Schwarz methods. In this section we introduce a parallel solution algorithm for the discrete shape optimization problem (2.10). We define a Lagrangian functional associated with problem (2.10) as follows

$$\begin{aligned}
\mathbf{L}(\mathbf{u},\mathbf{p},\mathbf{x},\mathbf{a},\lambda^{\mathbf{u}},\lambda^{\mathbf{p}},\lambda^{\mathbf{x}},\lambda^{\mathbf{a}}) = J_o(\mathbf{u},\mathbf{a},\delta_{\mathbf{x}}) + (\lambda^{\mathbf{u}})^T \cdot (\mathbf{K}\mathbf{u} + \mathbf{B}(\mathbf{u})\mathbf{u} - \mathbf{Q}p - \mathbf{F_f} - \mathbf{F_u}) \\
+(\lambda^{\mathbf{p}})^T \cdot (\mathbf{Q^T u}) + (\lambda^{\mathbf{x}})^T \cdot (\mathbf{D}\delta_{\mathbf{x}} - \mathbf{F_x}) + (\lambda^{\mathbf{a}})^T \cdot (\mathbf{A_a} - \mathbf{F_a}),
\end{aligned}$$

where $\lambda^{\mathbf{u}}$, λ^p, $\lambda^{\mathbf{x}}$ and $\lambda^{\mathbf{a}}$ are the Lagrange multipliers for the equality constraints. According to [36], one can find the solution of the optimization problem (2.10) by consider the first-order necessary optimality condition, or the KKT system, given by

$$\mathbf{G}(\mathbf{X}) \equiv \nabla_X \mathbf{L}(\mathbf{X}) = \mathbf{0}, \tag{3.1}$$

where $\mathbf{X} \equiv (\mathbf{u},\ p,\ \delta_{\mathbf{x}},\ \lambda^{\mathbf{u}},\ \lambda^p,\ \lambda^{\mathbf{x}},\ \mathbf{a},\ \lambda^{\mathbf{a}})^T$.

To form the algebraic system of the nonlinear equations from a finite element discretization, we need to order the unknowns and the corresponding functions. The ordering of unknowns, though irrelevant mathematically, can have a significant effect on the convergence properties of the solver and the parallel scalability of the overall algorithm. The unknowns are ordered element by element, in contrast to physical variable by physical variable as usually required by other methods. That is, we order the elements so that elements which are nearby geometrically are also as close as possible in the matrix. We order all the unknowns belonging to the first element together, and then all the unknowns that belong to the second but not the first, then all the unknowns that belong to the third element but have not already been placed, and so on [11]. The shape variables $\mathbf{a}$ and $\lambda^{\mathbf{a}}$, not belong to any elements, are placed at the end of the other variables. When we distribute these unknowns to parallel processors, we distribute them to the processors which include the mesh points on the moving boundary. The elements are ordered subdomain by subdomain, for the

purpose of parallel processing. The ordering of the subdomains is not important since we use additive Schwarz methods whose performance has nothing to do with the subdomain ordering. In order to avoid pivoting problem during the sparse LU factorization, for each element, the functions are ordered as [41, 42]

$$(\nabla_{\lambda^{\mathbf{u}}}\mathbf{L}, \nabla_{\lambda^{p}}\mathbf{L}, \nabla_{\lambda^{\mathbf{x}}}\mathbf{L}, \nabla_{\mathbf{u}}\mathbf{L}, \nabla_{p}\mathbf{L}, \nabla_{\delta_{\mathbf{x}}}\mathbf{L})^{T},$$

and the corresponding unknowns are ordered as

$$(\mathbf{u},\ p,\ \delta_{\mathbf{x}},\ \lambda^{\mathbf{u}},\ \lambda^{p},\ \lambda^{\mathbf{x}})^{T}.$$

As a result, the LU factorization can be carried out without running into the pivot problem. The structure of the nonlinear system (3.1) on each element looks like

$$\begin{pmatrix} \mathbf{K}+\mathbf{B(u)u} & -\mathbf{Q} & 0 & 0 & 0 & 0 & 0 & 0 \\ \mathbf{Q^T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{D} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{J} & 0 & 0 & \mathbf{K}+(\mathbf{B(u)u})^{\mathbf{T}}_{\mathbf{u}} & \mathbf{Q} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\mathbf{Q^T} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{M_u} & \mathbf{M_p} & \mathbf{D} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{A} & 0 & \\ 0 & 0 & 0 & 0 & 0 & \Gamma_{\mathbf{a}} & \beta\overline{\mathbf{J}} & \mathbf{A^T} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \\ \delta_{\mathbf{x}} \\ \lambda^{\mathbf{u}} \\ \lambda^{\mathbf{p}} \\ \lambda^{\mathbf{x}} \\ \mathbf{a} \\ \lambda^{\mathbf{a}} \end{pmatrix} - \begin{pmatrix} \mathbf{F_f}+\mathbf{F_u} \\ 0 \\ \mathbf{F_x} \\ 0 \\ 0 \\ -(J_o)_{\delta x} \\ \mathbf{F_a} \\ 0 \end{pmatrix} = \mathbf{0}. \tag{3.2}$$

Here $\mathbf{A}$ and $\overline{\mathbf{J}}$ refer to the geometric constraints and M_u and M_p refer to the derivative of the diffusion/convective terms (M_u) and the pressure term (M_p) with respect to the moving mesh variables $\delta_{\mathbf{x}}$. Though written in a matrix form, many of the operators are nonlinear. In particular the $\mathbf{B}(\mathbf{u})$ term depends on $\mathbf{u}$, and the $\mathbf{K}$, $\mathbf{B}(\mathbf{u})$ and $\mathbf{Q}$ terms depend on the moving mesh $\delta_{\mathbf{x}}$. Note that there are still two small zero blocks appearing on the diagonal of the Jacobian matrix (see the (2, 2) and (5, 5) blocks of the matrix in (3.2)). We deal with these two zero blocks by adding a small number ε during the LU factorization. For more details about the evaluation of the nonlinear function see Appendix A.

We solve the nonlinear system (3.1) by an inexact Newton method [17, 18]. Given an initial guess $\mathbf{X^0}$, at each iteration, $k = 0, 1, \cdots$, we approximately solve the right-preconditioned system

$$\mathbf{H}^k(\mathbf{M}^k)^{-1}(\mathbf{M}^k\mathbf{d}^k) = -\mathbf{G}^k, \tag{3.3}$$

in the senes of

$$\| \mathbf{H}^k(\mathbf{M}^k)^{-1}(\mathbf{M}^k\mathbf{d}^k) + \mathbf{G}^k \| \leq \max\{\eta_r \| \mathbf{G}^k \|, \eta_a\} \tag{3.4}$$

to find a search direction $\mathbf{d}^k$, where $\mathbf{H}^k = \nabla_X\mathbf{G}(\mathbf{X}^k)$ is the Jacobian matrix of the nonlinear system (3.1), $\mathbf{G}^k = \mathbf{G}(\mathbf{X}^k)$ and $(\mathbf{M}^k)^{-1}$ is an additive Schwarz preconditioner to be defined shortly, and η_r and η_a are relative and absolute tolerances for the linear solver. After approximately solving (3.3), one may use a globalization method such as *line search* or *trust region* [36] to update the solution. In this paper we focus on a line search approach, where the new approximate solution is

$$\mathbf{X}^{k+1} = \mathbf{X}^k + \tau^k\mathbf{d}^k,$$

and the step length τ^k is selected by a cubic line search method [16]. Newton methods can be implemented with or without the Jacobian matrix. A framework for implementing Newton methods without using the explicit form of the Jacobian is provided in [31]. In this paper, we do not use the matrix-free method since we need the explicit

form of the Jacobian to construct the preconditioner. The sparsity of $\mathbf{H}^k$ is similar to that of the matrix in (3.2) and it is a large, sparse nonsymmetric matrix and is very ill-conditioned. There are two saddle-point problems embedded in $\mathbf{H}^k$: incompressible Navier-Stokes equations and the constrained optimization problem. One of the most expensive steps of our algorithm is the construction of $\mathbf{H}^k$. Unfortunately, there is no reasonably cheap way to compute $\mathbf{H}^k$. Some researchers try to approximate $\mathbf{H}^k$ by dropping some terms (such as the shape derivatives) when taking the derivative of the nonlinear function $\mathbf{G}(\mathbf{X})$. After many attempts to use an approximate $\mathbf{H}^k$, we decide to use a full, analytically computed, Jacobian. For this class of problems, we find that the inexact Newton method converges faster and is most robust when equipped with the full Jacobian. The full Jacobian matrix can be obtained by either a hand-coded program which is accurate but programmer-time-consuming or a multi-colored finite difference method which is less accurate and computer-time-consuming. We choose to use a hand-coded Jacobian matrix since the finite difference based method is too slow. For more details about the evaluation of the Jacobian matrix $\mathbf{H}^k$ see Appendix A.

Most of the computing time for solving the optimization problem is spent on solving the Jacobian system at each Newton step. Therefore, one needs an efficient iterative method with a good preconditioner. We use a right-preconditioned GMRES method [45]. The nonlinear and linear solvers are standard, we focus here on the construction of the preconditioner.

To define the additive Schwarz preconditioner, we need an overlapping partition of Ω_α. Since the mesh topology doesn't change when the shape of the domain changes, we obtain the partition using the initial mesh on Ω_{α_0}. We first partition the domain Ω_{α_0} into non-overlapping subdomains $\Omega_{\alpha_l},\ l = 1, \cdots, N_p$ (see Fig. 4.2) and then extend each subdomain Ω_{α_l} to $\Omega^\delta_{\alpha_l}$ which overlaps its neighbors, i.e., $\Omega_{\alpha_l} \subset \Omega^\delta_{\alpha_l}$. Here N_p is the number of processors which is equal to the number of subdomains. The parameter δ represents the size of the overlap. Note that the shape vectors $\mathbf{a}$ and $\lambda^{\mathbf{a}}$ are not partitioned and assigned to all subdomains intersecting with the moving boundary. When extending a subdomain to increase the overlap, the shape variables are not considered.

Let N and N_l be the total degrees of freedom (DOF) in Ω_α and $\Omega^\delta_{\alpha_l}$, respectively. We define a $N_l \times N$ restriction matrix R^δ_l which maps the global vector of unknowns to those belonging to a subdomain. The component $(R^\delta_l)_{ij}$ is either 1 if the indices $1 < i < N,\ 1 < j < N_l$ are related to unknowns defined at the grid points belonging to $\Omega^\delta_{\alpha_l}$ or 0. For example, for $\mathbf{X} \in R^N$, $\mathbf{X}_l = R^\delta_l \mathbf{X}$ by throwing away all the components of $\mathbf{X} \in R^N$ corresponding to mesh points outside $\Omega^\delta_{\alpha_l}$.

For each of the overlapping subdomain we define $\mathbf{H}_l$ as the restriction of $\mathbf{H}$ to the overlapping subdomain $\Omega^\delta_{\alpha_l}$, i.e.,

$$\mathbf{H}_l = R^\delta_l\ \mathbf{H}\ (R^\delta_l)^T,$$

where $\mathbf{H}$ is the Jacobian matrix of $\mathbf{G}(\mathbf{X})$. This is equivalent to assuming homogeneous Dirichlet boundary conditions for all variables on the interior part of the boundary of the overlapping subdomain. Then the classical one-level additive Schwarz preconditioner [47] is defined as

$$\mathbf{M}^{-1}_{asm} = \sum_{l=1}^{N_p} (R^\delta_l)^T \mathbf{B}^{-1}_l R^\delta_l.$$

Here $\mathbf{B}_l^{-1}$ is either the inverse of $\mathbf{H}_l$ or a preconditioner for $\mathbf{H}_l$. In this paper, $\mathbf{B}_l^{-1}$ is the inverse of $\mathbf{H}_l$ which is computed by a sparse LU factorization.

The theory of classical one-level additive Schwarz preconditioner is very well developed for elliptic problems ([47]), and it says that the condition number of the preconditioned system satisfies

$$\kappa \leq \frac{C(1+H/\delta)}{H^2}, \tag{3.5}$$

where H is the subdomain diameter, δ is the size of the overlap and C is a constant which is independent of H, δ and the mesh size h. The $1/H^2$ term shows that the number of iterations increases with the number of subdomains. Our problem is not elliptic, but the performance of the preconditioner does show a similar dependence on $1/H^2$.

The overall algorithm can be described as follows:

Step 1: Set or compute the initial guess of the shape function $\alpha_0(x)$, the mesh displacement $\delta_{\mathbf{x}}$, the velocity $\mathbf{u}$, the pressure p and the Lagrange multipliers $\lambda_{\mathbf{u}}$, λ_p, $\lambda_{\mathbf{x}}$ and $\lambda_{\mathbf{a}}$.

Step 2: Form the Jacobian matrix of the nonlinear system $\mathbf{G}(\mathbf{X}^k)$.

Step 3: Inexactly solve the preconditioned Jacobian system (3.3) for $\mathbf{d}^k$ using a GMRES.

Step 4: Update the solution $\mathbf{X}^{k+1} = \mathbf{X}^k + \tau^k \mathbf{d}^k$, where the step length τ^k is selected by a cubic back-tracking line search method.

Step 5: Update the mesh using the grid displacements in $\mathbf{d}^k$ and return to *Step 2* unless a stopping condition is met.

In *Step 5*, the reason that we can update the mesh using the grid displacements in $\mathbf{d}^k$ is that the moving mesh equations are included in the Jacobian system at each Newton step and they are solved when the Jacobian system is solved in *Step 3*. The initial mesh displacement and the Lagrange multipliers are often set to zero. The initial velocity and pressure are computed by solving the state equations on the initial computational domain.

4. Numerical experiments. The algorithm introduced in the previous sections is applicable to general shape optimization problems governed by incompressible Navier-Stokes equations. Here we study an application of the algorithm for a simplified artery bypass problem [2] as shown in Fig. 4.1. The basic assumption is that there is a complete blockage in the artery, and a bypass needs to be built. The goal is to find the shape of the bypass such that the energy loss is minimized. We only consider a 2D version of the problem and the algorithms developed in this paper can be extended to 3D and generalized in a number of ways. For example, in the current approach the moving boundary is assumed to be a polynomial. This assumption can be replaced by any function that can be represented as a linear combination of some local or global basis functions. The moving mesh equation can be replaced by an elasticity equation which is more suitable for the case of large deformation. Our solver is implemented using the Portable Extensible Toolkit for Scientific computing (PETSc) [10]. All computations are performed on an IBM BlueGene/L supercomputer. Meshes are generated with CUBIT [37] from Sandia National Laboratory and partitioned with ParMETIS [29]. The purposes of the numerical experiments are to understand the convergence and the parallel scalability of the algorithm. Special attention is paid to the performance of the domain decomposition preconditioner which is the key component of the one-shot approach.

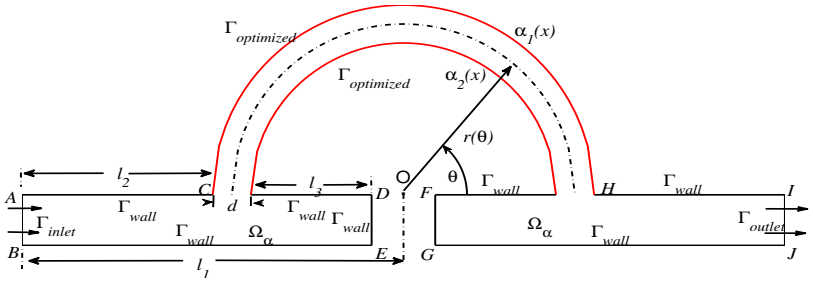

FIG. 4.1. *Simplified two dimensional bypass model; The red boundary* $\Gamma_{optimized}$ *denotes the part of the boundary whose shape is to be determined by the optimization process. Here* $l_1 = 6.0$, $l_2 = 3.0$, $l_3 = 1.9$, $d = 0.6$ *and* $|AB| = 0.8$.

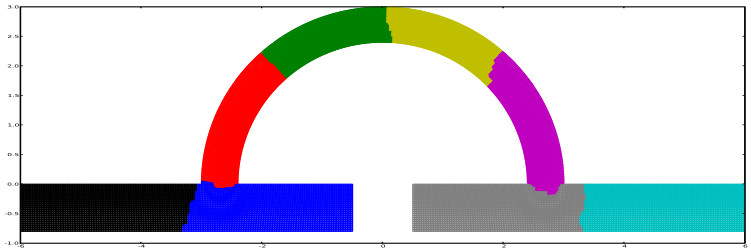

FIG. 4.2. *A sample partition of the computational domain, each color representing a different subdomain. The initial computational domain* Ω_{α_0} *is partitioned into 8 non-overlapping subdomains by Parmetis [29].*

We consider a simplified bypass problem as shown in Fig. 4.1. Without the blockage, the flow is supposed to go from AB to IJ, but now we assume that the artery is blocked at DE and the flow has to go through CH. For simplicity, the thickness of CH is fixed. The goal is to find the best shape of the bypass CH, such that the energy dissipation of the fluid in the entire computational domain Ω_α is minimized in

$$J_o(\mathbf{u}, \alpha) = 2\mu \int_{\Omega_\alpha} \epsilon(\mathbf{u}) : \epsilon(\mathbf{u}) dxdy + \frac{\beta}{2} \int_I (\alpha'')^2 dx.$$

For simplicity, we let the body forces $\mathbf{f} = \mathbf{0}$ in the state equations (2.1). The boundary condition on the inlet Γ_{inlet} is chosen as a constant v_{in}, no-slip boundary conditions are used on the walls Γ_{wall} and on the outlet boundary Γ_{outlet} the stress-free boundary conditions are imposed; see (2.1). We use a polynomial function

$$r(\theta) = \sum_{i=1}^{p} r_i \theta^i,$$

with $p = 5$ to represent the centerline of the bypass (see the dashed line in Fig. 4.1) whose shape is to be determined by the optimization process. Other shape functions can be used, but here we simply follow the paper [2].

In all experiments, we use a hand-coded Jacobian matrix. The Jacobian system in each Newton step is solved by a right-preconditioned GMRES method with an

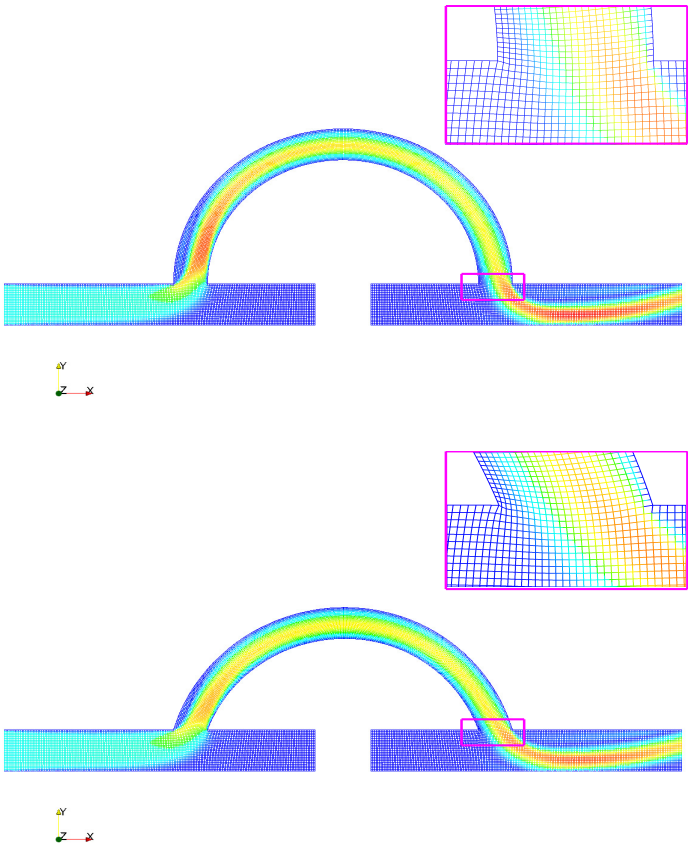

FIG. 4.3. *An example of the mesh deformation. The top figure is the initial mesh and the bottom one is the deformed mesh.*

absolute tolerance equal to 10^{-10} and a relative tolerance of 10^{-3}. We stop the Newton iteration when the nonlinear residual is decreased by a factor of 10^{-6}. For the one-level additive Schwarz preconditioner, the number of subdomains is equal to the number of processors and the extended subdomain problems have zero Dirichlet interior boundary conditions and are solved with a sparse LU factorization. The overlapping size δ is understood in terms of the number of elements; i.e., $\delta = 8$ means the overlapping size is 8 layers of elements.

For the initial guess of the Newton method, we first solve the state equations (2.1) on the initial computational domain Ω_{α_0} and use this solution as the initial guess of the velocity $\mathbf{u}$ and pressure p. We use zero initial guess for the other unknowns.

In the first test case, we set the Reynolds number $Re = \frac{Lv_{in}}{\mu}$ to 300, where $L = 0.8$ is the artery diameter, $v_{in} = 3.75$ is the inlet velocity and $\mu = 0.01$ is the kinematic viscosity. We solve the problem on a mesh with about 18,000 elements and 73,000 mesh points with a nearly uniform distribution in the flow domain. The parameter $\beta = 10.0$ and the DOF for this test case is 620,946. The centerline of the initial shape of the bypass is a semi-circle with center O and radius 2.7, and Fig. 4.4 shows the velocity distribution of the initial (top) and optimal shapes (bottom). The energy

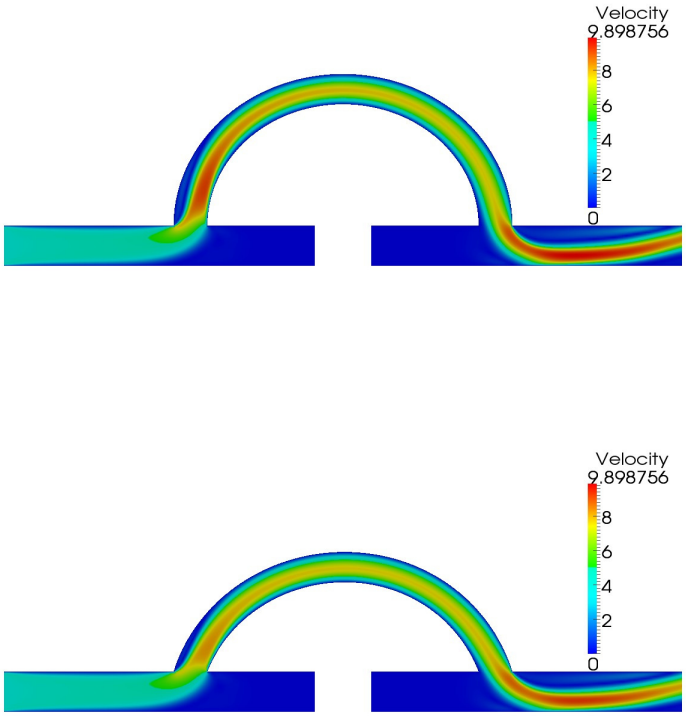

FIG. 4.4. *Velocity distribution of the initial (top) and optimal shapes (bottom). Here* $\beta = 10.0$ *and* $Re = 300$.

dissipation of the optimized shape (top) is reduced by about 16.23% compared to the initial shape (bottom). The centerline of the optimized shape is represented by the computed polynomial

$$r(\theta) = 2.700 - 0.844\theta + 0.377\theta^2 - 0.108\theta^3 + 0.036\theta^4 - 0.004\theta^5.$$

In some publications, e.g., [2, 40], the first term $\epsilon(\mathbf{u}) : \epsilon(\mathbf{u})$ in the objective function (2.2) is referred to as the squared shear rate. We show the distribution of the shear rate of the initial (top) and optimal (bottom) shapes in Fig. 4.5.

The regularization parameter β in the objective function (2.2) is very important for shape optimization problems. From Table 4.1 we can see that reducing β can increase the reduction of the energy dissipation ("Init.", "Opt." and "Reduction" are the initial, optimized and reduction of the energy dissipation in the table), but the number of Newton (Newton) and the average number of GMRES iterations per Newton (GMRES) and the total compute time in seconds (Time) increase, which means that the nonlinear algebraic system is harder to solve when β is small. This is because the boundary of Ω_α is more flexible and can become very irregular when β is small. Fig. 4.6 shows the initial shape and the optimized shapes obtained with

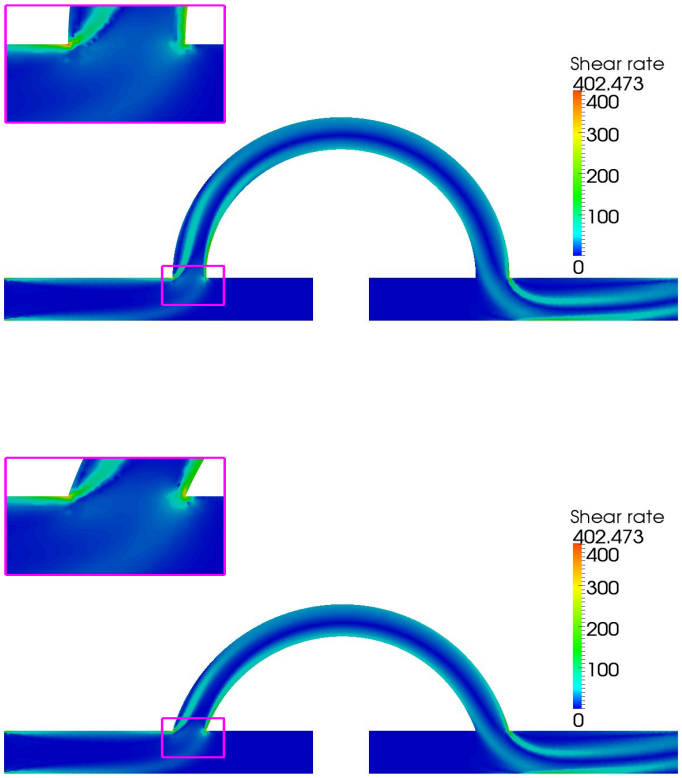

FIG. 4.5. *The shear rate distribution of the initial (top) and final shapes (bottom). Here* $\beta = 10.0$ *and* $Re = 300$.

TABLE 4.1
Effect of the regularization parameter β *in the cost functional (2.2). Here* $DOF = 620,946$, $Re = 300$.

β	Newton	GMRES	Time	Energy Dissipation		
				Init.	Opt.	Reduction
20.0	5	262.00	1573.58	106.51	92.17	13.47%
15.0	6	271.17	1804.33	106.51	90.78	14.77%
10.0	9	271.33	2450.19	106.51	89.22	16.23%
8.0	12	270.17	3101.40	106.51	88.55	16.86%
5.0	17	293.76	4275.80	106.51	87.45	17.90%

different values of β. From this figure we can see that β can control the boundary deformation.

To show the parallel scalability of the algorithm, two meshes with $DOF = 620,946$ and $DOF = 893,714$ are considered. The strong scalability of the algorithm is given in Fig. 4.7, which shows that the speedup is close to be linear when the number of processors is small. The straight solid lines are the ideal scalability which means that the compute time decreases in proportion to the increase of the number of proces-

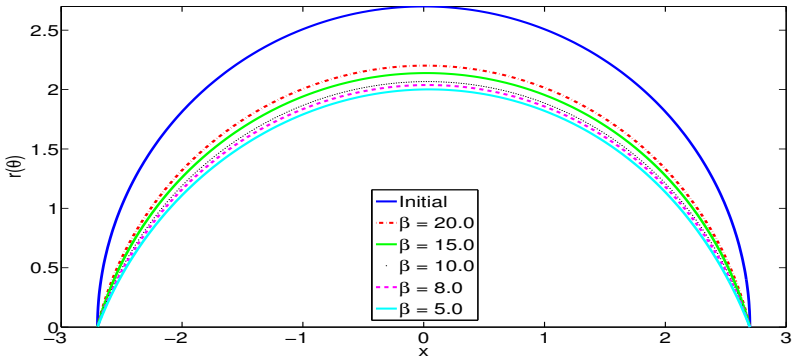

FIG. 4.6. *The initial and optimized shapes with different values of parameter* β. *Here* $DOF = 620,946$ *and* $Re = 300$.

TABLE 4.2
Parallel scalability. Here $\beta = 10$ *and overlap* $= 8$ *for the first grid* ($DOF = 620,946$) *and* $\beta = 20$ *and overlap* $= 6$ *for the second grid* ($DOF = 893,714$) *and* $Re = 300$ *for both grids.*

np	$DOF = 620,946$			np	$DOF = 893,714$		
	Newton	GMRES	Time		Newton	GMRES	Time
45	9	287.22	3390.83	80	8	341.75	2373.75
64	9	271.44	2454.14	120	8	429.75	1741.29
90	9	366.11	1926.30	160	8	493.63	1621.78
128	9	393.89	1545.37	240	8	672.50	1308.43

sors. Refer to [30] for the definition of the speedup. As expected in one-level Schwarz methods, the preconditioner becomes worse as the number of subdomains increases; overlap between adjacent subdomains is not sufficient for the global transfer of information. The number of GMRES iterations increases with number of processors, as shown in Table 4.2, where (as in all the tables) "np" is the number of processors. This suggests the need for a two-level or multi-level Schwarz algorithm which we plan to develop in the future.

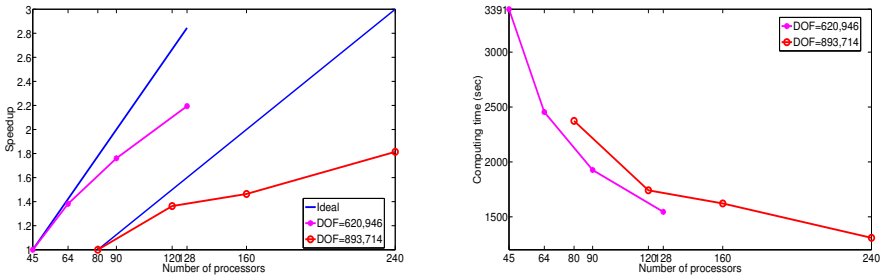

FIG. 4.7. *The speedup and the total compute time for two different mesh sizes. Here* $Re = 300$.

In our approach, we use a line search method to globalize the Newton method. But not every Newton step requires a line search. In Table 4.3, we present the number of line search (# Line-Search) and the compute time spent on the line search (Line-Search). We also show the compute time for the preconditioner on the slowest

TABLE 4.3
Time spent on the preconditioning and line search steps. Here $\beta = 10$, overlap $= 8$, $DOF = 620,946$ and $Re = 300$.

np	dof	Newton	# Line-Search	Time		
				Total	Preconditioner	Line-Search
64	63,208	9	4	2453.48	1562.94	39.56
90	60,328	9	4	1940.48	1372.32	27.91
128	52,808	9	5	1540.51	1173.84	19.85

TABLE 4.4
Test results using different overlapping factors with respect to the number of processors. Here $\beta = 10.0$, $DOF = 620,946$, $Re = 300$.

Overlap	Newton		GMRES		Time	
	np=64	*np*=128	*np*=64	*np*=128	*np*=64	*np*=128
5	9	9	393.67	825.11	2218.38	1821.93
6	9	9	334.00	586.22	2026.75	1563.14
7	9	9	282.11	459.22	2363.58	1669.32
8	9	9	271.89	407.11	2444.80	1567.20
9	9	9	261.89	344.67	2992.58	1818.65
10	9	9	233.00	330.00	2886.06	2437.13

processor (Preconditioner) and the degrees of freedom on this processor (dof) in Table 4.3. Since we use the overlapping domain decomposition preconditioner, the degrees of freedom on each processor is not halved when the number of processors is doubled. We find that about 70% of the total compute time (Total) is spent on the preconditioning step.

In overlapping domain decomposition methods, the overlapping parameter δ plays an important role in controlling the number of iterations of the linear solver. From Table 4.4, we see that the number of Newton iterations does not change when we change the overlapping size and the number of GMRES iterations decreases as we increase the overlapping size. The total compute time first decreases and then increases as the overlapping size increases over a certain value. This suggests that an optimal overlapping size exists if the goal is to minimize the total compute time for a given number of processors on a particular machine.

Table 4.5 shows some results for different Reynolds numbers and regularization parameters. Judging from the increase of the number of linear and nonlinear iterations, it is clear that the problem becomes harder as we increase the Reynolds number or decrease the regularization parameter. On the other hand, we achieve higher percentage of reduction of energy dissipation in the harder to solve situations.

5. Conclusions and future work. In this paper, we developed a parallel one-shot Lagrange-Newton-Krylov-Schwarz method for two-dimensional shape optimization problems governed by incompressible Navier-Stokes equations. Our approach is based on a finite element discretization on a moving unstructured mesh covering the computational domain whose shape determines the value of the objective function. This approach first reformulates the optimization problem into a large coupled nonlinear algebraic system of equations and then solves the system with a one-level Newton-Krylov-Schwarz algorithm. Even though the coupled system is highly ill-conditioned, NKS converges well thanks to the powerful Schwarz preconditioner. We tested the algorithms for an artery bypass design problem with more than 800,000 DOF and up to 128 processors. The numerical results show that our method is quite robust with respect to the Reynolds number and the regularization parameter. The

TABLE 4.5
The impact of Reynolds number. Here overlap = 8, DOF = 620,946, np = 64.

Re	Newton		GMRES		Time		Reduction	
	β=10.0	β= 5.0	β=10.0	β= 5.0	β=10.0	β= 5.0	β=10.0	β= 5.0
50	5	9	216.00	226.44	1524.26	2365.37	7.72%	7.92%
100	5	10	231.60	242.70	1541.14	2606.56	10.16%	10.72%
200	6	11	241.50	264.00	1761.78	2863.26	13.79%	14.97%
300	9	17	271.33	293.76	2450.19	4275.80	16.23%	17.90%

strong scalability is almost ideal when the number of processors is not too large. In the future, we plan to study some multi-level Schwarz methods which may improve the scalability when the number of processors is large.

Acknowledgments. We would like to thank Andrew T. Barker, Yuqi Wu, Alfio Quarteroni, and Chao Yang for many discussions and comments and the PETSc team of Argonne National Laboratory for their help on using the PETSc library.

REFERENCES

[1] V. AKCELIK, G. BIROS, O. GHATTAS, J. HILL, D. KEYES, AND B. WAANDERS, *Parallel algorithms for PDE-constrained optimization*, In: Parallel Processing in Scientific Computing, M. Heroux, P. Raghaven, and H. Simon (Eds.), SIAM, Philadelphia, 2006.

[2] F. ABRAHAM, M. BEHR, AND M. HEINKENSCHLOSS, *Shape optimization in stationary blood flow: A numerical study of non-Newtonian effects*, Comput. Methods Biomech. Biomed. Engrg. 8 (2005), pp. 127-137.

[3] F. ABRAHAM, M. BEHR, AND M. HEINKENSCHLOSS, *Shape optimization in unsteady blood flow: A numerical study of non-Newtonian effects*, Comput. Methods Biomech. Biomed. Engrg. 8 (2005), pp. 201-212.

[4] H. ANTIL, M. HEINKENSCHLOSS, AND R. H. W. HOPPE, *Domain decomposition and balanced truncation model reduction for shape optimization of the Stokes system*, CAAM Technical Report TR09-24 (2009).

[5] H. ANTIL, R. H. W. HOPPE, AND C. LINSENMANN, *Path-following primal-dual interior-point methods for shape optimization of stationary flow problems*, J. Numer. Math., 15 (2007), pp. 81-100.

[6] V. AGOSHKOV, A. QUARTERONI, AND G. ROZZA, *Shape design in aorto-coronaric bypass anastomoses using perturbation theory*, SIAM J. Numer. Anal., 44 (2006), pp. 367-384.

[7] V. AGOSHKOV, A. QUARTERONI, AND G. ROZZA, *A mathematical approach in the design of arterial bypass using unsteady Stokes equations*, J. Sci. Comput., 28 (2006), pp. 139-165.

[8] E. ARIAN AND S. TAASAN, *Shape optimization in one-shot*, In: Optimal Design and Control, J. Boggaard et al (Eds.), Birkhauser, Boston, (1995), pp. 273-294.

[9] J. ARORA AND Q. WANG, *Review of formulations for structural and mechanical system optimization*, Struct. Multidisciplin. Optim., 30 (2005), pp. 251-272.

[10] S. BALAY, K. BUSCHELMAN, V. EIJKHOUT, W. D. GROPP, D. KAUSHIK, M. G. KNEPLEY, L. C. MCINNES, B. F. SMITH, AND H. ZHANG, *PETSc Users Manual*, Technical report, Argonne National Laboratory, 2010.

[11] A. BARKER AND X.-C. CAI, *Scalable parallel methods for monolithic coupling in fluid-structure interaction with application to blood flow modeling*, J. Comput. Phys., 229 (2010), pp. 642-659.

[12] G. BIROS AND O. GHATTAS, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part I: The Krylov-Schur solver*, SIAM J. Sci. Comput., 27 (2005), pp. 687-713.

[13] G. BIROS AND O. GHATTAS, *Parallel Lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. Part II: The Lagrange-Newton solver, and its application to optimal control of steady viscous flows*, SIAM J. Sci. Comput., 27 (2005), pp. 714-739.

[14] P. T. BOGGS AND J. W. TOLLE, *Sequential quadratic programming*, Acta Numerica, 4 (1995), pp. 1-50.

[15] X.-C. CAI, W. D. GROPP, D. E. KEYES, R. G. MELVIN, AND D. P. YOUNG, *Parallel Newton-Krylov-Schwarz algorithms for the transonic full potential equation*, SIAM J. Sci. Comput.,

19 (1998), pp. 246-265.
[16] J. E. Dennis and R. B. Schnable, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, SIAM, Philadelphia, 1996.
[17] S. C. Eisenstat and H. F. Walker, *Globally convergent inexact Newton method*, SIAM J. Optim., 4 (1994), pp. 393-422.
[18] S. C. Eisenstat and H. F. Walker, *Choosing the forcing terms in an inexact Newton method*, SIAM J. Sci. Comput., 17 (1996), pp. 16-32.
[19] M. D. Gunzburger, *Perspectives in Flow Control and Optimization—Advances in Design and Control*, SIAM, Philadelphia, 2003.
[20] M. D. Gunzburger and H. Kim, *Existence of an optimal solution of a shape control problem for the stationary Navier-Stokes equations*, SIAM J. Cont. Optim., 36 (1998), pp. 895-909.
[21] M. D. Gunzburger, H. Kim, and S. Manservisi, *On a shape control problem for the stationary Navier-Stokes equations*, Math. Model. Numer. Anal., 34 (2000), pp. 1233-1258.
[22] O. Ghattas and C. Orozco, *A parallel reduced Hessian SQP method for shape optimization*, In: Multidisciplinary Design Optimization: State of the Art, N. M. Alexandrov and M. Y. Hussaini (Eds.), SIAM, Philadelphia (1997), pp. 133-152.
[23] P. M. Gresho and R. L. Sani, *Incompressible Flow and the Finite Element Method*, John Wiley and Sons, Inc., New York, 1998.
[24] A. Harbir, *Optimization and Model Reduction of Time Dependent PDE-constrained Optimization Problems: Applications to Surface Acoustic Wave Driven Microfluidic Biochips*, Ph.D. Thesis, University of Houston, 2009.
[25] S. B. Hazra, *Multigrid one-shot method for aerodynamic shape optimization*, SIAM J. Sci. Comput., 30 (2008), pp. 1527-1547.
[26] B. He, O. Ghattas, and J. F. Antaki, *Computational strategies for shape optimization of time dependent Navier Stokes flow*, Technical Report CMU-CML-97-102, Carnegie Mellon University, 1997.
[27] J. Haslinger and R. A. E. Makinen, *Introduction to Shape Optimization: Theory, Approximation and Computation*, SIAM, Philadelphia, 2003.
[28] W. Huang and R. D. Russell, *Adaptive Moving Mesh Methods*, Springer-Verlag, Berlin, 2011.
[29] G. Karypis, *METIS/ParMETIS web page*, University of Minnesota, 2008. http://glaros.dtc.umn.edu/gkhome/views/metis.
[30] V. Kumar and A. Gupta, *Analyzing Scalability of Parallel Algorithms and Architectures*, J. Parallel Distrib. Comput., 22 (1994), pp. 379-391.
[31] D. Knoll and D. E. Keyes, *Jacobian-free Newton-Krylov methods: A survey of approaches and applications*, J. Comput. Phys., 193 (2004), pp. 357-397.
[32] M. Laumen, *A comparison of numerical methods for optimal shape design problems*, Optim. Methods and Softw., 10 (1999), pp. 497-537.
[33] T. M. Lassila and G. Rozza, *Parametric free-form shape design with PDE models and reduced basis method*, Comput. Methods Appl. Mech. Engrg., 199 (2010), pp. 1583-1592.
[34] B. Mohammadi and O. Pironneau, *Applied Shape Optimization for Fluids*, Oxford University Press, Oxford, 2001.
[35] B. Mohammadi and O. Pironneau, *Optimal shape design for fluids*, Annu. Rev. Fluid Mech., 36 (2004), pp. 255-279.
[36] J. Nocedal and S. J. Wright, *Numerical Optimization*, First ed., Springer-Verlag, Berlin, 2000.
[37] S. J. Owen and J. F. Shepherd, *CUBIT project web page*, 2008, http://cubit.sandia.gov/.
[38] S. I. Petrova, *Applications of one-shot methods in PDEs constrained shape optimization*, Math. and Comput. Simulat., 80 (2009), pp. 581-597.
[39] O. Pironneau, *Optimal Shape Design for Elliptic Systems*, Springer-Verlag, Berlin, 1984.
[40] M. Probst, M. Llfesmann, M. Nicolai, H. Bcker, M. Behr, and C. Bischof, *Sensitivity of optimal shapes of artificial grafts with respect to flow parameters*, Comput. Methods Appl. Mech. Engrg., 199 (2010) pp. 997-1005.
[41] E. Prudencio, R. Byrd, and X.-C. Cai, *Parallel full space SQP Lagrange-Newton-Krylov-Schwarz algorithms for PDE-constrained optimization problems*, SIAM J. Sci. Comput., 27 (2006), pp. 1305-1328.
[42] E. Prudencio and X.-C. Cai, *Parallel multilevel restricted Schwarz preconditioners with pollution removing for PDE-constrained optimization*, SIAM J. Sci. Comput., 29 (2007), pp. 964-985.
[43] A. Quarteroni and G. Rozza, *Optimal control and shape optimization of aorto-coronaric bypass anastomoses*, Math. Models and Methods in Appl. Sci., 13 (2003), pp. 1801-1823.
[44] V. Schulz and I. Gherman, *One-shot methods for aerodynamic shape optimization*, in Notes on Numerical Fluid Mechanics and Multidisciplinary Design, N. Kroll et al eds., Springer-

Verlag, Berlin, 107 (2009), pp. 207-220.
[45] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856-869.
[46] J. Sokolowski and J.-P. Zolesio, *Introduction to Shape Optimization: Shape Sensitivity Analysis*, Springer-Verlag, Berlin, 1992.
[47] A. Toselli and O. Widlund, *Domain Decomposition Methods: Algorithms and Theory*, Springer-Verlag, Berlin, 2005.
[48] A. H. Truong, C. A. Oldfield, and D. W. Zingg, *Mesh movement for a discrete-adjoint Newton-Krylov algorithm for aerodynamic optimization*, AIAA Journal, 46 (2008), pp. 1695-1704.

Appendix A. Evaluation of the nonlinear KKT system and the construction of its Jacobian matrix. Providing an analytic hand-coded Jacobian is very helpful for the efficiency and the robustness of the nonlinear solver, but the actual construction of the Jacobian is often a challenge. In this appendix we provide some details for the evaluation of the KKT system (3.1) and the construction of the corresponding Jacobian matrix $\mathbf{H}$ in (3.3).

A.1. Evaluation of the nonlinear KKT system. First we compute the derivatives of the Lagrange functional with respect to the velocity $\mathbf{u}$. We note that in the Lagrange functional all the terms except the objective function term $J_o(\mathbf{u}, \mathbf{a})$ and the convective term $\mathbf{B}(\mathbf{u})\mathbf{u}$ are linear with respect to $\mathbf{u}$. Therefore we only need to compute these two terms. The derivative of the objective function with respect to $\mathbf{u}$ is

$$\frac{\partial(J_o(\mathbf{u}, \mathbf{a}))}{\partial \mathbf{u}} = 2\mu \mathbf{J}\mathbf{u}.$$

The nonlinear convective term $\mathbf{B}(\mathbf{u})\mathbf{u}$ in the x direction is

$$\sum_{i=1}^{n} u_i \int_{\Omega(\alpha)} \left(\frac{\partial \phi_i}{\partial x} \sum_{k=1}^{n} u_k \phi_k + \frac{\partial \phi_i}{\partial y} \sum_{k=1}^{n} v_k \phi_k \right) \phi_j dxdy, \quad 1 \le j \le n. \tag{A.1}$$

Taking the derivatives of (A.1) with respect to $u_q,\ 1 \le q \le n$, we obtain

$$\sum_{k=1}^{n} u_k \int_{\Omega(\alpha)} \left(\frac{\partial \phi_q}{\partial x} \phi_k + \frac{\partial \phi_k}{\partial x} \phi_q \right) \phi_j dxdy + \sum_{k=1}^{n} v_k \int_{\Omega(\alpha)} \frac{\partial \phi_q}{\partial y} \phi_k \phi_j dxdy,\ 1 \le j \le n.$$

And the derivative with respect to $v_q,\ 1 \le q \le n$ is

$$\sum_{k=1}^{n} u_k \int_{\Omega(\alpha)} \frac{\partial \phi_k}{\partial y} \phi_q \phi_j dxdy, \quad 1 \le j \le n. \tag{A.2}$$

The derivatives in the y direction are similar.

Next we compute the derivatives of the Lagrange functional with respect to the moving mesh $\delta_{\mathbf{x}}$. Because $\delta_{\mathbf{x}} = \mathbf{x} - \mathbf{x}^{\mathbf{0}}$, the derivatives to $\delta_{\mathbf{x}}$ are equal to that of $\mathbf{x}$. For simplicity, we compute $\nabla_{\mathbf{x}}$ instead of $\nabla_{\delta_{\mathbf{x}}}$. The dependence on the moving mesh is quite complicated because the integrals in the weak form are taken over a moving domain, which depends on the mesh variables. To deal with this, we map the physical domain Ω_α to a reference domain Ω^ξ. Let the coordinate on the physical domain be (x, y) and on the reference domain be (ξ, η). Then the transformation between the coordinate (x, y) and (ξ, η) is given by

$$x = \sum_{i=1}^{n} x_i \phi_i(\xi, \eta), \qquad y = \sum_{i=1}^{n} y_i \phi_i(\xi, \eta), \tag{A.3}$$

where (x_i, y_i) are the spatial coordinates at node i. The Jacobian matrix of the transformation is

$$J = \begin{bmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} \displaystyle\sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi} & \displaystyle\sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi} \\ \displaystyle\sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} & \displaystyle\sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} \end{bmatrix}, \tag{A.4}$$

and the determinant is

$$\begin{aligned} |J| &= \begin{vmatrix} \displaystyle\sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi} & \displaystyle\sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi} \\ \displaystyle\sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} & \displaystyle\sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} \end{vmatrix} \\ &= \sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi} \sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} - \sum_{i=1}^{n} x_i \frac{\partial \phi_i(\xi,\eta)}{\partial \eta} \sum_{i=1}^{n} y_i \frac{\partial \phi_i(\xi,\eta)}{\partial \xi}. \end{aligned} \tag{A.5}$$

Note that the curvilinear coordinate (ξ, η) is defined locally with respect to each element, so that each element has its own coordinate transformation. When we refer to (ξ, η) globally, what we have in mind is the local curvilinear coordinate system corresponding to the particular element that is referenced.

In order to change the integration domain, we need to replace the derivatives $\dfrac{\partial \phi_i}{\partial x}$ and $\dfrac{\partial \phi_i}{\partial y}$ with $\dfrac{\partial \phi_i}{\partial \xi}$ and $\dfrac{\partial \phi_i}{\partial \eta}$. Using the chain rule

$$\frac{\partial \phi_i}{\partial \xi} = \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \xi}, \qquad \frac{\partial \phi_i}{\partial \eta} = \frac{\partial \phi_i}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial \phi_i}{\partial y}\frac{\partial y}{\partial \eta},$$

we have

$$\begin{bmatrix} \dfrac{\partial \phi_i}{\partial x} \\ \dfrac{\partial \phi_i}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \dfrac{\partial \phi_i}{\partial \xi} \\ \dfrac{\partial \phi_i}{\partial \eta} \end{bmatrix}, \text{ where } \quad J^{-1} = \frac{1}{|J|} \begin{bmatrix} \dfrac{\partial y}{\partial \eta} & -\dfrac{\partial y}{\partial \xi} \\ -\dfrac{\partial x}{\partial \eta} & \dfrac{\partial x}{\partial \xi} \end{bmatrix}.$$

That is

$$\frac{\partial \phi_i}{\partial x} = \frac{1}{|J|}\left(\frac{\partial y}{\partial \eta}\frac{\partial \phi_i}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial \phi_i}{\partial \eta}\right), \quad \frac{\partial \phi_i}{\partial y} = \frac{1}{|J|}\left(-\frac{\partial x}{\partial \eta}\frac{\partial \phi_i}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial \phi_i}{\partial \eta}\right). \tag{A.6}$$

Below we give an example of the computation of the derivative. Let us take the Laplace term $\mathbf{Ku}$

$$(\mathbf{Ku})_j = \mu \sum_{i=1}^{n} u_i \int_{\Omega(\alpha)} \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \frac{\partial \phi_i}{\partial y}\frac{\partial \phi_j}{\partial y}\right) dxdy. \tag{A.7}$$

We transform the integral in (A.7) with (A.6) to obtain

$$\begin{aligned} (\mathbf{Ku})_j = \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big(&(\frac{\partial y}{\partial \eta}\frac{\partial \phi_i}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial \phi_i}{\partial \eta})(\frac{\partial y}{\partial \eta}\frac{\partial \phi_j}{\partial \xi} - \frac{\partial y}{\partial \xi}\frac{\partial \phi_j}{\partial \eta}) \\ &+(-\frac{\partial x}{\partial \eta}\frac{\partial \phi_i}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial \phi_i}{\partial \eta})(-\frac{\partial x}{\partial \eta}\frac{\partial \phi_j}{\partial \xi} + \frac{\partial x}{\partial \xi}\frac{\partial \phi_j}{\partial \eta})\Big)\frac{1}{|J|} d\xi d\eta. \end{aligned} \tag{A.8}$$

From the transformation (A.3), we have

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^{n} x_i \frac{\partial \phi_i}{\partial \xi}, \qquad \frac{\partial x}{\partial \eta} = \sum_{i=1}^{n} x_i \frac{\partial \phi_i}{\partial \eta} \tag{A.9}$$

and similarly for $\dfrac{\partial y}{\partial \xi}$ and $\dfrac{\partial y}{\partial \eta}$. We also need to compute the derivatives $\partial|J|/\partial x_q$ and $\partial|J|/\partial x_q$. From the definition of $|J|$ (A.5), we have

$$\begin{aligned} \frac{\partial |J|}{\partial x_q} &= \frac{\partial \phi_q}{\partial \xi} \sum_{i=1}^{n} y_i \frac{\partial \phi_i}{\partial \eta} - \frac{\partial \phi_q}{\partial \eta} \sum_{i=1}^{n} y_i \frac{\partial \phi_i}{\partial \xi}, \\ \frac{\partial |J|}{\partial y_q} &= \frac{\partial \phi_q}{\partial \eta} \sum_{i=1}^{n} x_i \frac{\partial \phi_i}{\partial \xi} - \frac{\partial \phi_q}{\partial \xi} \sum_{i=1}^{n} x_i \frac{\partial \phi_i}{\partial \eta}. \end{aligned} \tag{A.10}$$

Applying these to all the terms in (A.8), we obtain

$$\begin{aligned} \frac{\partial (\mathbf{K}u)_j}{\partial x_q} &= \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big(\frac{\partial \phi_i}{\partial x}|J| \frac{\partial \phi_j}{\partial x}|J| + \frac{\partial \phi_i}{\partial y}|J| \frac{\partial \phi_j}{\partial y}|J| \Big) \Big(\frac{-1}{|J|^2} \frac{\partial |J|}{\partial x_q} \Big) d\xi d\eta \\ &\quad + \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big((-\frac{\partial \phi_q}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} + \frac{\partial \phi_q}{\partial \xi} \frac{\partial \phi_i}{\partial \eta}) \frac{\partial \phi_j}{\partial y}|J| \\ &\qquad + (-\frac{\partial \phi_q}{\partial \eta} \frac{\partial \phi_j}{\partial \xi} + \frac{\partial \phi_q}{\partial \xi} \frac{\partial \phi_j}{\partial \eta}) \frac{\partial \phi_i}{\partial y}|J| \Big) \frac{1}{|J|} d\xi d\eta, \end{aligned} \tag{A.11}$$

which is an integral consisting of only quantities on the reference element—all dependence on x and y has been suppressed, and so this integral can be computed in the algorithm. The other terms can be computed similarly.

A.2. Construction of the Jacobian matrix. In this section we give an example for the construction of the Jacobian matrix. Let us consider the x direction of the Laplace term $\mathbf{Ku}$. We have shown how to compute the first derivative (A.11) of this term. We now compute the second derivative

$$\begin{aligned} \frac{\partial (\mathbf{K}u)_j}{\partial x_q \partial x_k} &= \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big(\frac{\partial \phi_i}{\partial x}|J| \frac{\partial \phi_j}{\partial x}|J| + \frac{\partial \phi_i}{\partial y}|J| \frac{\partial \phi_j}{\partial y}|J| \Big) \\ &\qquad \Big(\frac{2}{|J|} \frac{\partial |J|}{\partial x_q} \frac{\partial |J|}{\partial x_k} - \frac{\partial^2 |J|}{\partial x_q \partial x_k} \Big) \frac{1}{|J|^2} d\xi d\eta \\ &\quad + \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big((-\frac{\partial \phi_k}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} + \frac{\partial \phi_k}{\partial \xi} \frac{\partial \phi_i}{\partial \eta}) \frac{\partial \phi_j}{\partial y}|J| \\ &\qquad + (-\frac{\partial \phi_k}{\partial \eta} \frac{\partial \phi_j}{\partial \xi} + \frac{\partial \phi_k}{\partial \xi} \frac{\partial \phi_j}{\partial \eta}) \frac{\partial \phi_i}{\partial y}|J| \Big) \Big(\frac{-1}{|J|^2} \frac{\partial |J|}{\partial x_q} \Big) d\xi d\eta \\ &\quad + \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big((-\frac{\partial \phi_q}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} + \frac{\partial \phi_q}{\partial \xi} \frac{\partial \phi_i}{\partial \eta}) \frac{\partial \phi_j}{\partial y}|J| \\ &\qquad + (-\frac{\partial \phi_q}{\partial \eta} \frac{\partial \phi_j}{\partial \xi} + \frac{\partial \phi_q}{\partial \xi} \frac{\partial \phi_j}{\partial \eta}) \frac{\partial \phi_i}{\partial y}|J| \Big) \Big(\frac{-1}{|J|^2} \frac{\partial |J|}{\partial x_k} \Big) d\xi d\eta \\ &\quad + \mu \sum_{i=1}^{n} u_i \int_{\Omega^\xi} \Big((-\frac{\partial \phi_q}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} + \frac{\partial \phi_q}{\partial \xi} \frac{\partial \phi_i}{\partial \eta}) (-\frac{\partial \phi_k}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} + \frac{\partial \phi_k}{\partial \xi} \frac{\partial \phi_i}{\partial \eta}) \end{aligned}$$

$$+(-\frac{\partial\phi_q}{\partial\eta}\frac{\partial\phi_j}{\partial\xi}+\frac{\partial\phi_q}{\partial\xi}\frac{\partial\phi_j}{\partial\eta})(-\frac{\partial\phi_k}{\partial\eta}\frac{\partial\phi_j}{\partial\xi}+\frac{\partial\phi_k}{\partial\xi}\frac{\partial\phi_j}{\partial\eta})\Big)\frac{1}{|J|}d\xi d\eta$$

and $\frac{\partial^2|J|}{\partial x_q\partial x_k}=0$ from the definition of $|J|$ (A.5). The other terms can be computed similarly.