INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
*Int. J. Numer. Meth. Fluids* 2011; **65**:383–404
Published online 9 October 2009 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/fld.2188

# Fast solvers for models of ICEO microfluidic flows

Robert R. Shuttleworth[1], Howard C. Elman[2,*,†], Kevin R. Long[3] and Jeremy A. Templeton[4]

[1]*Applied Mathematics and Scientific Computing Program and Center for Scientific Computation and Mathematical Modeling, University of Maryland, College Park, MD 20742, U.S.A.*
[2]*Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, U.S.A.*
[3]*Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 79049, U.S.A.*
[4]*Sandia National Laboratories, PO Box 969, MS 9409, Livermore, CA 94551, U.S.A.*

SUMMARY

We demonstrate the performance of a fast computational algorithm for modeling the design of a microfluidic mixing device. The device uses an electrokinetic process, *induced charge electroosmosis* (*J. Fluid Mech.* 2004; **509**), by which a flow through the device is driven by a set of polarizable obstacles in it. Its design is realized by manipulating the shape and orientation of the obstacles in order to maximize the amount of fluid mixing within the device. The computation entails the solution of a constrained optimization problem in which function evaluations require the numerical solution of a set of partial differential equations: a potential equation, the incompressible Navier–Stokes equations, and a mass-transport equation. The most expensive component of the function evaluation (which must be performed at every step of an iteration for the optimization) is the solution of the Navier–Stokes equations. We show that by using some new robust algorithms for this task (*SIAM J. Sci. Comput.* 2002; **24**:237–256; *J. Comput. Appl. Math.* 2001; **128**:261–279), based on certain preconditioners that take advantage of the structure of the linearized problem, this computation can be done efficiently. Using this computational strategy, in conjunction with a derivative-free pattern search algorithm for the optimization, applied to a finite element discretization of the problem, we are able to determine optimal configurations of microfluidic devices. Copyright © 2009 John Wiley & Sons, Ltd.

Received 2 February 2009; Revised 29 June 2009; Accepted 17 August 2009

KEY WORDS: micro-fluids; Navier–Stokes; incompressible flow; linear solvers; optimization; laminar flow

## 1. INTRODUCTION

Improvements in techniques for manufacturing devices at small length scales have created a growing interest in the construction of miniature devices for use in biomedical screening and chemical analysis. These *microfluidic devices* manipulate fluid flows over small length scales, between 10 and 100 μm, with a low fluid volume, and correspondingly low Reynolds number. This results in laminar flow of the type commonly found in blood samples, bacterial cell suspensions, or protein/antibody solutions. Methods for controlling and manipulating fluids at such length scales

*Correspondence to: Howard C. Elman, Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, U.S.A.
†E-mail: elman@cs.umd.edu

Contract/grant sponsor: DOE Office of Science MICS Program; contract/grant number: DEFG0204ER25619
Contract/grant sponsor: ASC Program at Sandia National Laboratories; contract/grant number: DE-AC04-94AL85000

are a key ingredient in this process [1]. However, robust strategies for pumping and mixing in microfluidic devices are in short supply. Although mixing is one of the most time-consuming steps in biological agent detection, research and development of microfluidic mixing systems is relatively new. In this paper, we develop an efficient numerical algorithm for modeling this process using Induced Charge Electro-osmosis (ICEO) [2]. Our goal is to use this model to determine an optimal mixing design for a microfluidic device by manipulating the shape of the obstructions in the flow domain.

In the course of modeling the mixing process, we need to compute the numerical solution of a collection of partial differential equations (PDEs): a potential equation, a mass-transport equation, and the incompressible Navier–Stokes equations. Solving the third of these is by the far the most complex and time consuming and one of our aims is to demonstrate the utility of some new solution algorithms for performing this task efficiently. Moreover, the systems of equations have on the order $10^5$–$10^8$ unknowns, and these sets of numerical computations must be performed for the function evaluations required at every step of an algorithm used to optimize the structure of the ICEO device. Thus, it is critical that the solutions are computed efficiently.

The methods we use to solve the algebraic systems for the incompressible Navier–Stokes equations are built from preconditioners using 'approximate commutator' methods [3]. These methods are based on the approximation of the Schur complement operator by a technique proposed by Kay *et al.* [4], Silvester *et al.* [5], and Elman *et al.* [6]. They use multilevel multigrid methods and in our particular case, algebraic multilevel methods (AMG) as building blocks for the linear solver.

The paper is organized as follows. Section 2 gives a brief description and justification of the physical motivation for modeling ICEO mixing devices. Section 3 describes the steps necessary to model ICEO flows. Section 4 describes the Navier–Stokes solver used in this problem. Section 5 provides a brief overview of the parallel implementation of the optimization process including the choices of non-linear and linear solvers. Details of the numerical experiments and the results of these experiments are described in Section 6. Concluding remarks are provided in Section 7.

## 2. HISTORICAL CONTEXT AND BACKGROUND

We are concerned with mixing chemical or biological samples with reagents for the detection of specific agents. Microfluidic mixing strategies can be divided into two general classes, passive (pressure/capillary) and active (electric/magnetic) mixing. Passive mixing, which occurs when liquids are forced through winding paths (baffles, turns, etc.), continually dilutes the sample as long as the process continues. Such pressure-driven flows are commonly used in microfluidic devices and can be very effective when the channel dimensions are not too small ($>10\,\mu$m). However, these methods scale poorly with miniaturization, disperse the sample, and do not offer local control of flow direction.

Active mixing does not suffer from these difficulties because an independent source of motion is used to mix liquids. Strategies for active mixing include production of recirculating flows by ultrasonic means or by electrokinetic instabilities [7]. The drawback of ultrasonic methods is that these strategies are only useful for large and bulky mixing platforms. Generating flows by electrokinetic instabilities requires different conductivities in the two liquids being mixed and large voltages. ICEO [2, 8] has advantages over these approaches because it has been shown to mix dyes in a few seconds [9] and scales well for smaller devices.

ICEO occurs when an electrical conductor is placed in a liquid with dissolved electrolytes in the presence of an electric field. Consider a cylindrical conductor immersed in a liquid in the presence of an electric field, as shown in Figure 1 (left). The conductor is free of current and is electrically floating so it becomes polarized, thus making the field within it zero. Then the charge on the surface of the conductor attracts counter ions in the surrounding liquid so an electric double layer that acts like a capacitor is formed adjacent to the conductor surface. The applied field acts on this ionic charge layer, which has been created by the field, causing the ions to move. The mobile ions move in response to the electric field, and the ions drag the surrounding fluid with



DOI: 10.1002/fld

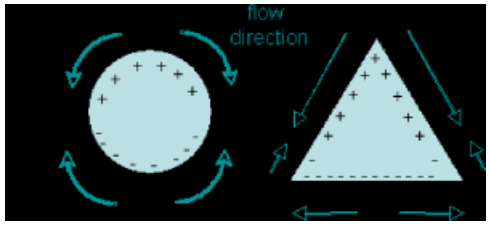


Figure 1. Double layer flows around a circular and triangular conductor as described in [9].

them by viscous forces. This produces an effective 'slip' velocity at the conductor surface which is proportional to the product of the electric field squared and the characteristic length of the conductor [2].

Both Adjari [10] and Ramos *et al.* [11] observed that electric double layers could form on charged electrodes and induce fluid motion. The theory that the polarization of conducting objects can lead to non-linear electro-osmotic flows for spherical polarizable colloids was first described by Murtsovkin and colleagues in [12, 13]. Squires and Bazant [2] extended this description to general shapes and derived analytical solutions for spheres and cylinders. They characterized both time-independent and dependent non-linear ICEO flows and suggested microfluidic pumping and mixing devices using symmetric bodies and electrodes. Subsequently they derived asymptotic solutions for symmetric shapes with asymmetric perturbations [14] and suggested microfluidic devices with asymmetric shapes [8]. ICEO has been experimentally observed in [9, 15, 16].

This ICEO process for mixing fluids is produced by placement of one or more electrically floating obstructions in a microchannel which are subjected to an applied voltage to create the electrokinetic motion. The shape and layout of the obstructions or posts are designed to generate streamlines that cross between the two fluids being mixed, effectively stretching their interface so diffusion can act more quickly. ICEO provides a bounty of desirable effects, including generating velocities proportional to the square of the voltage, scaling well to smaller devices, and enabling a range of possible configurations for different applications. Moreover, the posts can be charged to a fixed potential, allowing more control over the flow field, although this is more costly. Additional advantages of ICEO are that flows can be made to recirculate within a given volume, reducing dispersion [9], and that time-dependent electric fields can be used to create chaotic streamlines [17, 18]. Since we use background (Navier–Stokes) flow in our models, the motion in our flow field through the ICEO vortices acts like time modulation.

The ICEO flow pattern depends on the shape of the conductor(s). A symmetric shape typically results in symmetric recirculating flows surrounding the conductor. For a single cylindrical conductor such as that shown on the left of Figure 1, the flow will be composed of four symmetric vortices. If there are many of these conductors a periodic flow pattern is produced. An asymmetric shape, such as the triangle shown on the right of Figure 1, creates a non-symmetric flow which can transport fluid between the top and bottom halves to promote mixing [19]. Similar designs of ICEO flows around polarizable corners for triangular designs have been studied in [20, 21], whereas other recent work on general motion for elongated spherical shapes can be found in [22]. One initial configuration we have investigated can be found in Figure 2.

The objective of this work is to generate and solve numerically a model of an ICEO-driven microfluidic mixing device for combining a sample fluid with a reagent, building upon previous experimental, theoretical, and computational studies of ICEO flows and their applications. This device could be useful as part of a miniaturized biological detector. However, the best shapes and topology for the conductors that generate the ICEO flow is currently unknown. Our goal is to investigate this issue by solving a shape optimization problem to maximize the mixing of two fluids by manipulating the shape and topology of the charged region. Shape optimization has been applied to microfluidics in [9, 23, 24]. At each step of the optimization algorithm, a sequence of computationally expensive fluid problems must be solved requiring scalable linear solvers to effectively solve these optimization problems.



DOI: 10.1002/fld

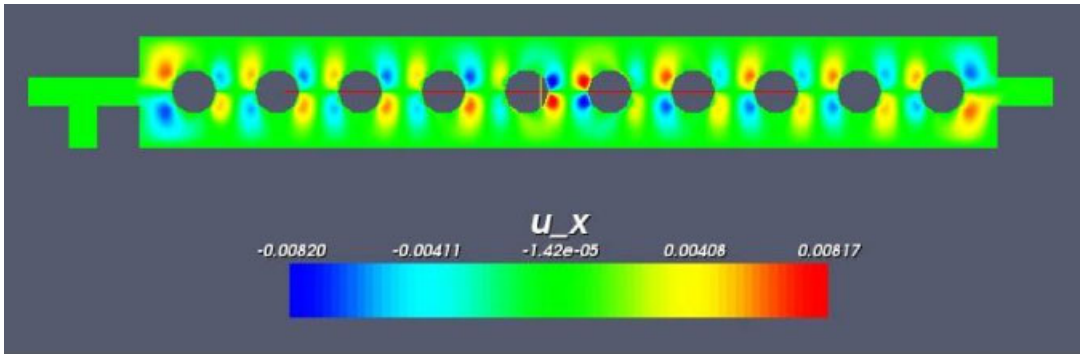


Figure 2. Sample initial domain for the multiple cylinder problem, with plot of the initial horizontal velocity.

## 3. MODEL DESCRIPTION

A finite element model was used to calculate the electric field, the ICEO flow as described in [2, 16], and the mass transport for a multispecies liquid. The DC field is assumed to be in a liquid with neutral charge. Under these conditions the electric field is governed by Laplace's equation,

$$E = \nabla^2 \phi = 0 \tag{1}$$

where $\phi$ is the electric potential and $E$ is the calculated electric field. The boundary conditions for (1) are Neumann conditions (zero normal gradient of the potential) at the channel boundaries and Dirichlet conditions (specified potential of 0.05) at the electrodes. Note that the insulating boundary condition applied on the surfaces of the posts is the same as that for the channel walls. The metallized posts are assumed to be completely shielded from the field by the double layer.

The electric field, $E$, induces a flow in the device, which is modeled by the incompressible Navier–Stokes equations

$$-v\nabla^2 \mathbf{u} + (\mathbf{u} \cdot \text{grad})\mathbf{u} + \text{grad}\, p = \mathbf{f} \tag{2}$$

$$-\text{div}\, \mathbf{u} = 0 \tag{3}$$

in $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) and used to calculate momentum transport. Here $\mathbf{u}$ is the fluid velocity, $p$ represents the hydrodynamic pressure, $v$ the kinematic viscosity, and $\mathbf{f}$ the body forces. No-slip velocity boundary conditions were used on all channel surfaces on $\partial\Omega$ except the metallized post surfaces, for which the boundary conditions, determined by $E$, are

$$\mathbf{u} = \frac{\varepsilon \zeta E_t}{\mu} \tag{4}$$

where $\varepsilon$ is the fluid permittivity, $\zeta$ is the potential drop across the electrical double layer, $E_t$ is the tangential electric field obtained from solving (1), and $\mu$ is the fluid viscosity [2]. For low concentration and low voltage ICEO flows, the flow is driven by the velocity boundary conditions along the posts, so the body forces, $\mathbf{f}$, are zero [2]. The above relationship is valid in the Debeye–Huckel limit of low surface charge (in practice, at low voltages and concentrations) in which no Stern layer forms and the diffuse layer is modeled by an exponentially decaying charge concentration. While practical devices may operate outside of these regimes, more complex models that can partially account for the Stern layer have demonstrated that the flow topology is largely independent of this boundary condition [9, 16], so it is reasonable to expect shapes generated by optimization using these boundary conditions would still perform well in actual devices. The difference would lie in the mixing time scale (i.e. the time taken to perform the mixing from an unmixed initial condition) and using boundary conditions that would completely describe this process is beyond the scope of this work.



DOI: 10.1002/fld

Once the velocities are obtained from the Navier–Stokes equations, the mass fraction of the solute is determined by the mass transport (or advection-diffusion) equation,

$$\mathbf{u}\cdot\nabla m = D\nabla^2 m \tag{5}$$

where $m$ is the mass fraction of solute and $D$ is the diffusivity.

The mass-transport equation in (5) is a useful formula to model mixing because it corresponds to a mass transfer process that occurs through a combination of convection and diffusion. The fluids of interest here are liquids, where diffusive mass transport is very slow over the distances typical of microchannels. Thus, convective transport is used to stretch and fold the liquids, that is, to increase interfacial area between the two liquid volumes and to reduce the distances over which diffusion must occur. We chose the diffusivity coefficient, $D = 1.8\times 10^{-9}\mathrm{cm}^2/\mathrm{s}$, which represents $\sim 3\,\mu\mathrm{m}$ particles in an aqueous solution. This value was used in [9]. It corresponds to a relatively small diffusivity constant, so that the problem is convection-dominated. A (mildly diffusive) model of this type creates a challenge for mixing and makes this a good test case for modeling a mixing device. A small value of $D$ results in a large ($\sim 10^5$) Peclet number, $Pe = uL/D$, where $L$ is the characteristic length scale of the device, so much of the mass transport needed for mixing occurs by advection. For the boundary conditions in this equation, we use Neumann zero flux conditions on the solid surfaces and Dirichlet conditions of 1 on one inflow boundary and 0 on the other inflow boundary.

A mixing metric, defined in [9], was used to quantify the extent of mixing based on the calculated results,

$$M = \frac{\int (m-\bar{m})^2\,\mathrm{d}V}{V} \tag{6}$$

where $\bar{m}$ is the average concentration of solute in the liquid mixture and the integral is over the volume, $V$, of the mixing domain. The initial value of this metric depends on the degree of segregation at the beginning of the mixing process and after the loading process for our initial configuration. As the shape of the obstructions are changed in the course of the optimization, the metric decreases. If perfect mixing is approached, the metric is zero.

Our goal is to determine an optimal geometry for mixing by varying the shape of the electrically charged posts. The shape of the posts are defined as functions parameterized by $N$ design variables, $d$. It is necessary to introduce constraints on the design parameters in order to avoid degenerate post shapes. We use simple bound constraints $L_i \leqslant d_i \leqslant U_i$. Both the choice of bounds and function parameterizations for different initial configurations are described further in Section 6.

The velocity field $\mathbf{u}$ depends on the design variables implicity through the boundary conditions for the Navier–Stokes equations. We consider the mixing metric $M$ to be an implicit function of the design variables, and write the problem as

$$\min M(d) \quad \text{s.t. } L_i \leqslant d_i \leqslant U_i \tag{7}$$

The algorithm used to solve this optimization problem is asynchronous parallel pattern search (APPS), described further in Section 5 where we discuss the software used in this study.

To compute $M(d)$ at each iteration of the optimization algorithm it is necessary to compute the velocity field, $\mathbf{u}(d)$, by solving problems (1), (2)–(3), and (5). Typically, hundreds of such solves are necessary to be able to solve these equations efficiently, motivating our use of fast solvers.

Finally, we note that the optimization problem is not convex and that multiple minima are possible. The APPS algorithm finds local minima, and different initial configurations may result in different locally optimal shapes. As will be seen in Section 6 below that is the case for this problem. To find a global minimum it would be necessary to embed the present problem in an outer global optimization algorithm such as stochastic tunneling [25]. We expect the change of optimization algorithm to have little impact on the fast computational algorithms that are the focus of this study.



DOI: 10.1002/fld

## 4. THE PRESSURE CONVECTION–DIFFUSION NAVIER–STOKES PRECONDITIONER

In the course of the optimization process (i.e. solving a series of problems (1), (2), (3), and (5)), the dominant cost in terms of CPU time is in solving the Navier–Stokes equations. We use a solution algorithm with convergence rates independent of the mesh size that we describe in this section.

We solve the nonlinear systems by Picard iteration, which is derived by lagging the convection coefficient in the quadratic term, $(\mathbf{u}\cdot\mathrm{grad})\mathbf{u}$. For a low Reynolds number flow such as this one, Picard's method is an adequate choice of a non-linear solver. This procedure begins with an initial guess $\mathbf{u}^{(0)}$ for the velocities and $p^{(0)}$ for the pressure, and updates to the velocities and pressures are computed by solving the *Oseen equations*

$$\begin{aligned} -\nu\nabla^2(\Delta\mathbf{u}^{(k)})+(\mathbf{u}^{(k)}\cdot\mathrm{grad})\Delta\mathbf{u}^{(k)}+\mathrm{grad}\,\Delta p^{(k)} &= \mathbf{f}-(-\nu\nabla^2(\mathbf{u}^{(k)})+(\mathbf{u}^{(k)}\cdot\mathrm{grad})\mathbf{u}^{(k)}+\mathrm{grad}\,p^{(k)}) \\ -\mathrm{div}\,\Delta\mathbf{u}^{(k)} &= \mathrm{div}\,\mathbf{u}^{(k)} \end{aligned} \tag{8}$$

The iterated sequence is determined by $\mathbf{u}^{(k+1)}=\mathbf{u}^{(k)}+\Delta\mathbf{u}^{(k)}$ and $p^{(k+1)}=p^{(k)}+\Delta p^{(k)}$. The discrete linear system has the form

$$\begin{pmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{pmatrix}\begin{pmatrix} \Delta\mathbf{u}^{\mathbf{k}} \\ \Delta p^{k} \end{pmatrix}=\begin{pmatrix} \mathbf{f}_{\mathbf{u}}^{\mathbf{k}} \\ f_p^k \end{pmatrix} \tag{9}$$

where $F$ is a discrete convection–diffusion operator, $B^{\mathrm{T}}$ is the discrete gradient operator, and $B$ is the discrete divergence operator. The right-hand side vector, $(\mathbf{f}_{\mathbf{u}}, f_p)^{\mathrm{T}}$, contains, respectively, the non-linear residual for the momentum and continuity equations.

The strategies we employ for solving (9) are derived from the LDU block factorization of this coefficient matrix where the diagonal ($D$) and upper triangular ($U$) factors are grouped together,

$$\begin{pmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{pmatrix}=\begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix}\begin{pmatrix} F & B^{\mathrm{T}} \\ 0 & -S \end{pmatrix} \tag{10}$$

and

$$S=BF^{-1}B^{\mathrm{T}} \tag{11}$$

is the Schur complement. We note that there is a similarity between the factorization (10) and methods developed for evolutionary problems in [26–28]. Connections among these approaches are discussed in [29] and [3, p. 376]. For large-scale computations, the Schur complement is a dense matrix, so its not feasible to use it in computations. For our preconditioner, we only use the upper triangular factor of (10), and replace the Schur complement $S$ by some approximation $\hat{S}$ (to be specified later).

We motivate this strategy by examining the computational issues associated with applying this upper triangular preconditioner in a Krylov subspace iteration. At each step, the application of the action of the inverse of this operator to a vector is needed. By expressing this operation in factored form,

$$\begin{pmatrix} F & B^{\mathrm{T}} \\ 0 & -S \end{pmatrix}^{-1}=\begin{pmatrix} F^{-1} & 0 \\ 0 & I \end{pmatrix}\begin{pmatrix} I & -B^{\mathrm{T}} \\ 0 & I \end{pmatrix}\begin{pmatrix} I & 0 \\ 0 & -S^{-1} \end{pmatrix}$$

two potentially difficult operations can be seen: $S^{-1}$ must be applied to a vector in the discrete pressure space and $F^{-1}$ must be applied to a vector in the discrete velocity space. The application of $F^{-1}$ can be performed relatively cheaply using an iterative technique, such as multigrid. However, applying $S^{-1}$ to a vector is too expensive. An effective preconditioner can be built by replacing this operation with an inexpensive approximation. We discuss the pressure convection–diffusion (P–CD) preconditioners where the basic idea hinges on the notion of an approximate commutator. Consider a convection–diffusion operator of the form

$$(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad})) \tag{12}$$



DOI: 10.1002/fld

When **w** is an approximation to the velocity obtained from the previous nonlinear step, (12) is an Oseen linearization of the non-linear term in (2). Suppose there is an analogous operator defined on the pressure space,

$$(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad}))_p$$

where the subscript $p$ here and below is intended to emphasize that operators are defined on the pressure space. Consider the commutator of these operators with the gradient:

$$\varepsilon=(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad}))\nabla-\nabla(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad}))_p \tag{13}$$

Supposing that $\varepsilon$ is small, multiplication on both sides of (13) by the divergence operator gives

$$\nabla^2(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad}))_p^{-1}\approx\nabla\cdot(\nu\nabla^2+(\mathbf{w}\cdot\mathrm{grad}))^{-1}\nabla \tag{14}$$

In discrete form, using finite elements, this usually takes the form

$$(Q_p^{-1}A_p)(Q_p^{-1}F_p)^{-1}\approx(Q_p^{-1}B)(Q_v^{-1}F)^{-1}(Q_v^{-1}B^{\mathrm{T}})$$

$$A_pF_p^{-1}Q_p\approx(BF^{-1}B^T)$$

where here $F$ represents a discrete convection–diffusion operator on the velocity space, $F_p$ is the discrete convection–diffusion operator defined on the pressure space, $A_p$ is a discrete Laplacian operator, $Q_v$ the velocity mass matrix, and $Q_p$ is a pressure mass matrix (or a lumped version of it). This suggests the approximation for the Schur complement

$$S\approx\hat{S}=A_pF_p^{-1}Q_p \tag{15}$$

for a stable finite element discretization. A similar approximation can be made for stabilized finite element discretizations [3, 29].

Applying the action of the inverse of $A_pF_p^{-1}Q_p$ to a vector requires solving a system of equations with a discrete Laplacian operator, then multiplication by the matrix $F_p$, and solving a system of equations with the pressure mass matrix. In practice, $Q_p$ can be replaced by its lumped approximation with little deterioration of effectiveness. Both the convection–diffusion system, $F$, and the Laplace system, $A_p$, can also be handled using multigrid with little deterioration of effectiveness. Considerable evidence for two and three-dimensional problems indicates that this preconditioning strategy is effective, leading to convergence rates that are independent of mesh size and mildly dependent on Reynolds numbers for steady flow problems [4, 5, 30, 31]. A proof that convergence rates are independent of the mesh is given in [32]. For this microfluidic problem, this new methodology enables the efficient solution of the ICEO model.

## 5. IMPLEMENTATION AND TESTING ENVIRONMENT

We have modeled the ICEO mixing process using Sundance, a finite element code developed at Sandia National Laboratory [33]. To minimize the objective function we use APPSPACK, which is an Asynchronous Parallel Pattern Search code also developed at Sandia National Laboratory. We describe both Sundance and APPSPACK in this section.

At each step of the optimization loop we need to perform a series of computations. Given the new set of design variables, which determine the domain $\Omega$, we automatically generate a mesh on $\Omega$ from a template. We use that mesh to solve a series of problems to model the ICEO flow and the mixing process. We generate the mesh using the software package CUBIT, which is developed at Sandia National Laboratory [34]. The unstructured meshes use triangular elements with an extra level of refinement around the conducting surfaces. This is done to accurately capture the wall-parallel flow and effect of the post on the potential field. Then we model the ICEO flow, by solving a potential equation, (1), which we use to implement a slip velocity boundary condition, (4), for the Navier–Stokes Equations (2)–(3). The calculated velocity value from the solution of the



DOI: 10.1002/fld

Navier–Stokes equations is used in the mass-transport equation, (5). The mass fraction, calculated from the mass-transport equation, is used to evaluate the mixing metric, (6), which is the value we want to minimize. These are the major calculations required at each step of the optimization algorithm. In the remainder of this section we describe the discretization details for each equation, our solver choice for each of the discrete systems of equations, and the software used for modeling the ICEO optimization process.

We discretize the *potential equation* using piecewise quadratic, $P_2$, finite elements integrated with second-order Gaussian quadrature. For solving the linear system resulting from the discretization of the potential equation we use conjugate gradient (CG) preconditioned with two levels of algebraic multigrid. The smoother for this problem is an incomplete *LU* factorization. For the coarsest level in the multigrid scheme, we used a direct *LU* solve. We terminate this iteration when the residual is reduced by a factor of $10^{-10}$, that is,

$$\|b - A\phi\| \leqslant 10^{-10}\|b\| \tag{16}$$

We discretize the *incompressible Navier–Stokes equations* using Taylor–Hood $P_2 - P_1$ finite elements with fourth-order Gaussian quadrature [3]. This is a div-stable finite element discretization, so no pressure stabilization is required. Moreover, these problems have Reynolds numbers on the order of 1, and stabilization of the transport term is also not needed. The non-linear system is solved by Picard's method where the structure of a two-dimensional steady version of $F$ is a $2 \times 2$ block matrix consisting of a discrete version of the operator

$$\begin{pmatrix} -\nu\Delta + u^{(k)} \cdot \nabla & 0 \\ 0 & -\nu\Delta + u^{(k)} \cdot \nabla \end{pmatrix} \tag{17}$$

where $u^{(k)}$ is a velocity value from a previous iteration. We terminate the non-linear iteration when the relative error in the residual is $10^{-4}$, that is,

$$\left\| \begin{pmatrix} \mathbf{f_u} - (F(\mathbf{u})\mathbf{u} + B^{\mathrm{T}} p) \\ f_p - B\mathbf{u} \end{pmatrix} \right\| \leqslant 10^{-4} \left\| \begin{pmatrix} \mathbf{f_u} \\ f_p \end{pmatrix} \right\| \tag{18}$$

At each step of the non-linear iteration, we terminate the linear iteration with the Oseen system, when the residual is reduced be a factor of $10^{-5}$, that is,

$$\left\| \begin{pmatrix} \mathbf{f_u^k} \\ f_p^k \end{pmatrix} - \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{\Delta u^k} \\ \Delta p^k \end{pmatrix} \right\| \leqslant 10^{-5} \left\| \begin{pmatrix} \mathbf{f_u^k} \\ f_p^k \end{pmatrix} \right\| \tag{19}$$

with zero initial guess. We solve the resulting linear system using GMRES with a Krylov subspace size of 300 and a maximum of 600 iterations, preconditioned with the pressure convection–diffusion preconditioner. We described this method in Section 4 and have found it to work well on some realistic benchmark problems in [3, 29]. This method is scalable, mesh independent and is built using algebraic multigrid for its core operations. This makes this strategy straightforward to construct and apply. Moreover, this strategy is robust to grids and grid spacing, so it is advantageous for a problem like this one where the grid is automatically generated with parameters from the optimization code.

The operators $F_p$, $A_p$, and $Q_p$ required by the pressure convection–diffusion strategy are generated by the application code, Sundance. For the pressure convection–diffusion preconditioner, we solve the subsidiary pressure Poisson type and convection–diffusion subproblems to a tolerance of $10^{-5}$, that is, this iteration for the convection–diffusion problem is terminated when

$$\|\mathbf{y} - F\mathbf{u}\| \leqslant 10^{-5}\|\mathbf{y}\| \tag{20}$$

For this system, we use GMRES preconditioned with four levels of smoothed aggregation algebraic multigrid, and for the pressure Poisson problem (with coefficient matrix $A_p$), we use CG preconditioned with four levels of smoothed aggregation algebraic multigrid. For both the



DOI: 10.1002/fld

convection–diffusion and pressure Poisson problem, a traditional point GS smoother is used for the smoothing operations. For the coarsest level in the multigrid scheme we used a direct *LU* solve.

We discretize the *mass-transport equation* (5) using $P_2$ finite elements with fourth-order Gaussian quadrature. For solving (5), we use GMRES preconditioned with three levels of smoothed aggregation algebraic multigrid. The smoother at the finest two levels is an incomplete *LU* factorization. On the coarsest level, we use a direct *LU* solve. We terminate this iteration when the residual is reduced by a factor of $10^{-5}$, that is,

$$\|b - Am\| \leqslant 10^{-5}\|b\| \tag{21}$$

For the optimization loop, where we want to minimize the objective function found in (7) and determine the optimal mixing strategy for a microfluidic device by manipulating the shape of the obstruction we use APPSPACK, which is a derivative-free Asynchronous Parallel Pattern Search code developed at Sandia National Laboratory. This code minimizes the objective function by asynchronous parallel Generating Set Search (GSS), which is an extension of pattern search to handle linear constraints. To prevent false convergence to suboptimal points, GSS methods use a core set of search directions that conform to the local geometry of the feasible region, permitting tangential movement along nearby constraints. A bound on optimality conditions is derived in terms of the maximum step size and convergence is determined when the step size for each search direction drops below a user specified tolerance. The ability to perform computationally expensive function evaluations asynchronously in parallel can dramatically reduce solve time and CPU inefficiencies due to load imbalance. APPSPACK is written in *C++* and uses MPI for parallelism. Our approach for using APPSPACK to solve optimization problems is that only function values are required for the optimization, so it can be applied easily. We have a small number of design variables (i.e. $n \leqslant 100$), but expensive objective function evaluations. Parallelism is achieved by assigning the individual function evaluations to different processors. The asynchrony enables better load balancing.

APPSPACK can solve optimization problems of the basic form

$$\min \quad M(d) \tag{22}$$

$$\text{s.t.} \quad c_L \leqslant A_{\mathrm{I}} d \leqslant c_u \tag{23}$$

$$A_E d = b \tag{24}$$

$$l \leqslant d \leqslant u \tag{25}$$

where $M(d)$ is the objective function, the inequality constraints are denoted by the matrix $A_{\mathrm{I}}$ and the upper and lower bounds by $c_{\mathrm{L}}$ and $c_{\mathrm{U}}$, respectively. The equality constraints are denoted by the matrix $A_{\mathrm{E}}$ and the right-hand side, $b$. Finally, $l$ and $u$ denote lower and upper bounds on the component variables [35, 36]. For our problem, we only use the linear constraints found in (25), which we describe further in Section 6. The objective function, $M(d)$, is the expression found in (6). The evaluation of this expression requires the solution of (1), (2)–(3), and (5), which all depend on the shape and orientation of the charged posts that are varied by the optimization algorithm. We have a non-linear constraint that is not directly handled by APPSPACK. This constraint is the requirement that the shape being meshed by Cubit is realistic. If Cubit successfully meshes the shape, then we evaluate the function and solve the ICEO flow. If Cubit fails to mesh the shape, then we return a large value to APPSPACK.

We use Sundance [33] for the finite element discretization, which is a tool developed at Sandia National Laboratory for specifying, building, and developing finite element solutions of PDEs. It uses automatic differentiation for symbolic objects, which allows the user to create differentiable simulations for use in optimization problems. Another feature of Sundance is that it allows a user to abstractly code a finite element problem, while providing a set of components with which the user can set up, describe, and solve a problem without worrying about bookkeeping details. This approach allows a high degree of flexibility in the formulation, design, discretization, and solution of a problem [33].



DOI: 10.1002/fld

Our implementation of the pressure convection–diffusion preconditioner and the other solvers for the discrete systems of equations uses *Trilinos* [37], a software environment developed at Sandia National Laboratories for implementing parallel solution algorithms using a collection of object-oriented software packages for large-scale, parallel multiphysics simulations. The main Trilinos components we use are Meros, Epetra, TSF/Thyra, AztecOO, and ML. *Meros* provides scalable block preconditioning for problems with coupled simultaneous solution variables. The pressure convection–diffusion preconditioner studied here is implemented in this package. *Epetra* provides the fundamental routines and operations needed for serial and parallel linear algebra libraries. Epetra also facilitates matrix construction on parallel distributed machines. *TSF/Thyra* provides an abstract interface to other Trilinos packages. The *AztecOO* package is a massively parallel iterative solver library for sparse linear systems. It supplies all of the Krylov methods used in solving (9), the $F$, and Schur complement approximation subsystems. We use the multilevel algebraic multigrid preconditioning package, *ML* with AztecOO to solve the potential equation system, the mass-transport system, as well as the subsidiary systems required for the preconditioner for (9).

We conclude this section with some comments on the relation between this work and other, related, approaches. First, we note that problems of the type considered here can be solved using commercial software packages such as COMSOL. One of our main aims was to explore a specific new class of algorithms for performing the function evaluations ($M(d)$ of (22)) needed to perform the optimization. The objective function, $M(d)$, is the expression found in (6) (where it should be understood that $m$ of (6) depends implicitly on the design variables $d$). This could in principal be done using COMSOL. Sundance, or more generally, the complete suite of Sandia software available to us, allowed us a great deal of flexibility in designing and modifying our solution strategy. This includes specifying the problem, gridding, and development of the solution algorithm. COMSOL uses Matlab's solvers, which are not specifically tailored for the problems (i.e. Poisson, convection–diffusion and Navier–Stokes equations) embedded in the function evaluations. These will be significantly less efficient for large-scale problems than the Trilinos-implemented solvers we are using here. In addition, COMSOL uses a derivative-based, primarily serial algorithm, SNOPT [38], for optimization. We do not have derivatives available for the optimization strategy and, moreover, the pattern search algorithm in APPSPACK that we used is naturally parallelizable and very robust. A direct comparison between the two approaches for optimization is beyond the scope of this project, but the technique we are studying offers the advantage of very natural parallelism, and it avoids the computational overhead associated with computing or estimating gradients (of the objective function $M(d)$) for a gradient-based optimization strategy.

In Section 6, we include details on the optimization process and choice of objective function, and include a few sample meshes and numerical results that were created in the course of the optimization loop. The results were obtained in parallel on Sandia's Institutional Computing Cluster (ICC) using 8–100 processors per run. Each of this cluster's compute nodes is dual Intel 3.6 GHz Xenon processors with 2 GB of RAM.

## 6. SIMULATION AND NUMERICAL RESULTS

Our goal is to optimize the shape of the microfluidic mixing device to maximize the amount of mixing being done in the channel. We have tested two different initial configurations consisting of circular posts (Figure 2) and alternating triangular posts as described in [9]. All of these designs use the mixing metric described in (6). We have also tested a continuous flow mixer, which we describe in Section 6.3, where the flow field is driven by both ICEO and an inflow boundary condition.

In Section 3, we described the steps needed to solve our optimization problem. In this section, we show a variety of flow fields obtained from various steps of the optimization loop and include the value of the mixing metric to show the quality of mixing for that particular mesh. We show the performance of the solvers with tabulated listings of iteration counts and CPU times for the various steps of the computation together with other costs such as mesh generation and matrix assembly.



DOI: 10.1002/fld

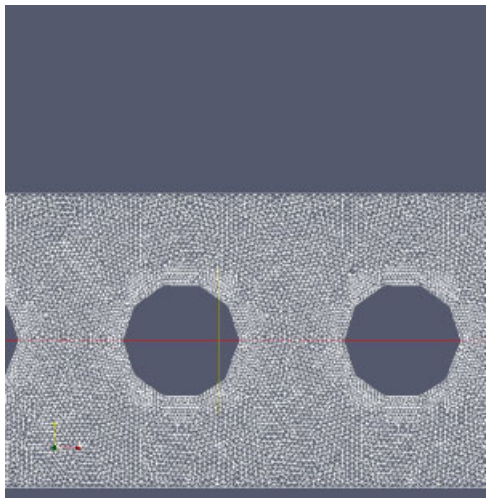

Figure 3. Sample mesh for the multiple cylinder domain.

The solver for the Navier–Stokes component of the ICEO flow was GMRES preconditioned with the pressure convection–diffusion preconditioner. This method generated scalable results in other applied settings [29] and we see similar trends when applying this technology to this problem. We validated our workflow by calculating the resultant ICEO flow for a model of a single post in a uniform field. Qualitatively the flow structure (four vortex structures emerging from the main post) between our calculated flow and the solution of a similar model in [2] matched. Quantitatively, the calculated velocity of this single post model problem matched (to three significant digits) the velocity of a single post model computed using the computational simulation tools mentioned in [9]. Note that the computed velocity is able to accurately predict the size of ICEO flow features, which is important because ICEO drives our mixing metric and helps determine the quality of the mixing performed in a device. In this sample problem, we also verified that the velocity computed by our code recovers the linear ICEO model that was implemented in it since the computed velocity depends on $\nabla^2\phi$.

### *6.1. Circle initial configuration*

For our first configuration, we begin with 10 circular posts. We use the objective function (6) constrained to 38 design variables. We parameterize each post as a set of piecewise line segments that connect 10 points, as in Figure 3. Each of these points is characterized in polar coordinates by a distance from a reference point of the post together with an angle with respect to the horizontal axis of our system. This results in 20 design variables. In the initial configuration, the reference points are the centers of the circular posts, which lie on a common horizontal line, and the vertical coordinate of each reference point is fixed throughout the simulation. The 10 posts are required to have the same shape, and each post other than the leftmost one is offset by a distance from its reference point to the reference point for the post to its left and rotated by an angle. This gives 18 more variables. The 38 variables are linearly constrained (25) as follows. The angles for points defining the posts are constrained to be between 0 and 360°, and the radii are constrained to have a value between 0.001 and 0.005 μm. Likewise, the rotation angle is constrained by 0 and 360°, and the linear constraint for the offset distance from one reference point to that of its neighbor to the left is bounded between 0.01 and 0.02 μm. Each corner is smoothed to a radius of 5 μm which allows the points to be collocated, helps with mesh generation, and is a tolerance to permit an acceptable physically manufacturable device. Note that the height of the domain is fixed at 0.01 μm, while the length of the domain is variable because the distance from the inlet to the first post and outlet to the last post is held constant at 0.005 μm. Therefore, the domain can expand or contract in the streamwise direction (horizontally), but not vertically.



DOI: 10.1002/fld

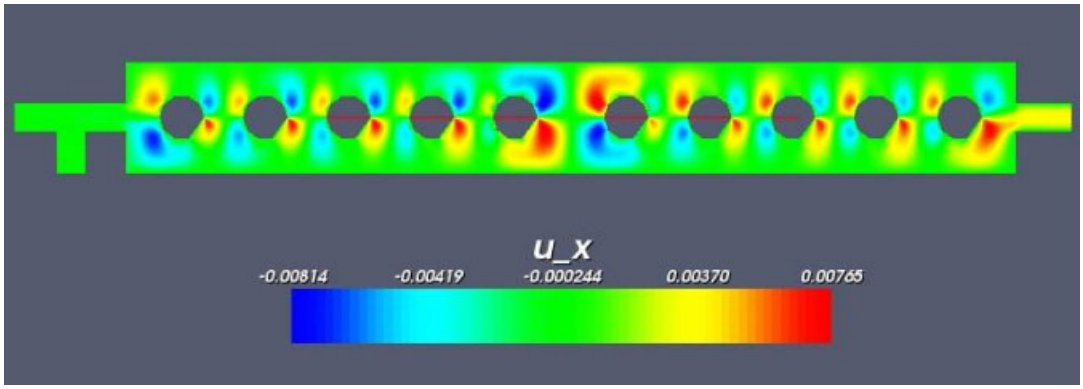


Figure 4. Preliminary design with mixing value of 0.032451.

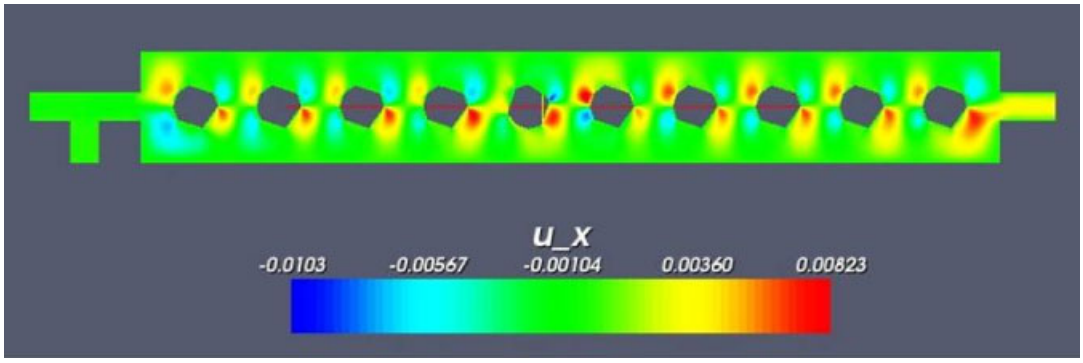


Figure 5. Intermediate design with mixing value of 0.0249871.

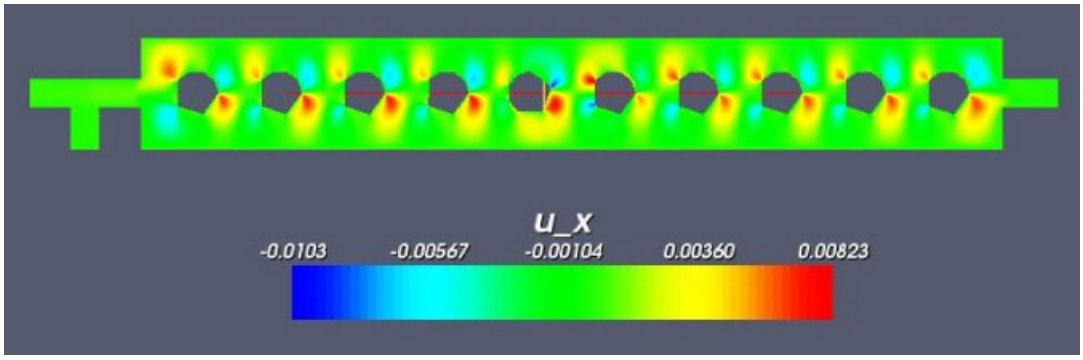


Figure 6. Intermediate design with mixing value of 0.018406.

The original configuration of the circular posts, shown in Figure 2, has an initial mixing metric value of 0.0287106. The optimization strategy improves on this value by manipulating the posts. In Figures 4–8 we show the flow field at various points of the optimization and list the value of the mixing metric in the caption of each figure. Notice that the posts are dimpled. The optimization strategy tests some configurations that increase the metric and rejects them. Figure 4 is one of these. This configuration produces a flow field where the fifth and sixth posts have been stretched apart; this resulted in an increase in the mixing metric from the initial value. Owing to this increase, the pattern search algorithm tended to stay away from configurations of this type. Figures 5–7 show a few sample flow fields where the mixing function value is decreasing, but the obstructions are



DOI: 10.1002/fld

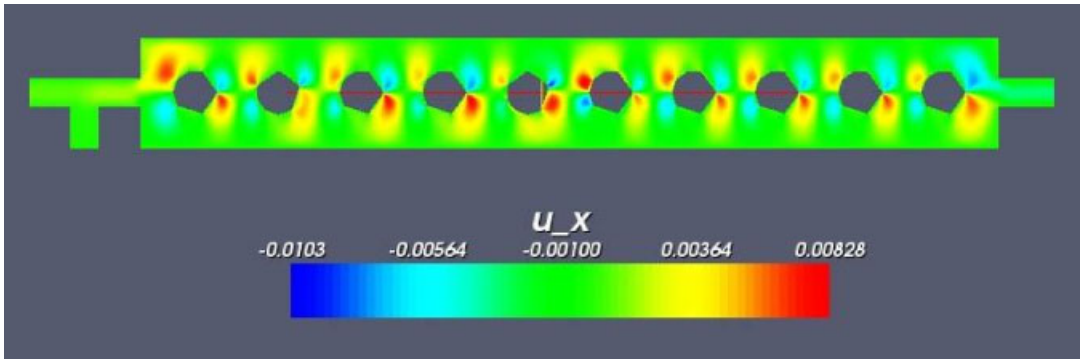


Figure 7. Intermediate design with mixing value of 0.00127773.

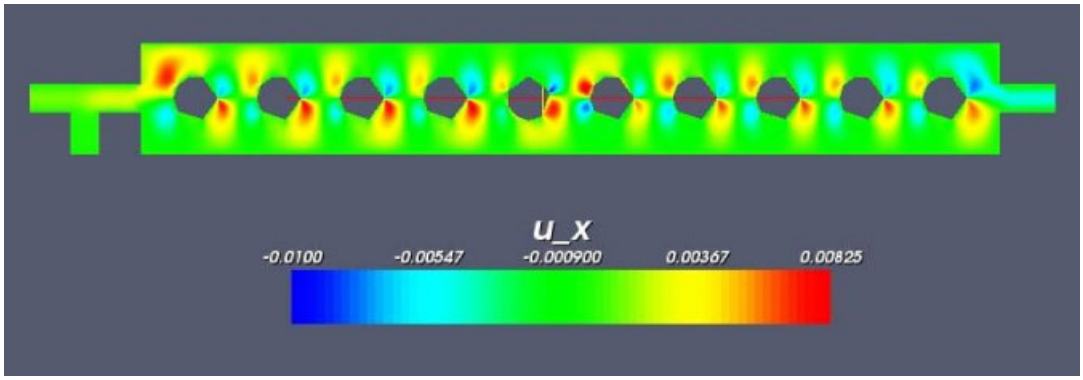


Figure 8. Intermediate design with mixing value of 0.000923394.

not aligned for optimal mixing. Figures 8 and 9 show two configurations for a low mixing metric (values of 0.000811796 and 0.00092394). The value of the mixing metric in these two examples is significantly lower than the value of the original mixing metric. It is interesting to note that the final configurations retained a strong memory to the initial configuration. This suggests that this initial configuration (symmetric circles) leads to a local minimum. We consider this to be an adequate reduction in the cost function. However, we expect the circle configuration to perform poorly because it allows little cross-flow between the two liquids. In the next section, we change the initial post configuration from circles to alternating triangles and see what change this has on the final post configuration and ICEO mixing process.

In Table I, we list the CPU costs for each major component of the function evaluation required for these computations. In column one of this table we list the figure number, followed by the total CPU time for a given function evaluation in column two, the total CPU time for Cubit to generate the mesh in column three, followed by the time to assemble the matrices in column four and the solver CPU time in column five. The CPU times are very consistent from one type of configuration to another. The dominant costs are in solving the discretized PDE systems required by the ICEO mixing process. We further break down these times in Table II.

In this table, we list the iteration counts and CPU time required for each computation required in the ICEO mixing process. We list the figure number in column one, the total solver CPU time in column two, followed by the number of iterations and CPU time required for the potential equation in column three. In column four we list the number of iterations and CPU time required to solve the Navier–Stokes equations, followed by the iterations and CPU time required to solve the mass-transport equation in column five. The CPU time required to solve the potential equation and mass-transport equations is very modest when compared with the time to solve the Navier–Stokes equations. The iteration counts for this problem are all in the range of 60–70 average iterations



DOI: 10.1002/fld

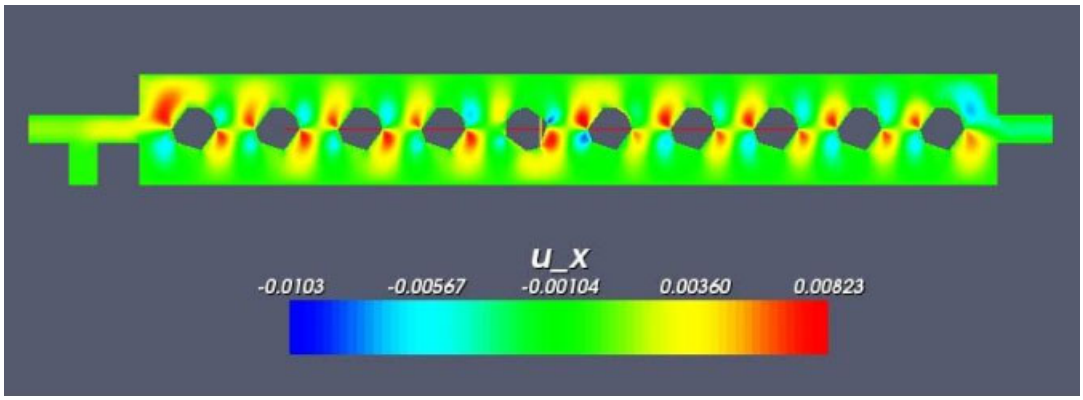


Figure 9. Final design with mixing value of 0.000811796.

Table I. CPU time for each major computational component.

| Figure number | Total CPU time (s) | Mesh generation time (s) | Matrix assembly (s) | Solver CPU time (s) |
|---|---|---|---|---|
| 2 | 20 765.1 | 907.1 | 680.1 | 17 714.1 |
| 4 | 20 874.1 | 909.2 | 679.1 | 17 987.2 |
| 5 | 19 923.9 | 991.7 | 684.5 | 16 505.4 |
| 6 | 19 643.1 | 947.2 | 691.2 | 16 410.9 |
| 7 | 19 173.8 | 958.1 | 672.9 | 16 008.4 |
| 9 | 19 515.5 | 932.3 | 668.1 | 16 689.1 |
| 8 | 19 488.9 | 899.1 | 690.1 | 16 340.1 |

Table II. CPU time and iteration count break down for solving the ICEO optimization of a multiple circular post microfluidic problem.

| Figure | Solver | Potential equation | | Navier–Stokes equation | | Mass-transport | |
|---|---|---|---|---|---|---|---|
| Number | CPU Time | Iters | Time | Iters | Time | Iters | Time |
| 2 | 17 714.1 | 21 | 2.6 | 64.0 | 17 412.1 | 6 | 2.5 |
| 4 | 17 987.2 | 17 | 2.2 | 67.1 | 17 643.2 | 8 | 2.8 |
| 5 | 16 505.4 | 18 | 2.3 | 66.1 | 16 284.2 | 5 | 2.3 |
| 6 | 16 410.9 | 14 | 1.8 | 68.2 | 16 105.1 | 6 | 2.4 |
| 7 | 16 008.4 | 16 | 1.9 | 69.2 | 15 698.2 | 7 | 2.7 |
| 9 | 16 689.1 | 20 | 2.5 | 60.4 | 16 300.1 | 5 | 2.4 |
| 8 | 16 340.1 | 18 | 2.3 | 67.3 | 15 901.1 | 8 | 2.8 |

per nonlinear Picard step. This suggests that changes in the obstruction have little effect on the solver for the discrete Navier–Stokes linear systems of equations. Note that in the iteration counts for the Navier–Stokes equations the non-linear iteration requires between 5 and 8 non-linear steps to converge to the specified tolerance of (18).

### *6.2. Triangle initial configuration*

Here we begin with an initial configuration of 10 alternating triangular posts. We use the objective function found in (6) constrained to 38 design variables. We parameterize each triangular post in a similar way as the circle initial configuration, that is, as a set of piecewise line segments that connect 10 points. Each of these points is defined in polar coordinates using a distance and angle with respect to the origin of our system. The difference from the circular configuration is that we place three of these points at two of the triangle vertices and four at the alternate vertex.



DOI: 10.1002/fld

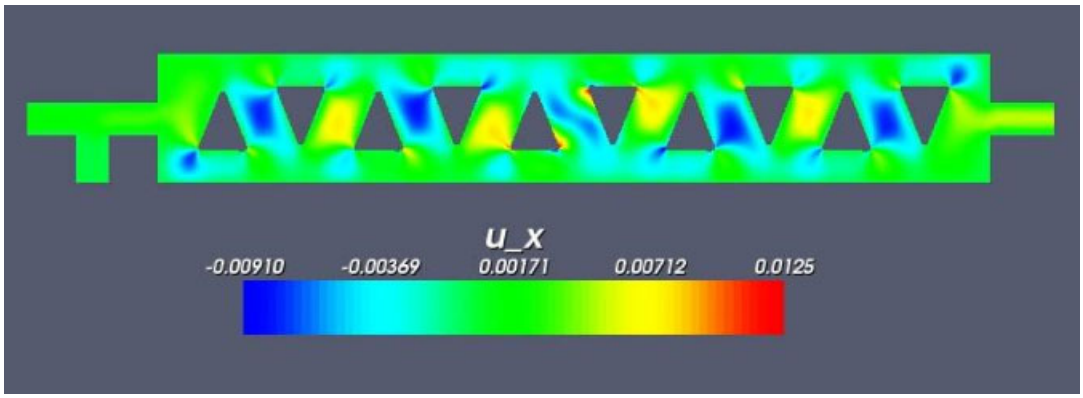


Figure 10. Initial design with mixing value of 0.00610832.

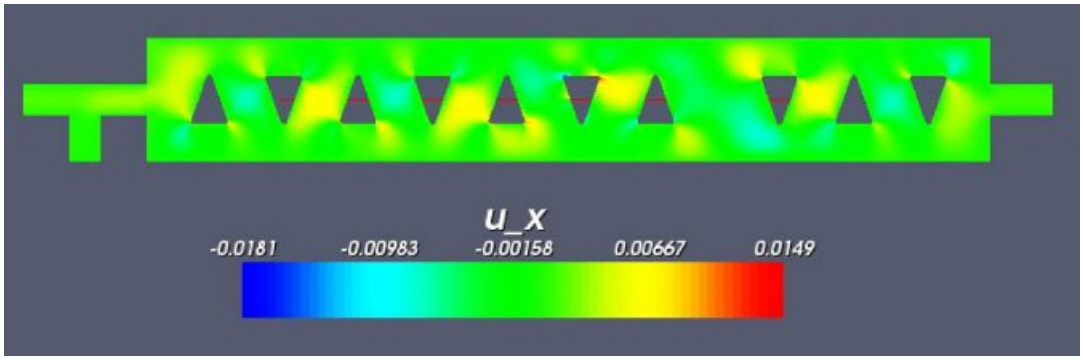


Figure 11. Intermediate design with mixing value of 0.00489152.

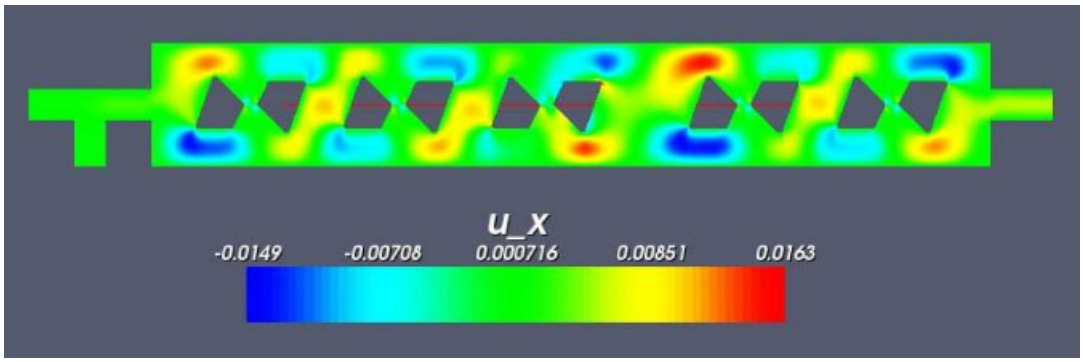


Figure 12. Intermediate design with mixing value of 0.00086896.

This results in 20 design variables. Again, each of the other 9 posts is offset by a distance from its reference point to that of its left neighbor and rotated by an angle, giving 18 more variables.

Figures 10–15 show examples of design configurations produced during the optimization of this initial configuration. Note that the optimized design (Figure 15) consists of non-convex obstacles. In Table III, we list the CPU costs for each major component of the function evaluation. In column one of this table we list the figure number, followed by the total CPU time for a given function evaluation in column two, the total CPU time for Cubit to generate the mesh in column three, followed by the time to assemble the matrices in column four and the solver CPU time in column



DOI: 10.1002/fld

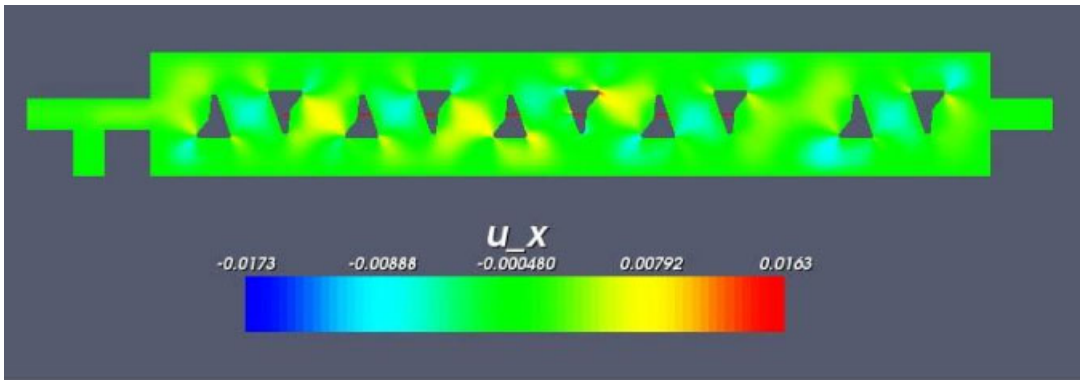


Figure 13. Intermediate design with mixing value of 0.00079312.

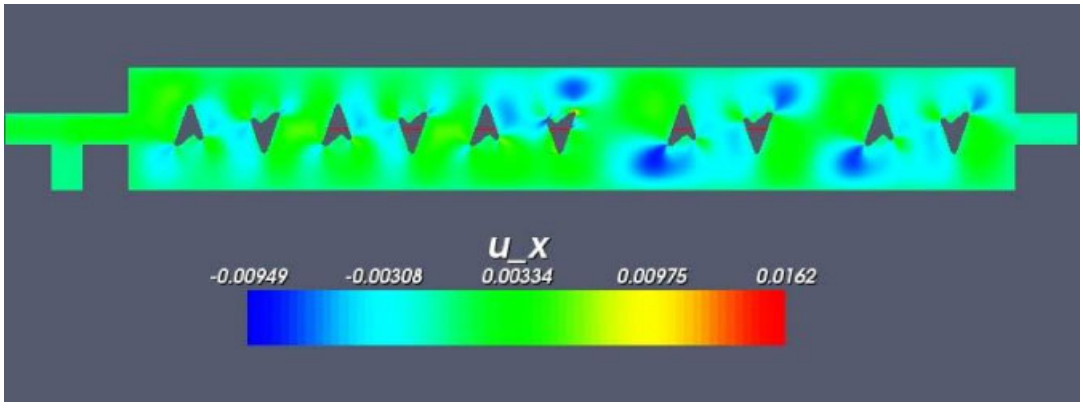


Figure 14. Intermediate design with mixing value of 0.000626595.

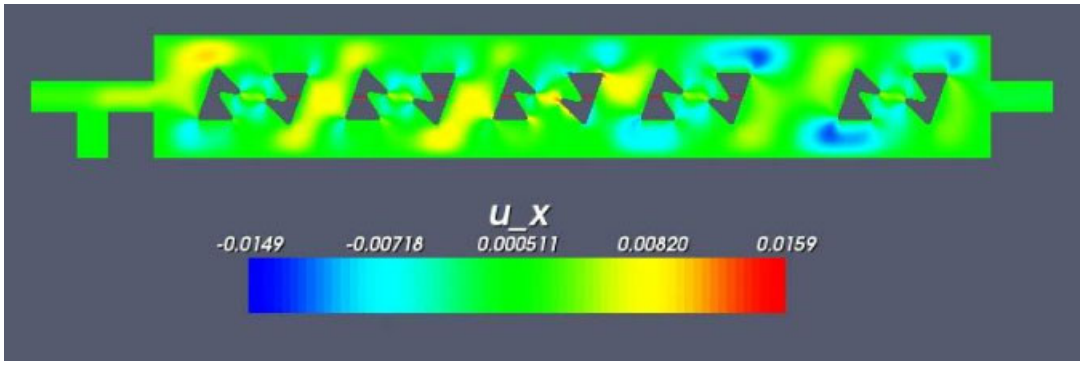


Figure 15. Final design with mixing value of 0.000489152.

five. The CPU times are very consistent from one type of configuration to another. The dominant costs are in solving the discretized PDE systems required by the ICEO mixing process.

In Table IV, we list the number of iterations and CPU time required for each subsequent computation required in the ICEO mixing process. We list the figure number in column one, the total solver CPU time in column two, followed by the number of iterations and CPU time required for the potential equation in column three. In column four we list the number of iterations and CPU time required to solve the Navier–Stokes equations, followed by the number of iterations and CPU time required to solve the mass-transport equation in column five. The CPU time required



DOI: 10.1002/fld

Table III. CPU time for each major computational component.

| Figure number | Total CPU time (s) | Mesh generation time (s) | Matrix assembly (s) | Solver CPU time (s) |
|---|---|---|---|---|
| 10 | 19 876.8 | 997.2 | 730.6 | 17 949.2 |
| 11 | 20 786.8 | 1007.2 | 704.3 | 18 321.2 |
| 12 | 23 474.2 | 1031.7 | 743.2 | 19 867.2 |
| 13 | 19 912.7 | 1207.8 | 709.1 | 17 692.1 |
| 14 | 20 643.4 | 1186.2 | 720.1 | 17 810.1 |
| 15 | 21 710.3 | 1020.0 | 705.1 | 18 615.2 |

Table IV. CPU time and iteration count break down for solving the ICEO optimization of a multiple cylinder microfluidic problem.

| Figure Number | Solver CPU Time | Potential equation | | Navier–Stokes equation | | Mass-transport | |
|---|---|---|---|---|---|---|---|
| | | Iters | Time | Iters | Time | Iters | Time |
| 10 | 17 949.2 | 19 | 4.1 | 60.1 | 17 591.1 | 5 | 2.2 |
| 11 | 18 321.2 | 27 | 5.6 | 62.1 | 17 892.1 | 6 | 2.3 |
| 12 | 19 867.2 | 21 | 4.4 | 67.1 | 19 302.1 | 7 | 2.9 |
| 13 | 17 692.1 | 24 | 4.8 | 61.2 | 17 092.1 | 6 | 2.2 |
| 14 | 17 810.1 | 15 | 4.1 | 62.2 | 17 110.2 | 6 | 2.2 |
| 15 | 18 615.2 | 25 | 5.2 | 63.2 | 18 005.1 | 5 | 2.1 |

to solve the potential equation and mass-transport equations is relatively modest when compared with the time to solve the Navier–Stokes equations. The number of iterations for this problem are all in the range of 60–70 average iterations per non-linear Picard step. The shape of the obstacle has no impact on the performance.

### *6.3. Continuous flow mixer*

In the previous two sections, we examined the quality of mixing for two fixed mode initial configurations. Here we explore a continuous flow ICEO mixer and examine the quality of mixing for this mode. For this example, we begin with the triangle configuration discussed in Section 6.2 with a fixed inflow boundary condition (i.e. $u_x = 5\,\mu$m per second and $u_y = 0$) on the left inlet for the Navier–Stokes equations. The quality of mixing is measured using a mixing metric similar to (6), but defined only at the outflow. In other words,

$$M = \frac{\int (m - \bar{m})^2 \, \mathrm{d}A}{A} \tag{26}$$

where $\bar{m}$ is the average concentration of solute in the liquid mixture and the integral is evaluated over the outflow area, $A$, of the mixing domain. Being concerned only with the quality of mixing along the outflow is a realistic goal for a designer of a microfluidic device since this is the place where the fluid is to be analyzed.

In the process of optimizing this microfluidic device, we have seen a significant reduction in the mixing metric using the continuous flow mixer compared with the fixed volume configurations discussed in the previous subsections. For the continuous flow case, in the course of the optimization algorithm, we were able to reduce the mixing metric from 0.00073088 (Figure 16) to $8.562 \times 10^{-9}$ (Figure 20), where Figures 17, 18 and 19 show the results at intermediate steps of the optimization. It seems that the addition of a cross-flow component to the ICEO flow has helped to mix the fluids at the outlet. We also tested the mixing metric defined in (6) which is defined over the entire flow domain and saw a similar reduction in this metric for the continuous flow problem.

In Tables V and VI, we describe the performance of the solver for the various components of the ICEO flow. We have found that the solvers perform in a similar manner to the fixed volume case.



DOI: 10.1002/fld

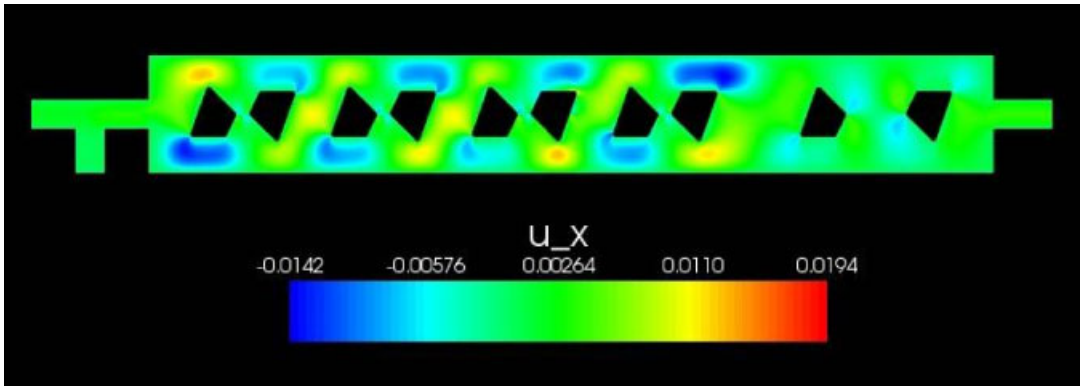


Figure 16. Mixing Value: 0.00073088.

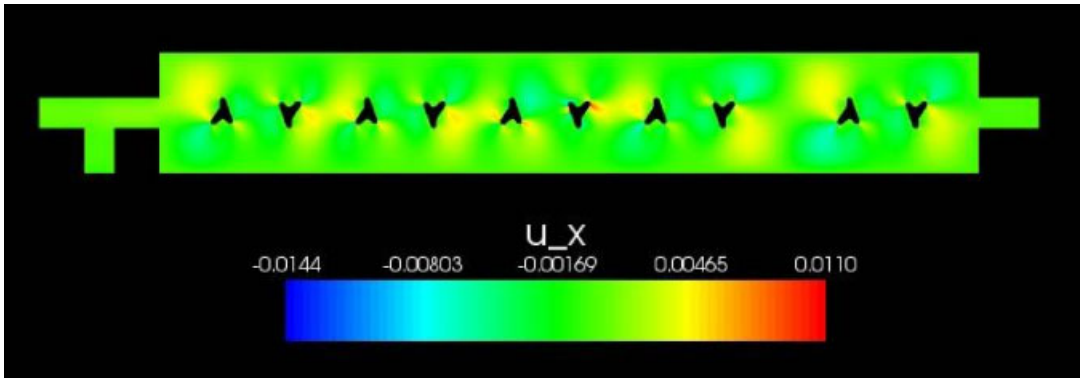


Figure 17. Mixing Value: 0.000032701.

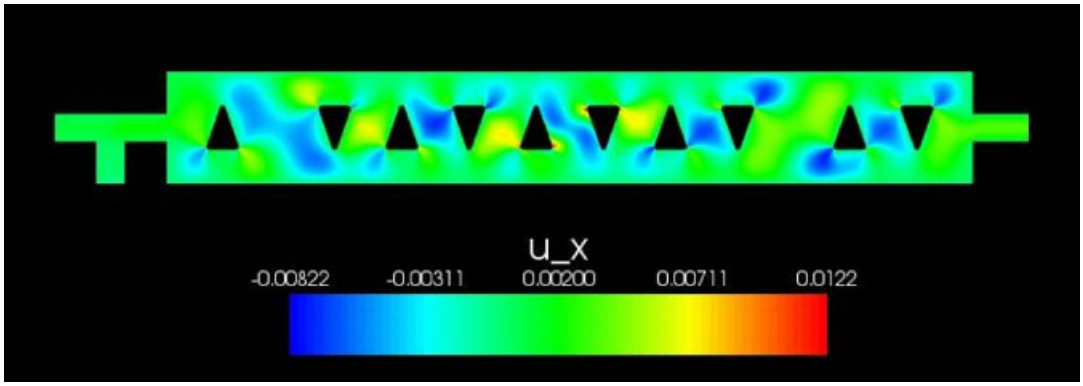


Figure 18. Mixing Value: 0.0000006605.

In Table V we list the CPU costs for each major component of the function evaluation required for these computations. The dominant costs are in solving the discretized PDE systems required by the ICEO mixing process, which we discuss further in Table VI. In this table, we list the iteration counts and CPU time required for each subsequent computation required in the ICEO mixing process. The CPU time required to solve the potential equation and mass-transport equations is relatively modest when compared with the time to solve the Navier–Stokes equations. The iteration counts for solving the Navier–Stokes problem with the pressure convection–diffusion preconditioner are all in the range of 50–60 average iterations per non-linear Picard step. This suggests that



DOI: 10.1002/fld

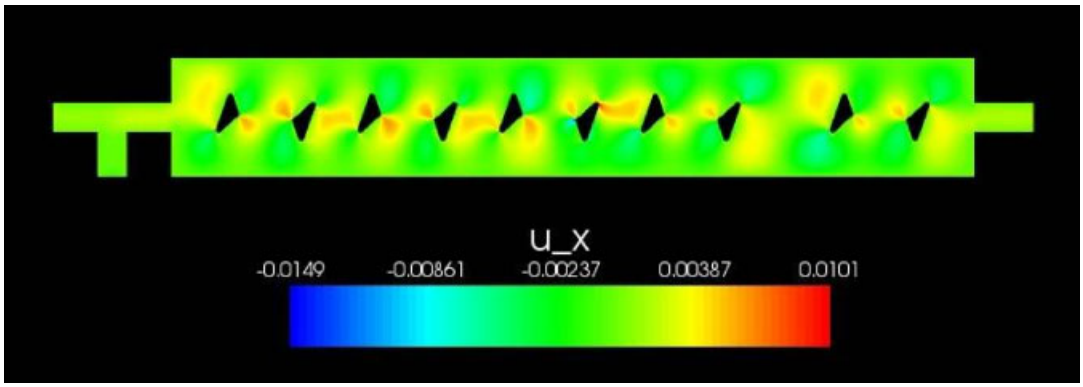


Figure 19. Mixing Value: 0.000000297.

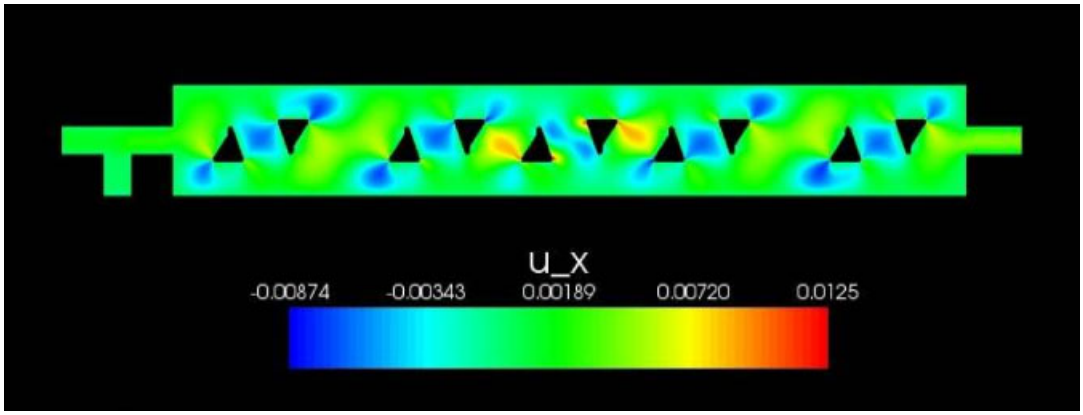


Figure 20. Mixing Value: 0.000000008562.

changes in the obstruction have little effect on the solver for the discrete Navier–Stokes linear systems of equations. Note that in the non-linear (Picard) iteration for the Navier–Stokes equations, 5–7 non-linear steps are needed to converge to the specified tolerance found in (18).

### *6.4. Extensions to the physical model*

The results presented above are focused on optimizing the geometry of the microchannel for an ICEO mixing device. Recently, there has been a large amount of work on designing AC electro-osmosis (ACEO) devices. Similar applications of numerical methods have been applied to designing effective pumping devices driven by ACEO around micro-electrodes [11]. Ajdari [10] predicted that microfluidic pumps could be designed using directional flows that are created by breaking the spatial symmetry of the device. Bazant and Ben [39] predicted that the flow of ACEO around asymmetric pairs of electrodes [40] can be improved by creating a 'fluid conveyor belt' of opposing slip velocities that more effectively drives the flow field [41, 42]. The theory behind this work has been validated experimentally in [43].

The work described in this paper, that is, using ICEO to optimize a mixing device, could be easily expanded to optimize designs for ACEO-driven pumping devices. In ACEO, the linear response time averages the solution over many AC oscillations to produce an equation identical to ICEO except that a complex impedance representing the equivalent circuit response is used as a boundary condition on the electric field equation. We expect the use of AC current to affect the magnitude of the flow field, but not the topology, so that the flow topology designs obtained from ICEO and ACEO will be similar. Therefore, the main difference from what we have described above is that the boundary conditions for the electric field equation (1) change, and the function being



DOI: 10.1002/fld

Table V. CPU time for each major computational component.

| Figure number | Total CPU time (s) | Mesh generation time (s) | Matrix assembly (s) | Solver CPU time (s) |
|---|---|---|---|---|
| 16 | 15 212.3 | 907.1 | 656.3 | 13 721.3 |
| 17 | 15 859.9 | 897.4 | 698.2 | 14 214.1 |
| 18 | 14 486.9 | 912.3 | 604.3 | 12 821.6 |
| 19 | 15 765.1 | 951.8 | 675.2 | 14 044.9 |
| 20 | 15 045.1 | 964.2 | 665.4 | 13 331.8 |

Table VI. CPU time and iteration count break down for solving the ICEO optimization of a multiple circular post microfluidic problem.

| Figure number | Solver CPU time | Potential equation | | Navier–Stokes equation | | Mass-transport | |
|---|---|---|---|---|---|---|---|
| | | Iters | Time | Iters | Time | Iters | Time |
| 16 | 13 721.3 | 19 | 2.3 | 54.0 | 13 212.1 | 6 | 2.5 |
| 17 | 14 214.1 | 21 | 2.6 | 52.0 | 13 703.2 | 7 | 2.6 |
| 18 | 12 821.6 | 22 | 2.7 | 53.4 | 12 298.9 | 8 | 2.8 |
| 19 | 14 044.9 | 18 | 2.2 | 56.1 | 13 542.7 | 6 | 2.5 |
| 20 | 13 331.8 | 20 | 2.5 | 55.0 | 12 892.3 | 7 | 2.6 |

optimized would be slightly different. (Additional optimization parameters for the post voltages would have to be added.) Since Sundance is extremely flexible it can easily handle the different boundary conditions. Moreover, given the simplicity and robustness of APPSPACK, the additional optimization parameters should have little effect on its ability to find a solution.

Finally, other studies [44, 45] have considered non-linear effects that go beyond the circuit model at the electrodes that was used in this project. Modeling all of the non-linear effects is still an open question. We expect that modeling non-linear effects would take the form of a non-linear dependence of the current as a function of voltage (or changes in the relationship between current and slip velocity) in the boundary conditions of the Navier–Stokes equations; these can be easily added to our model. Modeling non-dilute solutions is also an open question [46] and would require finding the right relationship between voltage and current for the boundary conditions of the flow equations.

## 7. CONCLUSIONS

In this paper, we have explored the numerical solution of the optimization problems that arise in models in of ICEO mixing in microfluidic mixing devices. We have used a combination of derivative-free optimization together with iterative solution of the collection of PDEs that determine function values. We have explored several models of devices, including different configurations of obstacle shapes defining the devices and several mixing metrics, and we have shown the solution algorithms used to optimize mixing metrics to be robust and efficient with respect to device topology and choice of metric. The numerical solution strategies are based on effective preconditioned Krylov subspace solvers for the incompressible Navier–Stokes equations, and the computations were performed using a derivative-free optimization code, APPSPACK, together with two software environments, Sundance and Trilinos.

ACKNOWLEDGEMENTS

This work was partially supported by the DOE Office of Science MICS Program under grant DEFG0204ER25619 and by the ASC Program at Sandia National Laboratories. Sandia is a multiprogram



DOI: 10.1002/fld

laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

## REFERENCES


1. Stone H, Strock A, Ajdari A, Squires T, Bazant M. Engineering flows in small devices: microfluidics towards a lab on a chip. *Annual Review of Fluid Mechanics* 2004; **36**:381–411.
2. Squires T, Bazant M. Induced charge electro-osmosis. *Journal of Fluid Mechanics* 2004; **509**:217–252.
3. Elman H, Silvester D, Wathen A. *Finite Elements and Fast Iterative Solvers*. Oxford University Press: Oxford, U.K., 2005.
4. Kay D, Loghin D, Wathen AJ. A preconditioner for the steady-state Navier–Stokes equations. *SIAM Journal on Scientific Computing* 2002; **24**:237–256.
5. Silvester D, Elman H, Kay D, Wathen A. Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow. *Journal on Computational and Applied Mathematics* 2001; **128**:261–279.
6. Elman HC, Howle V, Shadid J, Shuttleworth R, Tuminaro R. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing* 2005; **27**:1651–1668.
7. Oddy M, Santiago J, Mikkelsen J. Electrokinetic instability micromixing. *Analytical Chemistry* 2001; **73**: 5822–5832.
8. Bazant M, Squires T. Induced-charge electrokinetic phenomena: theory and microfluidic applications. *Physical Review Letters* 2004; **92**:066101.
9. Harnett C, Dunphy-Guzman K, Templeton J, Senousy Y, Kanouff M. Model based design of a microfluidic mixer driven by induced charge electroosmosis. *Lab Chip* 2008; **8**:565–572.
10. Ajdari A. Pumping liquids using asymmetric electrode arrays. *Physical Review E* 2000; **61**:R45–R48.
11. Ramos A, Morgan H, Green NG, Castellanos A. AC electrokinetics: a review of forces in microelectrode structures. *Journal of Physics D*: *Applied Physics* 1998; **31**:2338–2353.
12. Gamayunov NI, Murtsovkin VA, Dukhin A. Pair interaction of particles in electric field. *Colloid Journal USSR* 1986; **48**:197–203.
13. Murtsovkin VA. Nonlinear flows near polarized disperse particles. *Colloid Journal USSR* 1996; **58**:341–349.
14. Squires T, Bazant M. Breaking symmetries in induced-charge electro-osmosis and electrophoresis. *Journal of Fluid Mechanics* 2006; **560**:65–101.
15. Gangwal S, Cayre OJ, Bazant M, Velev OD. Induced-charge electrophoresis of metallodielectric particles. *Physical Review Letters* 2008; **100**:058302.
16. Levitan J, Devasenathipathy S, Studer V, Ben Y, Thorsen T, Squires T, Bazant M. Experimental observation of induced-charge electro-osmosis around a metal wire in a microchannel. *Colloids and Surfaces* 2005; **267**:122–132.
17. Kanouff M, Harnett C, Dunphy-Guzman K, Templeton J, Senousy Y, Skulan A. Science based engineering of a sample preparation device for biological agent detection. *Technical Report*, *Sandia National Laboratory*, 2007.
18. Zhao H, Bau HH. Microfluidic chaotic stirrer utilizing induced-charge electroosmosis. *Physical Review E* 2007; **75**:066217.
19. Yariv M. Induced-charge electrophoresis of non-spherical particles. *Physics of Fluids* 2005; **17**:051702.
20. Thamida S, Chang H. Nonlinear electrokinetic ejection and entrainment due to polarization at nearly insulated wedges. *Physics of Fluids* 2002; **14**:4315.
21. Yossifon G, Frankel I, Miloh T. On electro-osmotic flows through microchannel junctions. *Physics of Fluids* 2006; **18**:117108.
22. Saintillan D, Darve E, Shaqfeh ES. Hydrodynamic interactions in the induced-charge electrophoresis of colloidal rod dispersions. *Journal of Fluid Mechanics* 2006; **563**:223–259.
23. Bharadwaj R, Santiago J, Mohammadi B. Design and optimization of on-chip capillary electrophoresis. *Electrophoresis* 2002; **23**:2729–2744.
24. Mohammadi B, Santiago J. Incomplete sensitivities in design and control of fluidic channels. *Computer Assisted Mechanics and Engineering Sciences* 2003; **10**:201–210.
25. Wenzel W, Hamacher K. Stochastic tunneling approach for global minimization of complex potential energy landscapes. *Physical Review Letters* 1999; **82**:3003–3007.
26. Henriksen MO, Holmen J. Algebraic splitting for incompressible Navier–Stokes equations. *Journal of Computational Physics* 2002; **175**:438–453.
27. Perot JB. An analysis of the fractional step method. *Journal of Computational Physics* 1993; **108**:51–58.
28. Quarteroni A, Saleri F, Veneziani A. Factorization methods for the numerical approximation of Navier–Stokes equations. *Computational Methods in Applied Mechanical Engineering* 2000; **188**:505–526.
29. Elman HC, Howle VE, Shadid J, Shuttleworth R, Tuminaro R. A taxonomy and comparison of parallel block preconditioners for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 2007; **227**:1790–1808.
30. Elman HC. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM Journal on Scientific Computing* 1999; **20**:1299–1316.
31. Elman HC, Silvester DJ, Wathen AJ. Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations. *Numerische Mathematik* 2002; **90**:665–688.




DOI: 10.1002/fld

32. Loghin D, Wathen A, Elman H. Preconditioning techniques for Newton's method for the incompressible Navier–Stokes equations. *BIT* 2003; **43**:961–974.
33. Long K. *Sundance 2.0. Technical Report*, Sandia National Laboratories, SAND2004-4793, 2004.
34. Sandia National Laboratory. *Cubit Geometry and Mesh Generation Toolkit*, Sandia National Laboratory. Available from http://www.cubit.sandia.gov/index.html, 2006.
35. Gray GA, Kolda TG. Algorithm 8xx: APPSPACK 4.0: asynchronous parallel pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software* 2006; **32**(3):485–507.
36. Griffin JD, Kolda TG. *Asynchronous Parallel Generating Set Search for Linearly-Constrained Optimization. Technical Report*, Sandia National Laboratories, Albuquerque, NM, Livermore, CA, July 2006.
37. Heroux MA, Bartlett RA, Howle VE, Hoekstra RJ, Hu JJ, Kolda TG, Lehoucq RB, Long KR, Pawlowski RP, Phipps ET, Salinger AG, Thornquist HK, Tuminaro RS, Willenbring JM, Williams A, Stanley KS. An overview of the Trilinos Project. *ACM Transactions on Mathematical Software* 2005; **31**:397–423.
38. Gill PE, Murray W, Saunders MA. SNOPT: an sqp algorithm for large-scale constrained optimization. *SIAM Journal on Optimization* 2002; **12**:979–1006.
39. Bazant MZ, Ben Y. Theoretical prediction of fast 3d ac electro-osmotic pumps. *Lab Chip* 2006; **6**:1451–1461.
40. Brown ABD, Smith C, Rennie AR. Pumping of water with ac electric fields applied to asymmetric pairs of microelectrodes. *Physical Review E* 2000; **63**:016305.
41. Burch D, Bazant M. Design principle for improved three-dimensional ac electro-osmosis pumps. *Physical Review E* 2008; **77**:055303.
42. Olesen LH, Bruus H, Ajdari A. Ac electrokinetic micropumps: the effect of geometrical confinement, faradaic current injection, and nonlinear surface capacitance. *Physical Review E* 2006; **73**.
43. Urbanski JP, Levitan J, Burch DN, Thorsen T, Bazant MZ. The effect of step height on the performance of ac electro-osmotic microfluidic pumps. *Journal of Interface and Colloid Science* 2007; **309**:332–341.
44. Bazant M, Thornton K, Ajdari A. Diffuse-charge dynamics in electrochemical systems. *Physical Review E* 2004; **70**:021506.
45. Chu K, Bazant M. Nonlinear electrochemical relaxation around conductors. *Physical Review E* 2006; **74**:011501.
46. Storey BD, Edwards L, Kilic M, Bazant M. Steric effects on ac electro-osmosis in dilute electrolytes. *Physical Review E* 2008; **77**:06317.



DOI: 10.1002/fld