An Optimization-Based Domain Decomposition Method for Numerical Simulation of the Incompressible Navier-Stokes Flows

Fande Kong,[1] **Yichen Ma,**[1] **Junxiang Lu**[2]
[1] *School of Science, Xi'an Jiaotong University, Shaanxi, People's Republic of China, 710049*

[2] *School of Science, Xi'an Polytechnic University, Shaanxi, People's Republic of China, 710048*

Received 30 March 2009; accepted 18 May 2009
Published online 16 October 2009 in Wiley Online Library (wileyonlinelibrary.com).
DOI 10.1002/num.20519

This article is concerned about an optimization-based domain decomposition method for numerical simulation of the incompressible Navier-Stokes flows. Using the method, an classical domain decomposition problem is transformed into a constrained minimization problem for which the objective functional is chosen to measure the jump in the dependent variables across the common interfaces between subdomains. The Lagrange multiplier rule is used to transform the constrained optimization problem into an unconstrained one and that rule is applied to derive an optimality system from which optimal solutions may be obtained. The optimality system is also derived using "sensitivity" derivatives instead of the Lagrange multiplier rule. We consider a gradient-type approach to the solution of domain decomposition problem. The results of some numerical experiments are presented to demonstrate the feasibility and applicability of the algorithm developed in this article. Numer Methods Partial Differential Eq 27: 255–276, 2011

Keywords: domain decomposition method; finite element method; lagrange multiplier rule; Navier-Stokes flows

I. INTRODUCTION

Numerical methods based on domain decomposition are used in a wide variety of applications as a powerful technique to compute solutions of partial differential equations. The basic idea of a domain decomposition method is to split the whole domain into smaller ones so that the overall solution of a large problem can be obtained by solving smaller problems in subdomains. The methods are especially good when a computer's memory is not large enough for the complete problem or when a domain has an irregular shape, which often happens in practical applications. Because of the obvious implication for parallel processing, domain decomposition methods have been greatly studied [1–5].

Correspondence to: Fande Kong, School of Science, Xi'an Jiaotong University, P. O. Box 1748, Xi'an, Shaanxi, People's Republic of China, 710049 (e-mail: fdkong.jd@gmail.com)
Contract grant sponsor: The National Natural Science Fund of China; contract grant numbers: 10671153, 40730424

Domain decomposition methods can be classified as either a nonoverlapping or an overlapping subdomains problem. The classic domain decoposition method for overlapping subdomains is the Schwarz alternating method, in which subdomain problems are solved successively with boundary conditions obtained in the previous iteration. Schwarz's algorithm has been studied and developed from various points of view; e.g., [6–10]. Based on Lions' work [9, 10], Dryja and Widlund developed and analyzed the additive Schwarz methods for more effective parallel computing in [7, 8]. Another well-known family of domain decomposition method is the so-called substructuring iteration methods or Schur methods, which are for nonoverlapping subdomain problems. The main technique of these methods is to reduce an elliptic problem to an operator equation for lower-dimensional interfaces between subdomains. The main focus in the study of these methods has been to find and analyze related preconditioners [11]. Of course, it was shown by Dryja and Widlund [12] that substructuring methods and Schwarz methods can be unified in certain cases.

In this article, we propose a new optimization-based domain decomposition method by which we develop and implement an effective numerical algorithm for the Navier-Stokes equations, which describe the motion of incompressible viscous fluids. Domain decomposition methods based on optimization strategies have been previously proposed in [13–19]. The basic idea of these methods is that an appropriate cost functional is minimized so that the optimal solution satisfies the partial differential equations, which are linear or nonlinear and the constraints force the solutions on the two subdomains to agree on the common interface. In this article, differently from the previous articles [14–19], two penalty factors for the cost functional are introduced to improve the convergence rate, and Robin boundary conditions are used to be the interfaces conditions (Neumann or Dirichlet boundary conditions used in [14–18], and Nonlinear boundary conditions used in [19]). The optimization-based domain decomposition method considered here offers some advantageous features. First, compared with some similar methods discussed by other articles, the method can greatly improve the convergence rate, which is very important in practical applications. Second, the method provides an effective way to introduce parallelism into practical problem which may not exhibit obvious, inherent parallelism. In addition, an numerical algorithm developed by this method is so symmetrical that the same program code can be used in different subdomains. We have developed a significant amount of research on numerical methods for steady and unsteady Euler and Navier-Stokes equations [20].

The plan of the article is as follows. In Section II, the optimization-based domain decomposition method for the Navier-Stokes flows is introduced, and then we give some notations and weak formulations that will be used throughout the article. In Section III, we show that the Lagrange multiplier rule may be used to transform the constrained optimization problem into an unconstrained one and that rule is applied to derive an optimality system from which optimal solutions may be obtained. The optimality system is also derived using "sensitivity" derivatives instead of the Lagrange multiplier rule, and then we consider a gradient-type approach to the solution of domain decomposition problem. In Section IV, we present some numerical results, which show the algorithm constructed here is efficient.

II. STATEMENT OF THE DOMAIN DECOMPOSITION METHOD

A. The Problem Model

Let Ω denote a bounded open set in $\mathbb{R}^2$ with boundary Γ. Let $\boldsymbol{u}$ denote the velocity vector, p the pressure, $\boldsymbol{f}$ a given body force, and ν the constant kinematic viscosity; then the Navier-Stokes system which describes incompressible and viscous fluids is

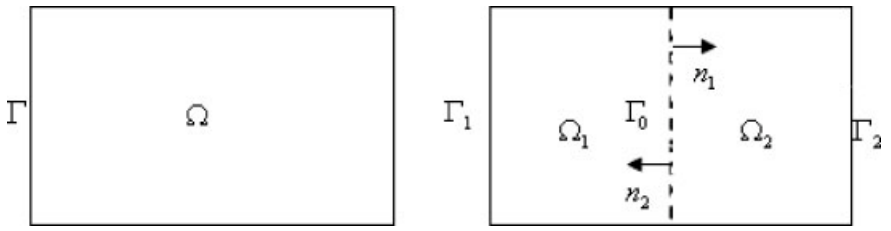

FIG. 1. The domain Ω divided into two nonoverlapping subdomains.

$$\begin{cases} -\nu\Delta \boldsymbol{u} + (\boldsymbol{u}\cdot\ \nabla)\boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{in } \Omega \\ \nabla\cdot\boldsymbol{u} = 0 & \text{in } \Omega \\ \boldsymbol{u} = 0 & \text{on } \Gamma. \end{cases} \tag{2.1}$$

In the simplest case, Ω is partitioned into two simply connected nonoverlapping subdomains Ω_1 and Ω_2, so that $\bar{\Omega} = \bar{\Omega}_1 \cup \bar{\Omega}_2$. The interface between the two subdomains is denoted by Γ_0 so that $\Gamma_0 = \bar{\Omega}_1 \cap \bar{\Omega}_2$. Let $\Gamma_1 = \bar{\Omega}_1 \cap \Gamma$ and $\Gamma_2 = \bar{\Omega}_2 \cap \Gamma$ (Fig. 1). Regularity conditions on the common interface Γ_0 will be assumed later.

Consider the following pair of Navier-Stokes systems with mixed boundary conditions:

$$\begin{cases} -\nu\Delta \boldsymbol{u}_1 + (\boldsymbol{u}_1\cdot\ \nabla)\boldsymbol{u}_1 + \nabla p_1 = \boldsymbol{f} & \text{in } \Omega_1 \\ \nabla\cdot\boldsymbol{u}_1 = 0 & \text{in } \Omega_1 \\ \boldsymbol{u}_1 = 0 & \text{on } \Gamma_1 \\ -p_1\boldsymbol{n}_1 + \nu\nabla\boldsymbol{u}_1\cdot\boldsymbol{n}_1 + k\boldsymbol{u}_1 = \boldsymbol{g} & \text{on } \Gamma_0 \end{cases} \tag{2.2}$$

and

$$\begin{cases} -\nu\Delta \boldsymbol{u}_2 + (\boldsymbol{u}_2\cdot\ \nabla)\boldsymbol{u}_2 + \nabla p_2 = \boldsymbol{f} & \text{in } \Omega_2 \\ \nabla\cdot\boldsymbol{u}_2 = 0 & \text{in } \Omega_2 \\ \boldsymbol{u}_2 = 0 & \text{on } \Gamma_2 \\ -p_2\boldsymbol{n}_2 + \nu\nabla\boldsymbol{u}_2\cdot\boldsymbol{n}_2 - k\boldsymbol{u}_2 = -\boldsymbol{g} & \text{on } \Gamma_0. \end{cases} \tag{2.3}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ denote the outward unit normal vectors to Ω_1 and Ω_2 (Fig. 1), respectively. Note that the last equations in (2.2) and (2.3) are Robin boundary conditions for Navier-Stokes systems. Of course, other boundary conditions can be chosen, e.g., generalized stress conditions chosen in [16] and nonlinear transmission conditions chosen in [19]. For the sake of simplicity, we assume that the Robin coefficient k is independent of $(\boldsymbol{u}_i, p_i), i = 1$ and 2.

For an arbitrary choice of $\boldsymbol{g}$, solutions of (2.2) and (2.3) are not solutions of (2.1), e.g., $\boldsymbol{u}_1 \neq \boldsymbol{u}\ |_{\Omega_1}$ and $\boldsymbol{u}_2 \neq \boldsymbol{u}\ |_{\Omega_2}$. In particular, we have that, in general, $\boldsymbol{u}_1\ |_{\Gamma_0} \neq \boldsymbol{u}_2\ |_{\Gamma_0}$. However, we know that a $\boldsymbol{g}$ exists such that the solutions of (2.2) and (2.3) are indeed the solutions of (2.1); we merely choose

$$\boldsymbol{g} = -p\boldsymbol{n}_1 + \nu\nabla\boldsymbol{u}\cdot\boldsymbol{n}_1 + k\boldsymbol{u},$$

where $(\boldsymbol{u}, p)$ is a solution of (2.1).

The optimization-based domain decomposition method finds such a $\boldsymbol{g}$ by minimizing the cost functional, which measures the difference $\boldsymbol{u}_1 - \boldsymbol{u}_2$ along the common interface Γ_0. Various energy (cost, objective) functionals (of $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ and other auxiliary variables) are defined so that their minimizers correspond to solutions of (2.2) and (2.3) that satisfy (2.1). With different functionals

and solution strategies, various domain decomposition algorithms are obtained [14, 16, 19]. In fact, with special choices of the functionals and the solution strategies for the optimization problems (such as the approach of alternating variables or alternating directions), various well-known nonoverlapping algorithms can be derived from the optimization-based framework.

Here, we only restrict our interest to the velocity vector $\boldsymbol{u}$ for our domain decomposition method and try to find a $\boldsymbol{g}$ that minimizes the $\boldsymbol{L}^2(\Gamma_0)$ norm of $\boldsymbol{u}_1 - \boldsymbol{u}_2$ on the common interface Γ_0. Hence, the functional to be minimized is

$$\tilde{\mathcal{J}}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g})) = \frac{1}{2}\int_{\Gamma_0} (\boldsymbol{u}_1 - \boldsymbol{u}_2)^2 \, \mathrm{d}\Gamma_0, \tag{2.4}$$

where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are determined from $\boldsymbol{g}$ through (2.2) and (2.3), respectively. It is clear that there exists a minimizer for $\tilde{\mathcal{J}}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}))$; indeed, for the choice

$$\boldsymbol{g} = -p\boldsymbol{n}_1 + \nu \nabla \boldsymbol{u} \cdot \boldsymbol{n}_1 + k\boldsymbol{u},$$

where $(\boldsymbol{u}, p)$ is a solution of (2.1), we have that $\boldsymbol{u}_1 = \boldsymbol{u}\ |_{\Omega_1}$ and $\boldsymbol{u}_2 = \boldsymbol{u}\ |_{\Omega_2}$ so that $\tilde{\mathcal{J}}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g})) = 0$. To regularize the optimization problem and improve the convergence rate, instead of minimizing (2.4), we minimize the penalized functional

$$\mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g}) = \frac{\alpha}{2}\int_{\Gamma_0} (\boldsymbol{u}_1 - \boldsymbol{u}_2)^2 \, \mathrm{d}\Gamma_0 + \frac{\beta}{2}\int_{\Gamma_0} \boldsymbol{g}^2 \, \mathrm{d}\Gamma_0. \tag{2.5}$$

where α and β are positive constants that can be chosen to change the relative importance of the two terms appearing in (2.5). The convergence of the resulting algorithm is evidently affected by the chosen weights and the choice of independent variables [14]. Hence, a penalty parameter for the first term of the cost functional is introduced to adjust the convergence. The penalized cost functional (2.5) we consider here is generalized one from which others can be derived. For example, the cost functional used in [16] is a direct derivation of (2.5), where $\alpha = 1$.

In general, the optimization problem we propose to solve is given:

$$\min_{\boldsymbol{g}} \mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g}), \tag{2.6}$$

where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are solutions of (2.2) and (2.3), respectively, as a control $\boldsymbol{g}$ in some admissibility set is given. For the sake of simplicity, we only consider the case of only two subdomains. The extension of the algorithms and analysis to more subdomains is straightforward.

B. Notations and Weak Formulations for the Navier-Stokes Equations

We start by introducing some notations that will be used throughout this article. Let $H^r(\Omega)$ denote the standard Sobolev space of order r on a domain Ω, equipped with the standard norm $\|\cdot\|_{r,\Omega}$; let $\boldsymbol{H}^r(\Omega)$ denote the corresponding Sobolev space of vector-valued functions. We then define the subspaces

$$\boldsymbol{H}_0^r(\Omega) = \{\boldsymbol{v} \in \boldsymbol{H}^r(\Omega) : \boldsymbol{v} = 0 \text{ on } \Gamma\}$$

and, for $i = 1$ and 2,

$$\boldsymbol{H}_{\Gamma_i}^r(\Omega) = \left\{\boldsymbol{v} \in \boldsymbol{H}^r(\Omega_i) : \boldsymbol{v} = 0 \text{ on } \Gamma_i\right\}.$$

We also define, for $i = 1$ and 2 and whenever $\boldsymbol{u} \cdot \boldsymbol{v},\ pq \in L^2(\Omega_i)$,

$$(\boldsymbol{u}, \boldsymbol{v})_{\Omega_i} = \int_{\Omega_i} \boldsymbol{u} \cdot \boldsymbol{v}\, \mathrm{d}\Omega_i,$$

$$(p, q)_{\Omega_i} = \int_{\Omega_i} pq\, \mathrm{d}\Omega_i,$$

for $i = 0, 1$ and 2 and whenever $\boldsymbol{u} \cdot \boldsymbol{v},\ pq \in L^2(\Gamma_i)$,

$$(\boldsymbol{u}, \boldsymbol{v})_{\Gamma_i} = \int_{\Gamma_i} \boldsymbol{u} \cdot \boldsymbol{v}\, \mathrm{d}\Gamma_i,$$

$$(p, q)_{\Gamma_i} = \int_{\Gamma_i} pq\, \mathrm{d}\Gamma_i.$$

The following bilinear and trilinear forms on the domain Ω are defined as follows:

$$\mathcal{A}(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, \mathrm{d}\Omega, \qquad \forall \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega),$$

$$\mathcal{B}(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v}) = \int_{\Omega} (\boldsymbol{u} \cdot \nabla)\boldsymbol{w} \cdot \boldsymbol{v}\, \mathrm{d}\Omega, \quad \forall \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega),$$

$$\mathcal{C}(\boldsymbol{u}, p) = \int_{\Omega} p \operatorname{div} \boldsymbol{u}\, \mathrm{d}\Omega, \qquad \forall \boldsymbol{u} \in \boldsymbol{H}_0^1(\Omega), \forall p \in L^2(\Omega).$$

Analogously, we also define bilinear and trilinear forms corresponding to subdomains $\Omega_i, i = 1$ and 2:

$$\mathcal{A}_i(\boldsymbol{u}, \boldsymbol{v}) = \int_{\Omega_i} \nabla \boldsymbol{u} \cdot \nabla \boldsymbol{v}\, \mathrm{d}\Omega_i, \qquad \forall \boldsymbol{u}, \boldsymbol{v} \in \boldsymbol{H}_{\Gamma_i}^1(\Omega_i), i = 1, 2;$$

$$\mathcal{B}_i(\boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v}) = \int_{\Omega_i} (\boldsymbol{u} \cdot \nabla)\boldsymbol{w} \cdot \boldsymbol{v}\, \mathrm{d}\Omega_i, \quad \forall \boldsymbol{u}, \boldsymbol{w}, \boldsymbol{v} \in \boldsymbol{H}_{\Gamma_i}^1(\Omega_i), i = 1, 2;$$

$$\mathcal{C}_i(\boldsymbol{u}, p) = \int_{\Omega_i} p \operatorname{div} \boldsymbol{u}\, \mathrm{d}\Omega_i, \qquad \forall \boldsymbol{u} \in \boldsymbol{H}_{\Gamma_i}^1(\Omega_i), \forall p \in L^2(\Omega_i), i = 1, 2.$$

It is well known that the forms $\mathcal{A}(\cdot,\cdot)$, $\mathcal{B}(\cdot,\cdot,\cdot)$, $\mathcal{C}(\cdot,\cdot)$, $\mathcal{A}_i(\cdot,\cdot)$, $\mathcal{B}_i(\cdot,\cdot,\cdot)$, and $\mathcal{C}_i(\cdot,\cdot)$, for $i = 1$ and 2, are continuous. Also, $\mathcal{A}(\cdot,\cdot)$ and $\mathcal{A}_i(\cdot,\cdot)$, for $i = 1$ and 2, satisfy the coercivity property. $\mathcal{C}(\cdot,\cdot)$ and $\mathcal{C}_i(\cdot,\cdot)$, for $i = 1$ and 2, satisfy the inf-sup (LBB) condition.

We assume that the given body force $\boldsymbol{f} \in \boldsymbol{L}^2(\Omega)$. Using the above notations, the weak formulations corresponding to (2.2) and (2.3) are given : find $\boldsymbol{u} \in \boldsymbol{H}_{\Gamma_i}^1(\Omega_i)$ and $p_i \in L^2(\Omega_i)$, for $i = 1$ and 2 such that

$$\begin{cases} \nu\mathcal{A}_1(\boldsymbol{u}_1, \boldsymbol{v}) + \mathcal{B}_1(\boldsymbol{u}_1, \boldsymbol{u}_1, \boldsymbol{v}) - \mathcal{C}_1(\boldsymbol{v}, p_1) + k(\boldsymbol{u}_1, \boldsymbol{v})_{\Gamma_0} = (\boldsymbol{f}, \boldsymbol{v})_{\Omega_1} + (\boldsymbol{g}, \boldsymbol{v})_{\Gamma_0}, & \forall \boldsymbol{v} \in \boldsymbol{H}_{\Gamma_1}^1(\Omega_1), \\ -\mathcal{C}_1(\boldsymbol{u}_1, q) = 0, & \forall q \in L^2(\Omega_1) \end{cases} \tag{2.7}$$

and

$$\begin{cases} \nu\mathcal{A}_2(\boldsymbol{u}_2, \boldsymbol{v}) + \mathcal{B}_2(\boldsymbol{u}_2, \boldsymbol{u}_2, \boldsymbol{v}) - \mathcal{C}_2(\boldsymbol{v}, p_2) - k(\boldsymbol{u}_2, \boldsymbol{v})_{\Gamma_0} = (\boldsymbol{f}, \boldsymbol{v})_{\Omega_2} - (\boldsymbol{g}, \boldsymbol{v})_{\Gamma_0}, & \forall \boldsymbol{v} \in \boldsymbol{H}_{\Gamma_2}^1(\Omega_2), \\ -\mathcal{C}_2(\boldsymbol{u}_2, q) = 0, & \forall q \in L^2(\Omega_2) \end{cases} \tag{2.8}$$

Similarly, the weak formulations associated with (2.1) are also given : find $\boldsymbol{u} \in \boldsymbol{H}_0^1(\Omega)$ and $p \in L^2(\Omega)$ such that

$$\begin{cases} \nu\mathcal{A}(\boldsymbol{u},\boldsymbol{v}) + \mathcal{B}(\boldsymbol{u},\boldsymbol{u},\boldsymbol{v}) - \mathcal{C}(\boldsymbol{v},p) = (\boldsymbol{f},\boldsymbol{v})_\Omega, \quad \forall \boldsymbol{v} \in \boldsymbol{H}_0^1(\Omega), \\ -\mathcal{C}(\boldsymbol{u},q) = 0, \quad \forall q \in L^2(\Omega). \end{cases} \tag{2.9}$$

To make a summary, in the remainder of this section we give a precise statement of the optimization problem for the domain decomposition method of the Navier-Stokes equations. Let the admissibility set $\mathcal{U}_{ad}$ be defined by:

$$\mathcal{U}_{ad} = \Big\{ \boldsymbol{g} \in \boldsymbol{L}^2(\Gamma_0) : \text{ there exist } (\boldsymbol{u}_i, p_i) \in \boldsymbol{H}^1_{\Gamma_i}(\Omega_i) \times L^2(\Omega_i) \text{ for } i = 1,2, \\ \text{so that (2.7) and (2.8) are satisfied and } \mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g}) < \infty. \Big\}$$

Thus, the domain decomposition method for Navier-Stokes equations is transformed into the following optimization problem:

$$\min_{\boldsymbol{g} \in \mathcal{U}_{ad}} \mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g}). \tag{2.10}$$

Our goal is to find an optimal solution $\boldsymbol{g}$ for the optimization problem (2.10), such that the solutions of (2.7) and (2.8) are indeed solutions of (2.9).

III. THE OPTIMALITY SYSTEM

In this section, an optimality system is derived by the Lagrange multiplier rule and "sensitivity" derivatives, respectively. A gradient-type method is also considered.

A. The Lagrange Multiplier Rule

We use a Lagrange multiplier rule to reduce the constrained minimization problem (2.10) to an unconstrained one, deriving the first-order necessary conditions which the optimal solutions must satisfy. For $(\boldsymbol{u}_1, p_1, \boldsymbol{u}_2, p_2, \boldsymbol{g}; \boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2, \mu_2) \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1) \times L^2(\Omega_1) \times \boldsymbol{H}^1_{\Gamma_2}(\Omega_2) \times L^2(\Omega_2) \times \boldsymbol{L}^2(\Gamma_0) \times \boldsymbol{H}^1_{\Gamma_1}(\Omega_1) \times L^2(\Omega_1) \times \boldsymbol{H}^1_{\Gamma_2}(\Omega_2) \times L^2(\Omega_2)$, define the Lagrangian functional:

$$\begin{aligned} \mathcal{L}(\boldsymbol{u}_1, p_1, \boldsymbol{u}_2, p_2, \boldsymbol{g}; \boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2, \mu_2) = {} & \mathcal{J}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{g}) + \sum_{i=1}^{2} (\nu\mathcal{A}_i(\boldsymbol{u}_i, \boldsymbol{\varphi}_i) + \mathcal{B}_i(\boldsymbol{u}_i, \boldsymbol{u}_i, \boldsymbol{\varphi}_i) \\ & - \mathcal{C}_i(\boldsymbol{\varphi}_i, p_i) + (-1)^{i-1} k(\boldsymbol{u}_i, \boldsymbol{\varphi}_i)_{\Gamma_0} \\ & - (\boldsymbol{f}, \boldsymbol{\varphi}_i)_{\Omega_i} + (-1)^i (\boldsymbol{g}, \boldsymbol{\varphi}_i)_{\Gamma_0} - \mathcal{C}_i(\boldsymbol{u}_i, \mu_i)) \end{aligned} \tag{3.1}$$

where $\boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2$, and μ_2 are Lagrange multipliers. Hence, the constrained problem (2.10) can now be transformed into the unconstrained one of finding stationary points of $\mathcal{L}(\boldsymbol{u}_1, p_1, \boldsymbol{u}_2, p_2, \boldsymbol{g}; \boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2, \mu_2)$ (3.1). We now apply the necessary conditions for the latter problem (3.1).

First, we are interesting in finding an expression for the Eulerian derivative of the Lagrangian functional $\mathcal{L}(\boldsymbol{u}_1, p_1, \boldsymbol{u}_2, p_2, \boldsymbol{g}; \boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2, \mu_2)$ at $\boldsymbol{\varphi}_1$ in the direction $\delta\boldsymbol{\varphi}_1 \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1)$, which is

denoted by $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_1}\delta\boldsymbol{\varphi}_1$. Easily, an expression for the Eulerian derivative of the Lagrangian functional $\mathcal{L}(\boldsymbol{u}_1, p_1, \boldsymbol{u}_2, p_2, \boldsymbol{g}; \boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2, \mu_2)$ (3.1) is given by:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_1}\delta\boldsymbol{\varphi}_1 = \nu\mathcal{A}_1(\boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1) + \mathcal{B}_1(\boldsymbol{u}_1, \boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1) - \mathcal{C}_1(\delta\boldsymbol{\varphi}_1, p_1) - (\boldsymbol{f}, \delta\boldsymbol{\varphi}_1)_{\Omega_1} - (\boldsymbol{g}, \delta\boldsymbol{\varphi}_1)_{\Gamma_0} + k(\boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1)_{\Gamma_0}. \tag{3.2}$$

Set $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_1}\delta\boldsymbol{\varphi}_1 = 0$, and then we have the following equations:

$$\nu\mathcal{A}_1(\boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1) + \mathcal{B}_1(\boldsymbol{u}_1, \boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1) - \mathcal{C}_1(\delta\boldsymbol{\varphi}_1, p_1) + k(\boldsymbol{u}_1, \delta\boldsymbol{\varphi}_1)_{\Gamma_0} = (\boldsymbol{f}, \delta\boldsymbol{\varphi}_1)_{\Omega_1} + (\boldsymbol{g}, \delta\boldsymbol{\varphi}_1)_{\Gamma_0}, \forall\delta\boldsymbol{\varphi}_1 \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1). \tag{3.3}$$

Obviously, the Eq. (3.3) is just the first relation of (2.7). In the same way, $\frac{\partial \mathcal{L}}{\partial \mu_1}\delta\mu_1$ can be obtained, and the related expression is shown by:

$$\frac{\partial \mathcal{L}}{\partial \mu_1}\delta\mu_1 = -\mathcal{C}_1(\boldsymbol{u}_1, \delta\mu_1). \tag{3.4}$$

Setting $\frac{\partial \mathcal{L}}{\partial \mu_1}\delta\mu_1$ (3.4) to be zero yields the following equations:

$$-\mathcal{C}_1(\boldsymbol{u}_1, \delta\mu_1) = 0, \forall\delta\mu_1 \in L^2(\Omega_1). \tag{3.5}$$

The earlier equations is "incompressibility" constraint for (3.3). Thus, the constraint equations (2.7) are an direct derivation from (3.3) and (3.5). Analogously, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_2}\delta\boldsymbol{\varphi}_2$ and $\frac{\partial \mathcal{L}}{\partial \mu_2}\delta\mu_2$ can be defined, respectively, for which the related expressions are shown by:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_2}\delta\boldsymbol{\varphi}_2 = \nu\mathcal{A}_2(\boldsymbol{u}_2, \delta\boldsymbol{\varphi}_2) + \mathcal{B}_2(\boldsymbol{u}_2, \boldsymbol{u}_2, \delta\boldsymbol{\varphi}_2) - \mathcal{C}_2(\delta\boldsymbol{\varphi}_2, p_2) - (\boldsymbol{f}, \delta\boldsymbol{\varphi}_2)_{\Omega_2} + (\boldsymbol{g}, \delta\boldsymbol{\varphi}_2)_{\Gamma_0} - k(\boldsymbol{u}_2, \delta\boldsymbol{\varphi}_2)_{\Gamma_0}, \tag{3.6}$$

and

$$\frac{\partial \mathcal{L}}{\partial \mu_2}\delta\mu_2 = -\mathcal{C}_2(\boldsymbol{u}_2, \delta\mu_2). \tag{3.7}$$

Setting $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\varphi}_2}\delta\boldsymbol{\varphi}_2$ and $\frac{\partial \mathcal{L}}{\partial \mu_2}\delta\mu_2$ to zero, we can get the following:

$$\nu\mathcal{A}_2(\boldsymbol{u}_2, \delta\boldsymbol{\varphi}_2) + \mathcal{B}_2(\boldsymbol{u}_2, \boldsymbol{u}_2, \delta\boldsymbol{\varphi}_2) - \mathcal{C}_2(\delta\boldsymbol{\varphi}_2, p_2) - k(\boldsymbol{u}_2, \boldsymbol{\varphi}_2)_{\Gamma_0} = (\boldsymbol{f}, \delta\boldsymbol{\varphi}_2)_{\Omega_2} - (\boldsymbol{g}, \delta\boldsymbol{\varphi}_2)_{\Gamma_0}, \forall\delta\boldsymbol{\varphi}_2 \in \boldsymbol{H}^2_{\Gamma_2}(\Omega_2), \tag{3.8}$$

and

$$-\mathcal{C}_2(\boldsymbol{u}_2, \delta\mu_2) = 0, \forall\delta\mu_2 \in L^2(\Omega_2). \tag{3.9}$$

From (3.8) and (3.9), the constraint equations (2.8) can be also directly derived.

Next, to find the adjoint momentum equations, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i}\delta \boldsymbol{u}_i$ and $\frac{\partial \mathcal{L}}{\partial p_i}\delta p_i$ for $i = 1, 2$ are given by:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_1}\delta \boldsymbol{u}_1 = \alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \delta \boldsymbol{u}_1)_{\Gamma_0} + \nu \mathcal{A}_1(\delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1) + \mathcal{B}_1(\delta \boldsymbol{u}_1, \boldsymbol{u}_1, \boldsymbol{\varphi}_1) + \mathcal{B}_1(\boldsymbol{u}_1, \delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1) - \mathcal{C}_1(\delta \boldsymbol{u}_1, \mu_1) + k(\delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1)_{\Gamma_0}, \tag{3.10}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_2}\delta \boldsymbol{u}_2 = -\alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \delta \boldsymbol{u}_2)_{\Gamma_0} + \nu \mathcal{A}_2(\delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2) + \mathcal{B}_2(\delta \boldsymbol{u}_2, \boldsymbol{u}_2, \boldsymbol{\varphi}_2) + \mathcal{B}_2(\boldsymbol{u}_2, \delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2) - \mathcal{C}_2(\delta \boldsymbol{u}_2, \mu_2) - k(\delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2)_{\Gamma_0}, \tag{3.11}$$

$$\frac{\partial \mathcal{L}}{\partial p_1}\delta p_1 = -\mathcal{C}_1(\boldsymbol{\varphi}_1, \delta p_1), \tag{3.12}$$

and

$$\frac{\partial \mathcal{L}}{\partial p_2}\delta p_2 = -\mathcal{C}_2(\boldsymbol{\varphi}_2, \delta p_2). \tag{3.13}$$

Similarly, setting $\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_i}\delta \boldsymbol{u}_i$ and $\frac{\partial \mathcal{L}}{\partial p_i}\delta p_i$ for $i = 1, 2$, to zero, we have the following adjoint equations:

$$\begin{cases} \nu \mathcal{A}_1(\delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1) + \mathcal{B}_1(\delta \boldsymbol{u}_1, \boldsymbol{u}_1, \boldsymbol{\varphi}_1) + \mathcal{B}_1(\boldsymbol{u}_1, \delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1) - \mathcal{C}_1(\delta \boldsymbol{u}_1, \mu_1) + k(\delta \boldsymbol{u}_1, \boldsymbol{\varphi}_1)_{\Gamma_0} \\ \quad = -\alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \delta \boldsymbol{u}_1)_{\Gamma_0}, \quad \forall \delta \boldsymbol{u}_1 \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1) \\ -\mathcal{C}_1(\boldsymbol{\varphi}_1, \delta p_1) = 0, \quad \forall \delta p_1 \in L^2(\Omega_1), \end{cases} \tag{3.14}$$

and

$$\begin{cases} \nu \mathcal{A}_2(\delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2) + \mathcal{B}_2(\delta \boldsymbol{u}_2, \boldsymbol{u}_2, \boldsymbol{\varphi}_2) + \mathcal{B}_2(\boldsymbol{u}_2, \delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2) - \mathcal{C}_2(\delta \boldsymbol{u}_2, \mu_2) - k(\delta \boldsymbol{u}_2, \boldsymbol{\varphi}_2)_{\Gamma_0} \\ \quad = \alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \delta \boldsymbol{u}_2)_{\Gamma_0}, \quad \forall \delta \boldsymbol{u}_2 \in \boldsymbol{H}^1_{\Gamma_2}(\Omega_2) \\ -\mathcal{C}_2(\boldsymbol{\varphi}_2, \delta p_2) = 0, \quad \forall \delta p_2 \in L^2(\Omega_2). \end{cases} \tag{3.15}$$

Finally, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{g}}\delta \boldsymbol{g}$ is shown as follows:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{g}}\delta \boldsymbol{g} = -(\boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2, \delta \boldsymbol{g})_{\Gamma_0} + \beta(\boldsymbol{g}, \delta \boldsymbol{g})_{\Gamma_0}. \tag{3.16}$$

Analogously, setting $\frac{\partial \mathcal{L}}{\partial \boldsymbol{g}}\delta \boldsymbol{g}$ to be zero yields the optimality condition:

$$(\boldsymbol{g}, \delta \boldsymbol{g})_{\Gamma_0} = \frac{1}{\beta}(\boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2, \delta \boldsymbol{g})_{\Gamma_0}, \quad \forall \delta \boldsymbol{g} \in \boldsymbol{L}^2(\Gamma_0). \tag{3.17}$$

To summarize, solutions of the optimization problem (2.10) may be determined by solving the optimality system (2.7), (2.8), (3.14), (3.15), and (3.17). This optimality system is a weak formulation corresponding to,

$$\begin{cases} -\nu \Delta \boldsymbol{u}_1 + (\boldsymbol{u}_1 \cdot \nabla)\boldsymbol{u}_1 + \nabla p_1 = \boldsymbol{f} & \text{in } \Omega_1 \\ \nabla \cdot \boldsymbol{u}_1 = 0 & \text{in } \Omega_1 \\ \boldsymbol{u}_1 = 0 & \text{on } \Gamma_1 \\ -p_1 \boldsymbol{n}_1 + \nu \nabla \boldsymbol{u}_1 \cdot \boldsymbol{n}_1 + k\boldsymbol{u}_1 = \frac{1}{\beta}(\boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2) & \text{on } \Gamma_0, \end{cases} \tag{3.18}$$

$$\begin{cases} -\nu\Delta \boldsymbol{u}_2 + (\boldsymbol{u}_2 \cdot \nabla)\boldsymbol{u}_2 + \nabla p_2 = \boldsymbol{f} & \text{in } \Omega_2 \\ \nabla \cdot \boldsymbol{u}_2 = 0 & \text{in } \Omega_2 \\ \boldsymbol{u}_2 = 0 & \text{on } \Gamma_2 \\ -p_2\boldsymbol{n}_2 + \nu\nabla \boldsymbol{u}_2 \cdot \boldsymbol{n}_2 - k\boldsymbol{u}_2 = -\frac{1}{\beta}(\boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2) & \text{on } \Gamma_0, \end{cases} \tag{3.19}$$

$$\begin{cases} -\nu\Delta \boldsymbol{\varphi}_1 + (\nabla \boldsymbol{u}_1)^T\boldsymbol{\varphi}_1 - (\boldsymbol{u}_1 \cdot \nabla)\boldsymbol{\varphi}_1 + \nabla\mu_1 = 0 & \text{in } \Omega_1 \\ \nabla \cdot \boldsymbol{\varphi}_1 = 0 & \text{in } \Omega_1 \\ \boldsymbol{\varphi}_1 = 0 & \text{on } \Gamma_1 \\ -\mu_1\boldsymbol{n}_1 + \nu\nabla\boldsymbol{\varphi}_1 \cdot \boldsymbol{n}_1 + k\boldsymbol{\varphi}_1 + (\boldsymbol{u}_1 \cdot \boldsymbol{n}_1)\boldsymbol{\varphi}_1 = -\alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2) & \text{on } \Gamma_0, \end{cases} \tag{3.20}$$

and

$$\begin{cases} -\nu\Delta \boldsymbol{\varphi}_2 + (\nabla \boldsymbol{u}_2)^T\boldsymbol{\varphi}_2 - (\boldsymbol{u}_2 \cdot \nabla)\boldsymbol{\varphi}_2 + \nabla\mu_2 = 0 & \text{in } \Omega_2 \\ \nabla \cdot \boldsymbol{\varphi}_2 = 0 & \text{in } \Omega_2 \\ \boldsymbol{\varphi}_2 = 0 & \text{on } \Gamma_2 \\ -\mu_2\boldsymbol{n}_2 + \nu\nabla\boldsymbol{\varphi}_2 \cdot \boldsymbol{n}_2 - k\boldsymbol{\varphi}_2 + (\boldsymbol{u}_2 \cdot \boldsymbol{n}_2)\boldsymbol{\varphi}_2 = \alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2) & \text{on } \Gamma_0. \end{cases} \tag{3.21}$$

Note that the adjoint system (3.14) and (3.15) or (3.20) and (3.21) is linear in the adjoint variables $\boldsymbol{\varphi}_i$ and μ_i, for $i = 1, 2$. This is important for the efficiency and practicality of the optimization-based domain decomposition method.

B. Sensitivity Derivatives

The optimality system (2.7), (2.8), (3.14), (3.15), and (3.17) may also be derived using "sensitivity" derivatives instead of the Lagrange multiplier rule. Let

$$\mathcal{M}(\boldsymbol{g}) = \mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g}), \tag{3.22}$$

where, for a given $\boldsymbol{g}$,

$$\boldsymbol{u}_i(\boldsymbol{g}) : \boldsymbol{g} \in \boldsymbol{L}^2(\Gamma_0) \to \boldsymbol{H}^1_{\Gamma_i}(\Omega_i), \text{ for } i = 1, 2$$

are defined as the solutions of (2.7) and (2.8), respectively.

Now, the first derivative of $\mathcal{M}(\boldsymbol{g})$, $\frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}}$, is defined through its action on variations $\tilde{\boldsymbol{g}}$ by

$$\left\langle \frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}}, \tilde{\boldsymbol{g}} \right\rangle = \alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \tilde{\boldsymbol{u}}_1 - \tilde{\boldsymbol{u}}_2)_{\Gamma_0} + \beta(\boldsymbol{g}, \tilde{\boldsymbol{g}})_{\Gamma_0} \tag{3.23}$$

where, for $i = 1$ and 2, $\tilde{\boldsymbol{u}}_i \in \boldsymbol{H}^1_{\Gamma_i}(\Omega_i)$ are solutions of the sensitivity system

$$\begin{cases} \nu\mathcal{A}_1(\tilde{\boldsymbol{u}}_1, \boldsymbol{v}) + \mathcal{B}_1(\tilde{\boldsymbol{u}}_1, \boldsymbol{u}_1, \boldsymbol{v}) + \mathcal{B}_1(\boldsymbol{u}_1, \tilde{\boldsymbol{u}}_1, \boldsymbol{v}) - \mathcal{C}_1(\boldsymbol{v}, \tilde{p}_1) + k(\tilde{\boldsymbol{u}}_1, \boldsymbol{v})_{\Gamma_0} \\ \quad = (\tilde{\boldsymbol{g}}, \boldsymbol{v})_{\Gamma_0}, \quad \forall \boldsymbol{v} \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1) \\ -\mathcal{C}_1(\tilde{\boldsymbol{u}}_1, q) = 0, \quad \forall q \in L^2(\Omega_1), \end{cases} \tag{3.24}$$

and

$$\begin{cases} \nu\mathcal{A}_2(\tilde{\boldsymbol{u}}_2, \boldsymbol{v}) + \mathcal{B}_2(\tilde{\boldsymbol{u}}_2, \boldsymbol{u}_2, \boldsymbol{v}) + \mathcal{B}_2(\boldsymbol{u}_2, \tilde{\boldsymbol{u}}_2, \boldsymbol{v}) - \mathcal{C}_2(\boldsymbol{v}, \tilde{p}_2) - k(\tilde{\boldsymbol{u}}_2, \boldsymbol{v})_{\Gamma_0} \\ \quad = -(\tilde{\boldsymbol{g}}, \boldsymbol{v})_{\Gamma_0}, \quad \forall \boldsymbol{v} \in \boldsymbol{H}^1_{\Gamma_2}(\Omega_2) \\ -\mathcal{C}_2(\tilde{\boldsymbol{u}}_2, q) = 0, \quad \forall q \in L^2(\Omega_2). \end{cases} \tag{3.25}$$

Thus, $\tilde{\boldsymbol{u}}_1$ and $\tilde{\boldsymbol{u}}_2$ give the changes in $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$, respectively, that result from the changes $\tilde{\boldsymbol{g}}$ in $\boldsymbol{g}$. In this sense, $\tilde{\boldsymbol{u}}_1$ and $\tilde{\boldsymbol{u}}_2$ can be viewed as sensitivities.

Let $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ be the solutions of (2.7) and (2.8), respectively, and define $\boldsymbol{\varphi}_1, \mu_1, \boldsymbol{\varphi}_2$ and μ_2 to be the solutions of the adjoint problems (3.14) and (3.15), respectively. Set $\boldsymbol{v} = \boldsymbol{\varphi}_1, q = \mu_1$ in (3.24),$\boldsymbol{v} = \boldsymbol{\varphi}_2, q = \mu_2$ in (3.25), $\delta\boldsymbol{u}_1 = \tilde{\boldsymbol{u}}_1, \delta p_1 = \tilde{p}_1$ in (3.14) and $\delta\boldsymbol{u}_2 = \tilde{\boldsymbol{u}}_2, \delta p_2 = \tilde{p}_2$ in (3.15). Combining the results yields that

$$(\tilde{\boldsymbol{g}}, \boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2)_{\Gamma_0} = -\alpha(\boldsymbol{u}_1 - \boldsymbol{u}_2, \tilde{\boldsymbol{u}}_1 - \tilde{\boldsymbol{u}}_2)_{\Gamma_0},$$

so that, from (3.23),

$$\left\langle \frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}}, \tilde{\boldsymbol{g}} \right\rangle = \beta(\boldsymbol{g}, \tilde{\boldsymbol{g}})_{\Gamma_0} - (\tilde{\boldsymbol{g}}, \boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2)_{\Gamma_0}. \tag{3.26}$$

Thus, the first-order necessary condition $\frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}} = 0$, yields that

$$\beta(\boldsymbol{g}, \tilde{\boldsymbol{g}})_{\Gamma_0} = (\tilde{\boldsymbol{g}}, \boldsymbol{\varphi}_1 - \boldsymbol{\varphi}_2)_{\Gamma_0}, \quad \forall \tilde{\boldsymbol{g}} \in \boldsymbol{L}^2(\Gamma_0)$$

which is exact (3.17), with $\delta\boldsymbol{g} = \tilde{\boldsymbol{g}}$, where again, $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ are determined as the solutions of (3.14) and (3.15), respectively.

Note that (3.26) yields an explicit formula for the gradient of $\mathcal{M}$, i.e.,

$$\frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}} = \beta\boldsymbol{g} - (\boldsymbol{\varphi}_1(\boldsymbol{g}) - \boldsymbol{\varphi}_2(\boldsymbol{g}))|_{\Gamma_0}. \tag{3.27}$$

where $\boldsymbol{\varphi}_1(\boldsymbol{g})$ and $\boldsymbol{\varphi}_2(\boldsymbol{g})$ are determined from $\boldsymbol{g}$ through (2.7), (2.8), (3.14), and (3.15). Thus, one has in hand the information needed if one were to use a gradient-type method, e.g., a method that requires $\mathcal{M}(\boldsymbol{g})$ and $\frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g})}{\mathrm{d}\boldsymbol{g}}$ for a given approximation of $\boldsymbol{g}$, to solve the optimization problem (2.10).

C. A Gradient-Type Method

The optimality system (2.7),(2.8), (3.14), (3.15), and (3.17) is a coupled system whose solutions yield solutions of the optimization problem (2.10). To achieve a parallel algorithm, we must at least decouple the subdomain problems; to make the individual subdomain problems tractable, we should uncouple the state and adjoint systems as well. One way of accomplishing this is through a gradient-type method iteration. Recall that our goal is to determine $\boldsymbol{g} \in \boldsymbol{L}^2(\Gamma_0)$ that minimizes $\mathcal{M}(\boldsymbol{g}) = \mathcal{J}(\boldsymbol{u}_1(\boldsymbol{g}), \boldsymbol{u}_2(\boldsymbol{g}), \boldsymbol{g})$, where $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are the solutions of (2.7) and (2.8), respectively.

The gradient-type method we consider is defined as follows. Given a starting guess $\boldsymbol{g}^{(0)}$, let

$$\boldsymbol{g}^{(n+1)} = \boldsymbol{g}^{(n)} - \frac{\gamma}{\beta}\frac{\mathrm{d}\,\mathcal{M}(\boldsymbol{g}^{(n)})}{\mathrm{d}\boldsymbol{g}}, \quad \text{for } n = 1, 2, \ldots, \tag{3.28}$$

where γ/β is a step size. Combining with (3.27) yields

$$\boldsymbol{g}^{(n+1)} = (1-\gamma)\boldsymbol{g}^{(n)} + \frac{\gamma}{\beta}\left(\boldsymbol{\varphi}_1^{(n)} - \boldsymbol{\varphi}_2^{(n)}\right)|_{\Gamma_0}.$$

where $\boldsymbol{\varphi}_1^{(n)}$ and $\boldsymbol{\varphi}_2^{(n)}$ are determined from (3.14) and (3.15), respectively, with $\boldsymbol{g}$ replaced by $\boldsymbol{g}^{(n)}$. In summary, the algorithm is given as follows.

Algorithm 3.1.

(1) *Choose a* $\boldsymbol{g}^{(0)}$.
(2) *For* $n = 1, 2, \ldots,$
A. compute $\boldsymbol{u}_1^{(n)}$, $p_1^{(n)}$, $\boldsymbol{u}_2^{(n)}$ *and* $p_2^{(n)}$ *by*

$$\begin{cases} \nu\mathcal{A}_1\left(\boldsymbol{u}_1^{(n)}, \boldsymbol{v}\right) + \mathcal{B}_1\left(\boldsymbol{u}_1^{(n)}, \boldsymbol{u}_1^{(n)}, \boldsymbol{v}\right) - \mathcal{C}_1\left(\boldsymbol{v}, p_1^{(n)}\right) + k\left(\boldsymbol{u}_1^{(n)}, \boldsymbol{v}\right)_{\Gamma_0} \\ \quad = (\boldsymbol{f}, \boldsymbol{v})_{\Omega_1} + \left(\boldsymbol{g}^{(n)}, \boldsymbol{v}\right)_{\Gamma_0}, \quad \forall \boldsymbol{v} \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1), \\ -\mathcal{C}_1\left(\boldsymbol{u}_1^{(n)}, q\right) = 0, \quad \forall q \in L^2(\Omega_1) \end{cases}$$

and

$$\begin{cases} \nu\mathcal{A}_2\left(\boldsymbol{u}_2^{(n)}, \boldsymbol{v}\right) + \mathcal{B}_2\left(\boldsymbol{u}_2^{(n)}, \boldsymbol{u}_2^{(n)}, \boldsymbol{v}\right) - \mathcal{C}_2\left(\boldsymbol{v}, p_2^{(n)}\right) - k\left(\boldsymbol{u}_2^{(n)}, \boldsymbol{v}\right)_{\Gamma_0} \\ \quad = (\boldsymbol{f}, \boldsymbol{v})_{\Omega_2} - (\boldsymbol{g}^{(n)}, \boldsymbol{v})_{\Gamma_0}, \quad \forall \boldsymbol{v} \in \boldsymbol{H}^1_{\Gamma_2}(\Omega_2), \\ -\mathcal{C}_2\left(\boldsymbol{u}_2^{(n)}, q\right) = 0, \quad \forall q \in L^2(\Omega_2) \end{cases}$$

B. compute $\boldsymbol{\varphi}_1^{(n)}$, $\mu_1^{(n)}$, $\boldsymbol{\varphi}_2^{(n)}$ *and* $\mu_2^{(n)}$ *by*

$$\begin{cases} \nu\mathcal{A}_1\left(\delta\boldsymbol{u}_1, \boldsymbol{\varphi}_1^{(n)}\right) + \mathcal{B}_1\left(\delta\boldsymbol{u}_1, \boldsymbol{u}_1^{(n)}, \boldsymbol{\varphi}_1^{(n)}\right) + \mathcal{B}_1\left(\boldsymbol{u}_1^{(n)}, \delta\boldsymbol{u}_1, \boldsymbol{\varphi}_1^{(n)}\right) - \mathcal{C}_1\left(\delta\boldsymbol{u}_1, \mu_1^{(n)}\right) \\ \quad + k\left(\delta\boldsymbol{u}_1, \boldsymbol{\varphi}_1^{(n)}\right)_{\Gamma_0} = -\alpha\left(\boldsymbol{u}_1^{(n)} - \boldsymbol{u}_2^{(n)}, \delta\boldsymbol{u}_1\right)_{\Gamma_0}, \quad \forall \delta\boldsymbol{u}_1 \in \boldsymbol{H}^1_{\Gamma_1}(\Omega_1) \\ -\mathcal{C}_1\left(\boldsymbol{\varphi}_1^{(n)}, \delta p_1\right) = 0, \quad \forall \delta p_1 \in L^2(\Omega_1), \end{cases}$$

and

$$\begin{cases} \nu\mathcal{A}_2\left(\delta\boldsymbol{u}_2, \boldsymbol{\varphi}_2^{(n)}\right) + \mathcal{B}_2\left(\delta\boldsymbol{u}_2, \boldsymbol{u}_2^{(n)}, \boldsymbol{\varphi}_2^{(n)}\right) + \mathcal{B}_2\left(\boldsymbol{u}_2^{(n)}, \delta\boldsymbol{u}_2, \boldsymbol{\varphi}_2^{(n)}\right) - \mathcal{C}_2\left(\delta\boldsymbol{u}_2, \mu_2^{(n)}\right) \\ \quad - k\left(\delta\boldsymbol{u}_2, \boldsymbol{\varphi}_2^{(n)}\right)_{\Gamma_0} = \alpha\left(\boldsymbol{u}_1^{(n)} - \boldsymbol{u}_2^{(n)}, \delta\boldsymbol{u}_2\right)_{\Gamma_0}, \quad \forall \delta\boldsymbol{u}_2 \in \boldsymbol{H}^1_{\Gamma_2}(\Omega_2) \\ -\mathcal{C}_2\left(\boldsymbol{\varphi}_2^{(n)}, \delta p_2\right) = 0, \quad \forall \delta p_2 \in L^2(\Omega_2). \end{cases}$$

C. compute $\boldsymbol{g}^{(n)}$ *by*

$$\boldsymbol{g}^{(n+1)} = (1-\gamma)\boldsymbol{g}^{(n)} + \frac{\gamma}{\beta}\left(\boldsymbol{\varphi}_1^{(n)} - \boldsymbol{\varphi}_2^{(n)}\right)|_{\Gamma_0}.$$

Of course, we also need to incorporate a stopping criteria into the algorithm and the whole algorithm is implemented at the discrete level.

Remark 3.1. The uncoupling of the subdomain problems and the parallelism of this algorithm are obvious. Within each of steps *(2)A* and *(2)B*, the subdomain problems may be solved in parallel. The only data that must be transferred between processors are the values of $\boldsymbol{u}_i$ and $\boldsymbol{\varphi}_i$, for $i = 1$ and 2, along the common interface.

Remark 3.2. In Algorithm 3.1, α and β are the penalty parameters inherited from the functional $\mathcal{J}(\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{g})$ and γ is an additional parameter that can be chosen for improving the convergent properties of the algorithm. We can choose a suitable step size by controlling γ if β is fixed. Note that, if $\gamma = 1$, the gradient method is equivalent to a simple iteration scheme of the optimality

system (2.7), (2.8), (3.14), (3.15), and (3.17). Of course, we can reconsider the minimization problem (2.10) from a point of view of nonlinear least squares problem, and use the conjugate gradient method to get a faster converging algorithm. But, it increases the complexity of algorithm so that the algorithm needs more code to implement. On the contrary, without losing the simplicity, the algorithm can be improved by adapting α, when γ and β are fixed.

Remark 3.3. The uncoupling of the state and adjoint systems is also obvious since we are now solving them sequentially in steps *(2)A* and *(2)B*. Thus, the systems that must be solved within each subdomain are of the size of a single Navier-Stokes system. However, note that the adjoint systems solved in step *(2)B* are linear in their dependent variables $\boldsymbol{\varphi}_i$ and μ_i, for $i = 1$ and 2. Hence, the costs associated with step *(2)B* are small compared with those for step *(2)A* which involves a nonlinear system for its dependent variables $\boldsymbol{u}_i$ and p_i, for $i = 1, 2$ and, as a result, the cost per iteration of the algorithm is pretty much the same as that for other methods.

IV. NUMERICAL EXPERIMENTS

In this section, numerical results for the incompressible Navier-Stokes flows based on the optimization-based domain decomposition method are reported, where the PDEs (2.7), (2.8), (3.14), and (3.15) are solved by finite element method. The play of the optimization-based domain decomposition method for different models is presented. The finite element grid for the fluid region uses triangular mesh which is generated by a Delaunay-Voronoi mesh generator (see B. Mohammadi and O. Pironneau [21]). The computations have been carried out on a home PC with AMD Athlon(tm) 64 $\times$ 2 Dual Core Processor 2.12 GHz and 1 GB memory.

A. Test 1: Square Cavity Flow Problem

In the first test, we consider a square cavity flow model with unit quare domain $\Omega = (0, 1) \times (0, 1) \in \mathbb{R}^2$ shown in Fig. 2. Ω is divided into two subdomains $\Omega_1 = (0, 1) \times (0.5, 1)$ and $\Omega_2 = (0, 1) \times (0, 0.5)$ with the common interface $\Gamma_0 = (0, 1) \times \{0.5\}$. For the square cavity flow problem, two cases are discussed. Case 1 is the square cavity vortex flow driven by a body force, and case 2 the lid-driven square cavity flow problem. In the numerical experiment, the Robin

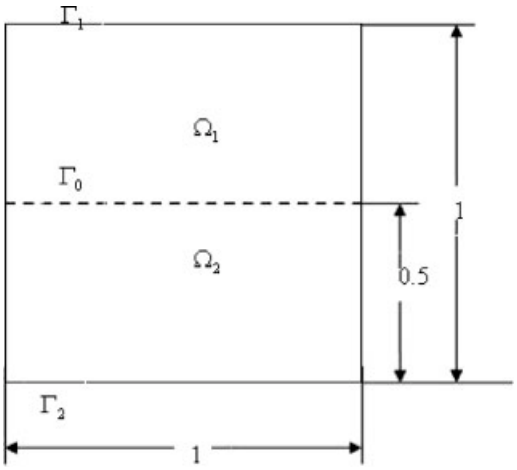

FIG. 2. The square cavity flow model.

coefficient k is dependent on $(\boldsymbol{u}_i, p_i), i = 1$ and 2. Thus, for the convenience of description, the Robin coefficient k in state equations (2.7) and (2.8) is denoted as k_1, whereas that in adjoint Eq. (3.14) and (3.15) is denoted to be k_2. The $\boldsymbol{L}^2(\Omega)$ norm is define by

$$|\boldsymbol{u}|_{0,\Omega} = \left(\int_\Omega \boldsymbol{u}^2 \, \mathrm{d}\Omega \right)^{\frac{1}{2}}. \tag{4.1}$$

Case 1. The square cavity vortex flow driven by body force. In this case, set Reynolds number $Re = \frac{1}{\nu} = 500$, $k_1 = 60, k_2 = 0.001$. We adjust the data $\boldsymbol{f}$ so that the Navier-Stokes system (2.1) has the following exact solution

$$\begin{pmatrix} \boldsymbol{u} \\ p \end{pmatrix} = \begin{pmatrix} 10x^2(x-1)^2y(y-1)(2y-1) \\ -10y^2(y-1)^2x(x-1)(2x-1) \\ 10(2x-1)(2y-1) \end{pmatrix}. \tag{4.2}$$

This model we choose for the numerical test has zero boundary conditions for $\boldsymbol{u}$. For the gradient-type method (algorithm 3.1), the step size γ/β is fixed as 2 for the penalty parameter $\beta = 10^{-8}$, i.e., γ is chosen to be 2×10^{-8}. We use the stopping criterion defined by the $\boldsymbol{L}^2(\Omega)$ relative error

$$\frac{|\boldsymbol{u} - \boldsymbol{u}_e|_{0,\Omega}}{|\boldsymbol{u}_e|_{0,\Omega}} < 6.1 \times 10^{-2}$$

between the domain decomposition solution $\boldsymbol{u}$ and the exact solution $\boldsymbol{u}_e$ (4.2). The domain decomposition solution $\boldsymbol{u}$ is obtained by solving the subdomain problems on a coarse grid shown in Fig. 3. To be compared with the domain decomposition solution $\boldsymbol{u}$, the direct solution $\boldsymbol{u}_d$ is also given via finding a solution to the Navier-Stokes system (2.9) on the same coarse grid as that for the domain decomposition problem.

Figure 3 shows a coarse grid with 1681 nodes and 3200 triangles. The performances of the individual algorithm we develop, in terms of iterations, CPU time and $\boldsymbol{L}^2(\Omega)$ relative error for

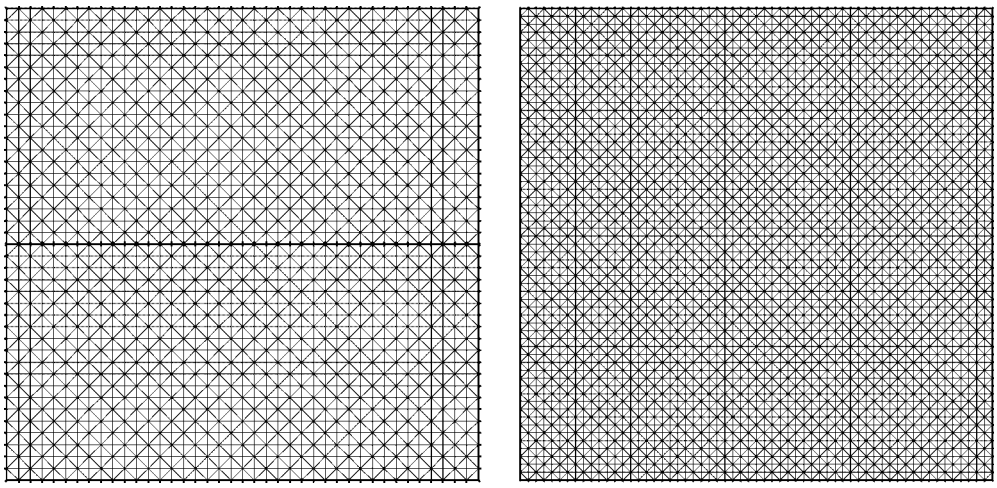

FIG. 3. Finite element mesh, left: the coarse grid with 1681 nodes and 3200 triangles; right: the fine grid with 3721 nodes and 7200 triangles.

TABLE I. The performances of the individual algorithm.

α	Iterations	CPU time (sec)	$\boldsymbol{L}^2(\Omega)$ relative error
1	2121	2×9572.6	0.0604706
50	42	2×212.219	0.0604159
100	22	2×108.929	0.0602501
150	15	2×77.312	0.0603425
200	12	2×61.633	0.0604321
250	Divergence	–	–
Direct solution	–	1×34.235	0.0602215

different α, are presented in Table I. The CPU time and the $\boldsymbol{L}^2(\Omega)$ relative error for the direct solution are also shown in the last row of Table I. It is observed that with the increase of α, less and less iterations are needed till the algorithm diverges at $\alpha = 250$. Only from a point of view of the cost functional, when $\alpha = 1$, our cost functional is the very same one constructed in [16], and the method converges most slowly, with 2121 iterations being needed. Hence, comparing our gradient-type method with the one used in [16], we note that the algorithm can be greatly improved by adjusting the penalty parameter α. On the other hand, the algorithm is parallel so that it can be carried on a distributed computer. If the algorithm is carried out on a distributed computer, Table I shows that the domain decomposition method would cost CPU time 61.633 sec while 34.235 sec is spent to find a direct solution. At first glance it would seem that the domain decomposition method costs more CPU time than the direct computation, but if an ideal speedup rate 2 is given and the domain is only divided into four subdomains, the CPU time and memory can be saved. In practical applications, the computational domain is always large enough so that it can be divided into more subdomains.

The comparison of the velocity fields in Ω between the exact solution and the domain decomposition solution is given in Fig. 4, when $\alpha = 200$. The horizontal component of the velocity fields for the exact solution and the domain decomposition solution is also given in Fig. 5, and the vertical one given in Fig. 6. Figs. 4–6 show that the domain decomposition solution has such a good approximation to the exact solution that the method we consider is effective and acceptable.

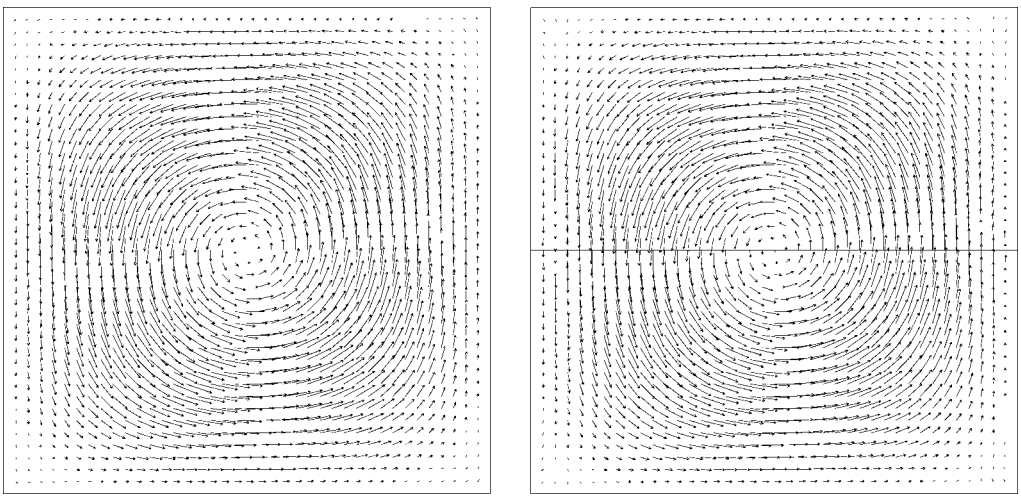

FIG. 4. Velocity field, left: the exact solution; right: the domain decomposition solution.

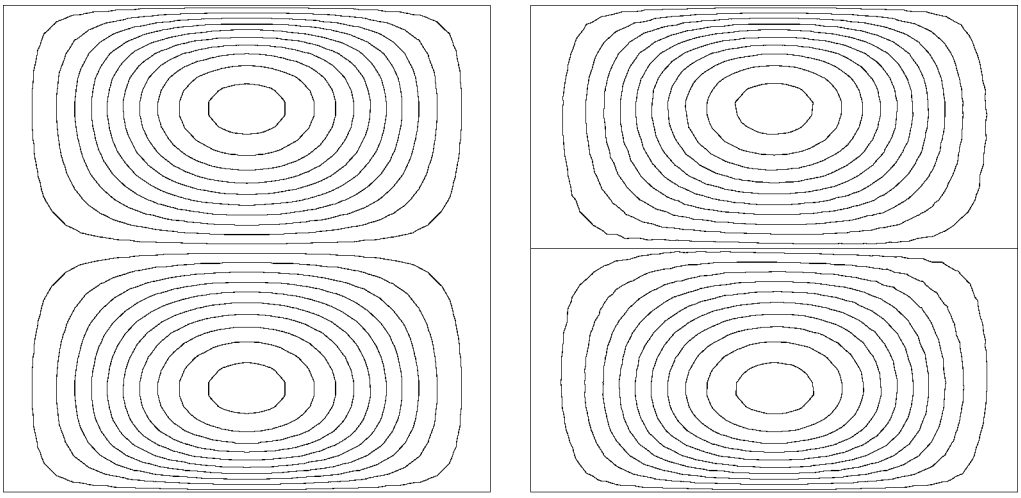

FIG. 5. The horizontal component of the velocity, left: the exact solution; right: the domain decomposition solution.

Case 2. The lid-driven square cavity flow. Analogously, set Reynolds number $Re = \frac{1}{\nu} = 100$, $k_1 = 60, k_2 = 0.001$ and $\boldsymbol{f} = (0, 0)^T$. The boundary conditions are

$$\begin{cases} \boldsymbol{u} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, & \text{on } (0,1) \times \{1\}, \\ \boldsymbol{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, & \text{on other.} \end{cases} \tag{4.3}$$

The step size γ/β is also fixed as 2 for the penalty parameter $\beta = 10^{-8}$, i.e., γ is chosen to be 2×10^{-8}. We also use the similar stopping criterion defined by the $\boldsymbol{L}^2(\Omega)$ relative error

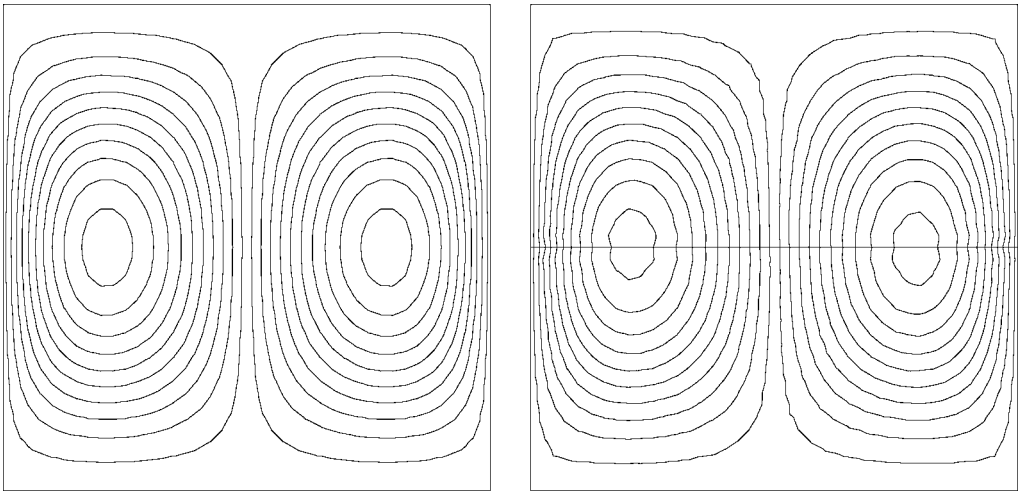

FIG. 6. The vertical component of the velocity, left: the exact solution; right: the domain decomposition solution.

TABLE II. The performances of the individual algorithm.

α	Iterations	CPU time (sec)	$\boldsymbol{L}^2(\Omega)$ relative error
50	151	2×673.915	0.00992334
80	110	2×518.525	0.00946079
100	94	2×417.289	0.00982460
120	98	2×443.273	0.00926018
150	107	2×466.328	0.00971019
200	Divergence	–	–
Direct solution	–	1×63.531	–

$$\frac{|\boldsymbol{u} - \boldsymbol{u}_d|_{0,\Omega}}{|\boldsymbol{u}_d|_{0,\Omega}} < 1.0 \times 10^{-2},$$

where $\boldsymbol{u}$ is a domain decomposition solution and $\boldsymbol{u}_d$ a direct solution witch is given by directly solving the Navier-Stokes system (2.9) with the boundary conditions (4.3) on the fine grid shown in Fig. 3. Similarly, the domain decomposition solution is obtained on the same course grid as that used in the case 1.

The fine grid with 3721 nodes and 7200 triangles is shown in Fig. 3. The performances of the individual algorithm, in terms of iterations, CPU time and $\boldsymbol{L}^2(\Omega)$ relative error for different α, are presented in Table II, and the CPU time for the direct solution is also placed in the last row of the table. Table II shows that with the increase of α, the iterations converge more and more fast until $\alpha = 120$, and after which more and more iterations are required till the method diverges at $\alpha = 200$. We see that the number of iterations can be greatly reduced when α changes from 50 to 100. Comparing the iterations in this case with those in the previous case, we find that more iterations for this case are needed to meet the tolerance criterion. If the computation is carried out on the parallel computer, Table II shows that at least 417.289 sec is required to solve the subdomain problems while only 63.531 sec is spent on finding a direct solution to the complete problem. In the same way, when enough subdomain problems are solved by our parallel algorithm, the CPU time will be less than that by the direct method.

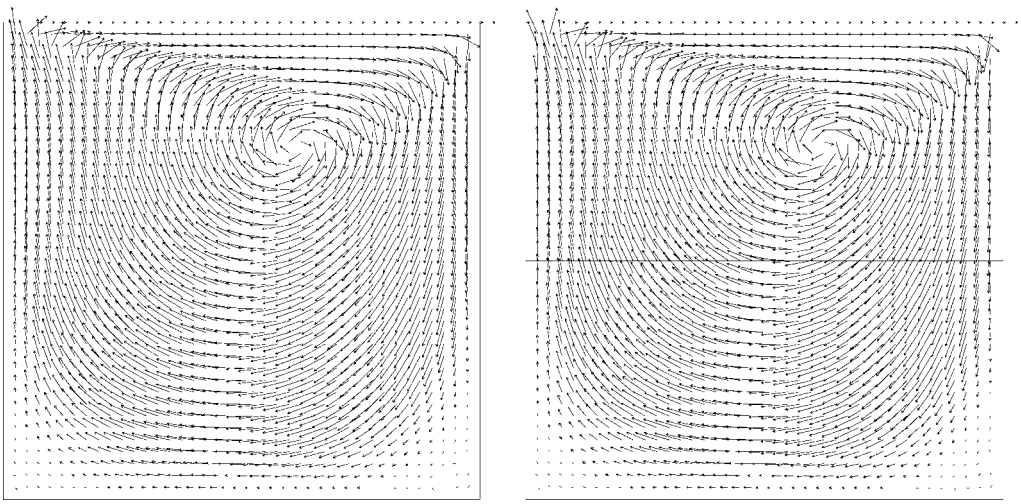

FIG. 7. Velocity field, left: the direct solution; right: the domain decomposition solution.

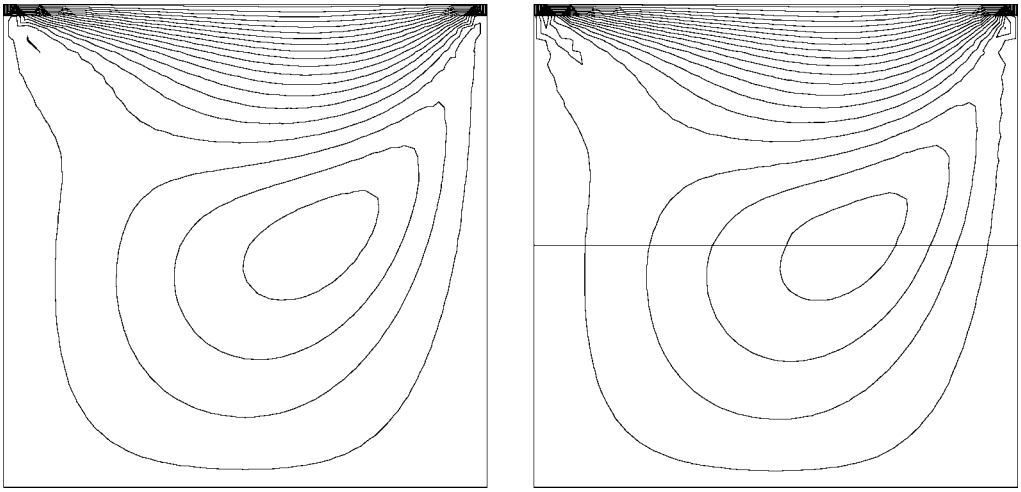

FIG. 8. The horizontal component of the velocity, left: the direct solution; right: the domain decomposition solution.

When $\alpha = 100$, the comparison of the velocity fields between the direct solution $\boldsymbol{u}_d$ and the domain decomposition $\boldsymbol{u}$ is given in Fig. 7. We also give the horizontal component of the velocity fields in Fig. 8, and the vertical one given in Fig. 9. Figures 7–9 show that the domain decomposition solution matches with the direct solution very well.

B. Test 2: Backward-Facing Step Problem

In the second example, an backward-facing step problem is considered. The domain $\Omega = (0, 20) \times (0, 1.5) \backslash (0, 2) \times (0, 0.5)$ shown in Fig. 10 is divided into two subdomains

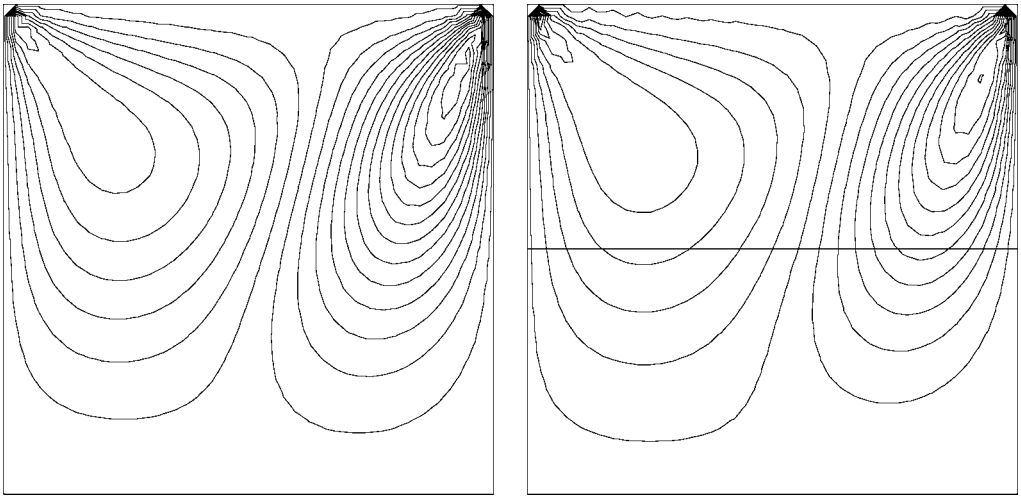

FIG. 9. The vertical component of the velocity, left: the direct solution; right: the domain decomposition solution.

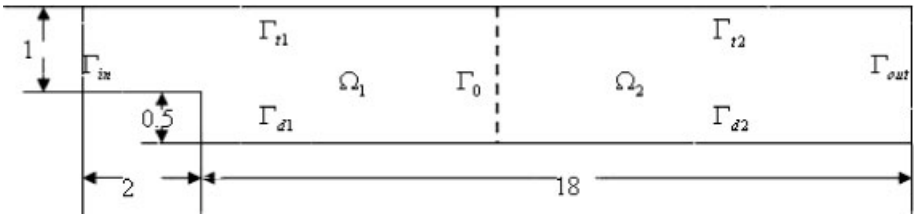

FIG. 10. The backward-facing step model.

$\Omega_1 = (0, 10) \times (0, 1.5) \backslash (0, 2) \times (0, 0.5)$ and $\Omega_2 = (10, 20) \times (0, 1.5)$. In Fig. 10, $\Gamma_0 = 10 \times (0, 1.5)$ denotes the common interface of subdomains Ω_1 and Ω_2, Γ_{ti} top boundaries, Γ_{di} bottom boundaries for Ω_i, $i = 1$ and 2, respectively. The boundaries corresponding to the subdomian Ω_i are $\partial\Omega_i = \Gamma_i \cup \Gamma_0$, for $i = 1$ and 2, where $\Gamma_1 = \Gamma_{\text{in}} \cup \Gamma_{t1} \cup \Gamma_{d1}$, $\Gamma_2 = \Gamma_{\text{out}} \cup \Gamma_{t2} \cup \Gamma_{d2}$.

The boundary conditions are

$$\begin{cases} \boldsymbol{u} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, & \text{on } \bigcup_{i=1}^{2} (\Gamma_{ti} \cup \Gamma_{di}), \\ \boldsymbol{u} = \begin{pmatrix} 4(y - 0.5)(1.5 - y) \\ 0 \end{pmatrix}, & \text{on } \Gamma_{\text{in}} \\ \begin{pmatrix} -pn_1 + \nu \frac{\partial u_1}{\partial x} n_1 = 0 \\ u_2 = 0 \end{pmatrix}, & \text{on } \Gamma_{\text{out}}, \end{cases} \tag{4.4}$$

where n_1 signifies the horizontal component of the outward normal $\boldsymbol{n}$ on the Γ_{out}, u_i, $i = 1, 2$, the horizontal and vertical component of velocity $\boldsymbol{u}$, respectively.

In this example, similarly, set Reynolds number $Re = \frac{UL}{\nu} = \frac{1 \times 0.5}{\nu} = 10$, $k_1 = 100, k_2 = 0.000001$, and $\boldsymbol{f} = (0, 0)^T$, where L is the height of the step and U the velocity at the entrance at the center of the parabolic profile. For the gradient-type method (algorithm 3.1), the step size γ/β is also fixed as 2 for the penalty parameter $\beta = 10^{-8}$, i.e., γ is chosen to be 2×10^{-8}. We still use the similar stopping criterion defined by the $\boldsymbol{L}^2(\Omega)$ relative error

$$\frac{|\boldsymbol{u} - \boldsymbol{u}_d|_{0,\Omega}}{|\boldsymbol{u}_d|_{0,\Omega}} < 3.5 \times 10^{-2}$$

between the domain decomposition solution $\boldsymbol{u}$ and the direct solution $\boldsymbol{u}_d$. We find the domain decomposition solution on the coarse grid shown in Fig. 11, whereas the direct solution to the

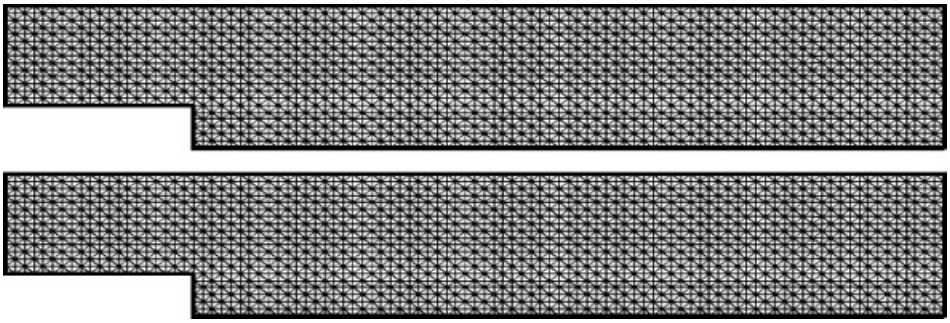

FIG. 11. The coarse grid with 4101 nodes and 7750 triangles, above: subdomain Ω_1, below: subdomain Ω_2.

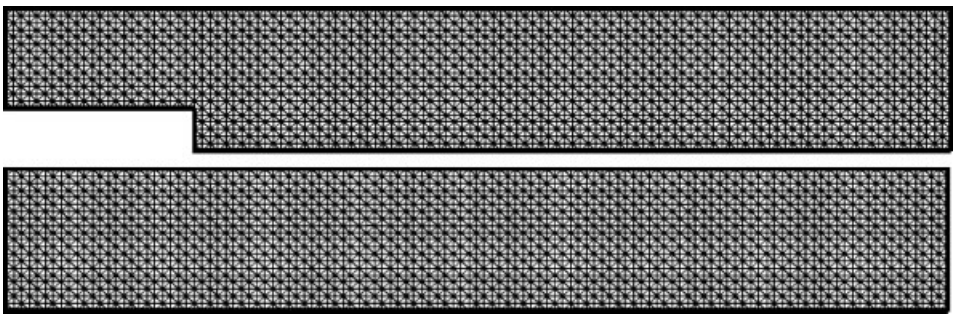

FIG. 12. The fine grid with 4917 nodes and 9312 triangles, above: subdomain Ω_1, below: subdomain Ω_2.

Navier-Stokes system (2.9) with the boundary conditions (4.4) is given on the fine grid shown in Fig. 12.

Figures 11 and 12 show the course grid with 4101 nodes and 7750 triangles, and the fine grid with 4917 nodes and 9312 triangles, respectively. The performances of the individual algorithm, in terms of iterations, CPU time and $\boldsymbol{L}^2(\Omega)$ relative error for different α, are also presented in Table III. The CPU time for the direct solution is also shown on the last row of the Table III. Comparing the CPU time for the direct solution with that for the domain decomposition solution, we can find that if more than four subdomain problems are solved with the parallel processing, the CUP time spent on the subproblems will less than that spent on the complete problem. In Table III, we very easily note that with the increase of α, less and less iterations are needed until $\alpha = 250$, and after this the gradient-type algorithm requires more iterations to meet the stopping criterion till iterates diverge at $\alpha = 300$. Therefore, there exists the best α so that the gradient-type algorithm has the fastest convergence rate, e.g., $\alpha = 250$ in this test.

Remark 4.1. Table I–III, all of them show that if α is too large, the gradient-type method would diverge. To seek out the reason why the algorithm diverges at large α, we should recall the cost functional used in this article. In the cost functional (2.5), the first term represents the gaol of our optimization and the second term is the penalty term necessary to regularize the solution. Obviously, if the second term for the cost functional vanishes, the corresponding optimization problem will be ill-conditioned. Thus, with the increase of α, the first term for the cost functional becomes more and more important, and then, with α going critical, the first term become relatively dominant in the cost functional such that the corresponding minimization problem is so ill-conditioned that the gradient-type algorithm diverges.

TABLE III. The performances of the individual algorithm.

α	Iterations	CPU time (sec)	$\boldsymbol{L}^2(\Omega)$ relative error
50	96	2×1021.48	0.03437008
100	48	2×510.74	0.03436998
150	30	2×329.234	0.03436998
200	20	2×217.031	0.03436988
250	15	2×161.172	0.03437008
300	23	2×235.836	0.03447972
350	Divergence	–	–
Direct solution	–	1×77.688	–

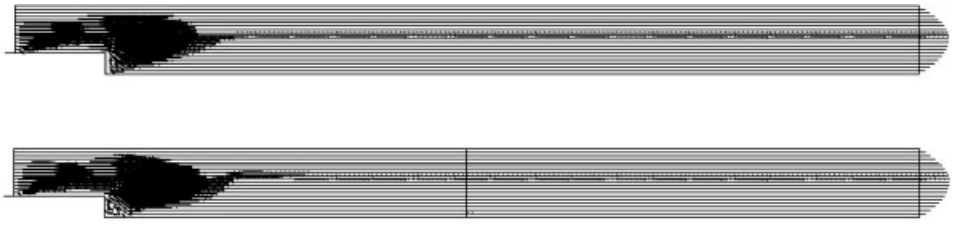

FIG. 13. Velocity field, above: the direct solution; below: the domain decomposition solution.

FIG. 14. The horizontal component of the velocity, above: the direct solution; below: the domain decomposition solution.

The comparison of the velocity fields in Ω between the direct solution $\boldsymbol{u}_d$ and the domain decomposition solution $\boldsymbol{u}$ is given in Fig. 13, when $\alpha = 250$. We also give the horizontal component of the velocity fields in Fig. 14, and the vertical one given in Fig. 15. Figs. 13–15 illustrate that the domain decomposition solution and the direct solution match very well.

V. CONCLUSION

An optimization-based domain decomposition method for numerical simulation of the incompressible Navier-Stokes flows has been studied. The nonoverlapping domain decomposition algorithm was remodeled into a constrained minimization problem for which the objective functional measures the jump in the dependent variables across the common boundaries between subdomains; the constraints are the Navier-Stokes equations in the subdomains with suitably chosen boundary conditions along the common interfaces. The optimality system was derived by the Lagrange multiplier rule and "sensitivity" derivatives, respectively. A gradient method-based approach to the solution of domain decomposition problem was considered. The results of

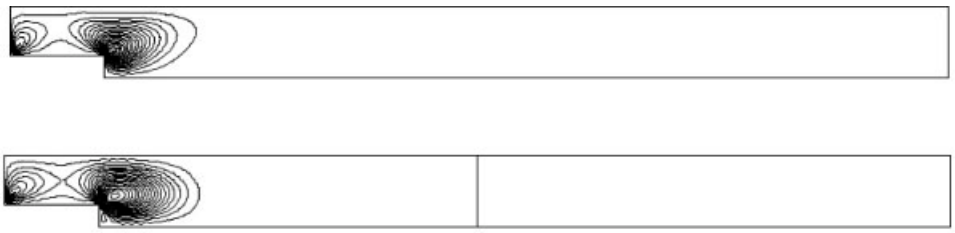

FIG. 15. The vertical component of the velocity, above: the direct solution; below: the domain decomposition solution.

some numerical experiments were also presented to demonstrate the feasibility and applicability of the algorithm we developed. However, to make the method practical and competitive with other methods, further studies including the dependence of Robin coefficient k on the solutions $(\boldsymbol{u}_i, p_i), i = 1, 2$, are needed, mostly in the realm of efficient implementations.

References

1. T. Chan, D. Keyes, G. Meurant, J. Scroggs, and R. Voigt, In: Proc. Fifth international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1992.
2. T. Chan, R. Glowinski, J. Periaux, and O. Widlund, In: Proc. Third international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1990.
3. T. Chan, G. Meurant, J. Periaux, and O. Widlund, Domain decomposition methods, SIAM, Philadelphia, 1989.
4. R. Glowinski, G. Golub, G. Meurant, and J. Periaux, In: Proc. First international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1988.
5. R. Glowinski, Y. Kuznetsov, G. Meurant, J. Periaux, and O. Widlund, In: Proc. Fourth international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1991.
6. C. Douglas, A variation of the Schwarz alternating method: the domain decomposition reduction method, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, editors, Proceedings of the third international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1990, pp. 191–201.
7. M. Dryja and O. Widlund, An additive variant of the Schwarz alternating method for the case of many subregions, Technical report 339, Department of Computer Science, Courant Institute, NYU, 1987.
8. M. Dryja and O. Widlund, Some recent results on Schwarz type domain decomposition algorithms, A. Quarteroni, J. Periaux, Y. Kuznetsov, and O. Widlund, editors, Domain decomposition methods in science and engineering, AMS, Providence, RJ, 1994.
9. P. L. Lions, On the Schwarz alternating methods, I, R. Glowinski, G. Golub, G. Meurant, and J. Periaux, editors, Proceedings of the first international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1988.
10. P. L. Lions, On the Schwarz alternating methods III, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, editors, Proceedings of the third international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1990.
11. O. Widlund, Iterative substructuring methods: algorithms and theory for elliptic problems in the plane, R. Glowinski, G. Golub, and G. Meurant, editors, Proceedings of the first international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1988.
12. M. Dryja and O. Widlund, Some domain decomposition algorithms for elliptic problems, The proceeding of the conference on iterative methods for large linear systems, Academic Press, Orlando, FL, 1989.
13. R. Glowinski and P. Le Tallec, Augmented Lagrangian interpretation of the nonoverlapping Schwarz alternating method, T. Chan, R. Glowinski, J. Periaux, and O. Widlund, editors, Proceedings of the third international symposium on domain decomposition methods for partial differential equations, SIAM, Philadelphia, 1990.
14. Q. Du, Optimization based nonoverlapping domain decomposition algorithms and their convergence, SIAM J Numer Anal 39 (2001), 1056–1077.
15. Q. Du and M. D. Gunzburger, A gradient method approach to optimization-based multidisciplinary simulations and nonoverlapping domain decomposition algorithms, SIAM J Numer Anal 37 (2000), 1513–1541.

16. M. D. Gunzburger and H. K. Lee, An optimization-based domain decomposition method for the Navier-Stokes equations, SIAM J Numer Anal 37 (2000), 1455–1480.

17. M. D. Gunzburger, M. Heinkenschloss, and H. K. Lee, Solution of elliptic partial differential equations by an optimization-based domain decomposition method, Appl Math Comput 113 (2000), 111–139.

18. M. D. Gunzburger, J. Peterson, and H. K. Lee, An optimization based domain decomposition method for partial differential equations, Comp Math Appl 37 (1999), 77–93.

19. D. Bresch and J. Koko, Operator-splitting and Lagrange multiplier domain decomposition methods for numerical simulation of two coupled Navier-Stokes fluids, Int J Appl Math Comput Sci 16 (2006), 419–429.

20. Y. C. Ma, L. Q. Mei, and A. X. Wang, Modern numerical methods for partial differential equations, Scientific Press, Beijing, 2006.

21. B. Mohammadi and O. Pironneau, Applied shape optimization for fluids, Clardendon press, Oxford, 2001.