A PARALLEL TWO-LEVEL DOMAIN DECOMPOSITION METHOD BASED INEXACT NEWTON METHOD FOR SHAPE OPTIMIZATION PROBLEMS

RONGLIANG CHEN* AND XIAO-CHUAN CAI*

Abstract. A two-level domain decomposition method is introduced for general shape optimization problems constrained by nonlinear partial differential equations. The problem is discretized with a finite element method on unstructured moving meshes and then solved by a parallel two-level one-shot Lagrange-Newton-Krylov-Schwarz algorithm. Due to the pollution effects of the coarse to fine interpolation, direct extensions of the one-level method to two-level do not work. To fix the pollution problem, a pollution removing coarse to fine interpolation scheme is introduced in this paper. As applications, we consider the shape optimization of a cannula problem and an artery bypass problem in 2D. Numerical experiments show that our algorithm performs well on a supercomputer with over one thousand processors for problems with millions of unknowns.

1. Introduction. As large scale computers become readily available for everyday use, engineers and scientists are not only interested in the detailed simulation of a physical process, such as a fluid flow, but also more interested in the optimization of a physical process, such as finding a fluid flow with the smallest energy dissipation [11]. Optimization problems are, generally speaking, more complicated than simulation problems and demand a lot more computing resources. Although many simulation algorithms and software for incompressible flows scale well on supercomputers with thousands or more processors [6], highly scalable algorithms and software for optimization problems constrained by the incompressible Navier-Stokes equations are still to be developed. In this paper we consider the shape optimization problems with equality constraints:

$$\begin{cases} \min\limits_{\mathbf{u},\alpha} & \mathbf{F}(\mathbf{u},\alpha) \\ \text{s.t.} & \mathbf{C}(\mathbf{u},\alpha)=\mathbf{0}, \end{cases} \tag{1.1}$$

where $\mathbf{u}$ is the state variable defined on the domain Ω_α, α is a variable that controls the shape of the computational domain. $\mathbf{F}(\mathbf{u},\alpha)$ and $\mathbf{C}(\mathbf{u},\alpha)$ are referred to as the objective function and the state equation, respectively. In shape optimization problems that we consider, the constraint is the stationary incompressible Navier-Stokes equations. Such problems arise in many industrial applications, for example, aerodynamic shape design [13, 17], artery bypass design [1, 2, 22], and microfluidic biochip design [3]. The difficulty of the problem is mainly due to the complicated constraints that have to be discretized on a sufficiently fine and moving grid. Such computations often require large scale parallel computers for their memory capacity and processing speed. In this paper, we introduce and study a parallel two-level one-shot Lagrange-Newton-Krylov-Schwarz (LNKSz) algorithm for shape optimization problems.

To numerically solve the shape optimization problem (1.1), one can either apply an optimization approach on the continuous level, e.g., the Lagrange multiplier method, to obtain the continuous optimal system and then discretize and solve it, or one can discretize problem (1.1) to obtain a finite dimensional constrained optimization problem and then use a discrete optimization method to solve it. We focus on the latter approach and denote the discretized shape optimization problem as

$$\begin{cases} \min\limits_{\mathbf{u}_h,\alpha} & \mathbf{F}_h(\mathbf{u}_h,\alpha) \\ \text{s.t.} & \mathbf{C}_h(\mathbf{u}_h,\alpha)=\mathbf{0}, \end{cases} \tag{1.2}$$

where h is a mesh size parameter. A common approach for solving the shape optimization problem (1.2) is the Lagrange multiplier method which first defines a Lagrange functional associated with (1.2)

$$L_h(\mathbf{u}_h,\alpha,\lambda_h)=\mathbf{F}_h(\mathbf{u}_h,\alpha)+\lambda_h^T\mathbf{C}_h(\mathbf{u}_h,\alpha), \tag{1.3}$$

where λ_h is a Lagrange multiplier, and then differentiate it to obtain the Karush-Kuhn-Tuker (KKT) system

$$\begin{cases} \nabla_{\lambda_h} L_h(\mathbf{u}_h,\alpha,\lambda_h) &= \mathbf{0} & \text{the state equation,} \\ \nabla_{\mathbf{u}_h} L_h(\mathbf{u}_h,\alpha,\lambda_h) &= \mathbf{0} & \text{the adjoint equation,} \\ \nabla_{\alpha}\ L_h(\mathbf{u}_h,\alpha,\lambda_h) &= \mathbf{0} & \text{the design equation.} \end{cases} \tag{1.4}$$

One popular method for solving the nonlinear system (1.4) is the so-called *nested analysis and design* (NAND) which solves the three equations in (1.4) one at a time, similar to the nonlinear block Gauss-Seidel

*Department of Computer Science, University of Colorado at Boulder, Boulder, CO 80309 (rongliang.chen@colorado.edu, cai@cs.colorado.edu)

method [1, 2, 17, 22]. The advantage of NAND is that one can apply well-developed state equation solvers directly and it requires less memory. The disadvantage of the NAND is that the nonlinear Gauss-Seidel type method often needs many iterations to converge, and sometimes, may not converge without using special scaling. Moreover, in each step of the Gauss-Seidel iterations, the large state equations have to be solved with sufficient accuracy and this is rather time consuming [23, 25]. In addition, the three components have to be solved one after another; such a sequential approach is not desirable on machines with a large number of processors. An alternative approach is the *simultaneous analysis and design* (SAND), or the so-called one-shot method which solves the three equations in (1.4) simultaneously by a nonlinear solver, e.g., Newton-Krylov methods [3, 4, 12, 13]. The main challenges of SAND are that the corresponding Jacobian system in the Newton step is very ill-conditioned and large but, in general, the method based on Newton-Krylov with a good preconditioner is more robust than the method based on nonlinear Gauss-Seidel. 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.

We apply an overlapping domain decomposition method (DDM) to partition the large coupled optimization problem into many independent subproblems. The scalability of DDM is well-studied for scalar elliptic equations ([24]) and some Schwarz preconditioning for boundary control problems can be found in [20, 21]. Through our numerical experiments, we find that the one-level Lagrange-Newton-Krylov-Schwarz method works well for shape optimization problems when the number of processors is small, but when the number of processors is large, unfortunately, the standard two-level method doesn't work as expected because of the computational domain on the coarse-level is not exactly the same as the computational domain on the fine-level. In this paper, we introduce a special interpolation method and a new two-level method that works well for shape optimization problems. Some scalability studies of multilevel preconditioners for boundary control problems can be found in [21], but as far as we know this kind of study has not been done for shape optimization problems. Although both boundary control and shape optimization problems are PDE-constrained optimization problems, the shape optimization problems are much more difficult than the boundary control problems due to the change of the computational domain during the optimization process.

In shape optimization problems, the computational domain changes during the optimization process, therefore one needs to generate a new mesh or allow the existing mesh to deform with the computational domain at each iteration. These two strategies are called mesh reconstruction and mesh perturbation, respectively. Mesh reconstruction often guarantees a good new mesh but is computationally expensive and the mesh perturbation is cheaper but the deformed mesh may become ill-conditioned when the boundary variation is large. In this paper, we focus on the mesh perturbation strategy where we only need to call the mesh generator once before the parallel solver begins. Another advantage of the mesh perturbation method is that when it is used together with an overlapping domain decomposition method the mesh topology doesn't change, so we can reuse the partition of the initial mesh for the entire computation. A common technique of mesh perturbation is to treat the mesh as a network of fictitious linear springs modeled by a system of Laplace's equations [10].

$$\begin{cases} -\Delta\delta_{\mathbf{x}} &= \mathbf{0} \quad \text{in} \quad \Omega_{\alpha_0}, \\ \delta_{\mathbf{x}} &= \mathbf{g}_\alpha \quad \text{on} \quad \partial\Omega_{\alpha_0}, \end{cases} \tag{1.5}$$

where $\delta_{\mathbf{x}}$ is the grid displacement and $\mathbf{g}_\alpha = (\mathbf{g}_\alpha^x, \mathbf{g}_\alpha^y)$ is the displacement on the boundary. Note that $\mathbf{g}_\alpha$ is not a given function, but a function obtained computationally during the iterative solution process. For more moving mesh strategies see, e.g., [17]. We denote the discretized form of (1.5) as

$$\mathbf{D}_h(\delta_{\mathbf{x}_h}, \alpha) = \mathbf{0}. \tag{1.6}$$

In our one-shot method, the mesh variable $\delta_{\mathbf{x}_h}$ is treated as an optimization variable and the moving mesh equations (1.6) are viewed as constraints of the optimization problem which are solved simultaneously with the other equations. Putting (1.2) and (1.6) together, we obtain a new discretized optimization problem

$$\begin{aligned} &\min_{\mathbf{u}_h, \delta_{\mathbf{x}_h}, \alpha} \mathbf{F}_h(\mathbf{u}_h, \delta_{\mathbf{x}_h}, \alpha) \\ &\text{s.t.} \quad \begin{cases} \mathbf{C}_h(\mathbf{u}_h, \delta_{\mathbf{x}_h}, \alpha) = \mathbf{0}, \\ \mathbf{D}_h(\delta_{\mathbf{x}_h}, \alpha) = \mathbf{0}. \end{cases} \end{aligned} \tag{1.7}$$

We denote the KKT system of (1.7) as

$$\mathbf{G}_h(\mathbf{X}_h) = \mathbf{0}, \tag{1.8}$$

where $\mathbf{X}_h \equiv (\mathbf{u}_h, \delta_{\mathbf{x}_h}, \alpha, \lambda_h, \lambda_h^{\mathbf{x}})$ and $\lambda_h^{\mathbf{x}}$ is the Lagrange multiplier for the moving mesh equations.

2. Two-level inexact Newton method with a modified boundary layer interpolation. In this section, we introduce a two-level inexact Newton method for solving the nonlinear KKT system (1.8). In the method, the initial guess for solving the fine grid nonlinear system (1.8) is obtained by: (1) solving a coarse grid problem; (2) modify part of the coarse grid solution near the moving boundary layer; and (3) interpolate the solution to the fine grid. Since Newton type methods can be quite sensitive to the initial guess, our new approach can often reduce the number of Newton iterations and in some cases reduce significantly the number of linear iterations needed inside the Newton steps.

The two-level inexact Newton method begins with constructing an analogous system on the coarse grid

$$\mathbf{G}_H(\mathbf{X}_H) = \mathbf{0} \tag{2.1}$$

and then solves the nonlinear system (2.1) with an inexact Newton method globalized by a line search method. After the problem (2.1) is solved, we interpolate the coarse grid solution $\mathbf{X}_H$ to the fine grid, i.e.,

$$\mathbf{X}_h^0 = \widetilde{I}_H^h \mathbf{X}_H,$$

where $\widetilde{I}_H^h$ is a non-standard coarse to fine interpolation operator to be defined shortly. Finally we solve the fine grid problem (1.8) using an inexact Newton method with the initial guess $\mathbf{X}_h^0$.

The coarse to fine interpolation operator $\widetilde{I}_H^h$ is a $N_h \times N_H$ matrix, where N_h and N_H are the degrees of freedom on the fine grid and the coarse grid, respectively. We first define the standard interpolation operator I_H^h (a $N_h \times N_H$ matrix) which has the components

$$(I_H^h)_{ij} = \phi_H^j(\mathbf{x}_h^i), \tag{2.2}$$

i.e., the value of the j^{th} coarse grid function ϕ_H^j at the i^{th} fine grid point $\mathbf{x}_h^i$. In practice the function $\phi_H^j(\mathbf{x})$ doesn't have to be the same as the finite element basis function used to generate the coarse-level problem, in fact, we use here a multiquadric radial basis function [7]. We tested some other interpolation methods and found that the radial basis function based interpolation method is more efficient for our problem. Unfortunately, the initial guess $\mathbf{X}_h^0$ interpolated from the coarse-level solution $\mathbf{X}_H$ using the interpolation matrix (2.2) does not reduce the initial residual of the fine-level problem as expected. There are two reasons for the failure of the standard interpolation. First, the Lagrange multiplier $\lambda_h^{\mathbf{x}}$ has a sharp jump at the moving boundary which the interpolation operator (2.2) can not catch on the coarse grid. To catch this jump, we introduce a method that is similar to the "pollution removing technique" of [21]. Second, the value of $\lambda_h^{\mathbf{x}}$ on the moving boundary decreases following the mesh refinement. To illustrate the situation, we take the derivative of the coarse-level Lagrangian functional $L_H(\mathbf{X}_H)$ and fine-level Lagrangian functional $L_h(\mathbf{X}_h)$ with respect to the design variable α

$$\sum_{i=1}^{n_H^x} \left(\overline{\lambda}_{i,H}^x \frac{\partial \mathbf{g}_\alpha^x(\mathbf{x}_H^i)}{\partial \alpha} + \overline{\lambda}_{i,H}^y \frac{\partial \mathbf{g}_\alpha^y(\mathbf{x}_H^i)}{\partial \alpha} \right) = -\frac{\beta}{2} \frac{\partial \mathbf{J}_\alpha}{\partial \alpha}, \tag{2.3}$$

$$\sum_{i=1}^{n_h^x} \left(\overline{\lambda}_{i,h}^x \frac{\partial \mathbf{g}_\alpha^x(\mathbf{x}_h^i)}{\partial \alpha} + \overline{\lambda}_{i,h}^y \frac{\partial \mathbf{g}_\alpha^y(\mathbf{x}_h^i)}{\partial \alpha} \right) = -\frac{\beta}{2} \frac{\partial \mathbf{J}_\alpha}{\partial \alpha}, \tag{2.4}$$

where $\overline{\lambda}_{i,h}^x$ and $\overline{\lambda}_{i,h}^y$ $(i = 1, 2, \cdots, n_h^x)$ are the Lagrange multipliers related to the mesh displacement $\delta_{x_{i,h}}$ and $\delta_{y_{i,h}}$ $(i = 1, 2, \cdots, n_h^x)$ on the moving boundary of fine grid, n_h^x is the number of $\delta_{x_{i,h}}$ and $\mathbf{x}_h^i$ is the coordinate of the node associated with $\delta_{x_{i,h}}$. $\overline{\lambda}_{i,H}^x$, $\overline{\lambda}_{i,H}^y$, n_H^x and $\mathbf{x}_H^i$ are values on the coarse grid. Comparing the two equations (2.3) and (2.4), the right-hand side and the coefficients of $\overline{\lambda}_{i,h}^x$, $\overline{\lambda}_{i,h}^y$, $\overline{\lambda}_{i,H}^x$ and $\overline{\lambda}_{i,H}^y$ are functions of α that are supposed to change very little from (2.3) to (2.4), but because $n_h^x \gg n_H^x$, we actually have $\overline{\lambda}_{i,h}^x \ll \overline{\lambda}_{i,H}^x$ and $\overline{\lambda}_{i,h}^y \ll \overline{\lambda}_{i,H}^y$. The standard interpolation operator (2.2) does not respect this phenomenon.

In order to fix the problem, we modify the components of the interpolation operator (2.2) related to the Lagrange multipliers $\overline{\lambda}^x_{i,h}$ and $\overline{\lambda}^y_{i,h}$ by dividing a factor $\gamma = n^x_h/n^x_H$. As a summary, the modified interpolation operator is defined as

$$\widetilde{I}^h_H = R_h(I^h_H - Z_h I^h_H(\mathbf{I}_H - Z_H)), \tag{2.5}$$

where the operator Z_h is a $N_h \times N_h$ diagonal matrix (on the fine grid). The component $(Z_h)_{ii}$ is zero if the index i is for $\overline{\lambda}^x_{i,h}$ or $\overline{\lambda}^y_{i,h}$ $(i = 1, 2, \cdots, n_x)$, otherwise is one. Z_H is similarly defined as Z_h (on the coarse grid) [21]. $\mathbf{I}_H$ is a $N_H \times N_H$ identity matrix and R_h is a $N_h \times N_h$ diagonal matrix with the components $(R_h)_{ii} = \frac{1}{\gamma}$ if the index i is for $\overline{\lambda}^x_{i,h}$ or $\overline{\lambda}^y_{i,h}$ $(i = 1, 2, \cdots, n_x)$, otherwise is one. Fig. 2.1 gives an example of the cross section of the coarse solution, the solution interpolated by (2.2), the solution interpolated by the modified boundary layer interpolation operator (2.5), and the fine solution of λ^x_h near the moving boundary. We show by numerical experiments that the modified interpolation operator (2.5) works very well for the problem under consideration.

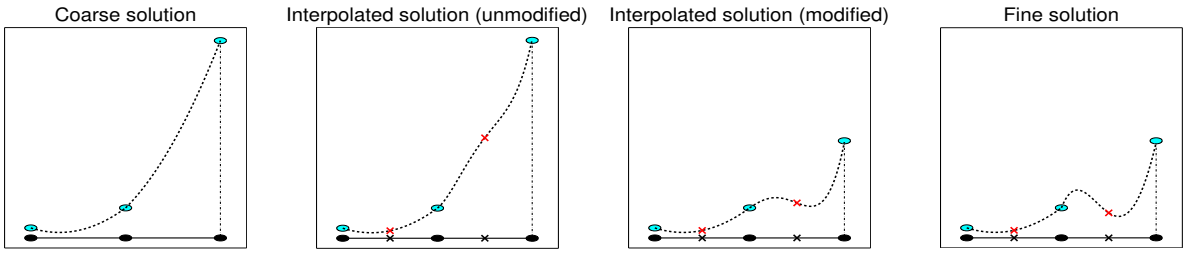

FIG. 2.1. *An example of a cross section of the coarse solution, the unmodified interpolated solution, the modified interpolated solution and the fine solution of λ^x_h near the moving boundary. The right most point is the boundary point. The black • is the grid point on the coarse gird and the black × is grid point on the fine gird. The dashed curve is the solution function and the cyan • and the red × on the solution curve are the values of the solution function on the coarse grid point and fine grid point, respectively.*

The overall two-level inexact Newton algorithm is described as follows:

Algorithm Two-level IN:

Step 1.
- *Given an initial guess $\mathbf{X}^0_H$ and solve the coarse grid problem (2.1) using the following inexact Newton method*
- *For $k = 0, 1, \cdots$ until convergence do:*
  - *Find $\mathbf{d}^k_H$ such that*

$$\| \mathbf{H}^k_H(\mathbf{M}^k_H)^{-1}(\mathbf{M}^k_H \mathbf{d}^k_H) + \mathbf{G}^k_H \| \leq \eta^k_H \| \mathbf{G}^k_H \| \tag{2.6}$$

  - *Set $\mathbf{X}^{k+1}_H = \mathbf{X}^k_H + \tau^k_H \mathbf{d}^k_H$*

Step 2.
- *Interpolate the coarse solution $\mathbf{X}_H$ to the fine grid: $\mathbf{X}^0_h = \widetilde{I}^h_H \mathbf{X}_H$*
- *Moving the initial fine mesh to match the changed computational domain by solving the mesh equations (1.5) with the boundary condition obtained on the coarse grid*

Step 3.
- *Solve the fine grid problem (1.8) using the following inexact Newton method with the initial guess $\mathbf{X}^0_h$*
- *For $k = 0, 1, \cdots$ until convergence do:*
  - *Find $\mathbf{d}^k_h$ such that*

$$\| \mathbf{H}^k_h(\mathbf{M}^k_h)^{-1}(\mathbf{M}^k_h \mathbf{d}^k_h) + \mathbf{G}^k_h \| \leq \eta^k_h \| \mathbf{G}^k_h \| \tag{2.7}$$

  - *Set $\mathbf{X}^{k+1}_h = \mathbf{X}^k_h + \tau^k_h \mathbf{d}^k_h$*

Here $\mathbf{H}^k_H$ and $\mathbf{H}^k_h$ are the Jacobian matrix of the nonlinear system (2.1) and (1.8), respectively, $\mathbf{G}^k_H = \mathbf{G}_H(\mathbf{X}^k_H)$, $\mathbf{G}^k_h = \mathbf{G}_h(\mathbf{X}^k_h)$, τ^k_H and τ^k_h are selected by a cubic line search method, η^k_H and η^k_h are the stopping conditions of the linear solvers for the coarse and fine problems, respectively, and $(\mathbf{M}^k_H)^{-1}$ is a one-level Schwarz preconditioner and $(\mathbf{M}^k_h)^{-1}$ is a two-level Schwarz preconditioner to be defined in the next section.

3. One-level and two-level Schwarz preconditioner. We write the Jacobian system in each Newton step introduced in the previous section as

$$\mathbf{H}\mathbf{x} = \mathbf{b}.$$

To define the 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 $\Omega_{\alpha_l},\ l = 1, \cdots, N_p$ and then extend each subdomain Ω_{α_l} to $\Omega^\delta_{\alpha_l}$ by including δ layers of elements belong to 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 and δ represents the overlap size.

The one-level restricted additive Schwarz preconditioner (RAS) [8] is defined as

$$\mathbf{M}^{-1}_{one} = \sum_{l=1}^{N_p} (R^0_l)^T \mathbf{H}^{-1}_l R^\delta_l, \tag{3.1}$$

where $\mathbf{H}_l$ is a submatrix which is the restriction of $\mathbf{H}$ to the overlapping subdomain $\Omega^\delta_{\alpha_l}$ and R^δ_l is a restriction matrix which maps the global vector of unknowns to those belonging to the subdomain $\Omega^\delta_{\alpha_l}$ by simply extract the unknowns that lie inside the subdomain $\Omega^\delta_{\alpha_l}$. This is equivalent to assuming homogeneous Dirichlet boundary conditions for all variables on the interior part of the boundary of the overlapping subdomain. R^0_l, which is similar to R^δ_l, only extracts the unknowns associated with the non-overlapping subdomain Ω_{α_l} and makes those in the extended subregion $\Omega^\delta_{\alpha_l} \backslash \Omega_{\alpha_l}$ vanish. The one-level preconditioner works well when the number of processors is small, however, when the number of processors is large the Jacobian solver converges very slowly and sometimes does not converge at all. This is because the condition number κ of the preconditioned matrix satisfies (for elliptic systems with sufficient regularity) [24]

$$\kappa \leq \frac{C(1 + \overline{H}/\delta)}{\overline{H}^2}, \tag{3.2}$$

where $\overline{H}$ is the subdomain diameter, δ is the overlap size and C is a constant independent of $\overline{H}$ and δ. The $1/\overline{H}^2$ term in (3.2) implies that the number of iterations of the Jacobian solver increases with the number of subdomains. Our problem is not elliptic, but the performance of the preconditioner does show a similar dependence on $1/\overline{H}^2$. To improve the convergence rate, we introduce a coarse-level preconditioner $\mathbf{M}^{-1}_c$. The two-level hybrid Schwarz preconditioner $\mathbf{M}^{-1}_{two}$ is obtained by multiplicatively combining the coarse-level preconditioner $\mathbf{M}^{-1}_c$ and the fine-level preconditioner $\mathbf{M}^{-1}_{one}$ defined in (3.1). In other words, the application of $\mathbf{M}^{-1}_{two}$ to a vector can be written in the following two steps

$$\mathbf{y} = I^h_H \mathbf{M}^{-1}_c (I^h_H)^T \mathbf{x}, \tag{3.3}$$

$$\mathbf{M}^{-1}_{two}\mathbf{x} = \mathbf{y} + \mathbf{M}^{-1}_{one}(\mathbf{x} - \mathbf{H}\mathbf{y}). \tag{3.4}$$

The two-level preconditioner has better convergence because the condition number κ of the preconditioned matrix satisfies (at least for elliptic problems) [24]

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

which shows that the number of iterations of the Jacobian solver is independent of the number of subdomains. Both (2.6) and (3.3) are linear problems on the same coarse grid and solved by a parallel GMRES with a one-level additive Schwarz preconditioner or a parallel sparse LU factorization.

4. Numerical experiments. The algorithm introduced in the previous sections is applicable to general shape optimization problems. In this section we consider a class of shape optimization problems governed by the stationary incompressible Navier-Stokes equations defined in a two-dimensional domain Ω_α and apply them to cannula optimization and artery bypass design problems. We consider

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

subject to

$$\left\{ \begin{array}{rcll} -\mu\Delta\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p &=& \mathbf{f} & \text{in} \quad \Omega_\alpha, \\ \nabla\cdot\mathbf{u} &=& 0 & \text{in} \quad \Omega_\alpha, \\ \mathbf{u} &=& \mathbf{g} & \text{on} \quad \Gamma_{inlet}, \\ \mathbf{u} &=& \mathbf{0} & \text{on} \quad \Gamma_{wall}, \\ \mu\dfrac{\partial\mathbf{u}}{\partial\mathbf{n}} - p\cdot\mathbf{n} &=& \mathbf{0} & \text{on} \quad \Gamma_{outlet}, \\ \alpha(a) &=& z_1, & \alpha(b) = z_2, \end{array} \right. \tag{4.1}$$

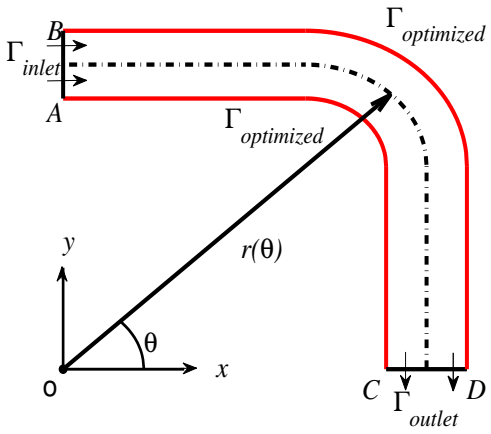

FIG. 4.1. *Simplified model for a blood flow cannula of an artificial heart device; The red boundary $\Gamma_{optimized}$ denotes the part of the boundary whose shape is to be determined by the optimization process.*

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. The first term of the objective function J_o describes the energy dissipation in the whole flow field, 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 the last term of the objective function is a regularization term which provides the regularity of the boundary of the domain Ω_α, where β is a nonnegative constant and $I = [a, b]$ is an interval in which the shape function $\alpha(s)$ is defined. In some approaches [1, 14], some restrictions of the geometry are included in the constraints instead of a regularization term in the objective function. In this paper, we use a regularization term as in [11]. The first five equations of the constraints in (4.1) are the stationary incompressible Navier-Stokes equations and boundary conditions. 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. 4.1. $\mathbf{f}$ is the given body force and $\mathbf{g}$ is the given velocity at the inlet Γ_{inlet}. The last two equations of the constraints in (4.1) are used to restrict the optimized boundary to be connected to the rest of the boundary, where z_1 and z_2 are two given constants [11]. We use a LBB-stable $Q_2 - Q_1$ finite element method to discretize problem (4.1) on a unstructured moving mesh and, Q_2 finite element method for the moving mesh equations (1.5).

Our solver is implemented using PETSc [5]. All computations are performed on a Dell PowerEdge C6100 supercomputer. Meshes are generated with CUBIT [19] and partitioned by ParMETIS [15]. 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. In all experiments, the Jacobian matrix is constructed analytically and the Jacobian system is solved by a right-preconditioned GMRES method. In the additive Schwarz preconditioner, the subdomain problems 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 overlap is 8 layers of elements.

4.1. Cannula optimization. Our first example is a simplified model for a blood flow cannula of an artificial heart device [14] shown in Fig. 4.1. The fluid enters horizontally from left, undergoes a 90^0 bend, and exits vertically downward. It is assumed that the thickness of the cannula is fixed. The goal is to design the shape of the cannula such that the energy dissipation of the fluid in the entire computational domain Ω_α is minimized. For simplicity, we let the body forces $\mathbf{f} = \mathbf{0}$ in the Navier-Stokes equations (4.1). 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} ($\Gamma_{optimized}$ is a wall boundary for the velocity $\mathbf{u}$) and on the outlet boundary Γ_{outlet} the free-stress boundary conditions are imposed; see (4.1). We use a polynomial function $r(\theta) = \sum_{i=1}^{d} r_i\theta^i$, with $d = 5$ to represent the centerline of the cannula (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 [1].

In the first test case, we set the Reynolds number $Re = \frac{Lv_{in}}{\mu}$ to 500, where $L = 1.0$ is the cannula diameter, $v_{in} = 5.0$ is the inlet velocity and $\mu = 0.01$ is the kinematic viscosity. We solve the problem on a mesh with about 14,000 elements and 4.7×10^5 DOFs. Fig. 4.2 shows the velocity distribution of the initial

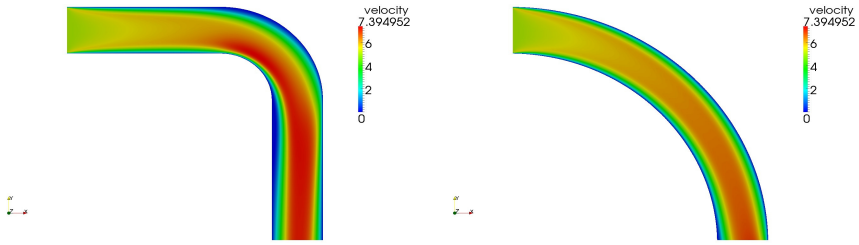

FIG. 4.2. *Velocity distribution of the initial shape (left) and the optimal shape (right) for the cannula design. Here* $\beta = 5.0$ *and* $Re = 500$.

TABLE 4.1

A comparison of the two-level Schwarz method (Two-S), the two-level inexact Newton method (Two-N) and the one-level method (One) for the cannula design. Here $\beta = 10$, $Re = 500$, $DOF(fine) = 1.1 \times 10^6$ *and* $DOF(coarse) = 7.0 \times 10^4$. *In the one-level method:* $\delta = 8$; *in the two-level inexact Newton method:* $\delta_h = 8$ *and* $\delta_H = 4$; *in the two-level Schwarz method:* $\delta_h = 0$ *and* $\delta_H = 2$.

np		32	36	64	72	128	144
Newton	One	6	6	6	6	6	7
	Two-N	4	4	4	4	4	4
	Two-S	4	4	4	4	4	4
GMRES	One	229.83	258.33	345.00	340.00	471.83	498.00
	Two-N	139.50	147.75	194.75	203.75	248.75	270.25
	Two-S	61.25	65.25	73.25	71.50	88.75	91.00
Time	One	756.92	751.93	498.01	439.32	353.49	350.97
	Two-N	481.44	434.72	272.89	249.66	173.03	176.06
	Two-S	345.80	310.58	223.46	191.49	180.25	161.79
Time ratio (coarse/fine)	One	0	0	0	0	0	0
	Two-N	3.3%	3.9%	3.9%	4.2%	4.4%	4.5%
	Two-S	19.5%	25.5%	46.5%	46.0%	66.2%	67.6%

(left) and optimal shapes (right). The energy dissipation of the optimized shape is reduced by about 22.4% compared to the initial shape.

Table 4.1 compares the one-level inexact Newton method (One), the two-level inexact Newton method (without any two-level linear preconditioning) (Two-N) and the two-level Schwarz method (plus two-level inexact Newton) (Two-S). In the rest of this paper, without special mention, the two-level Schwarz method refers to the two-level Schwarz plus the two-level inexact Newton method. In all the tables, the headings "Newton", "GMRES" and "Time" refer to the number of Newton iterations, the average number of GMRES iterations per Newton and the total compute time in seconds, respectively. "Time ratio (coarse/fine)" refer to the percentage of the total time spent on solving the coarse-level problem. "Linear" and "Nonlinear" refer to the coarse linear problem (3.3) and the coarse nonlinear problem (2.1), respectively. $DOF(fine)$ and $DOF(coarse)$ refer to the degrees of freedom on the fine-level and coarse-level, respectively. δ_h is the overlap size on the fine-level and δ_H is that on the coarse-level. From Table 4.1, we see that the parallel scalability for all the methods is good when the number of processors is not large. This table also shows the interesting effect of the two-level inexact Newton approach, namely it reduces not only the number of Newton iterations, but also the number of GMRES iterations. Table 4.2 presents the results of the two-level Schwarz method for a larger problem which the one-level method can not deal with, and from these tables we see that the two-level methods are more efficient than the one-level method and the two-level Schwarz method is the most efficient among all methods.

We mention here that the coarse preconditioner in the two-level Schwarz method, i.e., $\mathbf{M}_c^{-1}$ in (3.3), is solved by a parallel GMRES with a one-level additive Schwarz preconditioner in this subsection. For tests to be presented in the following subsection, we will solve this by a parallel sparse LU factorization SuperLU_DIST [16].

4.2. Artery bypass problem. In this section we study an application of the algorithm for a simplified artery bypass problem [1] as shown in Fig. 4.3. The basic assumption is that there is a complete blockage in the artery, and a bypass needs to be built. 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. Similar to the first example, the goal is to find the best shape of the bypass

TABLE 4.2
Two-level Schwarz method for the cannula design. Here $DOF(fine) = 7.5 \times 10^6$ and $DOF(coarse) = 1.2 \times 10^5$. The other parameters are $\delta_h = 0$, $\delta_H = 2$, $Re = 500$ and $\beta = 10$.

np	Newton	GMRES		Time		Time ratio (coarse/fine)	
		Fine	Coarse	Total	Coarse	Linear	Nonlinear
256	5	150.40	50.81	3222.46	240.55	5.5%	0.8%
512	5	159.20	45.79	959.68	209.89	16.6%	3.1%
1024	6	190.33	45.55	626.54	354.42	48.3%	5.3%

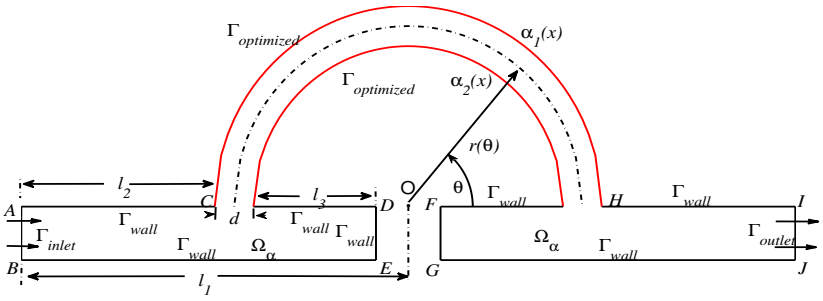

FIG. 4.3. *Simplified two-dimensional bypass model; the red boundary $\Gamma_{optimized}$ denotes the part of the boundary whose shape is to be determined by the optimization process. Here $l_1 = 6.0$, $l_2 = 3.0$, $l_3 = 1.9$, $d_1 = 0.6$, and $|AB| = 0.8$.*

CH, such that the energy dissipation of the fluid in the entire computational domain Ω_α is minimized. All boundary conditions and solver parameters are chosen to be the same as in the first example.

Fig. 4.4 shows the streamline of the initial and optimal shapes. Table 4.3 shows the performance of the two-level Schwarz method. Interestingly, from this table, we see that the average number of GMRES iterations per Newton step is nearly a constant for different number of processors, which matches the convergence theory of the two-level Schwarz method (3.5) for elliptic problems with sufficient shape regularity even though our problem is not elliptic. The speedup of the two-level Schwarz method for the artery bypass design problem is shown in Fig. 4.5. The two-level Schwarz method shows nearly superlinear speedup with up to 1536 processors. In Table 4.4, we present some comparisons of performance for different coarse grid sizes.

5. Concluding remarks. In this paper, we developed a parallel two-level domain decomposition method for shape optimization problems and studied in details for the two-dimensional shape optimization problems governed by incompressible Navier-Stokes equations. The one-level method is relatively easy to develop, but the standard two-level method does not work for shape optimization problems due to the pollution effects of the coarse to fine interpolation. To fix the problem, we introduced a modified boundary layer coarse to fine interpolation operator which works very well for the shape optimization problems. We tested the algorithm for a cannula design problem and an artery bypass design problem in 2D with more than nine million unknowns. The numerical experiments showed that the two-level method has a superlinear speedup with up to 1536 processors. Finally, we would like to mention that all elements of the algorithm can be extended to three-dimensional general shape optimization problems, and we plan to further test the algorithm and software on machines with larger number of processors.

REFERENCES

[1] 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.
[2] 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.
[3] H. ANTIL, M. HEINKENSCHLOSS, AND R. H. W. HOPPE, *Domain decomposition and balanced truncation model reduction for shape optimization of the Stokes system*, Optim. Methods Softw., 26 (2011), pp. 643-669.
[4] 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.
[5] 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, 2011.

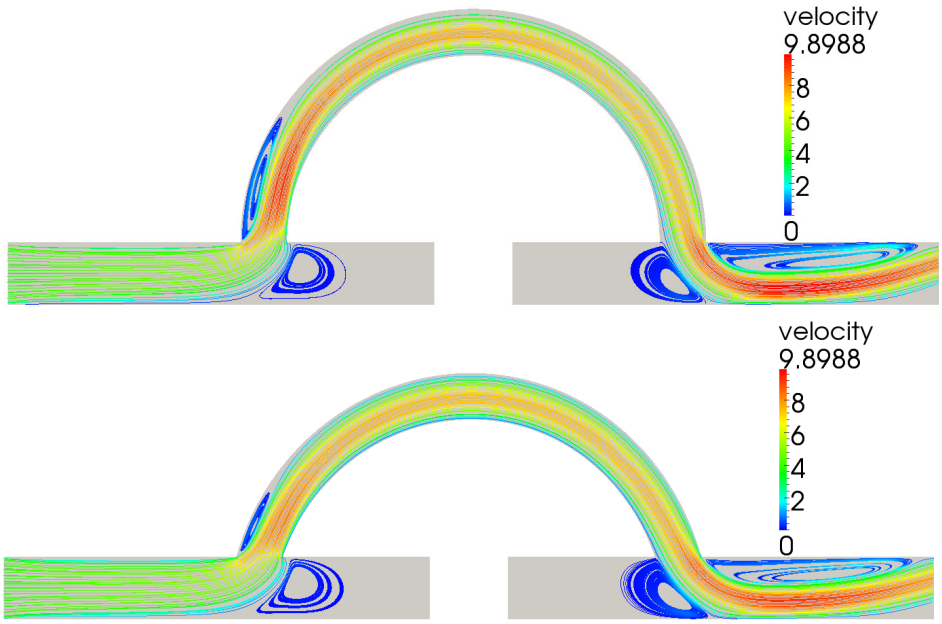

FIG. 4.4. *The streamline of the initial shape (top) and the optimal shape (bottom). Here* $\beta = 5.0$ *and* $Re = 300$.

TABLE 4.3
Two-level Schwarz method for the bypass design. Here $\delta_h = 0$, $Re = 300$, *and* $\beta = 50$.

np		Newton	GMRES	Time		Time ratio (coarse/fine)	
Fine	Coarse			Total	Coarse	Linear	Nonlinear
$DOF(fine) = 8.0 \times 10^6$ and $DOF(coarse) = 1.3 \times 10^5$							
384	96	8	227.00	1949.21	229.19	9.4%	1.6%
576	96	7	160.71	1224.51	385.51	28.7%	2.2%
768	96	7	145.86	707.03	184.52	20.3%	4.9%
1152	96	7	162.71	581.98	223.51	32.3%	4.8%
1536	96	7	196.00	528.18	213.78	30.6%	8.5%
$DOF(fine) = 9.8 \times 10^6$ and $DOF(coarse) = 1.6 \times 10^5$							
384	60	7	157.43	2665.08	238.89	6.5%	2.1%
576	60	7	143.57	1662.37	547.48	28.7%	3.7%
768	60	7	151.57	924.28	230.83	18.9%	5.2%
1152	60	7	169.57	677.37	290.01	34.3%	7.2%
1536	60	7	150.14	459.13	228.28	38.2%	9.8%

[6] A. BARKER AND X.-C. CAI, *Two-level Newton and hybrid Schwarz preconditioners for fluid-structure interaction*, SIAM J. Sci. Comput., 32 (2010), pp. 2395-2417.
[7] M.D. BUHMANN, *Radial Basis Function: Theory and Implementations*, Cambridge University Press, Cambridge, 2003.
[8] X.-C. CAI AND M. SARKIS, *A restricted additive Schwarz preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 21 (1999), pp. 792-797.
[9] S. C. EISENSTAT AND H. F. WALKER, *Globally convergent inexact Newton method*, SIAM J. Optim., 4 (1994), pp. 393-422.
[10] C. FARHAT, C. DEGAND, B. KOOBUS, AND M. LESOINNE, *Torsional springs for two-dimensional dynamic unstructured fluid meshes*, Comput. Methods Appl. Mech. Engrg., 163 (1998), pp. 231-245.
[11] M. D. GUNZBURGER, *Perspectives in Flow Control and Optimization—Advances in Design and Control*, SIAM, Philadelphia, 2003.
[12] O. GHATTAS AND C. OROZCO, *A parallel reduced Hessian SQP method for shape optimization*, In: Natalia M. Alexandrov and M.Y. Hussaini, eds., Multidisciplinary Design Optimization: State of the Art, SIAM, Philadelphia (1997), pp. 133-152.
[13] S. B. HAZRA, *Multigrid one-shot method for aerodynamic shape optimization*, SIAM J. Sci. Comput., 30 (2008), pp. 1527-1547.
[14] 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.
[15] G. KARYPIS, *METIS/ParMETIS web page*, University of Minnesota, 2011. `http://glaros.dtc.umn.edu/gkhome/views/metis`.
[16] X. S. LI AND J. W. DEMMEL, *SuperLU_DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems*, ACM Trans. Math. Software, 29 (2003), pp. 110-140.
[17] B. MOHAMMADI AND O. PIRONNEAU, *Applied Shape Optimization for Fluids*, Oxford University Press, Oxford, 2001.

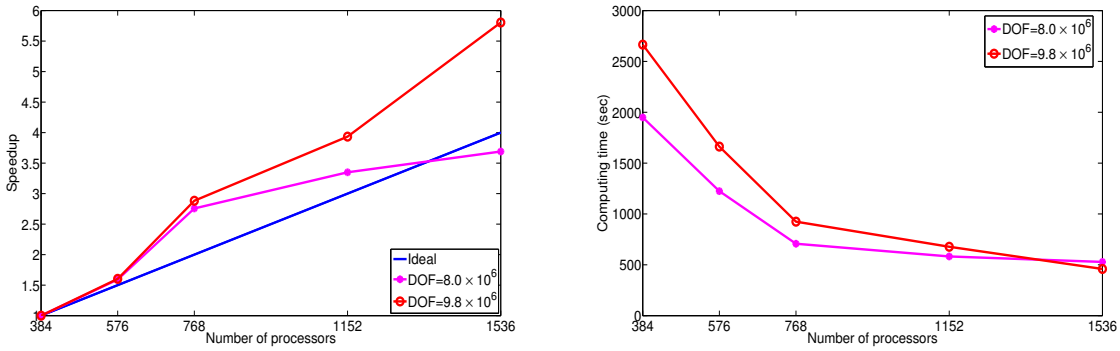

FIG. 4.5. *The speedup and the total compute time for two different mesh sizes.*

TABLE 4.4

The effect of various choices of the size of the coarse mesh for the bypass design. Here $DOF(fine) = 9.8 \times 10^6$, $\delta_h = 0$, $Re = 300$, *and* $\beta = 50$.

$DOF(coarse)$	Newton	GMRES	Time		Time ratio (coarse/fine)	
			Total	Coarse	Linear	Nonlinear
4.1×10^4	8	663.88	1748.55	343.05	15.4%	0.9%
7.3×10^4	9	585.00	1584.95	292.40	13.8%	2.2%
1.6×10^5	7	150.57	1019.95	312.67	24.6%	5.3%
2.0×10^5	7	234.86	1312.71	504.05	29.5%	6.6%

[18] J. NOCEDAL AND S. J. WRIGHT, *Numerical Optimization*, First ed., Springer-Verlag, Berlin, 2000.
[19] S. J. OWEN AND J. F. SHEPHERD, *CUBIT project web page*, 2011, `http://cubit.sandia.gov/`.
[20] 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.
[21] 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.
[22] 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.
[23] T. REES, M. STOLL, AND A. WATHEN, *All-at-once preconditioning in PDE-constrained optimization*, Kybernetika, 46 (2010), pp. 341-360.
[24] A. TOSELLI AND O. WIDLUND, *Domain Decomposition Methods: Algorithms and Theory*, Springer-Verlag, Berlin, 2005.
[25] A. WATHEN, *Preconditioning for PDE-constrained optimization*, SIAM News, 43 (2010).