University of Maryland Department of Computer Science TR-4901
University of Maryland Institute for Advanced Computer Studies TR-2008-01

FAST SOLVERS FOR MODELS OF ICEO MICROFLUIDIC FLOWS [*]

ROBERT R. SHUTTLEWORTH[†], HOWARD C. ELMAN[‡], KEVIN R. LONG[§], AND JEREMY A. TEMPLETON[¶]

Abstract. 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* [17], by which a flow through the device is driven by a set of charged 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 [11, 16], 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.

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 are a key ingredient in this process [18]. 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) [17]. 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: 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 to 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 [2]. These methods are based on the approximation of the Schur complement operator by a technique proposed by Kay, Loghin, Wathen [11], Silvester, Elman, Kay, Wathen [16], and Elman, Howle, Shadid, Shuttleworth, and Tuminaro [4]. They use multi-level multigrid methods and in our particular case, algebraic multi-level

[*]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 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.

[†]Applied Mathematics and Scientific Computing Program and Center for Scientific Computation and Mathematical Modeling, University of Maryland, College Park, MD 20742. rshuttle@math.umd.edu

[‡]Department of Computer Science and Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742, elman@cs.umd.edu.

[§]Department of Mathematics and Statistics, Texas Tech University, Lubbock, TX 79049, kevin.long@ttu.edu.

[¶]Sandia National Laboratories, PO Box 969, MS 9409, Livermore, CA 94551, jatempl@sandia.gov.

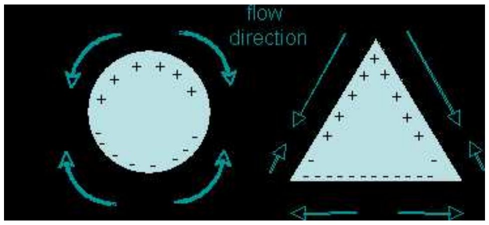

Fig. 1. *Double layer flows around a circular and triangular conductor as described in [10].*

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 nonlinear 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. 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 [10].

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 [15]. 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. From our perspective, Induced Charge Electro-osmosis (ICEO) has advantages over these approaches because it has been shown to mix dyes in a few seconds and scales well for smaller devices [10].

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 which 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 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 [17].

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. In addition, the posts can be charged to a fixed potential, allowing more control over the flow field, although this is more costly. A final advantage is that ICEO flows can be made to recirculate within a given volume to reduce dispersion in the mixing operation [10].

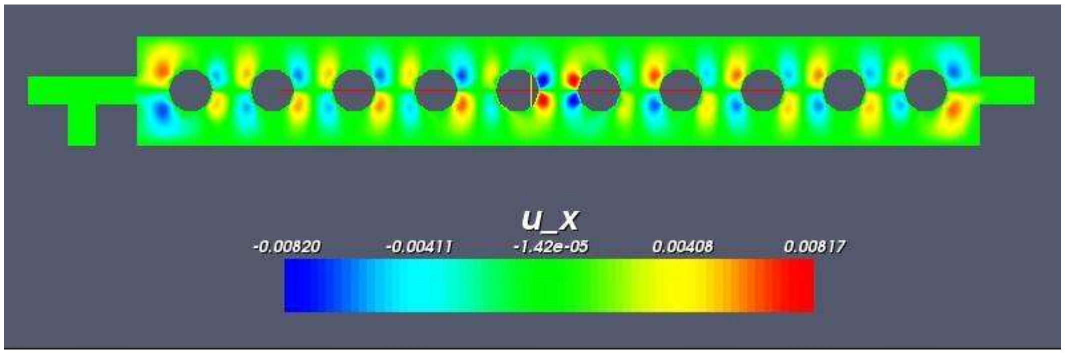

FIG. 2. *Sample initial domain for the multiple cylinder problem, with plot of the initial horizontal velocity.*

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. 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. This device could be useful as part of a miniaturized biological detector. However, the best shapes and topology for the conductors which generate the ICEO flow is currently unknown. Our goal is to investigate this issue by solving an optimization problem to maximize the mixing of two fluids by manipulating the shape and topology of the charged region. At each step of the optimization algorithm, a sequence of computationally expensive fluid problems must be solved. Therefore, scalable linear solvers are needed to effectively solve this optimization problem.

3. Model Description. A finite element model was used to calculate the electric field, the ICEO flow as described in [12, 17], and the mass transport for a multi-species 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 ϕ 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 remainder of 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

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

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

in $\Omega \subset \mathbb{R}^d (d = 2 \text{ or } 3)$ and used to calculate momentum transport. Here $\mathbf{u}$ is the fluid velocity, p represents the hydrodynamic pressure, ν 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

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

where ϵ is the fluid permittivity, ζ is the potential drop across the electrical double layer, E_t is the tangential electric field obtained from solving (1), and μ is the fluid viscosity [17].

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, i.e. 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 $= 1.8 \times 10^{-9} \frac{cm^2}{s}$, which represents ~ 3 μm particles in an aqueous solution. This value was used in [10]. 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 [10], was used to quantify the extent of mixing based on the calculated results,

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

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 mixing strategy for this microfluidic problem by varying the shape and orientation of the electrically charged posts. This requires us to solve a series of problems (1),(2)-(3), and (5) at each step of the optimization strategy where we want to

$$\text{minimize} \qquad M \qquad \text{subject to} \tag{7}$$
$$d_i \geq 0 \tag{8}$$

where d_i is a multi-dimensional component design variable related to the shape and orientation of the charged posts and the objective function, M, is the mixing metric defined in (6). We have tested a number of design configurations which we describe in Section 6.

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, $(\mathbf{u} \cdot \text{grad})\mathbf{u}$. For a low Reynolds number flow such as this one, Picard's method is an adequate choice of a nonlinear 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*

$$\begin{aligned} -\nu\nabla^2(\Delta\mathbf{u}^{(k)}) + (\mathbf{u}^{(k)}\cdot\text{grad})\Delta\mathbf{u}^{(k)} + \text{grad}\Delta p^{(k)} &= \mathbf{f} - (-\nu\nabla^2(\mathbf{u}^{(k)}) + (\mathbf{u}^{(k)}\cdot\text{grad})\mathbf{u}^{(k)} + \text{grad}p^{(k)}) \\ -\text{div}\Delta\mathbf{u}^{(k)} &= \text{div}\mathbf{u}^{(k)} \end{aligned} \tag{9}$$

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

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}\begin{pmatrix} \boldsymbol{\Delta}\mathbf{u}^{\mathbf{k}} \\ \Delta p^k \end{pmatrix} = \begin{pmatrix} \mathbf{f}_{\mathbf{u}}^{\mathbf{k}} \\ f_p^k \end{pmatrix}, \tag{10}$$

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

The strategies we employ for solving (10) are derived from the LDU block factorization of this coefficient matrix where the diagonal (D) and upper triangular (U) factors are grouped together,

$$\begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ BF^{-1} & I \end{pmatrix}\begin{pmatrix} F & B^T \\ 0 & -S \end{pmatrix} \tag{11}$$

and

$$S = BF^{-1}B^T \tag{12}$$

is the Schur complement (of F in A). 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 (11), 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,

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

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 preconditioners (P-CD) where the basic idea hinges on the notion of an approximate commutator. Consider a convection-diffusion operator of the form

$$(\nu\nabla^2 + (\mathbf{w}\cdot\text{grad})). \tag{13}$$

When $\mathbf{w}$ is an approximation to the velocity obtained from the previous nonlinear step, (13) is an Oseen linearization of the nonlinear term in (2). Suppose there is an analogous operator defined on the pressure space,

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

Consider the commutator of these operators with the gradient:

$$\epsilon = (\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))\nabla - \nabla(\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))_p. \tag{14}$$

Supposing that ϵ is small, multiplication on both sides of (14) by the divergence operator gives

$$\nabla^2(\nu\nabla^2 + (\mathbf{w} \cdot \text{grad}))_p^{-1} \approx \nabla \cdot (\nu\nabla^2 + \ (\mathbf{w} \cdot \text{grad}))^{-1}\nabla \tag{15}$$

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

$$\begin{aligned} (Q_p^{-1}A_p)(Q_p^{-1}F_p)^{-1} &\approx (Q_p^{-1}B)(Q_v^{-1}F)^{-1}(Q_v^{-1}B^T) \\ A_pF_p^{-1}Q_p &\approx (BF^{-1}B^T) \end{aligned}$$

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

$$S \approx \hat{S} = A_pF_p^{-1}Q_p \tag{16}$$

for a stable finite element discretization. A similiar approximation can be made for stabilized finite element discretizations [2, 5].

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 [3, 6, 11, 16]. A proof that convergence rates are independent of the mesh is given in [13]. 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 [14]. 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 Ω, we automatically generate a mesh on Ω 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 [1]. 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 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. We terminate this iteration when the residual is reduced by a factor of 10^{-10}, i.e.

$$\|b - A\phi\| \leq 10^{-10} \|b\| . \tag{17}$$

We discretize the **incompressible Navier Stokes equations** using Taylor-Hood $P_2 - P_1$ finite elements with fourth order Gaussian quadrature [2]. This is a stable choice of finite element pairs, so no stabilization is required. The nonlinear system is solved by Picard's method where the structure of a two-dimensional steady version of F is a 2×2 block matrix consisting of a discrete version of the operator

$$\begin{pmatrix} -\nu\Delta + u^{(k)} \cdot \nabla & 0 \\ 0 & -\nu\Delta + u^{(k)} \cdot \nabla \end{pmatrix} \tag{18}$$

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

$$\left\| \begin{pmatrix} \mathbf{f_u} - (F(\mathbf{u})\mathbf{u} + B^T p) \\ f_p - B\mathbf{u} \end{pmatrix} \right\| \leq 10^{-4} \left\| \begin{pmatrix} \mathbf{f_u} \\ f_p \end{pmatrix} \right\| . \tag{19}$$

At each step of the nonlinear iteration, we terminate the linear iteration with the Oseen system, when the residual is reduced be a factor of 10^{-5}, i.e.

$$\left\| \begin{pmatrix} \mathbf{f_u^k} \\ f_p^k \end{pmatrix} - \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{\Delta u^k} \\ \Delta p^k \end{pmatrix} \right\| \leq 10^{-5} \left\| \begin{pmatrix} \mathbf{f_u^k} \\ f_p^k \end{pmatrix} \right\| , \tag{20}$$

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 [2, 5]. 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}, i.e. this iteration for the convection-diffusion problem is terminated when

$$\|\mathbf{y} - F\mathbf{u}\| \leq 10^{-5} \|\mathbf{y}\|. \tag{21}$$

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 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}, i.e.

$$\|b - Am\| \leq 10^{-5} \|b\| . \tag{22}$$

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., $n \leq 100$), 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

$$\begin{aligned} \min \quad & M(m) && (23)\\ \text{subject to} \quad & c_L \leq A_I m \leq c_u && (24)\\ & A_E m = b && (25)\\ & l \leq m \leq u && (26) \end{aligned}$$

where $M(m)$ is the objective function, the inequality constraints are denoted by the matrix A_I and the upper and lower bounds by c_L and c_U respectively. The equality constraints are denoted by the matrix A_E and the right hand side, b . Finally, l and u denote lower and upper bounds on the component variables [7, 8]. For our problem, we only use the linear constraints found in (26) which we describe further in Section 6. The objective function, $M(m)$, 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 nonlinear 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 [14] for the finite element discretization, which is a tool for specifying, building, and developing finite-element solutions of partial differential equations developed at Sandia National Laboratory. 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 [14].

Our implementation of the pressure convection-diffusion preconditioner and the other solvers for the discrete systems of equations uses *Trilinos* [9], 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 (10), 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 (10).

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 to 100 processors per run. Each of this cluster's compute nodes are dual Intel 3.6 GHz Xenon processors with 2GB 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 [10]. 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. 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 [5] and we see similar trends when applying this technology to this problem.

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 radius from the center of the post together with an angle with respect to the horizontal axis of our system. This results in 20 design variables. The offset value is linearly constrained, see (26), to have a value between 0.001 and 0.005 microns. Likewise, the angle is constrained to be between 0 and 360 degrees. We require each of the 10 posts to have the same shape. Each of the posts is offset by a distance from the center of the previous post and rotated by an angle, giving 18 more variables. The linear constraint for the distance from the central post is bounded between 0.01 and 0.02 microns, and the rotation angle is bounded between 0 and 360 degrees. Each corner is smoothed to a radius of 5 microns which allows the points to be collocated, helps with mesh generation, and is a tolerance to permit an acceptable physically manufacturable device.

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 through 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. Due to this increase, the pattern search algorithm tended to stay away from configurations of this type. Figures 5, 6, and 7 show a few sample flow fields where the mixing function value is decreasing, but the obstructions are not aligned for optimal mixing. Figures 8 and 9show 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

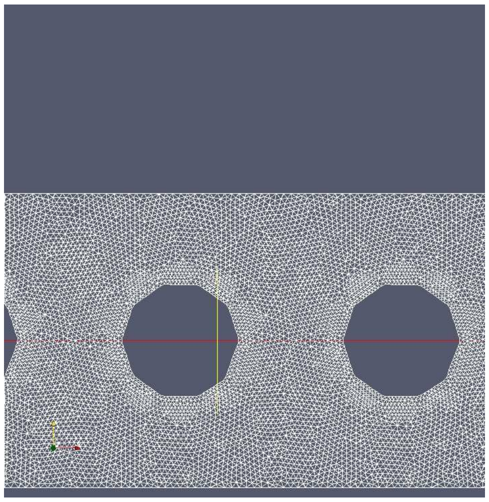

FIG. 3. *Sample mesh for the multiple cylinder domain.*

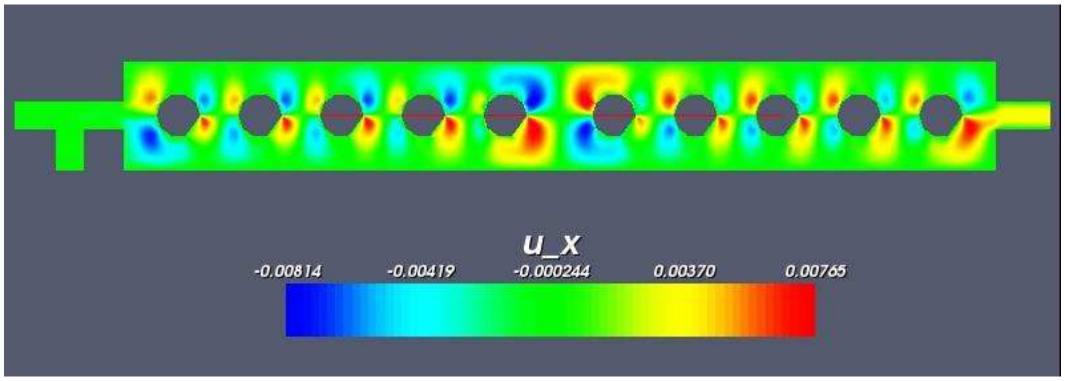

FIG. 4. *Preliminary design with mixing value of 0.032451.*

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.

Figure Number	Total CPU Time (sec)	Mesh Generation Time (sec)	Matrix Assembly (sec)	Solver CPU Time (sec)
2	20765.1	907.1	680.1	17714.1
4	20874.1	909.2	679.1	17987.2
5	19923.9	991.7	684.5	16505.4
6	19643.1	947.2	691.2	16410.9
7	19173.8	958.1	672.9	16008.4
9	19515.5	932.3	668.1	16689.1
8	19488.9	899.1	690.1	16340.1

TABLE 1
CPU time for each major computational component.

In Table 1, 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

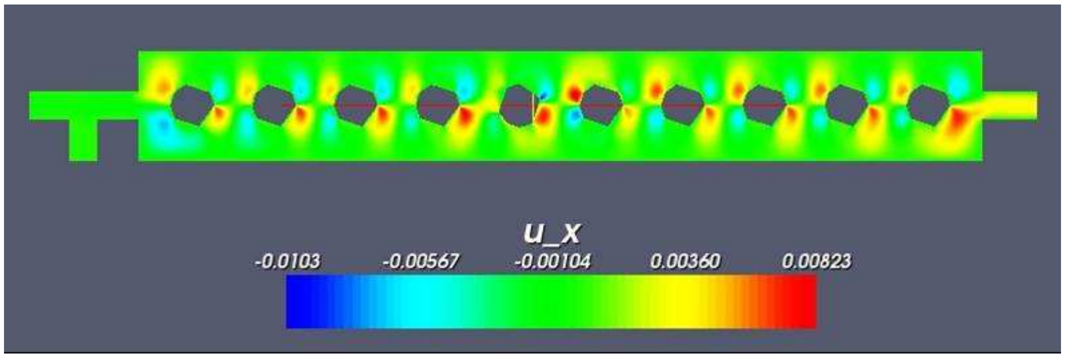

FIG. 5. *Intermediate design with mixing value of 0.0249871.*

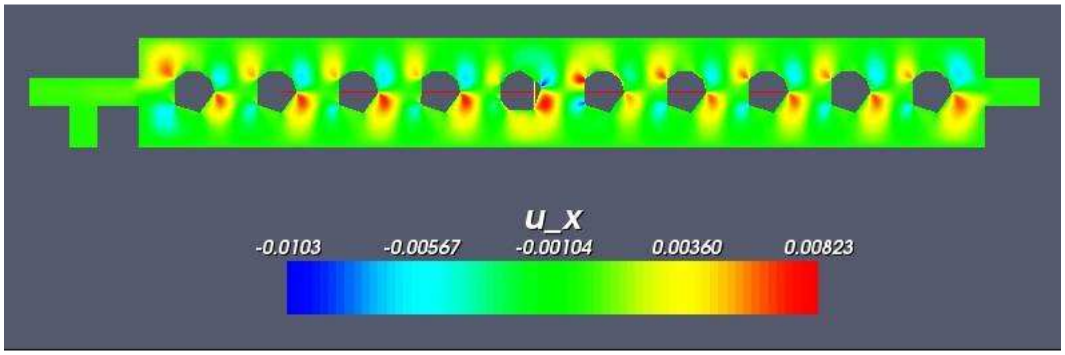

FIG. 6. *Intermediate design with mixing value of 0.018406.*

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 2.

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 to 70 average iterations 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 nonlinear iteration requires between 5 and 8 nonlinear steps to converge to the specified tolerance of (19).

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 similiar way as the circle initial configuration, i.e. 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. This results in 20 design variables. Again, each of the other 9 posts is offset by a distance from the central post and rotated by an angle, giving 18 more variables.

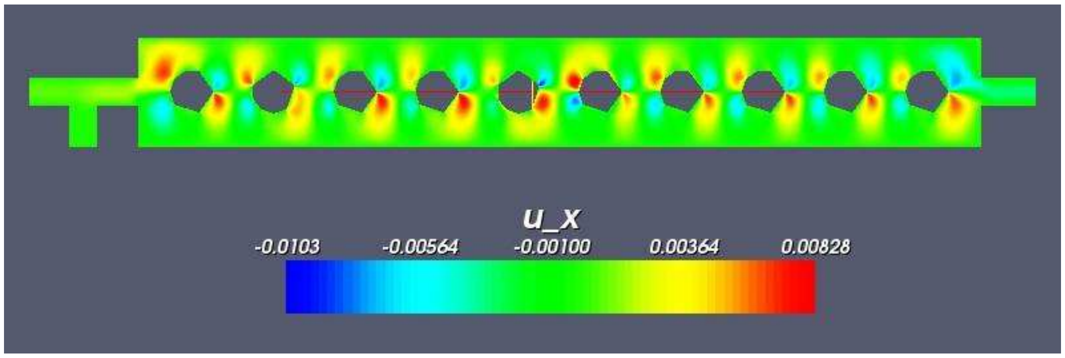

Fig. 7. *Intermediate design with mixing value of 0.00127773.*

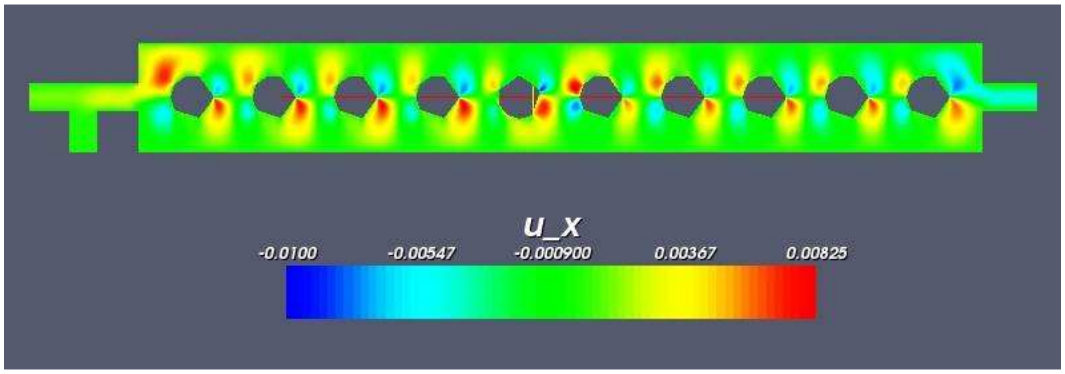

Fig. 8. *Intermediate design with mixing value of 0.000923394.*

Figures 10 through 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 3, 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 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 4, 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 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 to 70 average iterations per nonlinear Picard step. The shape of the obstacle has no impact on performance.

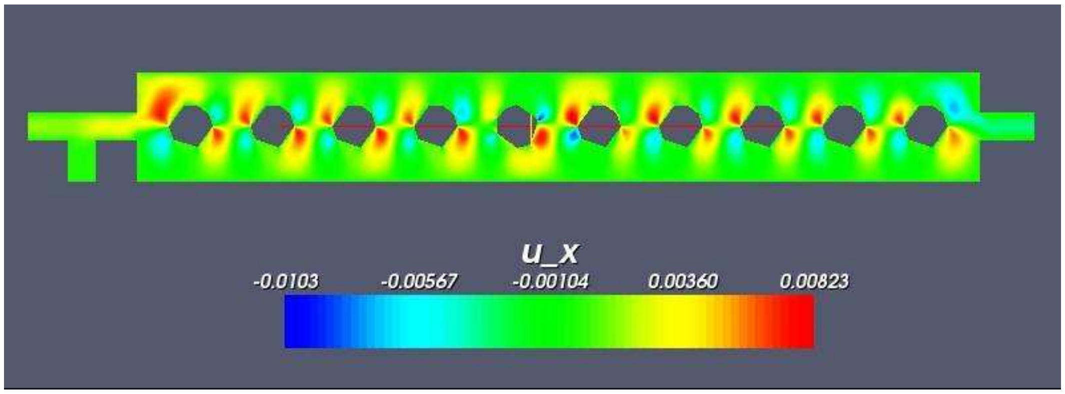

Fig. 9. *Final design with mixing value of 0.000811796.*

Figure	Solver	Potential Equation		Navier-Stokes Eq		Mass-transport	
Number	CPU Time	iters	time	iters	time	iters	time
2	17714.1	21	2.6	64.0	17412.1	6	2.5
4	17987.2	17	2.2	67.1	17643.2	8	2.8
5	16505.4	18	2.3	66.1	16284.2	5	2.3
6	16410.9	14	1.8	68.2	16105.1	6	2.4
7	16008.4	16	1.9	69.2	15698.2	7	2.7
9	16689.1	20	2.5	60.4	16300.1	5	2.4
8	16340.1	18	2.3	67.3	15901.1	8	2.8

Table 2

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

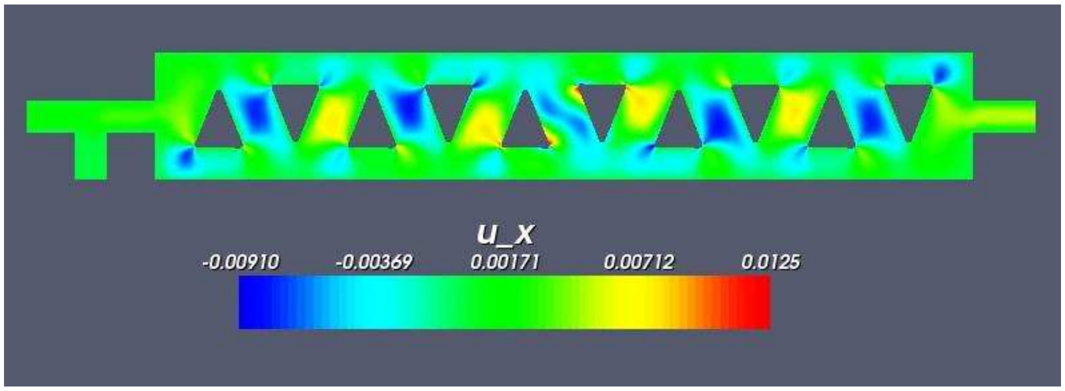

Fig. 10. *Initial design with mixing value of 0.00610832.*

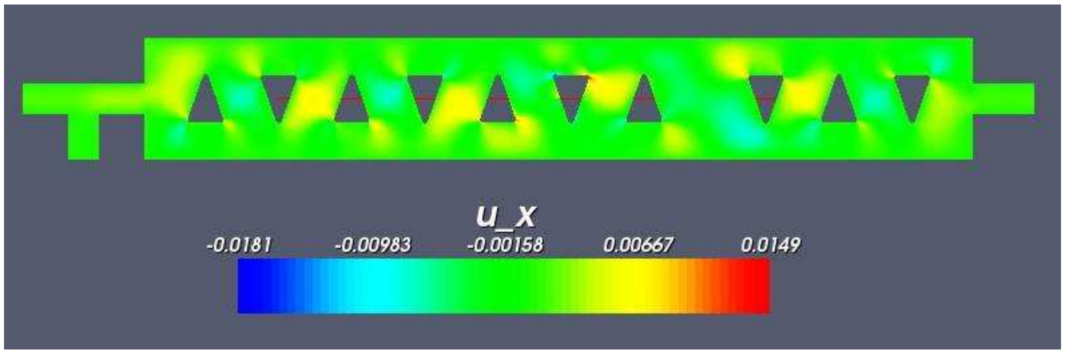

Fig. 11. *Intermediate design with mixing value of 0.00489152.*

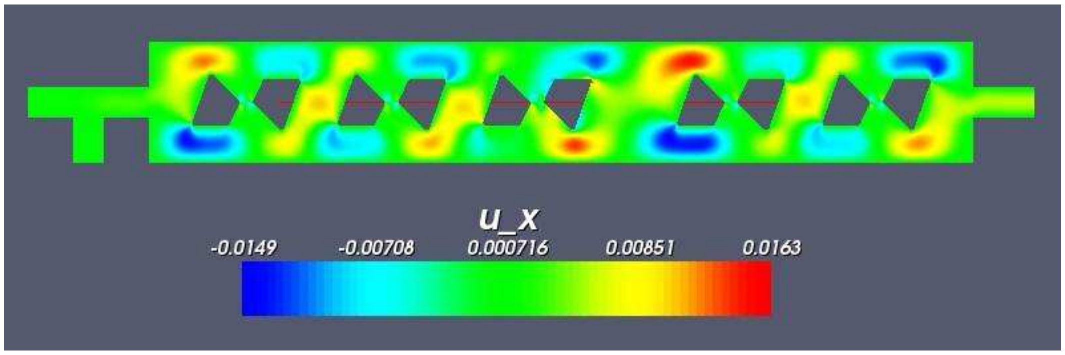

Fig. 12. *Intermediate design with mixing value of 0.00086896.*

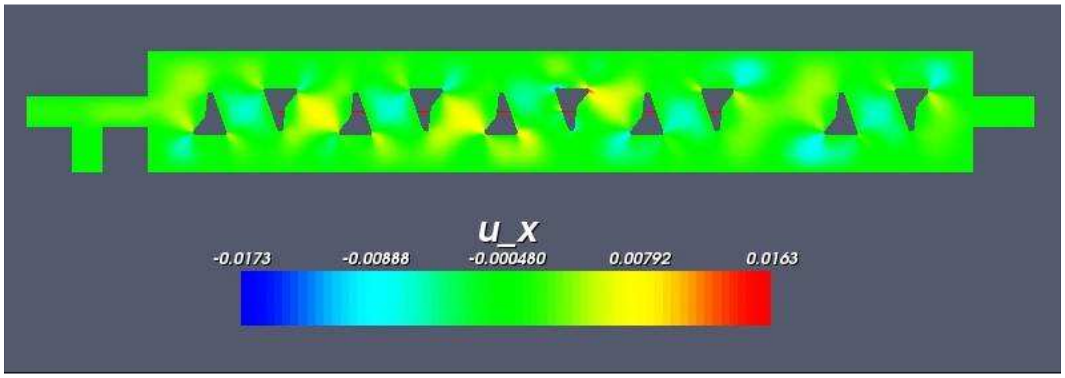

Fig. 13. *Intermediate design with mixing value of 0.00079312.*

Figure Number	Total CPU Time (sec)	Mesh Generation Time (sec)	Matrix Assembly (sec)	Solver CPU Time (sec)
10	19876.8	997.2	730.6	17949.2
11	20786.8	1007.2	704.3	18321.2
12	23474.2	1031.7	743.2	19867.2
13	19912.7	1207.8	709.1	17692.1
14	20643.4	1186.2	720.1	17810.1
15	21710.3	1020.0	705.1	18615.2

Table 3
CPU time for each major computational component.

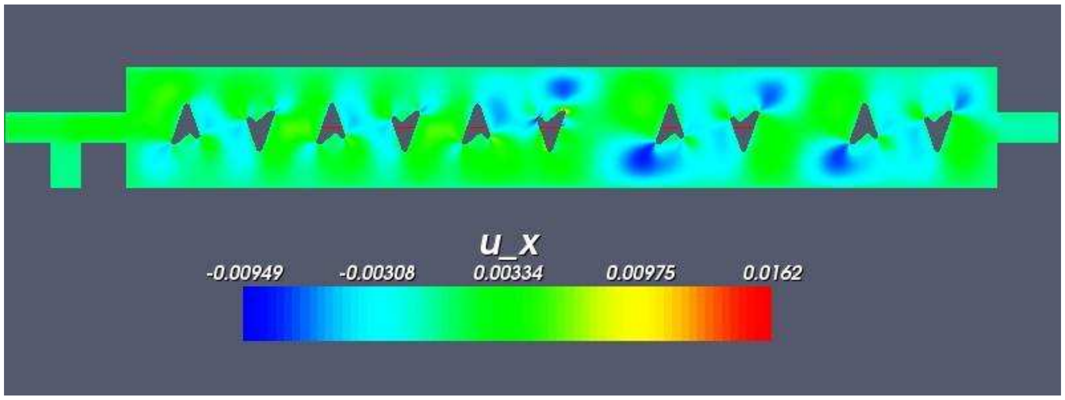

Fig. 14. *Intermediate design with mixing value of 0.000626595.*

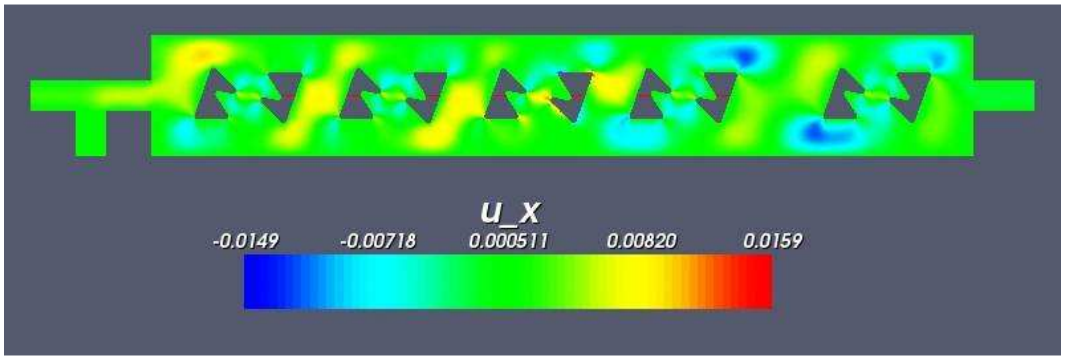

Fig. 15. *Final design with mixing value of 0.000489152.*

Figure Number	Solver CPU Time	Potential Equation iters	Potential Equation time	Navier-Stokes Eq iters	Navier-Stokes Eq time	Mass-transport iters	Mass-transport time
10	17949.2	19	4.1	60.1	17591.1	5	2.2
11	18321.2	27	5.6	62.1	17892.1	6	2.3
12	19867.2	21	4.4	67.1	19302.1	7	2.9
13	17692.1	24	4.8	61.2	17092.1	6	2.2
14	17810.1	15	4.1	62.2	17110.2	6	2.2
15	18615.2	25	5.2	63.2	18005.1	5	2.1

Table 4

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

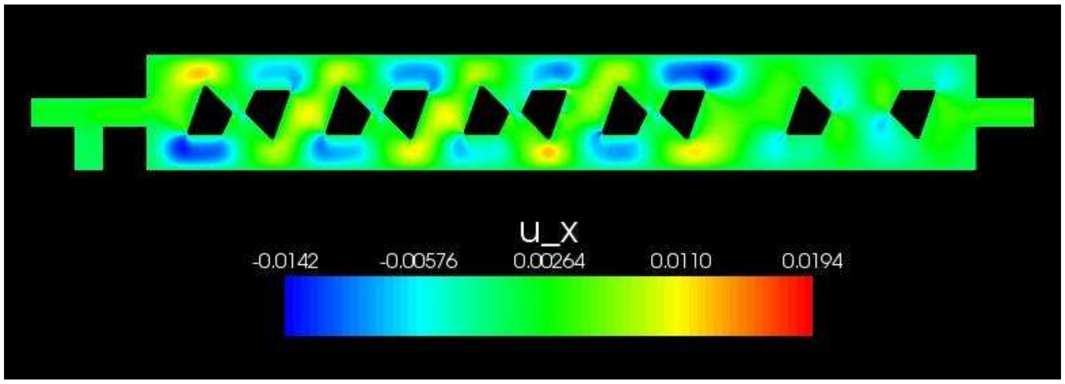

FIG. 16. *Mixing Value: 0.00073088*

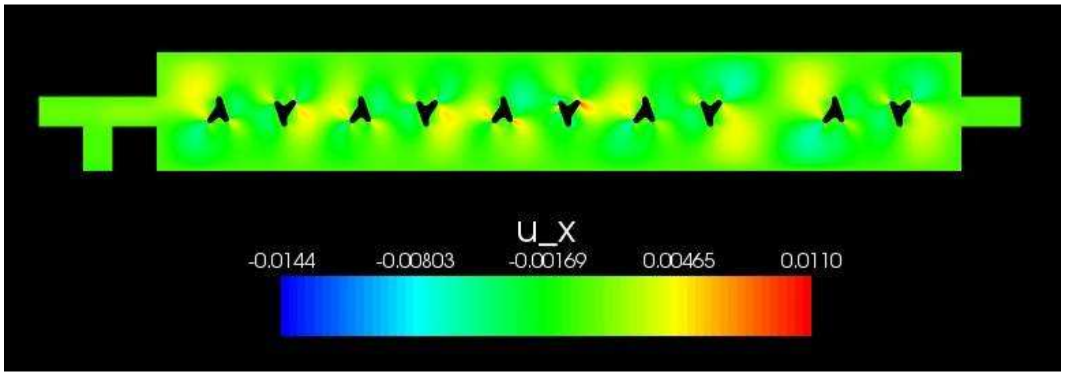

FIG. 17. *Mixing Value: 0.000032701*

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. $u_x = 5$ microns 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 similiar to (6), but defined only at the outflow. In other words,

$$M = \frac{\int (m - \bar{m})^2 dA}{A}, \tag{27}$$

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 of 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 8.562×10^{-9} (Figure 20). It seems that the addition of a crossflow 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 5 and 6, 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. In Table 5 we

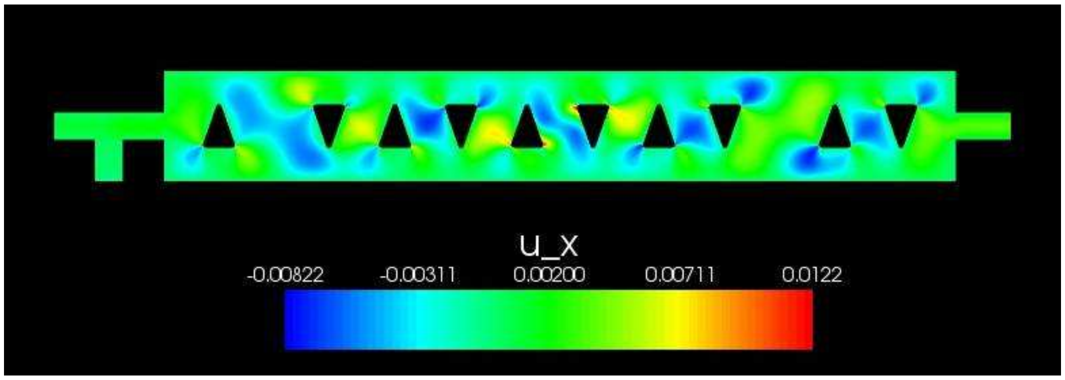

FIG. 18. *Mixing Value: 0.0000006605*

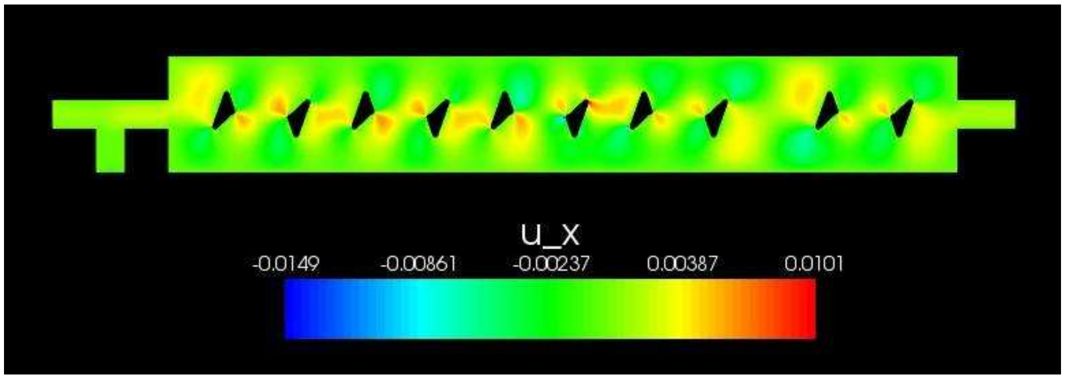

FIG. 19. *Mixing Value: 0.000000297*

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 6. 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 to 60 average iterations 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 nonlinear (Picard) iteration for the Navier-Stokes equations, 5-7 nonlinear steps are needed to converge to the specified tolerance found in (19).

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 partial differential equations 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.

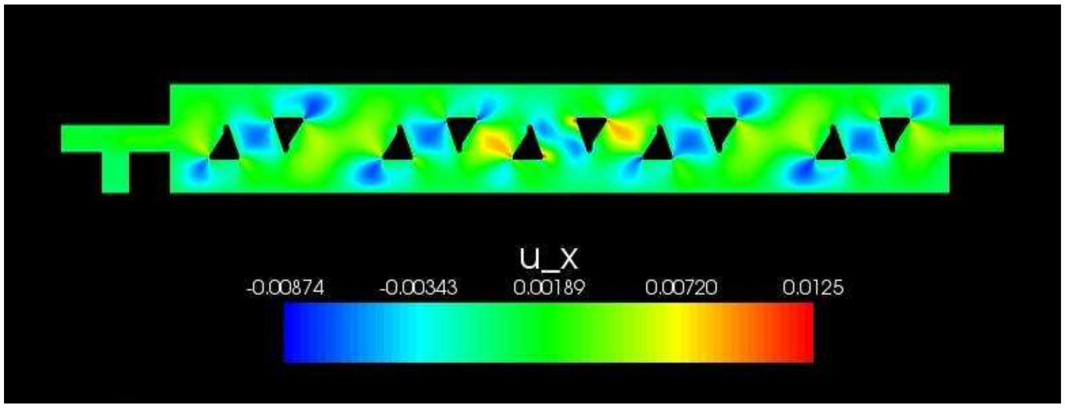

Fig. 20. *Mixing Value: 0.000000008562*

Figure Number	Total CPU Time (sec)	Mesh Generation Time (sec)	Matrix Assembly (sec)	Solver CPU Time (sec)
16	15212.3	907.1	656.3	13721.3
17	15859.9	897.4	698.2	14214.1
18	14486.9	912.3	604.3	12821.6
19	15765.1	951.8	675.2	14044.9
20	15045.1	964.2	665.4	13331.8

Table 5
CPU time for each major computational component.

REFERENCES

[1] *Cubit geometry and mesh generation toolkit*, 2006. Sandia National Laboratory, http://www.cubit.sandia.gov/index.html.

[2] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, UK, 2005.

[3] H. C. Elman, *Preconditioning for the steady-state Navier–Stokes equations with low viscosity*, SIAM Journal on Scientific Computing, 20 (1999), pp. 1299–1316.

[4] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *Block preconditioners based on approximate commutators*, SIAM Journal on Scientific Computing, 27 (2005), pp. 1651–1668.

[5] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro, *A taxonomy and comparison of parallel block preconditioners for the incompressible Navier-Stokes equations*, tech. rep., University of Maryland, College Park, 2007.

[6] H. C. Elman, D. J. Silvester, and A. J. Wathen, *Performance and analysis of saddle point preconditioners for the discrete steady-state Navier–Stokes equations*, Numerische Mathematik, 90 (2002), pp. 665–688.

[7] G. A. Gray and T. G. Kolda, *Algorithm 8xx: APPSPACK 4.0: Asynchronous parallel pattern search for derivative-free optimization*, ACM Transactions on Mathematical Software. To appear (Accepted Nov. 2005).

[8] J. D. Griffin and T. G. Kolda, *Asynchronous parallel generating set search for linearly-constrained optimization*, tech. rep., Sandia National Laboratories, Albuquerque, NM and Livermore, CA, July 2006.

[9] M. A. Heroux, R. A. Bartlett, V. E. Howle, R. J. Hoekstra, J. J. Hu, T. G. Kolda, R. B. Lehoucq, K. R. Long, R. P. Pawlowski, E. T. Phipps, A. G. Salinger, H. K. Thornquist, R. S. Tuminaro, J. M. Willenbring, A. Williams, and K. S. Stanley, *An Overview of the Trilinos Project*, ACM Transactions on Mathematical Software, 31 (2005), pp. 397–423.

[10] M. P. Kanouff, C. Harnett, K. Dunphy-Guzman, J. Templeton, Y. Senousy, and A. Skulan, *Science based engineering of a sample preparation device for biological agent detection*, tech. rep., Sandia National Laboratories, 2007.

[11] D. Kay, D. Loghin, and A. J. Wathen, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 237–256.

[12] J. Levitan, S. Devasenathipathy, V. Studer, Y. Ben, T. Thorsen, T. Squires, and M. Bazant, *Experimental observation of induced-charge electro-osmosis around a metal wire in a microchannel*, Colloids and Surfaces, 267 (2005), pp. 122–132.

Figure Number	Solver CPU Time	Potential Equation iters	Potential Equation time	Navier-Stokes Eq iters	Navier-Stokes Eq time	Mass-transport iters	Mass-transport time
16	13721.3	19	2.3	54.0	13212.1	6	2.5
17	14214.1	21	2.6	52.0	13703.2	7	2.6
18	12821.6	22	2.7	53.4	12298.9	8	2.8
19	14044.9	18	2.2	56.1	13542.7	6	2.5
20	13331.8	20	2.5	55.0	12892.3	7	2.6

Table 6

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

[13] D. Loghin, A. Wathen, and H. Elman, *Preconditioning techniques for Newton's method for the incompressible Navier-Stokes equations*, BIT, 43 (2003), pp. 961–974.

[14] K. Long, *Sundance 2.0*, tech. rep., Sandia National Laboratories, SAND2004-4793.

[15] M. Oddy, J. Santiago, and J. Mikkelsen, *Electrokinetic instability micromixing*, Analytical Chemistry, 73 (2001), pp. 5822–5832.

[16] D. Silvester, H. Elman, D. Kay, and A. Wathen, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, Journal on Computational and Applied Mathematics, 128 (2001), pp. 261–279.

[17] T. Squires and M. Bazant, *Induced Charge Electro-osmosis*, Journal of Fluid Mechanics, 509 (2004), pp. 217–252.

[18] H. Stone, A. Strock, A. Ajdari, T. Squires, and M. Bazant, *Engineering flows in small devices: microfluidics towards a lab on a chip*, Annual Review of Fluid Mechanics, 36 (2004).