Computer Methods in Applied Mechanics and Engineering 417 (2023) 116455

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Numerical algorithms and simulations of boundary dynamic control for optimal mixing in unsteady Stokes flows

Xiaoming Zheng [a,*], Weiwei Hu [b], Jiahong Wu [c]

[a] Department of Mathematics, Central Michigan University, Mount Pleasant, MI 48859, USA
[b] Department of Mathematics, University of Georgia, Athens, GA 30602, USA
[c] Department of Mathematics, University of Notre Dame, Notre Dame, IN 46556, USA

ARTICLE INFO

Keywords:
Optimal mixing
Boundary control
Unsteady Stokes flow
Gâteaux derivative
Steepest descent method
Conjugate gradient method

ABSTRACT

This work develops an efficient and accurate optimization algorithm to study the optimal mixing problem driven by boundary control of unsteady Stokes flows, based on the theoretical foundation laid by Hu and Wu in a series of work. The scalar being mixed is purely advected by the flow and the control is a force exerted tangentially on the domain boundary through the Navier slip conditions. The control design has potential applications in many industrial processes such as rotating wall driven mixing, mircomixers with acoustic waves, and artificial cilia mixing.

The numerical algorithms have high complexity, high accuracy demand, and high computing expense, due to the multiscale nature of the mixing problem and the optimization requirements. A crucial problem is the computation of the Gâteaux derivative of the cost functional. To this end, a hybrid approach based on variational formula and finite difference is built with high accuracy and efficiency to treat various types of control input functions. We have experimented with various optimization algorithms including the steepest descent algorithm, the conjugate gradient method and two line search options (backtracking and exact line search). We are able to identify and implement the best combinations.

The numerical simulations show that the mixing efficacy is limited when only one single type of control is applied, but can be enhanced when more diverse control types and more time segmentation are utilized. The mix-norm in the optimal mixings decays exponentially. The numerical study in this work demonstrates that boundary control alone could be an effective strategy for mixing in incompressible flows.

1. Introduction

Transport and mixing in fluids are of fundamental importance in many processes in nature and industry. A long-lasting and central problem is to design an optimal control that enhances transport and mixing or steers a scalar field to a desired distribution, which has drawn great attention to researchers in many fields.

1.1. Motivations and applications

Boundary control, by implementing energy sources through the boundary of the mixer, has been observed or used individually or synergistically with other approaches for transport and mixing in many scenarios. One straightforward boundary control protocol

* Corresponding author.
E-mail addresses: zheng1x@cmich.edu (X. Zheng), Weiwei.Hu@uga.edu (W. Hu), jwu29@nd.edu (J. Wu).

https://doi.org/10.1016/j.cma.2023.116455
Received 8 June 2023; Received in revised form 12 September 2023; Accepted 13 September 2023

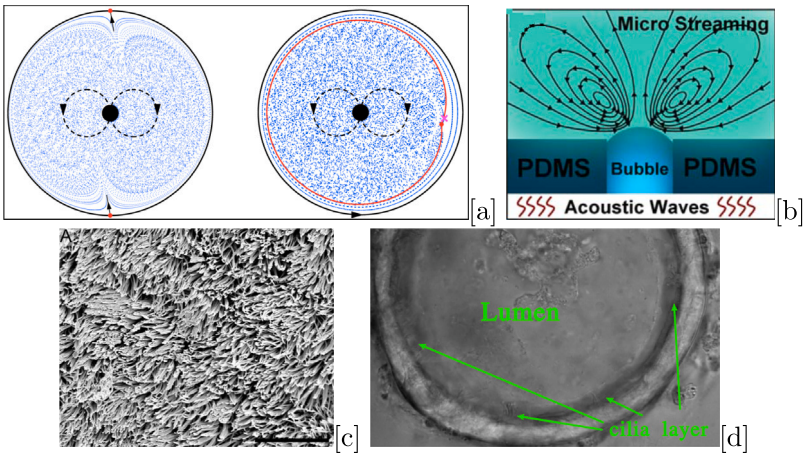

Fig. 1. [a] Left: numerical solution of figure-eight internal stirring with a fixed wall where the arrows point to wall separatrices. Right: rotating wall breaks the separatrices. Taken from [6] with permission. [b] Schematic of the velocity field generated by a micro air bubble activated by acoustic waves, which is embedded in the polydimethylsiloxane (PDMS) sidewall. Taken from [7] with permission. [c] Scanning electron microscopy image of human tracheal epithelial cilia. Taken from [8] with permission. [d] The mixing of mucus driven by cilia beating within the lumen of the airway organoids derived from human lung stem cells. The cilia layer is located on the boundary of the lumen. Taken from the video https://www.youtube.com/watch?v=1Q8RL1g9txk related to the paper [9].

is moving or rotating the container walls to facilitate mixing. In the mixing of two immiscible viscous fluids under low Reynolds numbers in a rectangular cavity [1,2], the top and bottom walls are moved where the moving velocity is employed as the control input to steer mixing, measured by the area or length of the fluid interface. In a series of studies [3–6], it is discovered that the fixed wall with no-slip boundary condition can slow down the internal mixing from exponential decay into power decay due to the separatrices near the wall; however, rotating walls with a constant angular velocity can recover the exponential decay by removing the separatrices (see Fig. 1[a]). These studies use theoretical analysis and/or scientific computing instead of real physical devices.

Instead of moving an entire piece of a sidewall, some boundary control strategies apply controls on individual spots of the fixed sidewall. For example, some micromixers use acoustic waves to perturb mircobubbles embedded in the sidewall of the mixer, whose oscillation can create high pressure and velocity in the bulk liquid in the mixer [7,10] (see Fig. 1[b]). This mixing method is considered simple and effective to overcome the low Reynolds numbers in microfluids due to high viscosity and long microchannel.

Another example of the boundary control is the cilia induced mixing [11]. Cilia are microscopic hair-like structures extensively present in vertebrates and they are located on the epithelial surfaces of internal organs such as the respiratory tract (see Fig. 1[c]). The cilia beating generates metachronal waves, which is an effective way to transport fluid and perform mixing [12,13]. Attracted by the functions of biological cilia, researchers have created artificial cilia, driven by magnetic or electric field, or pneumatics, to generate microfluidic flow, with possible practices in microfluidic devices like lab-on-chip [11]. There exist some numerical studies of cilia mixing such as [14–17], where all of these work consider the direct interaction between fluid and the cilium structure and the mixing is measured by the mixing number according to redistribution of tracer particles advected by the flow. When the cilium length is significantly smaller than the size of the mixer (see Fig. 1[d]), the cilia beating can be approximated as boundary conditions applied on the mixer.

1.2. Objectives and challenges

Despite the motivations and applications mentioned above, boundary control for transport and mixing is still a new field with very few studies. Recently, Hu and Wu in [18–21] have established a theoretical framework of boundary control for optimal mixing via the incompressible flows, where the boundary control is the tangential force exerted on the mixer boundary (2.4). In addition, the scalar or density being mixed is assumed to be driven by advection only and the diffusion is neglected, which corresponds to the case of large Péclet number (the ratio of the rate of advection to the rate of diffusion).

The objective of this work is to develop efficient numerical algorithms for the optimization problem proposed by Hu and Wu and then use them to investigate the efficacy of boundary control for fluid mixing. This work, to the authors' best knowledge, is the first numerical study of optimal mixing via boundary control of the unsteady Stokes flow. Indeed, there are barely any numerical algorithms developed for solving the optimal control for mixing governed by the coupled flow-transport system in a general open bounded domain. Although the optimal mixing and stirring of passive scalars via pure advection has been extensively discussed by means of theoretical analysis and numerical simulations in recent years (cf. [1,2,22–33]), all these studies focus on prescribed velocity fields and none of them consider the real-time control of the unsteady flow dynamics driven by control forces.

The current work on optimal control for fluid mixing problems features high complexity, high accuracy demand, and high computing expense. The first complexity is a cascade of four events from the control to the objective cost function as shown in (1.1), in contrast to 3 steps from flow to cost in the existing work mentioned above.

$$\text{control} \overset{\text{unsteady Stokes}}{\longrightarrow} \text{flow} \overset{\text{advection}}{\longrightarrow} \text{mixed scalar} \longrightarrow \text{cost functional}. \tag{1.1}$$

The entire cascade will be called repeatedly in optimization algorithms, which would entail a high computing expense. However, this can be partially relieved by utilizing a finite basis of the control space and the linear relation between flow velocity and the control (given zero initial velocity field). From the viewpoint of real world applications, a finite number of control inputs is a more realistic assumption since it is not practical to create arbitrarily distributed force fields for stirring. Through this approach, only the velocity fields corresponding to the control basis are needed and stored before the optimization process. Indeed, a control input is a linear combination of the control basis functions (see Eq. (3.2)) and the associated velocity field is a linear combination of the velocity basis with the same coefficients (see Eq. (3.5)).

The mixing problem is intrinsically multiscale, where the optimal mixed scalar has delicate structures of thin filaments everywhere in the domain. This complexity requires high accuracy in the flow and advection solvers. In the flow solver, one complexity is how to enforce the divergence free condition in the numerical methods of the unsteady Stokes equations, which is important in computing the transport equations and the gradient of the cost functional (see Eq. (3.13)). An iterative projection method for solving the Navier–Stokes equations [34] is applied in this work, which obtains the weakly divergence free velocity with the Taylor–Hood finite element method. In the evolution of the mixed scalar, the high order approximation is desirable due to its better ability to capture the microscale structures. However, high order approximations would slow down the evolution and thus the entire optimization process. Thus, a compromise between approximation order and evolution speed has to be made. Furthermore, a better mixing quality is often related to a larger control input and thus a larger flow velocity magnitude (see details in Section 4.2), which would induce small time steps in the advection solvers for stability reasons. If the velocity basis in all the time steps is stored in hard drive, it will result in a large amount of data storage, where a care is needed to balance the data storage quota and accuracy demand.

The development of optimization algorithms also has remarkable complexities. For instance, the accuracy of the gradient of the cost functional is crucial to the convergence of the optimization algorithms. The finite difference method is accurate but has high computing expense when the dimension of the control space is large. The variational formula is much more efficient but may give disastrous results for a certain type of control functions. A hybrid approach will be proposed to combine the advantages of these two methods based on extensive experiments. Another complexity is the choice of the line search method (for finding the step size in a given descent direction) and the descent direction method. The backtracking line search method is fast but may not provide a local minimizer. In the work [31], the exact line search is used with conjugate gradient method to solve an optimal mixing problem. The exact line search is computationally expensive because it needs many iterations of the cascade of (1.1) but it provides a local minimizer. In the optimal control problem of an advection–reaction–diffusion system, a linearization line search method is proposed in [35], which will be examined in this work (see details in Section 3.5.3). Both the steepest descent and conjugate gradient methods for finding descent directions, along with these line search choices, will be tested for convergence, efficiency, and robustness.

The rest of this paper is outlined as follows. Section 2 presents the optimization problem of boundary control design for optimal mixing in unsteady Stokes flows, along with the derivation of the Gâteaux derivative of the cost functional and the first-order necessary optimality conditions for solving the optimal control. Section 3 introduces the optimization algorithms, including the choice of the control input basis, the computation of the velocity basis, the transport equations, the cost functional and its Gâteaux derivative, the line search methods, and the descent direction methods. Section 4 first reports some basic properties of the control functions used in this work, such as flow patterns and mixing characteristics, and then applies the optimization algorithms to investigate the efficacy of boundary control in mixing optimization. The conclusions are presented in Section 5.

2. Boundary control design for optimal mixing

Here, we briefly introduce the mathematical model and the first-order optimality conditions established in [18].

2.1. Optimization problem

Consider a passive scalar field advected by an unsteady Stokes flow in an open bounded and connected domain $\Omega \subset \mathbb{R}^d$, $d = 2$, with a sufficiently smooth boundary Γ. The governing equations for the scalar density θ, velocity v, and pressure p are described by

$$\frac{\partial \theta}{\partial t} + v \cdot \nabla \theta = 0, \tag{2.1}$$

$$\frac{\partial v}{\partial t} - \Delta v + \nabla p = 0, \tag{2.2}$$

$$\nabla \cdot v = 0, \tag{2.3}$$

with the Navier slip boundary conditions (cf. [36]),

$$v \cdot n|_\Gamma = 0 \quad \text{and} \quad (2n \cdot \mathbb{D}(v) \cdot \tau + kv \cdot \tau)|_\Gamma = g, \tag{2.4}$$

and the initial condition

$$(\theta(0), v(0)) = (\theta_0, v_0). \tag{2.5}$$

Here, $\mathbb{D}(v) = (1/2)(\nabla v + (\nabla v)^T)$ is the strain rate tensor, and n and τ are the outward unit normal and tangential vectors to the domain boundary Γ. The Navier slip boundary conditions allow the fluid to slip along the boundary with resistance under the tangential

force and the friction between the fluid and the wall is proportional to $-v$ with the positive coefficient of proportionality k. In this model, the boundary control input g is specialized in the tangential direction, that is, $g\tau$ is the force exerted only in the tangential direction. Physically, this boundary condition can be regarded as a model in the tangential direction of the cilia beating in the inner membrane of vertebrate organs, as described at the end of Section 1.1.

The notation $L^2(G)$ is used to denote the Lebesgue space of square integrable functions over a set G, and $H^s(G), s \geq 0$, the subset of $L^2(G)$ of functions whose weak derivatives up to order s are also square integrable. Note $H^0(G) = L^2(G)$. Let

$$V_n^s(\Omega) = \{v \in H^s(\Omega) : \text{div } v = 0,\ v \cdot n|_\Gamma = 0\}, \quad \text{for } s \geq 0.$$

Throughout this paper, we use $(\cdot,\cdot)$ and $\langle\cdot,\cdot\rangle_\Gamma$ for the L^2-inner products in the interior of the domain Ω and on the boundary Γ, respectively.

The objective in this work is to seek a control input $g \in U_{ad}$ that minimizes the following cost functional at a given final time $T > 0$:

$$J(g) = \frac{1}{2}\|\theta(T)\|^2_{(H^1(\Omega))'} + \frac{\gamma}{2}\|g\|^2_{U_{\text{ad}}},$$

subject to the PDE constraints (2.1)–(2.5), where $\gamma > 0$ is the control weight parameter and $U_{ad} = L^2(0,T;L^2(\Gamma))$ is the set of admissible controls equipped with the norm $\|\cdot\|_{U_{ad}}$ given by

$$\|g\|_{U_{ad}} = \left(\int_0^T \int_\Gamma |g(x,t)|^2 dx dt\right)^{1/2}, \quad \forall g \in U_{ad}. \tag{2.6}$$

The choice of U_{ad} is often determined based on the physical properties as well as the need to guarantee the existence of an optimal solution. The detailed explanation can be found in [18]. In this work, we adopt the dual norm $\|\cdot\|_{(H^1(\Omega))'}$ that quantifies the weak convergence as the mix-norm to quantify mixing [24,26,37], where $(H^1(\Omega))'$ is the dual space of $H^1(\Omega)$. To make it explicit, we define f as the solution of

$$(-\Delta + I)f = \theta \quad \text{in } \Omega, \qquad \frac{\partial f}{\partial n} = 0 \quad \text{on } \Gamma. \tag{2.7}$$

Let

$$\Lambda = (-\Delta + I)^{1/2}.$$

Then Λ is a self-adjoint and positive operator. Thus $f = \Lambda^{-2}\theta$ and

$$\|\theta\|_{(H^1(\Omega))'} = (\Lambda^{-1}\theta, \Lambda^{-1}\theta)^{1/2} = (\Lambda^{-2}\theta, \theta)^{1/2} = (f,\theta)^{1/2}. \tag{2.8}$$

We impose θ_0 to be a spatially mean-zero function, that is, $\bar{\theta}_0 \triangleq \frac{1}{|\Omega|}\int_\Omega \theta_0(x)dx = 0$. Then when perfect mixing is achieved, the mix-norm is zero. This is the same treatment as in [24]. It is straightforward to show that the spatial mean value of θ is time-invariant, i.e., $\bar{\theta}(t) = \bar{\theta}_0, \forall t > 0$.

With the help of (2.7)–(2.8), J can be rewritten as

$$J(g) = \frac{1}{2}(\Lambda^{-2}\theta(T), \theta(T)) + \frac{\gamma}{2}\int_0^T \langle g, g\rangle_\Gamma \, dt. \tag{2.9}$$

Note that the boundary control of the velocity field gives rise to a nonlinear control problem of the scalar equation, due to the one-way coupling through the advective term $v \cdot \nabla\theta$. As a result, the problem (2.9) is no longer convex. The existence of an optimal solution $g \in U_{ad}$ is proven in [18]. Moreover, when $d = 2$ and γ is sufficiently large, the optimal solution is unique.

In this work, we set $v_0 = 0$ for simplicity. Since the state variables v and θ depend on g, we use the notations

$$v = v(g) \quad \text{and} \quad \theta = \theta(g). \tag{2.10}$$

Furthermore, we define the control-to-state operator

$$L : g \in U_{ad} \mapsto v(g) \in L^2(0,T;V_n^0(\Omega)), \tag{2.11}$$

where $v(g)$ is solution of (2.2)–(2.5) with nonhomogeneous boundary input g. With the zero initial velocity condition, it is easy to see that L is a linear operator, that is,

$$L(\alpha_1 g_1 + \alpha_2 g_2) = \alpha_1 L(g_1) + \alpha_2 L(g_2), \quad \forall \alpha_1, \alpha_2 \in \mathbb{R}, \forall g_1, g_2 \in U_{ad}. \tag{2.12}$$

The detailed properties of L are introduced in [18,20].

In this work, the domain is a two dimensional unit disk, i.e., $\Omega = \{(x,y) : x^2 + y^2 < 1\}$, the terminal time is $T = 1$, the friction coefficient is $k = 0.5$, and the control weight is $\gamma = 1\text{e}{-6}$. We adopt a scientific notation with 'e' in many programming languages to denote a very large or small floating point number, such as 6.23e−5 for 6.23×10^{-5}. The initial value of θ is $\theta_0 = \sin(2\pi y)$ (Fig. 12 at $t = 0$), the same as in [31]. The choice of control functions is discussed in Section 3.2.1.

2.2. First-order necessary optimality conditions

To solve the optimal control problem (2.9), we apply a variational inequality [38], that is, if g is an optimal solution, then

$$DJ(g;\varphi) \geq 0, \quad \forall \varphi \in U_{ad}, \tag{2.13}$$

where $DJ(g;\varphi)$ stands for the Gâteaux derivative of J with respect to g in the direction $\varphi \in U_{ad}$. A rigorous definition is given by

$$DJ(g;\varphi) = \lim_{\delta \to 0} \frac{J(g+\delta\varphi) - J(g)}{\delta} = \frac{dJ(g+\delta\varphi)}{d\delta}|_{\delta=0}, \quad \forall \varphi \in U_{ad}.$$

If the limit exists for all $\varphi \in U_{ad}$, then J is called Gâteaux differentiable at g. The Riesz representation of the Gâteaux derivative in U_{ad}, denoted as $DJ(g)$, which is the gradient of J at g [39], satisfies

$$(DJ(g),\varphi)_{U_{ad}} \triangleq DJ(g;\varphi) = \int_0^T \gamma \langle g,\varphi\rangle_\Gamma + (\theta(g)\nabla\rho(g), L\varphi)\, dt, \quad \forall \varphi \in U_{ad}, \tag{2.14}$$

where $\rho(g)$ is the adjoint state satisfying

$$\frac{\partial}{\partial t}\rho + v(g)\cdot\nabla\rho = 0, \tag{2.15}$$

$$\rho(T) = \Lambda^{-2}\theta(g)(T). \tag{2.16}$$

The derivation of (2.14)–(2.16) is briefly stated in Appendix A.1. Since there are no local constraints on U_{ad} [38], the first-order necessary optimality condition for g to be a local minimizer is given by

$$DJ(g) = 0 \text{ in } U_{ad}. \tag{2.17}$$

In addition, the following relation between θ and ρ holds, which is proven in Appendix A.2 and is used to verify the numerical code as shown in Appendix A.5.

Proposition 2.1. *For a fixed final time $T > 0$ and $g \in U_{ad}$, let ρ be the solution to the adjoint system (2.15)–(2.16). Then the quantity $\int_\Omega \rho(x,t)\theta(x,t)dx$ is invariant with respect to $t \in [0,T]$.*

3. Optimization algorithms

3.1. General optimization algorithm

The gradient descent based optimization strategies such as the steepest descent method and the conjugate gradient method will be used in solving the optimization problem. The fundamental idea used in this work is generating a sequence g^n, $n = 0, 1, \ldots$ with a recursive relation

$$g^{n+1} = g^n + \eta^n d^n, \tag{3.1}$$

where d^n is a descent search direction of J at g^n (i.e., $DJ(g^n;d^n) < 0$) and $\eta^n > 0$ is a step length. The entire optimization process is outlined in Algorithm 3.1.

Algorithm 3.1 General Optimization Algorithm for the Mixing Problem

- Input: mesh of size h, initial guess g^0, control basis (see Section 3.2.1).
- Output: solution g.

1. Compute and store velocity basis for the control basis (see Section 3.2.2).
2. Optimization. For $n = 0, 1, \ldots,$
   (1) If g^n is a local minimizer, stop and output it as the solution.
   (2) Compute a descent search direction d^n of J at g^n (see Section 3.6).
   (3) Compute a step length η^n in the direction d^n (see Section 3.5), then $g^{n+1} = g^n + \eta^n d^n$.

A relay approach through a sequence of refined meshes is used to improve computational efficiency. That is, the optimization problem is first solved on a coarse mesh, whose solution is passed as the initial guess for the optimization process on a finer mesh. The scheme is described in Algorithm 3.2. In this work, we use three meshes with resolution $h = 0.1, 0.05, 0.025$ in a unit disk domain.

Algorithm 3.2 Relay Algorithm for the Mixing Problem

1. Fix a control basis and create a sequence of meshes of mesh size $h_1 > h_2 > \cdots$.
2. Apply Algorithm 3.1 on mesh h_1 with initial guess g^0 and denote the solution as g_{h_1}.
3. Apply Algorithm 3.1 on mesh h_2 with initial guess g_{h_1} and denote the solution as g_{h_2}.
4. Relay from mesh h_2 to mesh h_3, ...

3.2. Control basis, velocity basis, and advection evolutions

3.2.1. Finite dimensional control basis

We focus on a finite dimensional control space $U_{ad}^M \triangleq \text{span}\{g_j^b\}_{j=1}^M$, where $\{g_1^b, \dots, g_M^b\} \subseteq U_{ad}$ is a basis of U_{ad}^M. Therefore, any control $g \in U_{ad}^M$ can be written as

$$g = \sum_{j=1}^{M} \alpha_j g_j^b, \tag{3.2}$$

where $\alpha_j \in \mathbb{R}$, $j = 1, \dots, M$ are the linear combination parameters. When the control basis is chosen, the true unknowns that we want to solve for are these parameter values. In the relay algorithm, the control basis is fixed and it is these parameter values that are relayed from a coarse grid to a fine grid.

In this work, the control basis functions g_j^b are built by time segmenting the elementary functions 1, $\cos(k\omega)$, $\sin(k\omega)$, where $k = 1, 2$ and ω is the polar angle of the point (x, y) on the unit circle. The time segmentation is defined as follows. Let N be the number of time segments, and $\Delta s = \frac{1}{N}$ is the uniform segment size. Define the time segmentation function $\chi_i^N(t)$ as

$$\chi_i^N(t) = \left\{ \begin{array}{ll} 1, & \text{if} \quad t \in ((i-1)\Delta s, i\Delta s), \\ 0, & \text{otherwise}, \end{array} \right\}, \quad i = 1, \dots, N. \tag{3.3}$$

A control basis function g^b is one of above elementary functions multiplying a time segmentation function, that is,

$$g^b(x, t) = \chi_i^N(t) \cdot \text{ one of } \{1, \cos(\omega), \sin(\omega), \cos(2\omega), \sin(2\omega)\}. \tag{3.4}$$

The control basis functions generated by the same elementary function are called of the same Type. For example, Type 1 is the set of functions generated by multiplying 1 with time segmentation functions, Type $\cos(\omega)$ is generated by multiplying $\cos(\omega)$ with time segmentation functions, etc.

3.2.2. Velocity basis: Generation and storage

Due to the linearity of the operator L in (2.12), the velocity field generated by g in (3.2) can be written as

$$v(g) = L(g) = \sum_{j=1}^{M} \alpha_j L(g_j^b). \tag{3.5}$$

This linear relation produces a big advantage in computations: we only need to compute the velocity basis

$$v_j^b = L(g_j^b), \quad j = 1, \dots, M, \tag{3.6}$$

before the optimization process and store it in the computer hard drive. Whenever there is a need to compute $L(g)$, the formula (3.5) will be used to compose the velocity for g from the stored velocity basis. An iterative projection method with Taylor–Hood finite elements is employed to solve the unsteady Stokes Eqs. (2.2)–(2.5), where the details are given in Appendix A.3.

The linearity of the operator L holds only when the initial velocity $v_0 = 0$. If $v_0 \neq 0$, we denote the velocity generated by v_0 and $g = 0$ as $v_{v_0,g=0}^b$. Then the full solution v can be written as $v = \sum_{j=1}^M \alpha_j v_j^b + v_{v_0,g=0}^b$. However, in our numerical experiments, we restrict our discussion to the cases with $v_0 = 0$.

Limited by storage, every basis velocity is saved with a not-too-small time step Δt_V, which is typically several folds of the time step used in the Stokes solver. Denote $N_V = \frac{T}{\Delta t_V}$. Thus, there are $N_V + 1$ moments of velocity storage in the time window $[0, 1]$. In other words, for each basis velocity v_j^b, $j = 1, \dots, M$, its values at time $t_V^i = i\Delta t_V$, $i = 0, 1, \dots, N_V$, are saved into files. In practice, we use $T = 1$ and $\Delta t_V = 0.01$, so $N_V + 1 = 101$.

If the Navier–Stokes equations with the nonlinear convection are considered, then the relation between v and g is no longer linear even when v_0 is zero, where a solver for the Navier–Stokes equations has to be called to obtain $v(g)$ whenever g changes. Therefore, the linearity of unsteady Stokes equation saves a lot of the computation time.

3.2.3. Evolution of advection equations with sparsely stored velocity data

A discontinuous Galerkin (DG) method is employed to solve the advection equations for the density θ and its adjoint state ρ, where a brief introduction is given in Appendix A.4. Due to the CFL condition (A.20), the time step of the DG method, Δt_{DG}, is generally far smaller than the velocity storage time step Δt_V, where Δt_V is often 20 to 40 folds larger than Δt_{DG}. Therefore, the stored velocity data is sparse relative to the requirement of the DG evolution method. We use the embedding and interpolation scheme in Algorithm 3.3 to evolve θ, where the one for ρ is the similar.

Algorithm 3.3 Evolution of transport equation for θ (or ρ) with sparsely stored velocity data

- Input: control $g = \sum_{j=1}^{M} \alpha_j g_j^b$, initial value θ_0, basis velocity data $v_j^b(t_V^i)$, $j = 1, \dots, M$ at time t_V^i, $i = 0, 1, \dots, N_V$. Note: $N_V = \frac{T}{\Delta t_V}$.
- Output: θ at time t_V^i, $i = 0, 1, \dots, N_V$.
- Evolution: at time t_V^i, $i = 0, 1, \dots, N_V - 1$,

  (1) Compose velocity $v(g)$ at t_V^i and t_V^{i+1}: $v(g)(t_V^i) = \sum_{j=1}^{M} \alpha_j v_j^b(t_V^i)$, $v(g)(t_V^{i+1}) = \sum_{j=1}^{M} \alpha_j v_j^b(t_V^{i+1})$.

  (2) Compute $V_{\max} = \max(||v(t_V^i)||_{max}, ||v(t_V^{i+1})||_{max})$.

  (3) Use the CFL condition (A.20) to compute a tentative DG time step $\widetilde{\Delta t}_{DG}^{\,i} = \frac{h \cdot \mathrm{CFL}_{L^2}}{V_{\max}}$. To get an integer number of steps of evolution in the time interval $[t_V^i, t_V^{i+1}]$, we let $N_i = \left\lceil \frac{\Delta t_V}{\widetilde{\Delta t}_{DG}^{\,i}} \right\rceil$, the ceiling function of the time steps ratio. Afterwards, define $\Delta t_{DG}^i = \frac{\Delta t_V}{N_i}$.

  (4) Interpolate the velocity at any time $t \in [t_V^i, t_V^{i+1}]$, $v_I(t)$, required by the DG method by $v_I(t) = \frac{t_V^{i+1} - t}{\Delta t_V} \cdot v(g)(t_V^i) + \frac{t - t_V^i}{\Delta t_V} \cdot v(g)(t_V^{i+1})$.

  (5) Use the DG method to evolve θ from t_V^i to t_V^{i+1} with time step size Δt_{DG}^i and the interpolated velocity $v_I(t)$.

The backward evolution of $\rho(t)$ from $t = T$ to $t = 0$ through the advection Eq. (2.15) can be reformulated to a forward evolution process by the following transformation. Let $s = T - t$ and $\tilde{\rho}(s) = \rho(t)$ and $\tilde{v}(s) = -v(t)$. Then $\tilde{\rho}$ satisfies

$$\frac{\partial \tilde{\rho}(s)}{\partial s} + \tilde{v}(s) \cdot \nabla \tilde{\rho}(s) = 0, \quad \tilde{\rho}(0) = \rho(T). \tag{3.7}$$

To evaluate the second integral in (3.13), both θ and ρ are stored at the same time moments as the velocity basis, that is, time t_V^i, $i = 0, 1, \dots, N_V$ as mentioned in Section 3.2.2. It turns out the majority time of the entire optimization process is spent on the simulation of θ and ρ, because whenever there is a need to compute the cost functional or its gradient, the evolution of θ and/or ρ will be computed. To balance the efficiency and accuracy, we choose to use a second order Runge–Kutta scheme in time for the transport equations and a second degree polynomial approximation for θ and ρ in space.

3.3. Computation of the cost functional $J(g)$

The cascade (1.1) or the computation from a control input to the cost functional is computed through Algorithm 3.4.

Algorithm 3.4 Computation of cost $J(g)$

- Input: control g, initial value θ_0, basis velocity data v_j^b, $j = 1, \dots, M$.
- Output: cost $J(g)$.
- Steps:

  (1) Evolve θ with g, θ_0, and the basis velocity data by Algorithm 3.3 to obtain $\theta(T)$.

  (2) Compute the adjoint state $\rho(T)$ from the Neumann elliptic problem (2.7). We use a continuous piecewise quadratic finite element method to solve this problem.

  (3) Compute the cost $J(g)$ by computing the integrals in the first formula of (2.9).

3.4. Computation of the gradient $DJ(g)$

Recall from Section 2.2 that $DJ(g)$ is the Riesz representation in the space U_{ad} of the Gâteaux derivative of J at g. When U_{ad} is chosen as U_{ad}^M, the representation of $DJ(g)$ becomes a linear combination of g_j^b, $j = 1, \dots, M$. That is,

$$DJ(g) = \sum_{j=1}^{M} DJ(g)_j \cdot g_j^b, \quad DJ(g)_j \in \mathbb{R}, \tag{3.8}$$

where $DJ(g)_j$, $j = 1, \dots, M$, are the linear combination coefficients.

Letting $\varphi = g_i^b$, $i = 1, \dots, M$, in (2.14) (using the first equality), we get the following linear system

$$\sum_{j=1}^{M} (g_i^b, g_j^b)_{U_{ad}} \cdot DJ(g)_j = DJ(g; g_i^b), \quad i = 1, \dots, M. \tag{3.9}$$

Let G be the matrix $G_{ij} \triangleq (g_i^b, g_j^b)_{U_{ad}}$, $i, j = 1, \dots, M$, and the vector $b = (DJ(g)_1, \dots, DJ(g)_M)^T$. Thereafter, the norm $\|DJ(g)\|_{U_{ad}}$ is given by

$$\|DJ(g)\|_{U_{ad}} = \sqrt{\left(\sum_{i=1}^{M} DJ(g)_i g_i^b, \sum_{j=1}^{M} DJ(g)_j g_j^b \right)} = \sqrt{b^T G b}. \tag{3.10}$$

3.4.1. Finite difference (FD) method

A simple method of computing the directional derivative is a Finite Difference (FD) approximation [40]:

$$DJ(g; \varphi) \approx \frac{J(g + \delta \cdot \varphi) - J(g)}{\delta}, \tag{3.11}$$

where δ is a small scalar. In our numerical implementations, a typical value of δ is 1e−5 or 1e−4. With this approach, the right side of linear system (3.9) is replaced by

$$DJ(g; g_i^b) = \frac{J(g + \delta \cdot g_i^b) - J(g)}{\delta}, \quad i = 1, \dots, M. \tag{3.12}$$

3.4.2. Variational formula (VF) with adjoint system

This method uses the Variational Formula (VF) (2.14) (the second equality), where the right side of (3.9) becomes

$$DJ(g; g_i^b) = \gamma \int_0^T \langle g, g_i^b \rangle_\Gamma \, dt + \int_0^T \left(\theta(g) \nabla \rho(g), L(g_i^b) \right) dt, \quad i = 1, \dots, M. \tag{3.13}$$

The second integral in (3.13) is evaluated with the trapezoidal rule in each interval $[t_V^i, t_V^{i+1}]$ for $i = 0, 1, \dots, N_V - 1$ by using the data of θ, ρ, and $L(g_i^b)$. The entire VF scheme is stated in Algorithm 3.5.

Algorithm 3.5 VF (Variational Formula) method of computing $DJ(g)$

- Input: control g, initial value θ_0, basis velocity data v_j^b, $j = 1, \dots, M$.
- Output: $DJ(g)$.
- Steps:
  1. Evolve θ from $t = 0$ to $t = T$ with Algorithm 3.3.
  2. Compute the adjoint state $\rho(T)$ from (2.16), that is, the Neumann elliptic problem (2.7) with $\theta = \theta(T)$.
  3. Evolve ρ with Algorithm 3.3 by solving the system (3.7).
  4. Compute $DJ(g)$ with equations (3.9) and (3.13).

3.4.3. Comparison of VF and FD methods in 1-D control spaces

The finite difference method requires to compute a forward evolution process for each basis function g_i^b, $i = 1, \dots, M$, in order to compute $J(g + \delta g_i^b)$. Plus another forward evolution of θ in $J(g)$, the FD scheme requires $M + 1$ forward evolutions to compute $DJ(g)$. In contrast, using the variational formula takes only two evolutions: one forward for θ and one backward for ρ. In this sense, the VF method is more appealing when M is large. However, the VF method has much higher complexity: one elliptic solver for $\rho(T)$ and the integration of $\int_0^T \left(\theta(g) \nabla \rho(g), L(g_i^b) \right) dt$. Especially, the calculation of $\theta \nabla \rho$ involves the spatial derivative of ρ, which has one less order accuracy than ρ itself. In certain cases, it may result in too large errors.

To compare the performance of the VF and FD methods, we give one experiment on the five elementary control functions used in this work: $g^b = 1$, $\cos(\omega)$, $\sin(\omega)$, $\cos(2\omega)$, $\sin(2\omega)$. Because the mix-norm in the cost functional, $J_\theta(g) \triangleq \frac{1}{2} \|\theta(g)\|^2_{(H^1(\Omega))'}$, is the only challenging part and the major source of error in the entire gradient calculation, this experiment just focuses on this term. The derivatives of this term computed by these two methods are shown in Fig. 2, where the computations are taken for integer values of $\alpha \in [0, 100]$ in $g = \alpha g^b$. Overall, both methods agree far better for the cosine and sine functions than the function 1. We denote $D_{VF} J_\theta(g)$ and $D_{FD} J_\theta(g)$ as the gradient of $J_\theta(g)$ with VF and FD methods, respectively. Let the average absolute error be $\mathrm{AAE}(g^b) = \frac{1}{101} \sum_{\alpha=0}^{100} |D_{VF} J_\theta(\alpha g^b) - D_{FD} J_\theta(\alpha g^b)|$ and the average relative error be $\mathrm{ARE}(g^b) = \frac{1}{101} \sum_{\alpha=0}^{100} |D_{VF} J_\theta(\alpha g^b) - D_{FD} J_\theta(\alpha g^b)| / |D_{FD} J_\theta(\alpha g^b)|$. These two errors for these control basis functions are shown in Table 1. We observe the first-order convergence of the average absolute errors when the mesh is refined, with the error of the control 1 is at least twice of the errors of other control basis functions. The average relative error is not a well-defined metric since it is not symmetric, so we cannot expect any convergence. However, it shows that the average relative error of the control 1 is far larger than those of other controls (at least 20 folds larger).

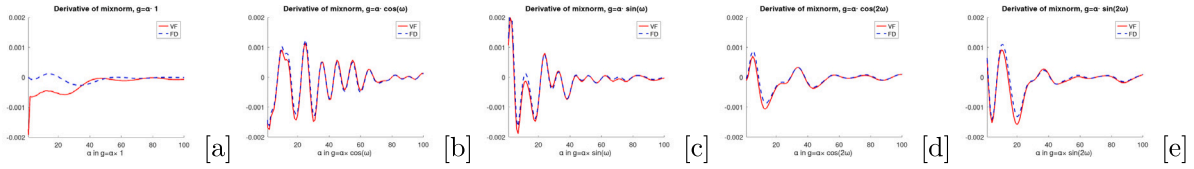

Fig. 2. Computed derivatives of the mix-norm $J_\theta(g) = \frac{1}{2}\|\theta(g)\|^2_{(H^1(\Omega))'}$ by the VF method (red solid line) and the FD method (blue dashline) on mesh $h = 0.1$ at $t = 1$. The control $g = \alpha, \alpha\cos(\omega), \alpha\sin(\omega), \alpha\cos(2\omega), \alpha\sin(2\omega)$, $\alpha \in [0, 100]$ from left to right.

Table 1
Errors of derivatives of the mix-norm $J_\theta(g) = \frac{1}{2}||\theta(g)||^2_{(H^1(\Omega))'}$ by VF and FD methods.

h	AAE 1	AAE $\cos(\omega)$	AAE $\sin(\omega)$	AAE $\cos(2\omega)$	AAE $\sin(2\omega)$	ARE 1	ARE $\cos(\omega)$	ARE $\sin(\omega)$	ARE $\cos(2\omega)$	ARE $\sin(2\omega)$
0.1	2.06e−4	6.47e−5	6.41e−5	4.36e−5	6.37e−5	3.65e+1	2.83e−1	1.25e0	5.45e−1	6.88e−1
0.05	8.01e−5	3.67e−5	2.58e−5	1.55e−5	2.81e−5	2.38e+2	2.05-e1	1.33e0	2.77e−1	6.71e−1
0.025	3.59e−5	1.69e−5	1.19e−5	7.77e−6	1.11e−5	2.33e+1	1.16e−1	7.05e−1	2.19e−1	6.74e−1

AAE = Average Absolute Error, ARE = Average Relative Error.

Table 2
Gradient approximated by VF and FD methods. The control $g = \alpha_1 1_{[0,0.5]} + \alpha_2 1_{[0.5,1]}$.

h	$\alpha = (15, 15)$ VF	$\alpha = (15, 15)$ FD	Relative error	$\alpha = (5, 5)$ VF	$\alpha = (5, 5)$ FD	Relative error
0.1	(−6.67e−4, −2.13e−4)	(8.49e−5, 1.64e−4)	455%	(−7.68e−4, −4.45e−4)	(−1.98e−5, −7.79e−5)	1036%
0.05	(−2.37e−4, 1.62e−6)	(8.96e−5, 1.65e−4)	194%	(−3.46e−4, −2.40e−4)	(−2.76e−5, −8.18e−5)	412%
0.025	(−6.09e−5, 9.09e−5)	(8.94e−5, 1.65e−4)	90%	(−1.74e−4, −1.55e−4)	(−2.77e−5, −8.17e−5)	190%

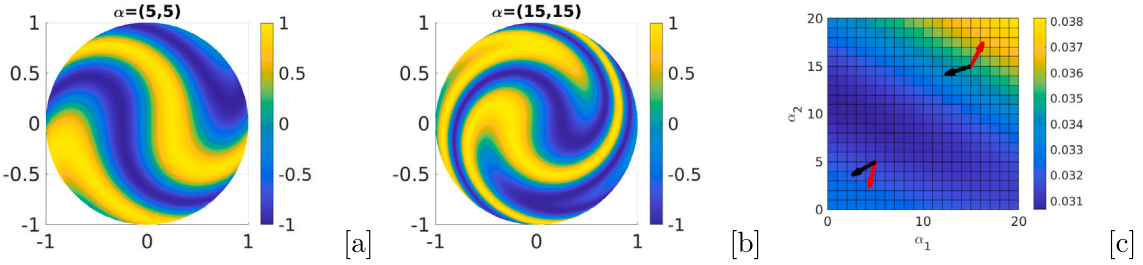

Fig. 3. [a]: $\theta(T)$ when $\alpha = (5, 5)$. [b]: $\theta(T)$ when $\alpha = (15, 15)$. [c]: cost map $J(g)$ when the mesh size $h = 0.1$ and $\gamma = 1e{-}6$. The red vectors are the derivatives from the FD method, and the black vectors are the derivatives from the VF method. The vectors are scaled to have the same length. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.4.4. Comparison of VF and FD methods in 2-D control spaces

We further explore the different performance between VF and FD methods in two tests where in each test, the control space is spanned by two time-segmented basis functions. We denote $g = \alpha_1 g_1^b + \alpha_2 g_2^b$ and $\alpha = (\alpha_1, \alpha_2)$. Here, we test on the whole gradient where $\gamma = 1e{-}6$.

In the first test, $g_1^b = 1_{[0,0.5]}$ (1 when $t \in [0, 0.5]$ and 0 when $t \in (0.5, 1]$) and $g_2^b = 1_{[0.5,1]}$. The results are shown in Table 2. In this table, the FD method gives consistent approximations when the mesh is refined. The FD results are also consistent when some different $\delta = 1e{-}5, 1e{-}4, 1e{-}3$ values are used in (3.12) (data not shown). This suggests the FD results are more reliable when the exact derivative is unknown. The VF results have huge relative errors compared with those of the FD method and they even have opposite directions when $\alpha = (15, 15)$ and $h = 0.1$ (see Figure 3, the VF derivative at $(15, 15)$). The correctness of the directional derivative from the FD method can be verified in Fig. 3 by checking with the cost map. The cost map is the colored plot of the costs computed on integer points of $\alpha = (\alpha_1, \alpha_2)$. Therefore, the VA result in this case does not give a descent direction. The wrong directional derivative is catastrophic in the optimization algorithms used in this work because the line search fails with a non-descent search direction.

The second test is given to $g_1^b = \cos(\omega) \cdot 1_{[0,0.5]}$ and $g_2^b = \sin(\omega) \cdot 1_{[0.5,1]}$. From the results shown in Table 3, the VF and FD methods are very close. The morphologies of θ at $t = T$ corresponding to two different α values are shown in Fig. 4. Similar observations are obtained when the control bases are $\cos(2\omega)$ and $\sin(2\omega)$ and their time segmentations (results not shown).

3.4.5. A hybrid approach

Because of the dramatically different performance of the VF method on Type 1 controls and other types of controls ($\cos(\omega)$, $\sin(\omega)$, $\cos(2\omega)$, $\sin(2\omega)$), we adopt an ad hoc hybrid approach: using the FD method to compute directional derivatives for Type 1

Table 3
Gradient approximated by VF and FD methods. The control $g = \alpha_1 \cos(\omega) \cdot 1_{[0,0.5]} + \alpha_2 \sin(\omega) \cdot 1_{[0.5,1]}$.

h	$\alpha = (50, 50)$ VF	$\alpha = (50, 50)$ FD	Relative error	$\alpha = (10, 10)$ VF	$\alpha = (10, 10)$ FD	Relative error
0.1	(3.79e−4, 9.31e−5)	(5.11e−4, 1.22e−4)	26%	(−6.99e−4, 2.20e−4)	(−6.27e−4, 2.84e−4)	14%
0.05	(6.31e−4, 1.39e−4)	(6.55e−4, 1.41e−4)	3.59%	(−6.51e−4, 2.44e−4)	(−6.20e−4, 2.87e−4)	7.9%
0.025	(6.42e−4, 1.48e−4)	(6.27e−4, 1.42e−4)	2.51%	(−6.33e−4, 2.73e−4)	(−6.20e−4, 2.86e−4)	2.69%

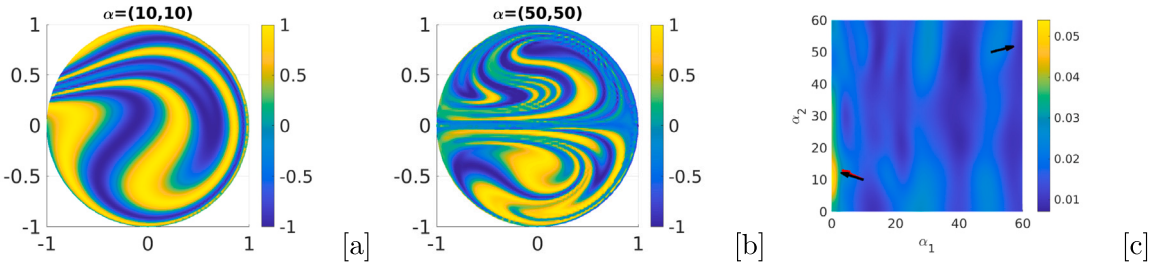

Fig. 4. [a]: $\theta(T)$ when $\alpha = (10, 10)$. [b]: $\theta(T)$ when $\alpha = (50, 50)$. [c]: cost map when the mesh size $h = 0.1$ and $\gamma = 1e{-}6$ and the computed derivative $DJ(g)$ at $\alpha = (10, 10)$ and $(50, 50)$. The red vectors are from the FD method and the black vectors from the VF method. At $\alpha = (50, 50)$, the vectors from the two methods are indistinguishable by eyes. The vectors are scaled to have the same length. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Comparison of different methods for computing $DJ(g)$. M is the dimension of control basis and M_{Type1} is the number of Type 1 control basis functions.

Method	VF (Variational Formula)	FD (Finite Difference)	Hybrid (FD for Type 1 and VF for Type 2)
Evolution of θ and/or ρ	2	$M + 1$	$3 + M_{Type1}$
Other computations	1 elliptic solver, M integrals of $\int_0^T \left(\theta(g)\nabla\rho(g), L(g_i^b)\right) dt$	None	1 elliptic solver, $M - M_{Type1}$ integrals of $\int_0^T \left(\theta(g)\nabla\rho(g), L(g_i^b)\right) dt$
Accuracy in mix-norm derivative	Poor in Type 1, accurate in other types	Accurate in all types	Accurate in all types

controls and the VA method for the other types. That is, in (3.9),

$$DJ(g; g_i^b) = \begin{cases} \frac{J(g+\delta\cdot g_i^b) - J(g)}{\delta}, & g_i^b \in \text{ Type 1 ;} \\ \gamma \int_0^T \langle g, g_i^b \rangle_\Gamma \, dt + \int_0^T \left(\theta(g)\nabla\rho(g), L(g_i^b)\right) dt, & g_i^b \in \text{ other types,} \end{cases} \tag{3.14}$$

3.4.6. Summary of numerical methods for computing $DJ(g)$

A summary of these three computation methods for the Gâteaux derivative is given in Table 4. Note that the derivative of the mix-norm is the only computationally demanding part and the main source of error.

3.5. Line search methods: Computation of step size η^n

The entire optimization algorithm, Algorithm 3.1, includes two essential parts: one is the descent direction method of finding the descent direction d^n of J at g^n, which is described in Section 3.6. The other is the line search method of solely computing the step size η^n in a given descent direction d^n, which is presented here.

3.5.1. Backtracking method and Armijo condition

The backtracking technique (e.g., [40]) is finding η^n such that it is the first value in the sequence

$$\{\eta_i^n = \frac{\epsilon_b}{2^i} : i = 0, 1, \dots\} \tag{3.15}$$

satisfying the following sufficient descent condition (also called Armijo condition),

$$J(g^n + \eta^n d^n) \le J(g^n) + \eta^n \cdot \mu \cdot DJ(g^n; d^n),$$

where ϵ_b^n, μ are positive constants. This method only guarantees the sufficient descent, not a local minimizer. Thus, it does not produce an exact line search. The value of the first value ϵ_b^n is empirically determined and in our work, the values between 1 and 8 are good candidates when d is a unit vector in the $U_{ad}-$norm. The parameter $\mu \in (0, 1)$ according to [40] and we use $\mu = 0.3$. The backtracking scheme is summarized in Algorithm 3.6.

Algorithm 3.6 Backtracking line search with Armijo condition

- Input: control g^n, search direction d^n, $J(g^n)$, parameters μ, ϵ_b, back_MAXITER.
- Output: η^n, g^{n+1}, $J(g^{n+1})$.
- Steps

(1) If d^n is not a descent direction (that is, $DJ(g^n; d^n) \geq 0$), then stop and report problem.
(2) Backtracking iteration. For $i = 1, 2, \dots,$ back_MAXITER,

1. $\eta_i^n = \epsilon_b / 2^{i-1}$.
2. $g_i^{n+1} = g^n + \eta_i^n d^n$.
3. Compute the cost $J(g_i^{n+1})$ by using Algorithm 3.4 with input g_i^{n+1}.
4. If $J(g_i^{n+1}) \leq J(g^n) + \eta_i^n \mu DJ(g^n; d^n)$, stop and return $\eta^n = \eta_i^n$, $g^{n+1} = g_i^{n+1}$, and $J(g^{n+1})$.

In this work, the backtracking method is typically combined with the steepest descent method, where $d^n = -\nabla J(g^n)$. Therefore, the Armijo formula for the steepest descent method becomes

$$J(g^n - \eta^n \nabla J(g^n)) \leq J(g^n) - \eta^n \mu \| DJ(g^n) \|_{U_{ad}}^2. \tag{3.16}$$

3.5.2. Exact line search: A coupled bisection-secant method

In some descent direction methods, an exact line search is needed, such as in the conjugate gradient method, to guarantee the new search direction d^{n+1} is a descent direction (see Section 3.6.2 Eq. (3.25)). That is, η^n is a minimizer of

$$\min_{\eta \geq 0} f(\eta) \triangleq J(g^n + \eta d^n). \tag{3.17}$$

Note $f'(\eta) = DJ(g^n + \eta d^n; d^n)$.

To get an exact solution η of (3.17), we use a coupled bisection and secant method. The strategy is first finding an interval $[0, \eta_1]$, as small as possible, where $f'(0) < 0$ and $f'(\eta_1) > 0$, and then searching for a root of $f'(\eta) = 0$ in this interval. The condition $f'(0) < 0$ is equivalent to that d^n is a descent direction of J at g^n. Because $\lim_{\|g\| \to \infty} J(g) = \infty$, a value η_1^n satisfying $f'(\eta_1) > 0$ must exist. To find η_1, we adopt a forward tracking process as shown in Algorithm 3.7 Step 2. When $f'(0) < 0$ and $f'(\eta_1) > 0$, there exists a root of $f'(\eta) = 0$ in $(0, \eta_1)$ with the continuity assumption of f'. To find a root, we first use several steps of bisection method in order to reduce the search interval size, defined by the distance between the last two bisection solutions ($|\eta_{BIS-1}^n - \eta_{BIS}^n|$), sufficiently small. This is important to the secant method that has faster convergence but requires that the initial guess values are sufficiently close to the exact root. The details of the bisection and secant methods of finding a root of a nonlinear function can be found, e.g., in [41]. The whole exact line search scheme is briefly described in Algorithm 3.7.

Algorithm 3.7 Exact line search with bisection-secant method

- Aim: finding a root of $f'(\eta) = 0$ in an interval $[0, \eta_1]$ where $f'(0) < 0$ and $f'(\eta_1) > 0$. The value of η_1 will be found in this algorithm.
- Input: control g^n, search direction d^n, $J(g^n)$, parameter $\epsilon_{bisection}$, ϵ_{secant}.
- Output: η^n, g^{n+1}, $J(g^{n+1})$.
- Steps

(1) If d^n is not a descent direction (that is, $DJ(g^n; d^n) \geq 0$), then stop and report problem.
(2) Find an $\eta_1^n > 0$ such that $f'(\eta_1^n) > 0$. This is done by a forward tracking process: η_1^n is the first value of the sequence $\eta = \{1, 2, 2^2, \dots\}$ that satisfies $f'(\eta) > 0$.
(3) Apply the bisection method of finding a root of $f'(\eta) = 0$ in $[0, \eta_1^n]$ and stop when $|\eta_{BIS-1}^n - \eta_{BIS}^n| \leq \epsilon_{bisection}$. Here, $\eta_{BIS-1}^n, \eta_{BIS}^n$ are the last two values of bisection solution. In practice, we use $\epsilon_{bisection} = 1$.
(4) Apply the secant method of finding a root of $f'(\eta) = 0$ with the initial values as $\eta_{BIS-1}^n, \eta_{BIS}^n$. Stop when $|DJ(g^n + \eta d^n; d^n)| < \epsilon_{secant}$ and return $\eta^n = \eta$, $g^{n+1} = g^n + \eta d^n$, and $J(g^{n+1})$. In practice, we choose $\epsilon_{secant} = 1e{-}10$.

The forward tracking of finding η_1, bisection, and secant are all iterative and in each iteration, the directional derivative $DJ(g^n + \eta_i^n d^n; d^n)$ is computed for an iterative index i. Because this derivative is only in one direction d^n, we adopt the FD method which uses two evolutions of θ, one for $J(g^n + \eta_i^n d^n)$, one for $J(g^n + (\eta_i^n + \delta) d^n)$. This is simpler than the VF method (see comparisons in Table 4 when $M = 1$). If N_{exact} steps are used in the whole algorithm, then there are $2N_{exact}$ evolutions. From our experience,

Table 5
Comparison of line search methods of computing the step size η.

Method	Backtracking	Exact line search	Linearization
Evolutions of transport equations	2 to 3 on average	15 on average	1
Guarantee descent?	Yes	Almost yes	No
Exact local minimizer?	No	Yes	No
Comments	Mainly used with steepest descent method	Mainly used with conjugate gradient method	Low efficiency: solution η is often too small. Not used in this work.

this whole process of the exact line search takes about 8 iterations on average, which is about 16 evolutions of θ. This is far more expensive than the backtracking method which uses only 2 evolutions on average in each line search.

To guarantee the step size η^n is a local minimizer instead of a local maximizer or saddle point, the interval $[0, \eta_1]$ should be small enough such that it does not contain any other roots of f'. But it is difficult to actualize it because it is too time consuming to find all the roots in this interval. Fortunately, among over thousands of exact line searches in this work, we only find only one case where the step size increases the cost value. Therefore, we claim this method "almost guarantees descent".

3.5.3. Linearization method

A linearization process has been proposed in [35] to approximate the step size in the line search in an optimal control problem subject to an reaction–advection–diffusion system. This motivates us to develop a similar approach. We first linearize the relation between θ and $g^n + \eta d^n$ as

$$\theta(g^n + \eta d^n) \approx \theta(g^n) + \eta \cdot D\theta(g^n; d^n) \tag{3.18}$$

and denote $z \triangleq D\theta(g^n; d^n)$. Then the objective function $J(g^n + \eta d^n)$ is replaced by the linearized version

$$J_L(g^n + \eta d^n) = \frac{1}{2}(\Lambda^{-2}(\theta(g^n) + \eta z), \theta(g^n) + \eta z)(T) + \frac{\gamma}{2}\int_0^T \langle g^n + \eta d^n, g^n + \eta d^n\rangle_\Gamma dt. \tag{3.19}$$

Its derivative on η is

$$DJ_L(g^n + \eta d^n; d^n) = (\Lambda^{-2}(\theta(g^n) + \eta z), z)(T) + \gamma\int_0^T \langle g^n + \eta d^n, d^n\rangle_\Gamma dt. \tag{3.20}$$

Letting it be zero, we get the critical value

$$\eta^n = -\frac{(\Lambda^{-2}\theta(g^n), z)(T) + \gamma\int_0^T \langle g^n, d^n\rangle_\Gamma dt}{(\Lambda^{-2}z, z)(T) + \gamma\int_0^T \langle d^n, d^n\rangle_\Gamma dt} = -\frac{(DJ(g^n), d^n)_{U_{ad}}}{(\Lambda^{-2}z, z)(T) + \gamma\int_0^T \langle d^n, d^n\rangle_\Gamma dt}. \tag{3.21}$$

To determine z, we take the Gâteaux derivative on Eq. (2.1) and the initial value (2.5) and obtain

$$\frac{\partial z}{\partial t} + v(g^n)\cdot\nabla z + v(d^n)\cdot\nabla\theta(g^n) = 0, \tag{3.22}$$

$$z(t=0) = 0. \tag{3.23}$$

Note $v(g^n) = L(g^n)$ and $v(d^n) = L(d^n)$. Thus, to evaluate η^n, we first evolve z with (3.22)–(3.23) and then compute it from (3.21).

In this method, the product $\eta^n d^n$ is scale invariant, i.e., if d^n multiplies a positive number a, then η^n value will be decreased by a. Indeed, if d^n is increased by a folds, then z will be also increased by a folds (because z is linear on d^n in (3.22)), and then η^n in (3.21) will be decreased by a.

There are two issues with this linearization methods based on our numerical tests. First, the step sizes obtained by this method are often ten to a few hundred times smaller than those computed by the backtracking and exact line search methods, which makes this method very inefficient. Second, when this method is combined with the conjugate gradient method, the cost value often increases. This is because the combined method cannot guarantee that the new search direction is descent, that is, $DJ(g^{n+1}; d^{n+1}) < 0$. Indeed, in the calculation in (3.24) and (3.25), $DJ(g^{n+1}; d^n)$ is not guaranteed to be zero. Instead, $DJ_L(g^{n+1}; d^n)$ is zero in this linearization method due to the choice of η^n in (3.21). That is, η^n is a local minimizer of $J_L(g^n + \eta d^n)$, instead of $J(g^n + \eta d^n)$. Due to the nature of linearization, this method should provide a good approximation of the exact line search only when the exact step size is sufficiently close to zero, which is not often the case. Therefore, this method is not used in our work.

3.5.4. Summary of line search methods

Table 5 summarizes the performance of these line search methods based on the simulations of this work. The linearization method is not used extensively in this work due to its low efficiency. We mainly use the backtracking and exact line search methods.

3.6. Descent direction d^n and the entire optimization algorithms

This section describes two choices of the descent directions: the steepest descent direction and the conjugate gradient direction. To simplify the presentation, we use these two directions to name the entire optimization algorithms, that is, the steepest descent method and the conjugate gradient method.

3.6.1. Steepest descent (SD) method

The steepest descent method uses the negative Gâteaux derivative as the descent search direction, i.e., $d^n = -DJ(g^n)$. This method is described in Algorithm 3.8. Through trials, we find the exact line search applied to the SD method not only requires many evolutions in each line search, but also takes many steepest descent steps to converge. Therefore, we will only use backtracking with steepest descent method. In this work, we set MAXITER = 10000 and ϵ = 1e−5 for both steepest descent and conjugate gradient methods in most cases.

Algorithm 3.8 Steepest descent method

- Input: initial control g^0, maximum iteration number MAXITER, stopping criterion ϵ.
- Output: a local minimizer of J.
- Before iteration: compute $J(g^0)$.
- For $n = 0, 1, \dots,$ MAXITER,
  (1) Compute $DJ(g^n)$ with FD or VF or Hybrid method.
  (2) If $||DJ(g^n)||_{U_{ad}}/(1 + J(g^n)) < \epsilon$, stop and output g^n as a local minimizer.
  (3) Let $d^n = -DJ(g^n)$.
  (4) Use a line search method with g^n and d^n to compute η^n and then obtain $g^{n+1} = g^n + \eta^n d^n$ and $J(g^{n+1})$.

3.6.2. Conjugate gradient (CG) method

The conjugate gradient method (e.g. [40, Section 13.4]) is widely used in optimization and its application in this work is given in Algorithm 3.9. Note the exact line search is used with the conjugate gradient method to ensure that d^{n+1} is a descent direction. Indeed,

$$DJ(g^{n+1}; d^{n+1}) = DJ(g^{n+1}; -DJ(g^{n+1}) + \beta^n d^n) \tag{3.24}$$

$$= -\|DJ(g^{n+1})\|^2_{U_{ad}} + \beta^n DJ(g^{n+1}; d^n). \tag{3.25}$$

To guarantee the negativity of $DJ(g^{n+1}; d^{n+1})$ when $\|DJ(g^{n+1})\|_{U_{ad}}$ approaches the tolerance ϵ, the value $|\beta^n DJ(g^{n+1}; d^n)|$ should be smaller than ϵ^2. In this work, we use ϵ = 1e−5 and the tolerance in the exact line search as 1e−10, that is, g^{n+1} is accepted when $|DJ(g^{n+1}; d^n)|$ <1e−10 in the exact line search.

Algorithm 3.9 Conjugate gradient method

- Input: initial control g^0, maximum iteration number MAXITER, stopping criterion ϵ,
- Output: a local minimizer of J.
- Before iteration: compute $J(g^0)$, $DJ(g^0)$, and let $d^0 = -DJ(g^0)$.
- For $n = 0, 1, \dots,$ MAXITER,
  (1) If $||DJ(g^n)||_{U_{ad}}/(1 + J(g^n)) < \epsilon$, stop and output g^n as a local minimizer.
  (2) Use the exact line search Algorithm 3.7 with g^n and d^n to compute η^n and then obtain $g^{n+1} = g^n + \eta^n d^n$ and $J(g^{n+1})$.
  (3) Compute $DJ(g^{n+1})$ with FD or VF or Hybrid method.
  (4) Compute the parameter $\beta^n = \frac{||DJ(g^{n+1})||^2_{U_{ad}}}{||DJ(g^n)||^2_{U_{ad}}}$.
  (5) Compute the new search direction $d^{n+1} = -DJ(g^{n+1}) + \beta^n d^n$.

3.7. A convergence test of the optimization algorithms

In this convergence study, we compare the steepest descent method with the backtracking line search and the conjugate gradient method with the exact line search. The control function space is chosen as $U^2_{ad} = \text{span}\{g^b_1 = 1_{[0,0.5]}, g^b_2 = 1_{[0.5,1]}\}$. In the stopping criterion, $\|DJ(g^n)\|_{U_{ad}}/(1 + J(g^n)) < \epsilon$ of Algorithms 3.8 and 3.9, ϵ is set as 1e−5, 5e−6, 1e−6 when $h = 0.1, 0.05, 0.025$ respectively. The initial guess is $\alpha^0 = (15, 15)$. Both the steepest descent and conjugate gradient solutions converge approximately to the same local minimizer $\alpha = (-2.06, 11.81)$ (see Table 6), where the corresponding θ at $t = 1$ is plotted in Fig. 5[c]. The steepest descent method shows the typical zigzag path of solutions (Fig. 5[b]), as seen in [40, page 408]. Although the relayed conjugate gradient method uses far less iteration steps towards the minimizer than the relayed steepest descent method (5 steps vs 44 steps), it indeed takes roughly the same amount of CPU time (between 7 and 9 h). This is because the conjugate gradient method uses many more evolutions in each line search, which results in roughly the same number of total evolutions (roughly 90). Most importantly, the relay schemes significantly save the computational time: they converge within 9 h but the non-relayed schemes take one or two

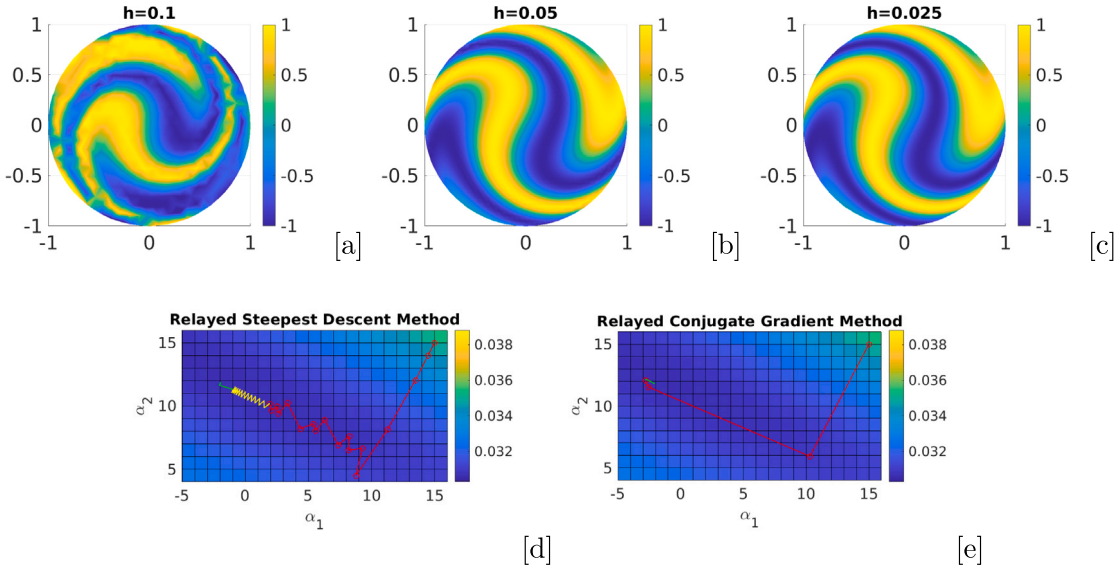

Fig. 5. [a, b, c]: Plots of θ at $t = 1$ in the conjugate gradient algorithm. [d, e]: paths of iterative solutions $\alpha^n = (\alpha_1^n, \alpha_2^n)$ of the relayed methods on the cost map. The red marker and line refer to the solutions on the mesh with $h = 0.1$, the yellow ones with $h = 0.05$, and the green ones with $h = 0.025$. The iteration information is given in Table 6. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 6
Convergence information of SD with backtracking line search and CG with exact line search.

	non-relay SD	relay SD	non-relay CG	relay CG
$h = 0.1$ CPU time:	15 min	⇐ same as left	15 min	⇐ same as left
steps:	17 steps		3 steps	
minimizer:	(1.94, 10.19)		(−2.81, 12.09)	
cost:	3.07013e−02		3.06672e−02	
$h = 0.05$, CPU time:	5 hr	1 hr 49 min	2 hr 2 min	6 min
steps:	39 steps	24 steps	2 steps	0 steps
minimizer:	(−1.01, 11.33)	(−0.97, 11.31)	(−2.11, 11.81)	(−2.81, 12.09)
cost:	3.10956e−02	3.10957e−02	3.10929e−02	3.10941e−02
$h = 0.025$, CPU time:	48 hr 41 min	5 hr 22 min	24 hr	8 hr 24 min
steps:	40 steps	3 steps	3 steps	2 steps
minimizer:	(−1.66, 11.61)	(−2.00, 11.77)	(−2.06, 11.81)	(−2.06, 11.81)
cost:	3.119671e−2	3.119639e−2	3.119638e−2	3.119638e−2
Relay total CPU time,		7 hr 26 min, 44		8 hr 45 min, 5
steps, evols in LS:		steps, 93 evols		steps, 88 evols

Abbreviations: SD = steepest descent, CG = conjugate gradient, 2 hr 30 min = 2 h 30 min, steps = SD or CG iteration steps, 93 eolvs in LS = 93 evolutions of θ in line search.

days on the finest mesh used in the relayed schemes. The solution paths (Fig. 5[de]) manifest the search on the coarsest mesh gets close to the final solution, which makes the remaining search on the finer meshes much easier. Although the mixed scalar looks rough on the coarsest mesh (Fig. 5[a], it does not prevent the relay algorithm to converge to a local minimizer. This is a hallmark of all the relay simulations in this work.

4. Optimization simulations

This section applies the optimization algorithms developed in this work to study the boundary controlled mixing with the control basis functions mentioned in Section 3.2.1 through extensive numerical experiments. We first describe the flow patterns and mixing features of each control basis function, and then combine them together to study the optimal mixing. All the numerical simulations are performed in Michigan State University's High Performance Computing Center (HPCC).

The physical setup and the initial values are introduced at the end of Section 2.1. When the optimization algorithms are called for a set of control basis $\{g_i^b : i = 1, \dots, M\}$, we apply the relay Algorithm 3.2 with three meshes of $h = 0.1, 0.05, 0.025$. To handle multiple local minimizers, 5 different initial guesses of $\alpha = (\alpha_1, \dots, \alpha_M)$ are tested on the coarsest mesh for each set of control basis. These initial vectors α are randomly chosen where each component $\alpha_i, i = 1, \dots, M$, is uniformly distributed from −100 to 100. For each initial guess of α, we apply both the steepest descent and conjugate gradient methods to find the optimal solutions, where

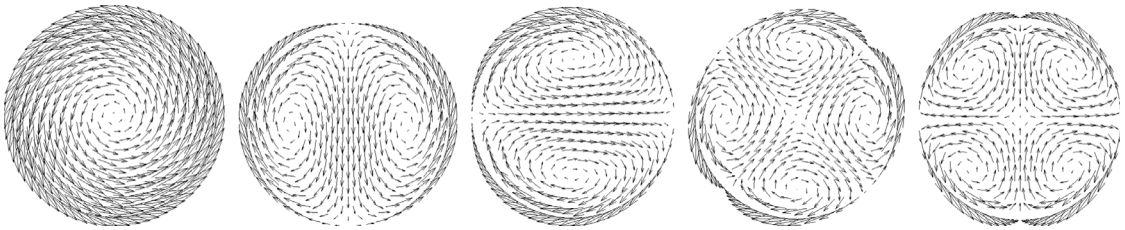

Fig. 6. Flow patterns for $g = 1, \cos(\omega), \sin(\omega), \cos(2\omega), \sin(2\omega)$ at time $t = 1$ (from left to right).

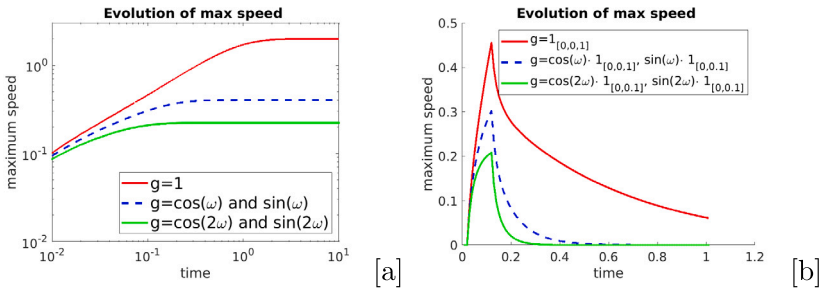

Fig. 7. Evolution of the maximum speed of velocity for five elementary control functions. [a]: the controls are applied in the entire time interval $[0, 1]$. [b]: the controls are only applied in the time interval $[0, 0.1]$. The fluid-wall friction parameter $k = 0.5$.

these two solutions are generally different. Afterwards, the one with the smallest cost is relayed to the intermediate mesh and finally is sent to the finest mesh. The computation of the cost gradient is by the hybrid method in Section 3.4.5. The line search for the steepest descent method is the backtracking Algorithm 3.6 and the one for the conjugate gradient method is the exact line search Algorithm 3.7. The parameter ϵ in the stopping criterion of SD and CG methods is $\epsilon = 1e{-}5$ for all the simulations in this section.

4.1. Flow patterns of control basis functions

In this work, the controls are divided into five types based on five elementary functions: 1, $\cos(\omega)$, $\sin(\omega)$, $\cos(2\omega)$, $\sin(2\omega)$, where $\omega \in [0, 2\pi)$ (see details in Section 3.2.1). The U_{ad}-norm is $\sqrt{2\pi}$ for $g = 1$ and $\sqrt{\pi}$ for other functions. Their flow patterns at time $t = 1$ are shown in Fig. 6. There exist one vortex for $g = 1$, two vortices for $\cos(\omega)$ and $\sin(\omega)$, and four vortices for $\cos(2\omega)$ and $\sin(2\omega)$.

When $g = 1$, a radially symmetric analysis (Appendix A.6) shows the velocity field has a unique steady state with radial component $v_r = 0$ and angular component $v_\varphi = g/kr$, along with a zero pressure field. Apparently this steady state velocity does not induce any mixing because it is simply a rigid rotation. Therefore, the mixing for $g = 1$ occurs only when the velocity is unsteady. The evolution of the maximum speeds in the domain of these elementary control functions are shown in Fig. 7[a]. The flow of $g = 1$ reaches the steady state around $t = 3.8$, while the flows of $\cos(\omega)$ and $\sin(\omega)$ reach the steady states with maximum speed 0.4 around $t = 0.6$. The flows of $\cos(2\omega)$ and $\sin(2\omega)$ reach the steady states with maximum speed 0.22 around $t = 0.3$. Note when the initial velocity is zero, the time scale for the flow induced by αg, $\alpha \neq 0$, to reach the steady state is independent of α, due to the linearity of the flow to the control.

When the control is only applied in a time segmentation interval, the flow velocity will decay to zero over time after the control is turned off due to viscous dissipation and boundary wall friction. Fig. 7[b] shows the evolution of the maximum speed where the five elementary functions are applied only in the time interval $[0, 0.1]$. When the time segmentation interval is $[0.1n, 0.1(n+1)]$, $n = 1, \ldots, 9$, the corresponding flow can be obtained by simply shifting by $0.1n$ units to the positive time direction the flow generated by the same elementary function applied on $[0, 0.1]$. It is noticed that the flow decays to zero far faster when it is generated by a cosine or sine function than by the function 1. This is produced by the extra dissipation between multiple vortices from a cosine or sine control function, in contrast to only one vortex from the control 1 (see Fig. 6).

4.2. Optimization by each single control type

This part is devoted to the mixing properties of each of the five types of control basis functions. First, we compute the mix-norms and costs at $t = 1$ with Algorithm 3.4 for the controls $g = \alpha g^b$, where g^b is one of the five elementary functions and α takes integer values in $[0, 100]$. This corresponds to the time segmentation number $N = 1$. Afterwards, we use the optimization algorithms to compute the optimal solution when $N = 10$ for each type of control basis functions.

The most striking property is the existence of multiple local minimizers of the mix-norm for most control basis functions when the coefficient α varies, according to Fig. 8[a]. When $\gamma = 1e{-}6$, the cost also has multiple local minimizers (Fig. 8[b]). Because one

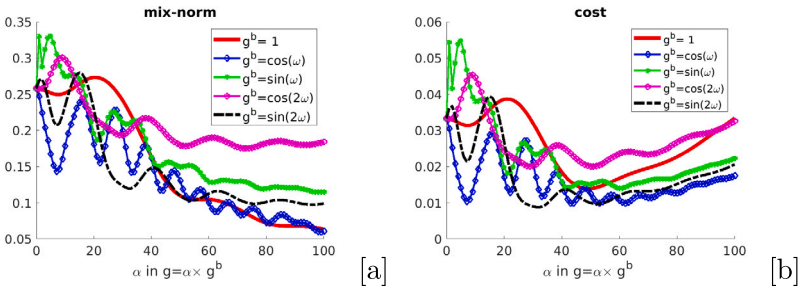

Fig. 8. Mix-norms and costs of five elementary functions when $\alpha \in [0, 100]$.

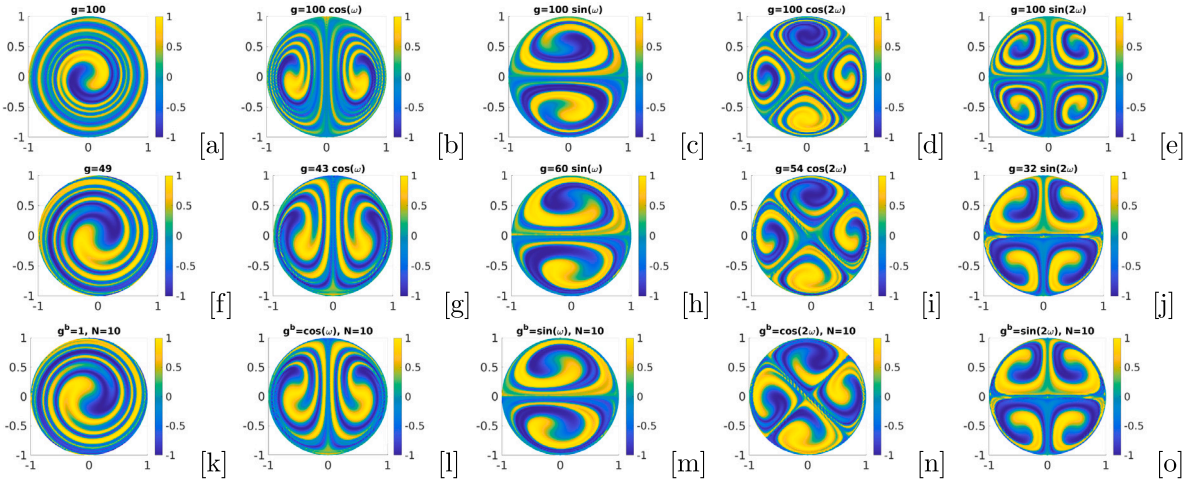

Fig. 9. Plotings of θ at $t = 1$ on the mesh of $h = 0.025$. The first row is when $\alpha = 100$ in $g = \alpha g^b$, the second row is for the optimal solutions when $N = 1$, and the third row is for the optimal solutions when $N = 10$.

Table 7
Mixing information of single type of control basis functions.

	$\alpha = 100$ and $N = 1$			Optimal solution when $N = 1$			Optimal solution when $N = 10$			CRP
	Mix-norm	g-norm	Cost	Mix-norm	g-norm	cost	Mix-norm	g-norm	Cost	
1	6.34e−2	2.51e+2	3.34e−2	1.12e−1	1.23e+2	1.39e−2	1.10e−1	1.17e+2	1.30e−2	6%
$\cos(\omega)$	6.07e−2	1.77e+2	1.76e−2	1.18e−1	7.62e+1	9.84e−3	9.32e−2	9.44e+1	8.79e−3	11%
$\sin(\omega)$	1.15e−1	1.77e+2	2.24e−2	1.30e−1	1.06e+2	1.41e−2	1.11e−1	1.10e+2	1.22e−2	13%
$\cos(2\omega)$	1.84e−1	1.77e+2	3.27e−2	1.80e−1	9.57e+1	2.08e−2	1.82e−1	6.87e+1	1.89e−2	9%
$\sin(2\omega)$	9.93e−2	1.77e+2	2.06e−2	1.20e−1	5.67e+1	8.78e−3	1.13e−1	6.46e+1	8.42e−3	4%

CRP = Cost Reduction Percentage from $N = 1$ case to $N = 10$ case.

initial guess only leads to one local minimizer in an optimization algorithm, multiple initial guesses are needed in order to achieve the global minimizer.

The second property is that the better mixing quality, identified with the lower mix-norm, corresponds to the larger control strength in general (Fig. 8[a]), and thus the larger velocity magnitude because the flow velocity is linearly dependent on the control. The mixed scalar fields at $t = 1$ when $\alpha = 100$, the largest control strength considered, are shown in Fig. 9[a–e], each of which has almost the smallest mix-norm in the same control type. On the other hand, the scalar fields at $t = 1$ with the smallest costs in the same control type when $N = 1$ are shown in Fig. 9[f–j]. The data of the mix-norms, g-norms, and costs of these simulations are displayed in Table 7. From the relation between the scalar field renderings and their mix-norms, it appears that a better mixed field is characterized by thinner and longer filaments.

The third property is that the mixing quality of one control type is limited by its specific flow pattern. By comparing Figs. 6 and 9, we can tell Type 1 takes the entire domain as a single mixing zone, Type $\cos(\omega)$ and Type $\sin(\omega)$ divide the domain into two separate mixing zones, and Type $\cos(2\omega)$ and Type $\sin(2\omega)$ divide the domain into four isolated mixing zones. In each mixing zone, the mixing is performed by rotating the scalar around the center. If a mixing zone is predominantly occupied by one value or one color, then the mixing would not be effective due to the lack of mass exchange between different zones. For example, in the four mixing zones of the control $\cos(2\omega)$, the color of θ is predominantly blue in the upper zone and predominantly yellow in the lower

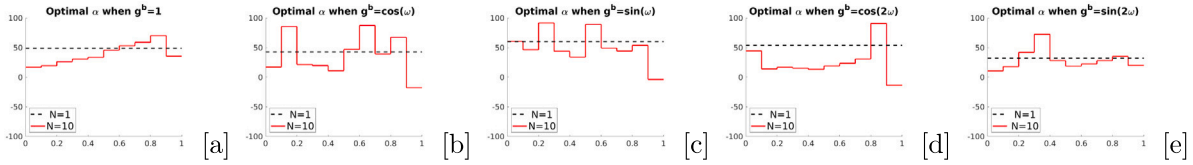

Fig. 10. Optimal solutions α for $N = 1$ and $N = 10$ for different control basis functions. The black dashlines are for $N = 1$ and the red solid lines are for $N = 10$. When $N = 1$, $\alpha = 49, 43, 60, 54, 32$ from left to right.

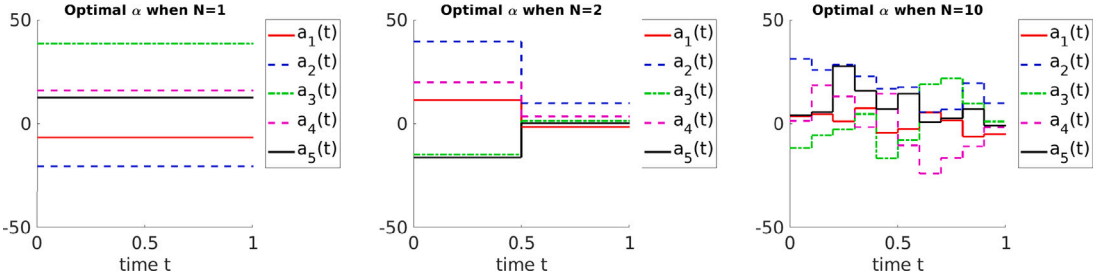

Fig. 11. Optimal solutions $\alpha = (\alpha_1(t), \ldots, \alpha_5(t))$ in $g = \sum_{i=1}^{5} \alpha_i(t) g_i^b$, where $g_1^b = 1$, $g_2^b = \cos(\omega)$, $g_3^b = \sin(\omega)$, $g_4^b = \cos(2\omega)$, and $g_5^b = \sin(2\omega)$.

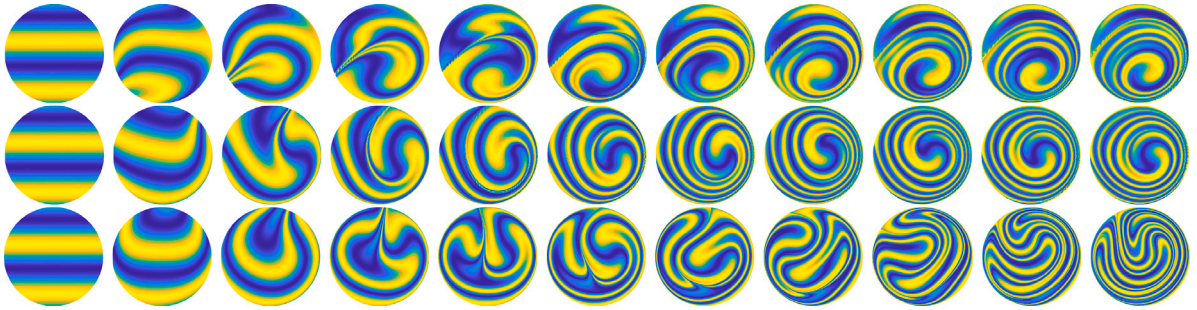

Fig. 12. Snapshots of θ at $t = 0, 0.1, \ldots, 1$ of the optimal solutions with five types of control. First row: $N = 1$. Second row: $N = 2$. Third row: $N = 10$.

zone all the time during the mixing process no matter how α changes (see Fig. 12 at $t = 0$ and Fig. 9[d, i, n]). This is why the mix-norm refuses to decrease when α exceeds 50 for $\cos(2\omega)$ (see Fig. 8[a]).

The purpose of time segmenting is to provide control flexibility in time to reduce cost. This is modestly successful because the cost reduction rates from $N = 1$ to $N = 10$ are only between 4% and 13%, as seen in Table 7. The mix-norms when $N = 10$ are also smaller than those when $N = 1$ in the same type of control except for Type $\cos(2\omega)$. To easily plot the control solution, we express the control as $g = \alpha g^b$ where $\alpha = \sum_{i=1}^{N} \alpha_i \chi_i^N(t)$ and g^b is one of the five elementary functions. The optimal solutions α are plotted in Fig. 10, which are very different between $N = 1$ and $N = 10$ cases for the same type of control. The scalar field of the optimal solution when $N = 10$ does not differ much from that when $N = 1$ of the same control type (see Fig. 9 second and third rows). All of these facts indicate that under single control types investigated in this work, modulating the time segmentation is not very efficient in cost reduction.

4.3. Optimization by combined control types

In this section, all the five types of controls used in the last section are combined together to steer mixing, where the time segmentation number N is chosen as $N = 1, 2, 10$. The optimal solutions of the control are shown in Fig. 11. The snapshots of time evolution of the density in the optimal mixing of each value of N are illustrated in Fig. 12, which show that the morphology is more complicated when N is larger. Furthermore, when N is larger, the mix-norm, g-norm, and the cost of the optimal solution become smaller (Table 8). The minimum cost of the combined control types is 2.37e−3, which is 28% of the smallest cost 8.42e−3 of only one control type, corresponding to $g^b = \sin(2\omega)$ in Table 7. This supports the usage of multiple control types to reduce the cost. The mix-norms of these optimal solutions demonstrate the exponential decay in the time window $[0.6, 1]$ (Figire 13). Fig. 14 illustrates the details how the mix-norm, g-norm, and cost decrease with the iteration number in the relay algorithm.

Table 8
Information of optimal solutions when all five types of controls are combined.

N	Mix-norm	g-norm	Cost
1	7.55e−2	8.71e+1	6.64e−3
2	6.20e−2	6.70e+1	4.17e−3
10	3.68e−2	5.83e+1	2.37e−3

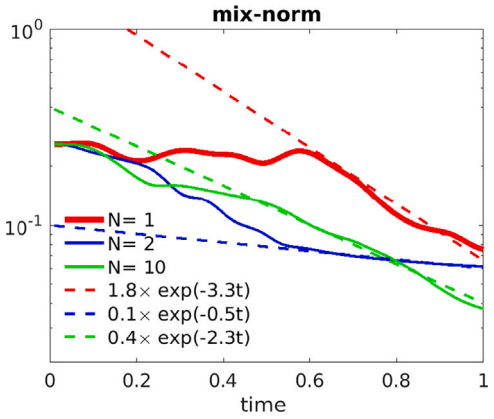

Fig. 13. Mix-norm decay over time of three optimal solutions with the combined control types.

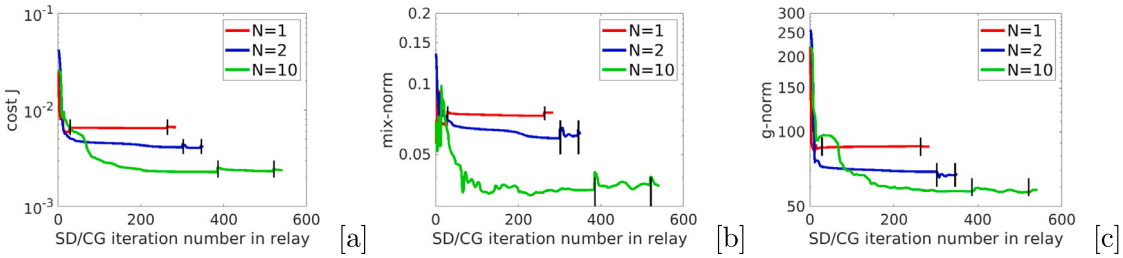

Fig. 14. Decays of the cost [a], mix-norm [b], and g-norm [c] with respect to iterations in the relay algorithm. The small vertical black bars represents the relay moments when a coarse mesh is replaced with a fine mesh.

5. Conclusions

This work is the first numerical study of optimal mixing through tangential force exerted on the boundary in the unsteady Stokes flows. In the absence of diffusion, transport and mixing occur due to pure advection. Built upon the theoretical foundation laid by Hu and Wu, an accurate and efficient optimization algorithm is proposed. The entire algorithm is sophisticated due to the nature of the problem and has many new techniques, which are summarized below.

(1) The boundary control is focused on a finite number of basis functions with time segmentation. Given the zero initial velocity field, the linear relation between the flow and the control allows the generation of the velocity basis before the optimization process, thus saving the simulation time.
(2) The computation of the gradient of the cost functional is crucial to the numerical accuracy, where a hybrid method is developed to treat different control basis functions with appropriate methods (finite difference or variational formula).
(3) The combination of several line search methods and descent direction choices are investigated. Specifically, the following two pairs work well: the steepest descent method with the backtracking line search, and the conjugate gradient method with the exact line search. The simulations demonstrate that the latter performs slightly better than the former in most simulations, but not significantly.
(4) A relay process is placed on the top of this optimization algorithm by repeatedly refining the search from a coarser mesh to a finer one. Numerical tests in Section 3.7 show that this process produces accuracy results while significantly saving the computational time.

The numerical simulations reveal the following physical features of mixing by the boundary control design.

(1) The mixing efficacy of only one single type of control function may be limited by the fixed flow pattern, as shown in Section 4.2. The different control types derived from $\cos(\omega)$, $\sin(\omega)$, $\cos(2\omega)$, $\sin(2\omega)$ have separatrices in the domain. But when these types are combined and added the Type 1 control, the separatrices are eliminated. This is consistent with the observation in [6], where the wall rotation removes the separatrices produced by the internal mixing. Furthermore, the time segmentation of a control, similar to the chaotic mixing strategy, can furthermore increase the mixing efficacy.
(2) The result of the boundary control can be comparable to that of the internal control, which can be seen from the comparison of the mixed density in Section 4.3 with those in [31], where the velocity field is generated by the internal stirring. In addition, it is observed that the mix-norm of the scalar field under the optimal boundary control reaches the exponential decay rate.

Another unique feature of this work is the use of the dynamic control, where a force is modulated to steer mixing. In contrast, all the existing works from other researchers mentioned at the beginning of Section 1.2 have employed the kinematic control, that is, a velocity field is directly modulated. One intrinsic difference between these two types of controls is the inertia, i.e., the perseverance of the motion until it is changed by a force. In the case of dynamic controls, the velocity takes a certain time to accelerate from zero to a field with effective mixing when the force is started, and another time duration to decelerate to negligible magnitude after the force is withdrawn. This can be seen clearly in Fig. 7. However, in the case of kinematic controls, a prescribed velocity field is modulated in an arbitrary manner without consideration of any inertia effects. Therefore, the dynamic control would better represent the reality in the mixing problems where the inertia effect is significant.

The optimal control model and numerical methods can be straightforwardly extended to the three dimensional case. First, the boundary control design for the optimal mixing has been shown to be valid in both two and three spatial dimensions [18]. Second, the numerical methods, including the optimization algorithms and the solvers for the unsteady Stokes equations and transport equations can be directly extended to three dimensional space. The apparent challenge will be the more storage and computational costs when one extra spatial dimension is added.

This work focuses on the tangential boundary force control with the Navier slip boundary conditions, which can model the tangential cilia beating in the inner membrane of vertebrate organs. As described in Section 1.1, there are many examples of boundary driven mixing in nature and industry, including rotating wall driven mixing, mircomixers with acoustic waves, and artificial cilia mixing. Therefore, there is a big potential to extend this work to these applications and beyond. Furthermore, it is interesting to study the effects of combining it with internal controls for optimal mixing problems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

X. Zheng was partially supported by the National Science Foundation of USA Grant DMS-2309747. W. Hu was partially supported by the National Science Foundation of USA grants DMS-2005696 (previously DMS-1813570), DMS-2111486 and DMS-2205117. J. Wu was partially supported by the National Science Foundation of USA grants DMS-2104682 and DMS-2309748. This work was supported in part by computational resources and services provided by HPCC of the Institute for Cyber-Enabled Research at Michigan State University, USA through a collaboration program of Central Michigan University, USA.

Appendix

A.1. Derivation of the Gâteaux derivative

The rigorous derivation of the first-order optimality system for $U_{ad} = L^2(0,T;L^2(\Gamma))$ has been addressed in [18], using an approximating control approach. Here we provide a short and formal derivation by assuming that all the involved functions are sufficiently smooth and all the operations are valid.

Theorem A.1. *With the governing Eqs. (2.1)–(2.5), the Gâteaux derivative of J with respect to g in the direction $\varphi \in U_{ad}$ is given by*

$$DJ(g;\varphi) = \int_0^T (\theta(g)\nabla\rho(g), L\varphi)\,dt + \gamma \int_0^T \langle g, \varphi\rangle_\Gamma\,dt, \tag{A.1}$$

where $\rho(g)$ is the adjoint state satisfying (2.15)–(2.16) and $L\varphi$ is the velocity field governed by the Stokes system (2.2)–(2.5) with $v_0 = 0$ and the tangential boundary control g replaced by φ.

Proof. Recall from (2.9) that $J(g) = \frac{1}{2}(\Lambda^{-2}\theta(T), \theta(T)) + \frac{\gamma}{2}\int_0^T \langle g, g\rangle_\Gamma dt$. Taking the Gâteaux derivative of J at g in the direction φ gives

$$DJ(g;\varphi) = (\Lambda^{-2}\theta(T), D\theta(g;\varphi)(T)) + \gamma \int_0^T \langle g, \varphi\rangle_\Gamma dt, \tag{A.2}$$

where $D\theta(g;\varphi)(T)$ is the Gâteaux derivative of θ at g in the direction φ at time T. Let $z \triangleq D\theta(g;\varphi)$ and $w \triangleq Dv(g;\varphi)$. Then z satisfies

$$\frac{\partial z}{\partial t} + v\cdot\nabla z + w\cdot\nabla\theta = 0, \tag{A.3}$$

$$z(0) = 0. \tag{A.4}$$

Using the notation $v(g) = L(g)$ and the linearity of L when $v_0 = 0$, we have $w = DL(g;\varphi) = L(\varphi)$, which is also divergence free and $L(\varphi)\cdot n|_\Gamma = 0$. Next, taking the inner produce of (A.3) with ρ and integrating with respect to t over $[0,T]$, we get

$$\int_0^T \left(\frac{\partial z}{\partial t}, \rho\right) dt + \int_0^T (v\cdot\nabla z, \rho)\, dt + \int_0^T (L(\varphi)\cdot\nabla\theta, \rho)\, dt = 0. \tag{A.5}$$

Using $(v\cdot\nabla z, \rho) = -(v\cdot\nabla\rho, z)$ and (A.4), the above equation becomes

$$(\rho(T), z(T)) - \int_0^T \left(z, \frac{\partial\rho}{\partial t}\right) dt - \int_0^T (v\cdot\nabla\rho, z)\, dt + \int_0^T (L(\varphi)\cdot\nabla\theta, \rho)\, dt = 0. \tag{A.6}$$

Since ρ satisfies (2.15) and (2.16), it follows from (A.6) that

$$(\Lambda^{-2}\theta(T), D\theta(g,\varphi)(T)) = (\rho(T), z(T)) = -\int_0^T (L(\varphi)\cdot\nabla\theta, \rho) dt. \tag{A.7}$$

Finally, plugging (A.7) into (A.2) yields

$$DJ(g;\varphi) = -\int_0^T (L(\varphi)\cdot\nabla\theta, \rho) dt + \gamma\int_0^T \langle g, \varphi\rangle_\Gamma dt = \int_0^T (\theta\nabla\rho, L(\varphi)) dt + \gamma\int_0^T \langle g, \varphi\rangle_\Gamma dt.$$

Remark A.2. This theorem still holds when the initial velocity $v_0 \neq 0$. In this case, $v(g) = L(g) + \tilde{v}$ where $\tilde{v}$ is the velocity field generated by v_0 through the Stokes system (2.2)–(2.4) with $g = 0$. Since $\tilde{v}$ is independent of g, $D\tilde{v}(g;\varphi) = 0$. Thus, $Dv(g;\varphi) = DL(\varphi)(g,\varphi) + D\tilde{v}(g,\varphi) = L(\varphi)$. Then the same proof holds.

A.2. Proof of Proposition 2.1

Proof. For any $s \in [0,T]$, taking the inner product of (2.1) with ρ over Ω and integrating in time from s to T gives

$$\int_s^T (\theta_t, \rho)\, dt + \int_s^T (v\cdot\nabla\theta, \rho)\, dt = 0. \tag{A.8}$$

Integration by parts leads to

$$(v\cdot\nabla\theta, \rho) = \langle v\cdot n, \theta\rho\rangle_\Gamma - (\nabla\cdot v, \theta\rho) - (\theta, v\cdot\nabla\rho) = -(\theta, v\cdot\nabla\rho),$$

where the conditions $\nabla\cdot v = 0$ and $v\cdot n|_\Gamma = 0$ are used. Thus, (A.8) becomes

$$\int_s^T (\theta,\rho)_t - (\theta,\rho_t) dt - \int_s^T (\theta, v\cdot\nabla\rho) dt = 0.$$

This turns to

$$(\rho(T),\theta(T)) - (\rho(s),\theta(s)) = \int_s^T (\theta, \rho_t + v\cdot\nabla\rho)\, dt.$$

Since $\rho_t + v\cdot\nabla\rho = 0$,

$$(\rho(T),\theta(T)) = (\rho(s),\theta(s)). \tag{A.9}$$

A.3. Unsteady Stokes equations: Iterative projection/BDF2/Taylor–Hood finite element method

The standard Taylor–Hood P2/P1 elements are employed to approximate the velocity and pressure in the Stokes Eqs. (2.2)–(2.4). That is, the velocity is approximated by the continuous piecewise quadratic functions and the pressure by the continuous piecewise linear functions. Denote the triangulated domain as Ω_h where all the elements are triangles. The finite element spaces are defined as

$$V_h = \{w = (w_1, w_2) \in (C^0(\Omega))^2 : w\cdot n|_\Gamma = 0, w_i|_K \in P^2(K), i = 1,2, \forall K \subset \Omega_h\}, \tag{A.10}$$

$$Q_h = \{q \in C^0(\Omega) : q|_K \in P^1(K), \forall K \subset \Omega_h\}, \tag{A.11}$$

where n is the unit outward normal on the boundary.

The basis functions of V_h are chosen as follow. Denote the inner nodes of the mesh as x_i, $i = 1, \dots, N_I$ and the boundary nodes as x_j^B, $j = 1, \dots, N_B$. Denote ϕ_i as the scalar basis function that is continuous in Ω, piecewise quadratic in each element, taking value 1 at node i and zero on all other nodes. Let vectors $e_1 = (1, 0)^T$ and $e_2 = (0, 1)^T$. At an inner node x_i, there are two basis functions of velocity, which are $\phi_i e_1$ and $\phi_i e_2$. At a boundary node x_j^B, there is only one basis function, $\phi_j e_\tau$, where e_τ is the unit tangential vector at x_j^B.

The weak form of equations of (2.2)–(2.5) is finding $v \in V_h$ and $p \in Q_h$ such that for all $w \in V_h$ and $q \in Q_h$,

$$\int_\Omega \frac{\partial v}{\partial t} \cdot w + 2\int_\Omega \mathbb{D}(v) \cdot \mathbb{D}(w) + \int_\Gamma k(v \cdot \tau)(w \cdot \tau) - \int_\Omega p\nabla \cdot w = \int_\Gamma g(w \cdot \tau), \tag{A.12}$$

$$\int_\Omega q\nabla \cdot v = 0, \tag{A.13}$$

where $v = (v_1, v_2)^T$, $w = (w_1, w_2)^T$, $\mathbb{D}(v) \cdot \mathbb{D}(w) = \frac{1}{4}\sum_{i,j=1,2}(\partial_i v_j + \partial_j v_i)(\partial_i w_j + \partial_j w_i)$.

An iterative projection method with BDF2 time discretization is used to solve the velocity and pressure [34]. Denote the numerical solution at the time step t^s as (v^s, p^s). To obtain (v^{s+1}, p^{s+1}), we use the following iterations with index l. For $l = 0, 1, 2, \dots$, let $p^{s+1,0} = (2p^s - p^{s-1})$, and

$$\int_\Omega \frac{1.5\tilde{v}^{s+1,l+1} - 2v^s + 0.5v^{s-1}}{\Delta t} \cdot w + 2\int_\Omega \mathbb{D}(\tilde{v}^{s+1,l+1}) \cdot \mathbb{D}(w) + \int_\Gamma k(\tilde{v}^{s+1,l+1} \cdot \tau)(w \cdot \tau) = \int_\Omega p^{s+1,l}\nabla \cdot w + \int_\Gamma g(w \cdot \tau), \quad \forall w \in V_h, \tag{A.14}$$

$$\int_\Omega \nabla\phi^{l+1} \cdot \nabla q = -\frac{1}{\Delta t}\int_\Omega q(\nabla \cdot \tilde{v}^{s+1,l+1}), \quad \forall q \in Q_h, \tag{A.15}$$

$$\int_\Omega p^{s+1,l+1} q = \int_\Omega (p^{s+1,l} + 1.5\phi^{l+1} - \nabla \cdot \tilde{v}^{s+1,l+1})q, \quad \forall q \in Q_h, \tag{A.16}$$

$$\int_\Omega v^{s+1,l+1} \cdot w = \int_\Omega \tilde{v}^{s+1,l+1} \cdot w - \Delta t\phi^{l+1}(\nabla \cdot w), \quad \forall w \in V_h. \tag{A.17}$$

The stopping criterion for the iterations is chosen as when $\|p^{s+1,l+1} - p^{s+1,l}\|_{L^2(\Omega)} < \varepsilon_s$. When convergent, we let $(v^{s+1}, p^{s+1}) = (v^{s+1,l+1}, p^{s+1,l+1})$ and have the estimate

$$\left| \int_\Omega q(\nabla \cdot v^{s+1}) \right| < \varepsilon_s, \forall q \in Q_h. \tag{A.18}$$

The threshold ϵ_s is set as 10^{-10} in this work. Therefore, although the divergence of the numerical velocity is not pointwise zero, it is almost zero in the weak sense.

A.4. Transport equations: Discontinuous Galerkin method

A standard Runge–Kutta Discontinuous Galerkin (RKDG) scheme [42] is used to solve the scalar θ governed by the transport Eq. (2.1), and the adjoint quantity ρ from (2.15). Define the discontinuous finite element space

$$W_{h,M_{DG}}^{DG} = \{w_h \in P^{M_{DG}}(K), \forall K \subset \Omega_h\}, \tag{A.19}$$

where $P^{M_{DG}}(K)$ denotes the set of M_{DG}th degree polynomials in each triangle K of the discrete domain Ω_h. To ensure stability, a Courant–Fredrichs–Lewy (CFL) condition is used to determine the time step size Δt,

$$\|v\|_{\max} \cdot \frac{\Delta t}{h} \le \mathrm{CFL}_{L^2} \tag{A.20}$$

where the constant CFL_{L^2} for degree M_{DG} of polynomials is given in Table 2.2 of [42].

To show the idea, a first-order temporarily discretized numerical scheme is given as follows. Given the numerical solution $\theta^s \in W_{h,M_{DG}}^{DG}$ at time step t^s, we obtain $\theta^{s+1} \in W_{h,M_{DG}}^{DG}$ from

$$\int_K \frac{\theta^{s+1} - \theta^s}{\Delta t}\phi + \int_{e\subseteq\partial K}(v \cdot \hat{n})\theta^s\phi - \int_K \theta(v \cdot \nabla\phi) = \int_K \theta(\nabla \cdot v)\phi, \quad \forall\phi \in P^{M_{DG}}(K), \tag{A.21}$$

where $\hat{n}$ is the unit outward normal on edge e of K and $\hat{\theta}^s$ is the numerical flux. The Godunov flux (see [42] page 206) is used, i.e.,

$$\hat{\theta}^s|_{\partial K} = \begin{cases} \theta^s|_K & v \cdot \hat{n} > 0, \\ \theta^s|_{K+} & v \cdot \hat{n} < 0, \end{cases} \tag{A.22}$$

where $K+$ is the neighbor triangle that K bounds across the edge e. In practice, we use a second order TVD-RK scheme in time and quadratic DG approximations in space ($M_{DG} = 2$ in (A.19)), of which the details can be found in [42, page 190].

Table 9
Choices of basis functions and quadrature rule on a triangle for DG method.

M_{DG}, order of polynomial	0	1	2	3	4	5
$M_t = (M_{DG}+1)(M_{DG}+2)/2$, dimension of $P^{M_{DG}}(K)$	1	3	6	10	15	21
G, minimum number of quadrature points	1	3	6	10	15	21

A.4.1. Choices of basis functions of $P^{M_{DG}}(K)$ and quadrature rules

The basis functions of $P^{M_{DG}}(K)$, $M_{DG} \geq 0$, $K \subset \Omega_h$ are chosen as follows. Denote the center point of K as (x_0, y_0) and a generic basis function as $\phi_{i,j} = (x-x_0)^i (y-y_0)^j$, $i \geq 0, j \geq 0$, $i+j \leq M_{DG}$. There are $M_t = (M_{DG}+1)(M_{DG}+2)/2$ such basis functions, or $dim(P^{M_{DG}}(K)) = M_t$. For any smooth function $\theta(x,y)$, its representation $\theta_h \in P^{M_{DG}}(k)$ has the expression $\theta_h = \sum_{i,j\geq 0}^{i+j\leq M} \theta_{i,j}\phi_{i,j}$, where $\theta_{i,j} = \frac{1}{i!j!}\frac{\partial^{i+j}\theta(x_0,y_0)}{\partial x^i \partial y^j}$. Re-order these bases as $\psi_s = \phi_{i,j}$ where $s = (i+j)(i+j+1)/2 + (j+1)$, which is a one-to-one correspondence from the double-index set $\{(i,j) : i \geq 0, j \geq 0, i+j \leq M_{DG}\}$ to the single-index set $\{1, \ldots, M_t\}$.

The mass matrix $A_{M_t \times M_t}$ on each triangle K is

$$A_{i,j} = \int_K \psi_i(x,y)\psi_j(x,y)dxdy, \quad i,j = 1, \ldots, M_t. \tag{A.23}$$

Suppose the above integral is approximated by the following quadrature rule,

$$\int_K f(x,y) \approx \sum_{l=1}^{G} w_l f(x_l, y_l), \tag{A.24}$$

where all the weight $w_l > 0$. Denote the resulting matrix generated from the above quadrature rule as A^G. The next lemma provides a necessary condition to ensure the invertibility of A^G.

Lemma A.3. *For the matrix A^G to be invertible, the number of quadrature points in the triangular integral (A.24) which approximates (A.23) must be greater than or equal to the number of basis functions of $P^M(K)$, that is, $G \geq M_t$.*

Proof. For any $c \in \mathbb{R}^{M_t}$, $c^T A^G c = \sum_{l=1}^{G}\sum_{i,j=1}^{M_t} w_l c_i \psi_i(x_l,y_l)\psi_j(x_l,y_l)c_j$. Let $f_l = \sum_{i=1}^{M_t} c_i\psi_i(x_l,y_l)$. Then $c^T A^G c = \sum_{l=1}^{G} w_l f_l^2 \geq 0$ since $w_l > 0$. It is clear that the matrix A^G is symmetric and positive semi-definite. To be invertible, it requires that A^G is positive definite or $c^T A^G c = 0$ has only the zero solution $c = 0$. Because $w_l > 0$ for $l = 1, \ldots, G$, it leads to $f_l = 0$ for all l, i.e., $\sum_{i=1}^{M_t} \psi_i(x_l,y_l)c_i = 0$. This system has G linear equations and M_t variables (c_i). If $G < M_t$, then this system must have free variables and thus nonzero solutions.

Some choices of basis functions and quadrature rules are given in Table 9. In the implementations with $M_{DG} = 3$ or 4, a 16-point Gaussian quadrature rule on a triangle from [43] is used, which is exact for 8th degree polynomials. As for the line integral, a 16-point quadrature rule in [44] is used, which is exact for polynomials of degree ≤ 31. In the implementations with $M = 0$, 1, or 2, a 7-point Gaussian quadrature rule on a triangle is used, which is exact for 5th degree polynomials, and a 3-point quadrature rule is used for the line integral, which is exact for polynomials of degree ≤ 5.

A.5. A simple check of the numerical code for the solution of υ, θ and ρ

We make use of Proposition 2.1 to check the code that solves the velocity υ from the Stokes equations from given controls, evolves θ with υ from $t = 0$ to $t = T$, computes $\rho(T) = \Lambda^{-2}\theta(T)$, and transports $\rho(t)$ backward with υ from $t = T$ to $t = 0$. We set $\theta_0 = \sin(2\pi y)$ and choose control $g = 10\cos(2\omega)$ when $t \in [0, 0.5]$ and $g = 20\sin(2\omega)$ when $t \in [0.5, 1]$. The velocity υ is computed using the iterative projection scheme in Appendix A.3 and θ and ρ are solved by DGP2 ($M_{DG} = 2$) method in Appendix A.4. The test results are shown in Fig. 15, where

$$Mean_T = \frac{1}{T}\int_0^T \int_\Omega \rho(x,t)\theta(x,t)dxdt. \tag{A.25}$$

In this test, $T = 1$. The maximum error of $(\int_0^T \int_\Omega \rho^T(x,t)\theta(x,t)\,dx - Mean_T)$ over $t \in [0,1]$ is 1.05e−4 when $h = 0.1$, 3.15e−5 when $h = 0.05$, and 8.70e−6 when $h = 0.025$, which shows roughly second order convergence to zero when the mesh is refined. This partially verifies the code.

A.6. Radially symmetric steady flow in the unit disk when $g = 1$

In polar coordinates (r, φ), denote the velocity as $\upsilon = \upsilon_r \hat{e}_r + \upsilon_\varphi \hat{e}_\varphi$, where e_r and e_φ are unit vectors in the direction r and φ. The divergence free condition is $\nabla \cdot \upsilon = \frac{1}{r}\frac{\partial(r\upsilon_r)}{\partial r} + \frac{1}{r}\frac{\partial \upsilon_\varphi}{\partial \varphi} = 0$. Under the radial symmetry assumption, $\upsilon_r = \upsilon_r(r)$, $\upsilon_\varphi = \upsilon_\varphi(r)$, $p = p(r)$, and $\upsilon_r(0) = \upsilon_\varphi(0) = 0$. Thus, the divergence free condition becomes $\frac{1}{r}\frac{\partial(r\upsilon_r)}{\partial r} = 0$, which gives $\upsilon_r(r) = 0$ in the disk.

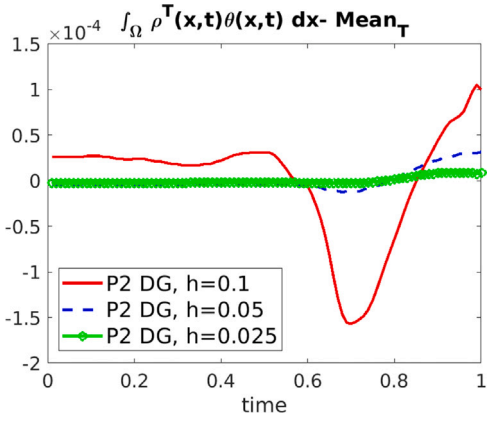

Fig. 15. A test for Proposition 2.1: $(\int_\Omega \rho_T(x,t)\theta_T(x,t)dx - Mean_T)$ over time. Initial value $\theta_0 = \sin(2\pi y)$, $g = 10\cos(2\omega)$ if $t \in [0, 0.5]$ and $g = 20\sin(2\omega)$ if $t \in [0.5, 1]$. $Mean_T$ is the mean value in time defined in (A.25).

In general, $\nabla v + (\nabla v)^T = \begin{pmatrix} 2\frac{\partial v_r}{\partial r} & \frac{1}{r}\frac{\partial v_r}{\partial \varphi} + \frac{\partial v_\varphi}{\partial r} - \frac{v_\varphi}{r} \\ \frac{1}{r}\frac{\partial v_r}{\partial \varphi} + \frac{\partial v_\varphi}{\partial r} - \frac{v_\varphi}{r} & \frac{2}{r}\frac{\partial v_\varphi}{\partial \varphi} + \frac{v_r}{r} \end{pmatrix}$. With radial symmetry and $v_r = 0$, the steady state momentum equations become $\frac{\partial p}{\partial r} = 0$ and $-\frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial (r v_\varphi)}{\partial r}\right) = 0$ when $0 < r < 1$. The Navier-slip boundary condition on the unit circle becomes $\frac{\partial v_\varphi}{\partial r} + (k-1)v_\varphi = g$. These three equations admit a unique solution: $v_\varphi = \frac{g}{k}r$ and p is a constant.

References

[1] V.S. Chakravarthy, J.M. Ottino, Mixing of two viscous fluids in a rectangular cavity, Chem. Eng. Sci. 51 (14) (1996) 3613–3622.
[2] A. Vikhansky, Enhancement of laminar mixing by optimal control methods, Chem. Eng. Sci. 57 (14) (2002) 2719–2725.
[3] E. Gouillart, O. Dauchot, B. Dubrulle, S. Roux, J.-L. Thiffeault, Slow decay of concentration variance due to no-slip walls in chaotic mixing, Phys. Rev. E 78 (2) (2008) 026211.
[4] E. Gouillart, N. Kuncio, O. Dauchot, B. Dubrulle, S. Roux, J.-L. Thiffeault, Walls inhibit chaotic mixing, Phys. Rev. Lett. 99 (11) (2007) 114501.
[5] E. Gouillart, J.-L. Thiffeault, O. Dauchot, Rotation shields chaotic mixing regions from no-slip walls, Phys. Rev. Lett. 104 (20) (2010) 204502.
[6] J.-L. Thiffeault, E. Gouillart, O. Dauchot, Moving walls accelerate mixing, Phys. Rev. E 84 (3) (2011) 036313.
[7] D. Ahmed, X. Mao, B.K. Juluri, et al., A fast microfluidic mixer based on acoustically driven sidewall-trapped microbubbles, Microfluid. Nanofluid. 7 (2009) 727.
[8] R. Nakamura, T. Katsunom, Y. Kishimoto, et al., A novel method for live imaging of human airway cilia using wheat germ agglutinin, Sci. Rep. 10 (2020) 14417.
[9] K.P.Y. Hui, R.H.H. Ching, S.K.H. Chan, et al., Tropism, replication competence, and innate immune responses of influenza virus: An analysis of human airway organoids and ex-vivo bronchus cultures, Lancet Respir. Med. 11 (2018) 846–854.
[10] Y. Li, X. Liu, Q. Huang, A.T. Ohta, T. Arai, Bubbles in microfluidics: An all-purpose tool for micromanipulation, Lab Chip 21 (2021) 1016–1035.
[11] T. ul Islam, Wang Y., I. Aggarwal, et al., Microscopic artificial cilia - a review, Lab Chip 22 (2022) 1650–1679.
[12] W. Supatto, S.E. Fraser, J. Vermot, An all-optical approach for probing microscopic flows in living embryos, Biophys. J. 95 (2008) L29.
[13] J.C. Nawroth, H. Guo, E. Koch, et al., Motile cilia create fluid-mechanical microhabitats for the active recruitment of the host microbiome, Proc. Natl. Acad. Sci. 114 (2017) 9510.
[14] S. Lukens, X. Yang, L. Fauci, Using Lagrangian coherent structures to analyze fluid mixing by cilia, Chaos 20 (2010) 017511.
[15] Y. Ding, J. Nawroth, M. McFall-Ngai, E. Kanso, Mixing and transport by ciliary carpets: A numerical study, J. Fluid Mech. 743 (2014) 124–140.
[16] S. Chateau, U. d'Ortona, S. Poncet, J. Favier, Transport and mixing induced by beating cilia in human airways, Front. Physiol. (2018) 161.
[17] H. Guo, H. Zhu, S. Veerapaneni, Simulating cilia-driven mixing and transport in complex geometries, Phys. Rev. Fluids 5 (2020) 053103.
[18] W. Hu, An approximating control design for optimal mixing by Stokes flows, Appl. Math. Optim. 82 (2020) 471–498.
[19] W. Hu, Boundary control for optimal mixing by Stokes flows, Appl. Math. Optim. 78 (1) (2018) 201–217.
[20] W. Hu, J. Wu, Boundary control for optimal mixing via Navier–Stokes flows, SIAM J. Control Optim. 56 (4) (2018) 2768–2801.
[21] W. Hu, J. Wu, An approximating approach for boundary control of optimal mixing via Navier–Stokes flows, J. Differential Equations 267 (10) (2019) 5809–5850.
[22] G. Alberti, G. Crippa, A. Mazzucato, Exponential self-similar mixing by incompressible flows, J. Amer. Math. Soc. 32 (2) (2019) 445–490.
[23] T.M. Elgindi, A. Zlatoš, Universal mixers in all dimensions, Adv. Math. 356 (2019) 106807.
[24] Z. Lin, J.-L. Thiffeault, C.R. Doering, Optimal stirring strategies for passive scalar mixing, J. Fluid Mech. 675 (2011) 465–476.
[25] E. Lunasin, Z. Lin, A. Novikov, A. Mazzucato, C.R. Doering, Optimal mixing and optimal stirring for fixed energy, fixed power, or fixed palenstrophy flows, J. Math. Phys. 53 (11) (2012) 115611.
[26] J.-L. Thiffeault, Using multiscale norms to quantify mixing and transport, Nonlinearity 25 (2) (2012) R1.
[27] Y. Yao, A. Zlatos, Mixing and un-mixing by incompressible flows, J. Eur. Math. Soc. 19 (7) (2017) 1911–1948.
[28] G. Crippa, R. Lucà, C. Schulze, Polynomial mixing under a certain stationary Euler flow, Physica D 394 (2019) 44–55.
[29] G. Iyer, A. Kiselev, X. Xu, Lower bounds on the mix norm of passive scalars advected by incompressible enstrophy-constrained flows, Nonlinearity 27 (5) (2014) 973.
[30] O. Gubanov, L. Cortelezzi, Towards the design of an optimal mixer, J. Fluid Mech. 651 (2010) 27–53.
[31] G. Mathew, I. Mezić, S. Grivopoulos, U. Vaidya, L. Petzold, Optimal control of mixing in Stokes fluid flows, J. Fluid Mech. 580 (2007) 261–281.
[32] W. Liu, Mixing enhancement by optimal flow advection, SIAM J. Control Optim. 47 (2) (2008) 624–638.
[33] C. Seis, Maximal mixing by incompressible fluid flows, Nonlinearity 26 (12) (2013) 3279.

[34] X. Zheng, K. Zhao, J. Wu, W. Hu, D. Du, Iterative projection method for unsteady Navier-Stokes equations with high Reynolds numbers, 2023, arXiv preprint arXiv:2304.07963.
[35] R. Glowinski, Y. Song, X. Yuan, H. Yue, Bilinear optimal control of an advection-reaction-diffusion system, SIAM Rev. 64 (2) (2022) 392–421.
[36] J.P. Kelliher, Navier–Stokes equations with Navier boundary conditions for a bounded domain in the plane, SIAM J. Math. Anal. 38 (1) (2006) 210–232.
[37] G. Mathew, I. Mezić, L. Petzold, A multiscale measure for mixing, Physica D 211 (1) (2005) 23–46.
[38] J.L. Lions, Optimal Control of Systems Governed By Partial Differential Equations, Springer-Verlag, Berlin Heidelberg, 1971.
[39] M. Hinze, R. Pinnau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, vol. 23, Springer Science & Business Media, 2008.
[40] I. Griva, S.G. Nash, A. Sofer, Linear and Nonlinear Optimization, SIAM, 2009.
[41] R.L. Burden, J.D. Faires, A.M. Burden, Numerical Analysis, tenth ed., Cengage Learning, 2016.
[42] B. Cockburn, C.-W. Shu, Runge-kutta discontinuous Galerkin methods for convection-dominated problems, J. Sci. Comput. 16 (3) (2001) 173–261.
[43] L. Zhang, T. Cui, H. Liu, A set of symmetric quadrature rules on triangles and tetrahedra, J. Comput. Math. 27 (2009) 89–96.
[44] P. Davis, P. Rabinowitz, Abscissas and weights for Gaussian quadratures of high order, J. Res. Natl. Bur. Stand. 56 (1956) 35–37, https://ia600701.us.archive.org/3/items/jresv56n1p35/jresv56n1p35_A1b.pdf.