Inverse Design of Complex Fluids with Fully-Differentiable Lagrangian Particle Dynamics

an undergraduate thesis presented
by
Kaylie S. Hausknecht
to
The Department of Physics

in partial fulfillment of the requirements
for the joint degree of
Bachelor of Arts
in the subjects of
Physics and Astrophysics

Harvard University
Cambridge, Massachusetts
May 2024

Contents

1 Introduction **1**
- 1.1 Fluid Mechanics . . . 1
- 1.2 Computational Fluid Dynamics . . . 2
- 1.3 Inverse Design Techniques for Flow Control . . . 6
- 1.4 Differentiable Physics Simulations . . . 9
- 1.5 Objectives . . . 13

2 Differentiable Lagrangian Fluid Simulations **16**
- 2.1 Steady Flow Problems . . . 17
- 2.2 Non-Steady Flow Problems . . . 21

3 Taylor Dispersion **22**
- 3.1 Background on Dispersion . . . 23
- 3.2 Taylor Dispersion Theory . . . 25
- 3.3 Simulations . . . 26
- 3.4 Results . . . 29

4 Porous Media **32**
- 4.1 Dispersion and Mixing in Porous Media . . . 34
- 4.2 Simulations . . . 35
- 4.3 Optimizations . . . 38

5 Mixing in Journal Bearing **47**
- 5.1 Theory . . . 48
- 5.2 Simulations . . . 49
- 5.3 Optimizations . . . 51

6 Flow Sculpting **56**
- 6.1 Optimizations . . . 58

6.2 Preliminary Results 60

7 Catheter Design **65**

8 Conclusion **71**

References **83**

Listing of figures

1.1 A schematic diagram showing the difference between the Eulerian approach to fluid mechanics and the Lagrangian approach to fluid mechanics 3

2.1 The fluid velocity field $u(x, y)$ corresponding to flow through a channel . . 18
2.2 The evolution of Lagrangian tracer particles in flow through a channel . . . 19
2.3 The evolution of tracer particles diffusing in a flow through a channel with no reinforcement of the wall boundaries 19
2.4 The evolution of tracer particles diffusing in a flow through a channel with the wall boundaries reinforced by static, repulsive particles 20

3.1 A simulation of Taylor Dispersion that shows the evolution of diffusing tracer particles in a channel at three different times 27
3.2 (a) The trajectories of Brownian particles with $D_m = 10^{-5}$ in the absence of a flow field, (b) the average MSD of the Brownian particles over time for various molecular diffusivities computed using the differentiable lagged MSD algorithm, (c) the actual molecular diffusivities set in the simulation vs. the molecular diffusivities inferred from the trajectory data 29
3.3 MSD$/2\tau$ vs. τ, the lag time, calculated with the lagged MSD algorithm from the trajectories of particles flowing through a channel (shown in Figure 3.1); the dispersion coefficient, $D*$, calculated based on the limiting behavior of MSD$/2\tau$ is marked with a dashed line 30
3.4 The ratio of the dispersion coefficient inferred from the channel flow simulations to the molecular diffusivities set in the simulation vs. the square of the Péclet number 31

4.1 Diagram of the flow set-up for the periodic porous media 36
4.2 The fluid velocity field for flow through a periodic porous media solved via JAX-CFD: (a) $u(x, y)$ and (b) $v(x, y)$ 37

4.3 (a) Sample particle trajectories in a flow through a periodic porous medium at Pe = 1000, (b) sample particle trajectories in a flow through a periodic porous medium at Pe = 1 38
4.4 Optimization trajectory of the scaled dispersion coefficient for porous media with a volume fraction of 0.7 and a Péclet number of 1. The dashed line represents the scaled dispersion coefficient for the monodisperse square array. . 40
4.5 Sample simulation of two groups of passive tracer particles initialized separately mixing in a periodic porous media over time 43
4.6 Optimization of the center of mass distance mixing indicator at a Péclet number of 100; the scaled dispersion coefficient, calculated via forward simulations but not optimized directly, is plotted over the average distance loss indicator 44
4.7 Optimization of the average distance mixing indicator at a Péclet number of 100; the scaled dispersion coefficient, calculated via forward simulations but not optimized directly, is plotted over the average distance loss indicator . . 45

5.1 Flow parameters for the mixing in journal bearing optimization 49
5.2 The evolution of Lagrangian tracer particles in time from (a) to (b) to (c) to (d) in a simulation of Taylor-Couette flow 52
5.3 The trajectory of the design parameters during the mixing efficiency optimization in eccentric Taylor-Couette flow 53
5.4 The normalized mixing indicator as a function of time for the initial guess parameters and the optimized parameters with images of the tracer particles shown at different points of the simulation; the inset plot shows the angular velocities corresponding to the optimal rotation protocols for the inner and outer cylinders 54

6.1 (a) Diagram of the set-up of flow sculpting problems with parameters including the initial distribution of the tracer particles, the sizes and positions of the obstacles, and the target configuration, (b) Diagram of the desired output of flow sculpting problems, which is an arrangement of obstacles such that the passive tracer particles form into the target configuration downstream from the tracer particles 59
6.2 Optimizing letter design: (a) Optimized velocity field $u(x, y)$, (b) Optimized velocity field $v(x, y)$, (c) Trajectory of tracer particles moving through flow, (d) Final position of tracer particles in the flow 62
6.3 Optimizing particle positions: (a) Optimized velocity field $u(x, y)$, (b) Optimized velocity field $v(x, y)$, (c) Trajectory of tracer particles moving through flow, (d) Final position of tracer particles in the flow 63

7.1 Trajectory of an upstream swimming bacteria in a channel in the absence of rotational and translational Brownian motion 67
7.2 (a) Velocity field $u(x, y)$ for a channel flow with elliptical obstacles embedded in the flow (solved for with JAX-CFD), (b) Velocity field $u(x, y)$ for a channel flow with triangular obstacles embedded in the flow (solved for with JAX-CFD) 69
7.3 Trajectory of a swimming bacterium in a catheter with elliptical obstacles that mitigate upstream travel 70

1

Introduction

1.1 Fluid Mechanics

From the air we breathe to the bodies of water that flow across our planet to the blood coursing through our veins, fluids play a fundamental role in our daily lives. Our ability to understand and model the dynamics of fluids has also been foundational to the development of important technologies from airplane flight to microfluidics, to biomedical de-

vices, and weather prediction software. Today, understanding the range of complex fluid behaviors we encounter remains an essential and unfinished project in physics and mathematics.

Incompressible fluids are governed by the Navier-Stokes equations:

$$\begin{aligned} \frac{\partial u}{\partial t} &= -u \cdot \nabla u - \nabla p + \frac{1}{\mathrm{Re}} \nabla^2 u \\ \nabla \cdot u &= 0 \end{aligned} \tag{1.1}$$

where u is the velocity field, p is the pressure, and Re is the Reynolds number. The Reynolds number characterizes the ratio of inertial to viscous forces in a flow and is given by $\mathrm{Re} = \frac{UL}{\nu}$ where U and L are a characteristic velocity and characteristic length for the flow, respectively, and ν is the kinematic viscosity. This is a non-linear partial differential equation (PDE) that quickly becomes analytically intractable to work with without making simplifying assumptions. Indeed, one of the open Millennium Prize problems in mathematics is to prove the existence and smoothness of solutions to the Navier-Stokes equations [Carlson et al., 2006]. Before the advent of computers, the Navier-Stokes equations were explored with perturbation methods and semi-analytical techniques. The development of numerical and computational methods for modeling fluids has become an active area of research called Computational Fluid Dynamics.

1.2 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is a branch of fluid mechanics that uses numerical techniques and algorithms to model and analyze fluid flows. As a discipline, it has both driven and evolved alongside advancements in computational technology.

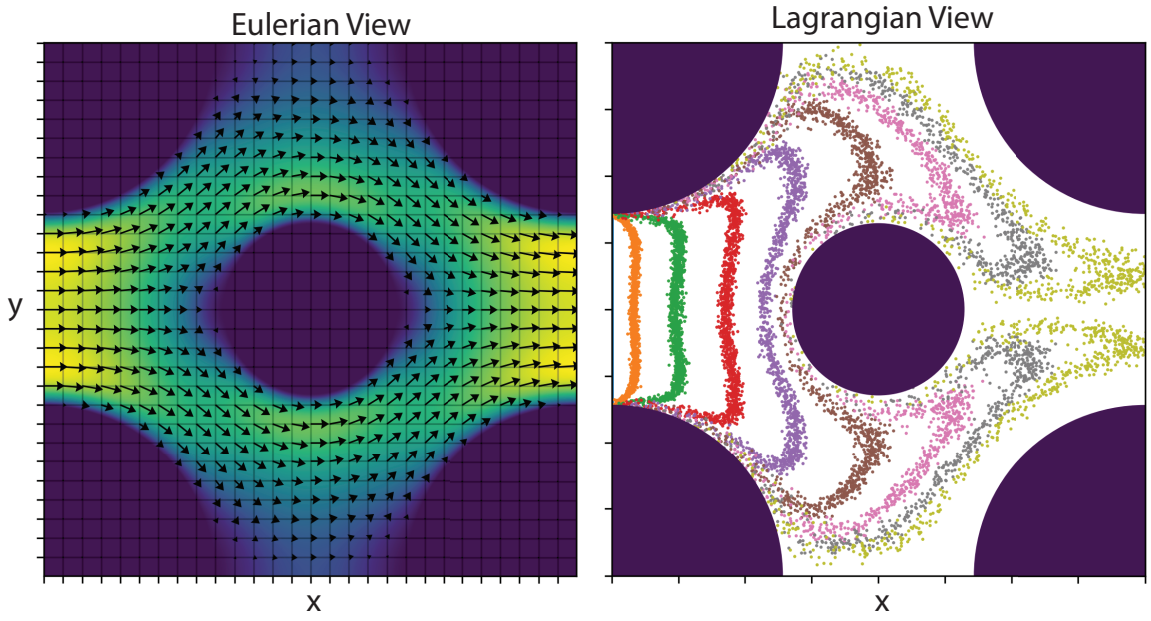

Figure 1.1: A schematic diagram showing the difference between the Eulerian approach to fluid mechanics and the Lagrangian approach to fluid mechanics

1.2.1 Eulerian vs. Lagrangian Fluid Mechanics

There are two paradigms for specifying and modeling fluid flows — the Eulerian framework and the Lagrangian framework. The Eulerian framework focuses on modeling fluids at the level of fields and looks at fluid velocity fields at fixed points in space as they evolve in time. The Lagrangian framework follows the trajectory of individual fluid parcels in time as they move spatially through a flow. A diagram showing the distinction between these two approaches to CFD is shown in Figure 1.1. In CFD, Eulerian methods are typically implemented with grid-based techniques, and Lagrangian methods are typically implemented

using meshfree, particle-based techniques. Although these two paradigms are ultimately equivalent to each other and are fundamentally capable of describing the same fluid dynamics, certain flow behaviors can be more effectively studied with one framework over the other.

A classic example of a problem where Lagrangian studies tend to be more informative than Eulerian ones is in the analysis of transport and mixing in fluids. Lagrangian studies enable direct calculation of quantities like dispersion and provide a detailed view of the distribution of tracer particles in a fluid over time. That is not to say that mixing cannot be studied from the Eulerian perspective: Several studies have developed so-called "Eulerian mixing indicators" that use heuristics like the shear rate and the intersection of streamlines to quantify mixing solely based on velocity fields [Sturman and Wiggins, 2009; McIlhany and Wiggins, 2012]. However, studying mixing from the Eulerian perspective limits the complexity of the mixing phenomena that can be explored. Another scenario where Lagrangian studies may be preferred over Eulerian ones is when linking simulations and experimental data. A common technique in experimental fluid mechanics is particle tracking velocimetry and the injection of passive tracer dyes into fluid flows to visualize flow properties. In such cases, Lagrangian models may align better with experiments if their outputs are passive particle trajectory data.

1.2.2 The Projection Method

While a wide variety of techniques exist for numerically solving the incompressible Navier-Stokes equations, in this thesis, we focus on one such method called the Projection Method since the CFD codebase utilized here is based on this technique [Kochkov et al., 2021]. The

Projection Method, developed by Chorin [1968], enables efficient calculation of numerical solutions to the Navier-Stokes equations by decoupling the computation of the velocity and pressure. The first step in the Projection Method is to calculate an intermediate velocity field by neglecting the pressure term in the Navier-Stokes momentum equation. The pressure field is then found by enforcing the divergence-free condition. This pressure is then used to solve for a correction to the velocity field. The procedure is repeated with the corrected velocity field serving as the intermediate velocity field. Corrections are iteratively computed until a convergence criterion is reached.

1.2.3 The Penalty Method

Many of the most interesting problems in fluid mechanics involve fluid-solid interactions. For example, CFD models of airplane flight must account for the dynamics between the air and the aircraft surface, just as simulations of flows in porous media must consider the interactions between the fluid and the solid obstacles embedded in the flow. Penalty Methods are a class of techniques used for simulating flows that contain solid particles, structures, or walls. In this work, we model fluid-structure interactions using the Brinkman Penalty Method [Brinkman, 1949a,b]. In the Brinkman Penalty Method, a forcing, or "penalty", term is introduced into the Navier-Stokes equations as

$$\frac{\partial u}{\partial t} = -u \cdot \nabla u - \nabla p + \frac{1}{\mathrm{Re}} \nabla^2 u + \lambda \chi (u_s - u) \tag{1.2}$$

to model the drag due to a solid obstacle. Here, λ is the penalization parameter, u_s is the velocity of the solid, and χ is a position- and geometry-dependent factor that is 0 outside of solid boundaries and smoothly transitions to 1 inside of solid boundaries [Khadra et al.,

2000]. The velocity field found from solving Equation 1.2 enforces the no-slip boundary conditions at the fluid-solid interface. The Brinkman Penalty method has been used extensively in flow modeling because it can be conveniently combined with most numerical techniques for solving the Navier-Stokes equations.

1.3 Inverse Design Techniques for Flow Control

One of the primary reasons that CFD as a field has garnered significant interest is that modeling flows with simulations reduces the need for expensive physical experiments and can aid in the development of improved experimental designs. As an example, in the aerospace industry, CFD practitioners simulate laminar-turbulent transition along airfoils to find more fuel-efficient airfoil geometries, which they then construct and test in wind tunnel experiments [e.g. Shams et al., 2020]. Despite one of the main motivations for modeling fluids being to solve important inverse design problems, much of the focus in CFD to date has been on solving forward modeling problems. This emphasis on forward problems is understandable, as inverse design problems are challenging and often intractable to solve due to the computationally intensive nature of many forward simulations in CFD and the vast parameter spaces that must be explored. However, new computational tools and methods have emerged over the past few decades that are making these inverse problems more tractable. The availability of tools to efficiently solve inverse problems will open up new avenues for design, optimization, and flow control for a wide range of applications in fluid dynamics.

1.3.1 Forward vs. Inverse Problems

Solving a forward problem in the context of CFD means designing a computational model $\mathcal{F}$ that takes in flow parameters $\vec{X}$ and, when run, outputs $\mathcal{F}(\vec{X}) = \vec{Y}$. The canonical example would be a Navier-Stokes solver $\mathcal{F}$, flow parameters $\vec{X}$, and the fluid velocity field corresponding to the given flow parameters $\vec{Y}$. On the other hand, solving an inverse problem in the context of CFD, means finding the specific flow parameters $\vec{X}^*$ such that when the model $\mathcal{F}$ is run with those parameters, it outputs $\mathcal{F}(\vec{X}^*) = \vec{Y}^*$ for some specifically targeted $\vec{Y}^*$. This is a much more difficult problem: It contains all the same modeling challenges as the forward problem (i.e. how to design $\mathcal{F}$) with the additional complications that it requires some form of parameter space exploration to find $\vec{X}^*$ with no guarantee that $\vec{X}^*$ exists or that it is unique.

1.3.2 Flow Control

Flow control inverse problems can be categorized into two types — passive flow control and active flow control. In passive flow control problems, the design of the flow (e.g the geometries involved or the arrangement of obstacles in a flow) is modified to naturally induce desired flow characteristics without external energy input. This method shapes the flow's behavior and properties through structural changes alone. A historical example of passive flow control was the agricultural use of windbreaks to divert wind currents from crops. More modern examples include the design of airfoil geometries, called natural laminar flow airfoils, that minimize drag and have extended laminar flow regimes [Yagiz et al., 2012]. In contrast, in active flow control problems, energy is used to dynamically alter flow behavior. A classic example of active flow control is the use of suctioning on airfoils as a form of

boundary layer control [Moghaddam and Banazadeh Neishabouri, 2017].

As in standard optimization problems, flow control problems are defined by an objective function, design parameters, and constraints. The objective function quantitatively defines the target behavior of a system (e.g. reducing drag or enhancing mixing). The design parameters are the variables that can be tuned to optimize the objective function, and the constraints limit the range or conditions under which the design parameters can be varied.

One approach for solving inverse problems in fluid mechanics has involved the use of adjoint-based methods, which convert PDE-constrained optimization problems to boundary value problems [Plessix, 2006]. Some examples of inverse problems that have successfully been solved with adjoint methods include the work by Roper et al. [2008] and Ogawa and Kawahara [2003]. However, using adjoint methods can be challenging for many problems in general because of the need to derive optimality conditions.

More recently, machine learning (ML) has been employed to solve inverse design problems in the physical sciences. The general approach taken by most studies has been to train surrogate ML models that emulate the results of more computationally intensive physics-based simulations. Here, the design parameters are provided as inputs to the ML model. Once the model is trained, gradient-based optimization techniques can be used to identify the input parameters that yield the desired output. Two examples include the use of graph neural network fluid simulators to manipulate fluid flows and optimize the design of airfoil shapes to minimize drag [Allen et al., 2022] and the use of reinforcement learning to design active flow control strategies to stabilize vortex alleys and reduce drag in simulations of the Karman Vortex Street [Rabault et al., 2019]. Two other passive flow control problems that have been approached with ML that will be discussed at length in this thesis are flow

sculpting [Stoecklein et al., 2017] and catheter design [Zhou et al., 2024].

The use of ML to solve scientific inverse problems comes with several drawbacks. First, the training of ML emulators demands large amounts of high-quality data, either from simulations or experiments. The process of obtaining training data may itself be computationally expensive. Furthermore, if the training data does not encompass the full parameter space of the design problem, the emulator may fail to generalize effectively. Finally, a general problem with the use of ML is that learned ML algorithms are not interpretable. Standard ML models trained to emulate physics do not inherently respect the conservation laws and symmetries intrinsic to specific problems. In the past few years, the field of physics-informed machine learning has become an active area of research aimed at encoding knowledge of physical laws into ML models [Hao et al., 2023]. While important progress has been made in this area, this thesis advocates for and describes the development of new techniques for solving inverse problems based on automatic differentiation that make direct use of physics and eliminate the need to learn from data at all.

1.4 Differentiable Physics Simulations

Differentiable physics is an emerging computational paradigm that enables the computation of exact gradients from physical simulations. The ability to compute gradients is important in optimization and inference problems. Four main methods exist for calculating gradients computationally — hard-coding analytical gradients, numerical differentiation, symbolic differentiation, and automatic differentiation. Hard-coding analytical gradients is precise but is quite restrictive as a methodology. Numerically computing gradients via finite differences is straightforward to implement with low computational costs but is sub-

ject to round-off errors and numerical instabilities. Symbolic differentiation methods work by decomposing functions into simpler operations and using differentiation rules to compute exact expressions for the gradients via the chain rule. For functions composed of long chains of operations, symbolic derivative expressions quickly explode in complexity and become computationally intensive to evaluate. Evaluating higher order derivatives only increases errors in the case of finite differences and the complexity of expressions in the case of symbolic differentiation [Gunes Baydin et al., 2015].

Automatic differentiation (AD) is capable of achieving machine precision in gradient computations without the complexity explosion characteristic of symbolic methods [Gunes Baydin et al., 2015]. AD is implemented by placing every operation involved in a calculation on a computational graph. Since each of these primitive operations has a defined analytical gradient, gradients at any point in the computational graph can be efficiently computed using the chain rule. In contrast to symbolic differentiation, it works with numerical values and not mathematical expressions. Automatic differentiation can be run in two modes — forward mode and reverse mode. Importantly, the computational cost of the gradient computation is independent of the number of parameters, making AD a powerful tool for solving high-dimensional optimization problems. The computational complexity of each gradient sweep is also directly proportional to the computational complexity of a forward evaluation. Therefore, it is also less expensive to compute gradients with automatic differentiation than it is with finite differences.

Backpropagation, the well-known algorithm that underpins the training process of modern ML models, is a special case of reverse-mode automatic differentiation [Hecht-Nielsen, 1989]. Backpropagation has made taking gradients with respect to loss functions

through neural networks (networks of weights and biases) efficient enough to enable extremely large neural networks to be trained via gradient descent algorithms. Although AD is most widely known in the context of ML, there is nothing inherent to AD that limits it to being used for differentiating through the computational graphs associated with neural networks. Maclaurin et al. [2015] did pioneering work in making standard numerical computing libraries, like NumPy, natively compatible with automatic differentiation. This work opened up the possibility of rewriting scientific computing codes to be fully differentiable.

Several hardware and software advances over the past few years have enhanced the efficiency of AD algorithms. On the software end, the development of PyTorch and JAX, high-level programming languages that abstract away much of the complexity that goes into putting operations on a computational graph and making code fully differentiable, have lowered the barriers for experimentation with differentiable programming [Paszke et al., 2017; Bradbury et al., 2018]. With these packages, over the last few years, AD has been leveraged to solve a new class of problems that do not require learning from data — computing exact gradients through scientific simulations. In this workflow, gradient information can be used for solving optimization problems and high-dimensional inference problems. Hamiltonian Monte Carlo methods, for example, use gradients to sample posteriors more effectively.

Although the use of automatic differentiation in the physical sciences has, to date, been quite limited, some recent studies have begun to showcase its wide-ranging potential. In photonics, automatic differentiation has been used to optimize the dispersion of a crystal waveguide [Minkov et al., 2020]. In cosmology, it has been used to efficiently compute

gradients with respect to cosmological likelihoods in order to accelerate parameter estimation [Campagne et al., 2023; Balkenhol et al., 2024]. In molecular dynamics, it has been used to tune self-assembly and design self-assembling patchy particles [Goodrich et al., 2021; King et al., 2023], design reaction pathways to control colloidal disassembly [Krueger et al., 2023], and control non-equilibrium systems [Engel et al., 2023]. In all of these works, physics-based simulations are constructed by piecing together primitive operations that are fully differentiable.

Importantly, writing simulations to be differentiable does not have to compromise the integrity of the simulation: Most differentiable codes replicate the results of standard simulation codes quite well [Schoenholz and Cubuk, 2021]. The main considerations for developing differentiable simulations are that (1) the computational graph must stay fixed through every optimization iteration, and (2) some operations that are inherently not differentiable (e.g. delta functions, hard sphere potentials, and histograms) must be replaced by smoother approximations. Neural networks themselves fall into the class of fully differentiable functions. As universal function approximators, neural networks provide enormous flexibility in the development of differentiable simulations [Cybenko, 1989]. Any functions that are too complex to be coded explicitly can easily be emulated by neural networks and integrated into simulations.

In this work, we build upon JAX-CFD, which implements a fully-differentiable Navier-Stokes solver. JAX-CFD was developed by Kochkov et al. [2021]. Previously, it has been used to accelerate turbulence modeling, but it provides a general framework for optimizing Eulerian objective functions — not Lagrangian ones, however. Recently, there have been other efforts toward building differentiable fluid dynamics frameworks. Another differ-

entiable codebase called JAX-FLUIDS was designed for Eulerian simulations of compressible, two-phase flows [Bezgin et al., 2023; Bezgin et al., 2024]. A framework called ODiL (Optimizing a Discrete Loss) has been developed in JAX for solving PDEs and has been applied to solving the inverse problem of flow reconstruction [Karnakov et al., 2022; Karnakov et al., 2023]. A differentiable implementation of Smoothed Particle Hydrodynamics models called JAX-SPH provides a meshfree solver of the Navier-Stokes equations that is well-equipped for modeling fluids with open or complex wall boundaries [Toshev et al., 2024].

1.5 Objectives

The objectives of this thesis are to (1) describe the development of a fully-differentiable framework for Lagrangian fluid dynamics in JAX and (2) demonstrate the versatility of this framework by applying it to solve a variety of complex fluid inverse problems. Methodologically, this work develops a framework for integrating the full suite of differentiable molecular dynamics tools from JAX-MD with a differentiable Navier-Stokes solver that accounts for the presence of solid obstacles. This enables significantly more complex flow control problems to be solved with automatic differentiation than was previously possible with JAX-CFD alone.

The differentiable Lagrangian fluid framework is first applied to the problem of optimizing dispersion and mixing in porous media. To demonstrate the versatility of this method for solving flow control problems for systems with non-steady velocity fields, we apply the framework to optimize mixing in journal bearing. This thesis describes ongoing work on flow sculpting, a very high-dimensional optimization problem for finely controlling the

configuration of tracer particles in a flow. It also describes preliminary work for designing catheter geometries to reduce the upstream swimming of bacteria that cause infections in patients. The flow sculpting problem and the catheter design problem were chosen specifically because they have been previously approached using ML techniques [Stoecklein et al., 2017; Zhou et al., 2024]. Thus, they provide direct points of comparison between ML and the automatic differentiation techniques presented here.

In a time when the physical sciences, and the field of fluid mechanics in particular, is grappling with what the role of machine learning should be in the field [Brenner et al., 2019; Brunton et al., 2020; Brenner and Koumoutsakos, 2021], one of the broader objectives of this work is to argue that we have not yet exhausted the potential of purely physics-based techniques for solving inverse design problems in fluid mechanics. In this work, we demonstrate through examples that we can solve flow control problems in their full complexity — that is, without replacing any of the physics with the use of machine learning or reduced order models. This work represents some of the first uses cases of differentiable fluid mechanics for solving flow control problems involving complex fluids. It sheds light on the immense computational capability we have without replacing the laws of physics with neural networks, which are uninterpretable and susceptible to generalization error.

The outline of this thesis is as follows. Chapter 2 details the development of a differentiable framework for Lagrangian particle dynamics. Chapter 3 describes the validation of this methodology on Taylor Dispersion, a canonical problem in fluid mechanics. Chapter 4 describes the application of this methodology to optimizing mixing in steady flows through porous media. Chapter 5 describes the application of this methodology to a non-steady flow problem, mixing in Journal bearing. Chapter 6 describes ongoing work on solving

high-dimensional flow sculpting optimization problems. Chapter 7 describes preliminary work for simulating the dynamics of bacteria in flows in order to optimize catheter geometries that prevent the upstream swimming of bacteria. In the conclusion in Chapter 8, the key takeaways from this research and the avenues for future exploration are discussed.

The material in Chapters4 and Chapter 5 of this thesis reflect work that, at the time of writing, has been submitted for publication.* My thesis research was advised by Dr. Mohammed Alhashim, a postdoctoral fellow, and Professor Michael Brenner. Chapter 2, Chapter 3, Chapter 4, and Chapter 6 of this thesis reflect work that was primarily led by me. The work in Chapter 5 was led by Dr. Mohammed Alhashim, which I contributed to by implementing the differentiable Lagrangian particle tracking framework. The work in Chapter 7 is an ongoing collaboration between me, Dr. Alp Sunol, and Dr. Mohammed Alhashim.

*The submitted paper is available on the arXiv: https://arxiv.org/pdf/2403.06257.pdf

2

Differentiable Lagrangian Fluid Simulations

The goal of developing differentiable Lagrangian fluid simulations is to enable us to define arbitrary objective functions based on particle trajectories in a flow that we can differentiate through and optimize over. We use JAX-CFD, a differentiable solver of the Navier-Stokes

equations, to simulate fluid velocity fields. To build a toolkit for differentiable Lagrangian fluid mechanics, we use JAX-MD, which is a differentiable molecular dynamics engine [Schoenholz and Cubuk, 2021], to implement Lagrangian particle tracking, to model Brownian dynamics and other custom molecular dynamics, and to compute pairwise interactions between particles.

2.1 Steady Flow Problems

For problems that involve steady flow fields, Lagrangian particle tracking can be implemented by sequentially coupling a CFD simulation and an MD simulation. First, the steady velocity field, $\mathbf{u}$, can be solved using JAX-CFD. The trajectories of tracer particles in the flow can then be modeled using the Langevin Equation with discrete time steps

$$\frac{d\mathbf{X}^i}{dt} = \mathbf{u}(t) + \nabla \sum_{j=1}^{N} U_j\left(\mathbf{X}^i, r_j\right) + \sqrt{2D_m \xi(t)} \tag{2.1}$$

where $U_j(\mathbf{X}^i, r_j)$ is the repulsive potential due to the j_{th} solid obstacle in the flow with radius r_j at a point $\mathbf{X}^i$, D_m represents the molecular diffusivity, and $\boldsymbol{\xi}(t)$ represents the white noise term.

The different steps required to implement Langevin dynamics can be visualized through the simple example of channel flow.

1. **Lagrangian Particle Tracking:** If we consider only advection and no diffusion, passive tracer particles can be simulated from a velocity field using

$$\frac{d\mathbf{X}^i}{dt} = \mathbf{u}(t). \tag{2.2}$$

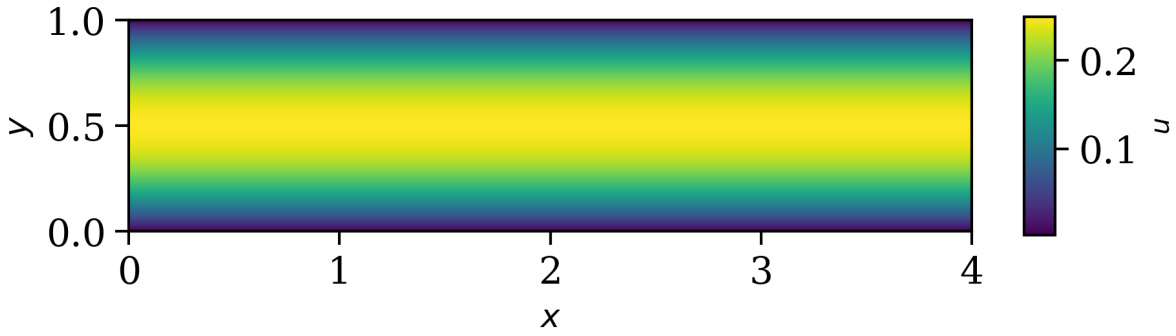

Figure 2.1: The fluid velocity field $u(x, y)$ corresponding to flow through a channel

We can simulate the particles in the flow by advecting particles with a position-dependent force. The force field is set to be the velocity field from the differentiable Navier-Stokes solver, and for steady flow problems, this velocity field does not change in time. An example of such a velocity field, $u(x, y)$ solved by JAX-CFD for flow through a two-dimensional channel is shown in Figure 2.1. For this flow, $v(x, y) = 0$ everywhere. At each time step, the force on each particle is interpolated from the velocity field and is used to advect the particles. An example of simulating particle trajectories according to the channel flow velocity field is shown in Figure 2.2.

2. **Molecular Diffusion:** If we introduce Brownian dynamics into the simulation, tracer particles can be simulated using

$$\frac{d\mathbf{X}^i}{dt} = \mathbf{u}(t) + \sqrt{2D_m}\xi(t). \tag{2.3}$$

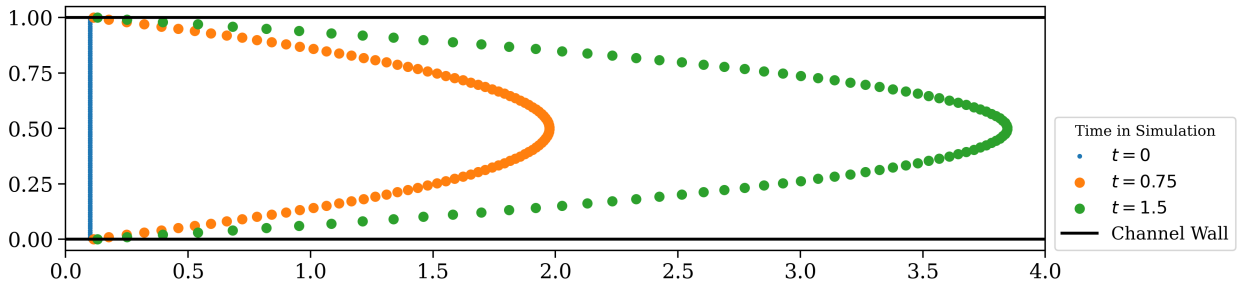

Figure 2.2: The evolution of Lagrangian tracer particles in flow through a channel

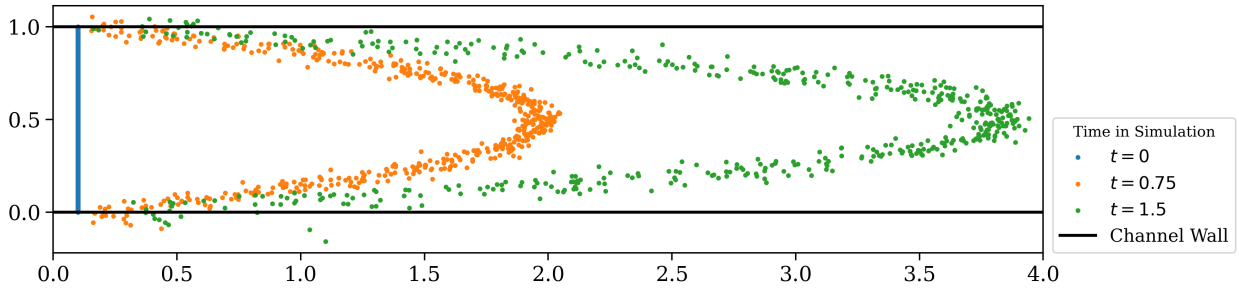

Figure 2.3: The evolution of tracer particles diffusing in a flow through a channel with no reinforcement of the wall boundaries

The results of including molecular diffusion in the simulation are shown in Figure 2.3. With this set up, we see that tracer particles diffuse into impermeable (zero velocity) regions of the flow (e.g. outside the channel walls) because the walls are not reinforced in the MD simulation.

3. **Particle Interactions:** To simulate the impermeability conditions at the walls, we line the wall with particles that within a small radius, repel the Lagrangian tracer particles. We model the interaction between each of the wall particles and the Lagrangian tracer particles using the repulsive side of the Morse potential, which is

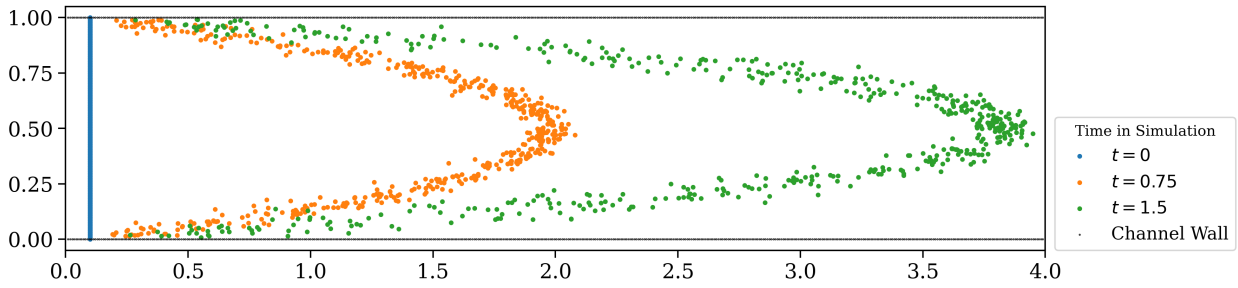

Figure 2.4: The evolution of tracer particles diffusing in a flow through a channel with the wall boundaries reinforced by static, repulsive particles

given by

$$U(r) = \begin{cases} \frac{1}{2}k\left(r - r_0\right)^2 - D_0, & r < r_0 \\ D_0\left(e^{-2\alpha(r-r_0)} - 2e^{-\alpha(r-r_0)}\right), & r \geq r_0 \end{cases}. \tag{2.4}$$

We set r_0 to be the radius of each wall particle. In practice, the number and radii of the wall particles has to be tuned for each simulation and will depend on the molecular diffusivity and time step. We modify the particle time stepping function to treat these wall particles as immobile. Figure 2.4 shows the result of using Equation 2.1 with this potential to simulate the motion of tracer particles in the channel flow. In contrast to Figure 2.3, the tracer particles remain confined between the two walls throughout this simulation.

Note that although the main MD feature used in this work is the ability to model interactions between particles to reinforce fluid impermeability conditions at solid interfaces, the framework developed here is compatible with the full suite of molecular dynamics tools available in JAX-MD. We first validate this simulation toolkit by demonstrating that it can

replicate the results of Taylor Dispersion Theory (Chapter 3). This simple framework alone enables us to explore a range of inverse design problems involving steady flows, from flow through porous media in Chapter 4 to flow sculpting in Chapter 6. For the catheter design problem in Chapter 7, a steady flow set-up is used, but we write a custom molecular dynamics function to simulate the motion of bacteria based on the vorticity of the flow field.

2.2 Non-Steady Flow Problems

We extend this framework to problems where the velocity field is not steady in time. To mitigate the memory constraints associated with having to store the velocity field at every step of the CFD simulation in order to later simulate tracer particles moving through the flow, we modify the code such that CFD steps and MD steps are taken in tandem and only MD particle trajectories are stored through the simulation. In these simulations, in addition to having to consider the spatial resolution of the CFD velocity field, which we interpolate to implement Langevin dynamics, we also have to consider the temporal resolution of the evolution of the velocity field in time. We describe the optimization of the design of a non-steady flow in the journal bearing example in Chapter 5.

3

Taylor Dispersion

To validate the differentiable framework for simulating diffusing Lagrangian tracer particles in a flow field, we apply it to a canonical problem in fluid mechanics called Taylor Dispersion.

3.1 Background on Dispersion

Dispersion characterizes the spread of a solute in a flow, and it is driven by the combined effects of molecular diffusion and the background velocity field. The diffusion equation in one dimension is given by

$$\frac{\partial c}{\partial t} = D\frac{\partial^2 c}{\partial x^2} \tag{3.1}$$

where c is the concentration of the solute and D is the diffusivity of the solute in the medium. Under the initial condition that all the solute is released at $x = 0$ at $t = 0$, the diffusion equation is solved by

$$c(x,t) = \frac{N}{\sqrt{4\pi D t}} e^{-\left(\frac{x^2}{4Dt}\right)}, \tag{3.2}$$

where N is the total amount of solute. This solution is a Gaussian with a variance that evolves with time as $\sigma^2 = 2Dt$. In n dimensions, $\sigma^2 = 2nDt$. For purely diffusive systems, the relevant diffusion coefficient is D_m, the molecular diffusivity, which is related to the molecular properties and surrounding temperature as

$$D_m = \mu k_B T \tag{3.3}$$

where μ is the mobility and T is the temperature.

In many flows, it is desirable to model both the effects of advection and diffusion. The Péclet number, Pe, is an important dimensionless quantity in these contexts. It quantifies the ratio of the advective transport rate to the diffusive transport rate. In a channel with

height h, the Péclet number is given by

$$\mathrm{Pe} = \frac{\bar{u}h}{D_m} \tag{3.4}$$

where $\bar{u}$ represents the average velocity and D_m represents the molecular diffusivity.

The advection-diffusion equation is given by

$$\frac{\partial c}{\partial t} + \mathbf{v} \cdot \nabla c = D\nabla^2 c. \tag{3.5}$$

When the velocity field is uniform with the same initial conditions as above, it is solved by

$$c(x,t) = \frac{N}{\sqrt{4\pi Dt}} e^{\left(\frac{x^2 - vt}{4Dt}\right)}, \tag{3.6}$$

which is now a Gaussian with a mean moving at the velocity of the flow and a standard deviation evolving in time as $\sigma^2 = 2Dt$. In set-ups where the velocity field is not uniform across the flow, as is the case in the following chapters on channel flow and flow through porous media, the solutions becomes more complicated and the width of the solute plume σ^2 reflects contributions from both the molecular diffusion and the gradients in the velocity field.

The relation between the width of the solute plume and the diffusion coefficient forms the basis for the Lagrangian study of dispersion. The dispersion coefficient, D^*, is given by

$$D^* = \frac{1}{2} \lim_{t \to \infty} \frac{d\sigma(t)^2}{dt}. \tag{3.7}$$

When the velocity field is zero or uniform, as in the cases above, $D^* = D_m$. In general,

dispersion is a tensorial quantity: The longitudinal, or axial dispersion, describes the spread of solute in the direction of the flow, and the transverse dispersion describes the spread of solute perpendicular to the direction of the flow.

3.2 Taylor Dispersion Theory

Taylor Dispersion Theory describes the mechanism by which laminar shear flows enhance the axial dispersion of solutes in a fluid. Here, we consider flow through a channel, one of the simplest shear flows, which has the parabolic velocity field shown in Figure 2.1.

Qualitatively, molecular diffusion causes particles traveling in a flow to jump across streamlines. In shear flows, crossing streamlines changes a particle's velocity. The characteristic parabolic concentration profile therefore becomes blurred as slow-moving particles near the wall diffuse inward and fast-moving particles near the center diffuse outward.

Specifically, Taylor Dispersion theory predicts that if we transform to a frame moving with the mean velocity of the flow, then on timescales $t \gg \frac{b^2}{D_m}$ the flow can be modeled as a pure diffusion process. That is, in the coordinate system given by

$$x' = x - \overline{u}t \tag{3.8}$$

where $\overline{u}$ is the mean velocity, the average solute concentration, $\bar{c}$ evolves according to

$$\frac{\partial \bar{c}}{\partial t} = D^* \frac{\partial^2 \bar{c}}{\partial x'^2}. \tag{3.9}$$

where

$$D^* = D_m \left(1 + \kappa \mathrm{Pe}^2\right). \tag{3.10}$$

This is exactly the one dimensional diffusion equation, matching Equation 3.1. Here, κ is a geometry-dependent factor that has been calculated for a wide variety of wall geometries. For flow between two infinite parallel plates, $\kappa = \frac{1}{210}$. This "effective diffusion coefficient" in the frame moving with the center of mass of the particles is the dispersion coefficient.

3.3 Simulations

In channel flows, Taylor Dispersion Theory provides a prediction of how the dispersion coefficient scales with the Péclet number. To validate the Lagrangian framework developed here, we perform simulations of particles diffusing in a channel at Péclet numbers ranging from 10 to 1000 with the goal of measuring the dispersion coefficient and comparing it to the theory. The simulation follows the procedure in Chapter 2.1, where we model the wall of the channel with repulsive particles. With this set-up, lower Péclet number simulations require denser packing of wall particles to confine the Lagrangian tracer particles. Figure 3.1 shows the evolution of diffusing tracer particles moving through a channel with the velocity field shown in Figure 2.1. At the first time point shown in the plot, we see the originally parabolic concentration profile of the tracer particles. By the third time point, the effect of the shear flow, which blurs out the concentration profile, becomes visible.

The goal of simulating the trajectories of the tracer particles in this flow is to calculate the dispersion coefficient of particles at each Péclet number and compare that with the scaling predicted by the theory. We calculate the axial dispersion coefficient as

$$D^* = \frac{1}{2} \lim_{t \to \infty} \frac{d\sigma_x^2(t)}{dt}. \tag{3.11}$$

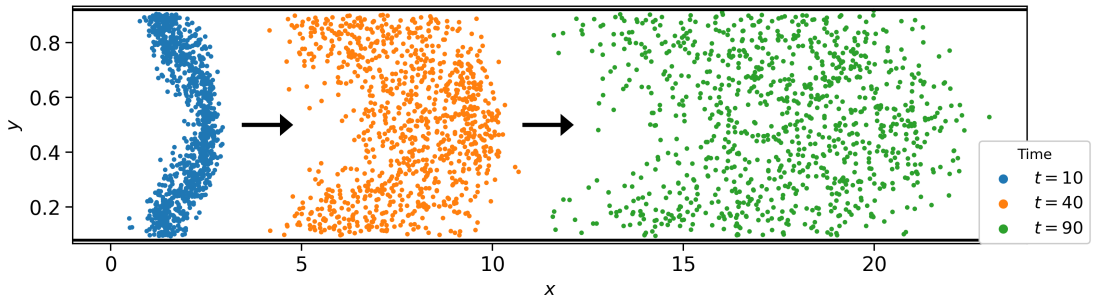

Figure 3.1: A simulation of Taylor Dispersion that shows the evolution of diffusing tracer particles in a channel at three different times

We can compute $\sigma_x^2(t)$ from the simulated particle trajectories as

$$\sigma_x^2(t) = \left\langle \left(x(t) - \overline{x(t)} \right)^2 \right\rangle . \tag{3.12}$$

In studies of dispersion, in order to obtain better statistical sampling and minimize the sensitivity to the local characteristics of the flow, it is common to compute lagged mean squared displacements (MSDs) [Calandrini et al., 2011].

Lagged Mean Squared Displacement

In the standard MSD calculation, displacements are calculated from a fixed reference point, typically $t = 0$. In contrast, lagged MSD calculations consider all particle displacements over specific time intervals or lags, τ, detached from any fixed reference point. The lagged MSD is thus given by

$$\mathrm{MSD}(\tau) = \left\langle [\mathbf{r}(t+\tau) - \mathbf{r}(t)]^2 \right\rangle \tag{3.13}$$

where

$$\mathbf{r}(t) = \mathbf{X}(t) - \langle \mathbf{X}(t) \rangle . \tag{3.14}$$

For N particles in a simulation with M timesteps of size Δt, if the lagged algorithm is used, the number of displacement data points that can be gathered for a lag time of $\tau = m\Delta t$ is $N(M - m + 1)$. If the straightforward approach of computing MSDs without lagging is used, the number of displacement data points that can be gathered for to a lag time of $\tau = m\Delta t$ is just N. Directly computing the displacements of all the particles, for every lag time, from every possible reference point in order to implement a lagged MSD would be quite computationally intensive. Instead, by algebraically manipulating the lagged MSD into the form

$$\begin{aligned} \mathrm{MSD}(\tau) &= \left\langle [\mathbf{r}(t+\tau) - \mathbf{r}(t)]^2 \right\rangle \\ &= \underbrace{\left\langle \mathbf{r}(t+\tau)^2 \right\rangle}_{\text{Term 1}} + \underbrace{\left\langle \mathbf{r}(t)^2 \right\rangle}_{\text{Term 2}} - \underbrace{2\left\langle \mathbf{r}(t+\tau)\mathbf{r}(t) \right\rangle}_{\text{Term 3}} \end{aligned} \tag{3.15}$$

a method for accelerating the calculation using a Fast Fourier Transform (FFT) becomes apparent. Here, Term 1 and Term 2 are straightforward to compute directly. Term 3 represents an autocorrelation, which can be calculated more efficiently using FFT algorithms. Differentiable FFT algorithms have already been written in JAX. Since the FFT introduces edge effects at high lag times, where there is greater noise, we discard the 5% of the data at the highest lag times when computing dispersion coefficients from particle MSDs.

To verify that this algorithm for computing lagged MSDs from particle trajectory data is accurate, we test how well it recovers the diffusion coefficient of a collection of Brownian particles diffusing in the absence of a velocity field. We simulate particle trajectories (examples shown in Figure 3.2a) at molecular diffusivities ranging from $D_m = 10^{-9} - 10^{-2}$ and demonstrate that by tracking the particle positions over time, we can recover their molec-

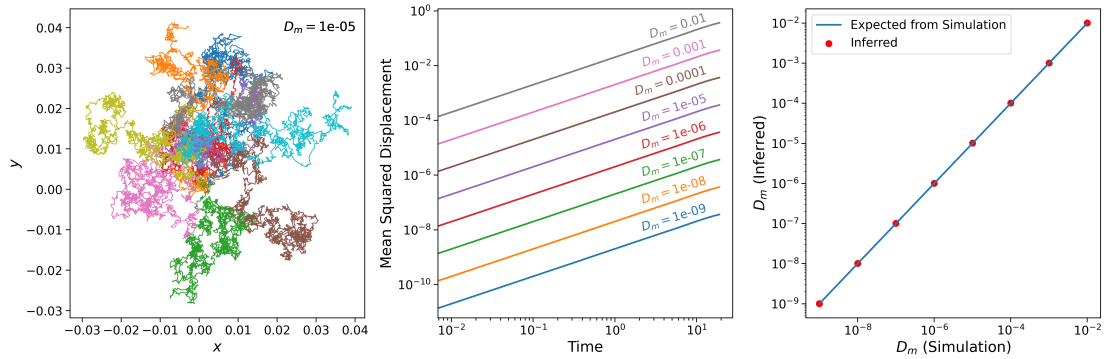

Figure 3.2: (a) The trajectories of Brownian particles with $D_m = 10^{-5}$ in the absence of a flow field, (b) the average MSD of the Brownian particles over time for various molecular diffusivities computed using the differentiable lagged MSD algorithm, (c) the actual molecular diffusivities set in the simulation vs. the molecular diffusivities inferred from the trajectory data

ular diffusivities. Figure 3.2b shows the MSD over time, calculated via the lagged MSD algorithm, for the Brownian particle trajectory data at each molecular diffusivity we considered. Figure 3.2c shows the molecular diffusivities inferred from the particle trajectories (calculated as the slope of the average MSD for each simulation), plotted against the actual molecular diffusivities set in the simulation. We find that the differentiable lagged MSD algorithm precisely recovers the molecular diffusivities. We use this algorithm to calculate the dispersion coefficients of particles flowing through a channel to compare our simulations with the predictions of Taylor Dispersion Theory, and we use it in Chapter 4 to calculate the dispersion coefficients for flows through porous media.

3.4 Results

After running the channel flow simulations across a range of Péclet number, we calculate the dispersion coefficients corresponding to each flow. Figure 3.3 plots MSD/2τ for the

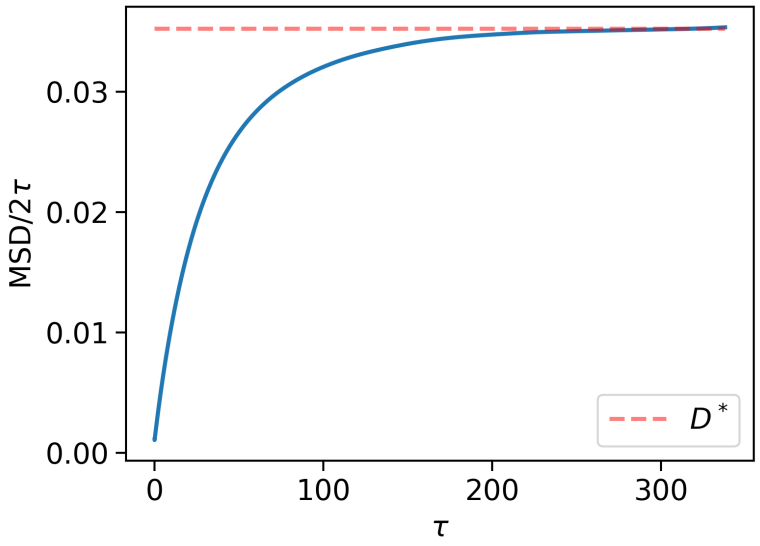

Figure 3.3: MSD$/2\tau$ vs. τ, the lag time, calculated with the lagged MSD algorithm from the trajectories of particles flowing through a channel (shown in Figure 3.1); the dispersion coefficient, $D*$, calculated based on the limiting behavior of MSD$/2\tau$ is marked with a dashed line

particle trajectories shown in Figure 3.1. For each simulation, we obtain the dispersion coefficient from the long-time behavior of this curve. In Figure 3.4, we plot Pe^2 vs. $\frac{D^*}{D_m} - 1$ where D_m is the molecular diffusivity we set in the simulation and D^* is the dispersion coefficient we calculate via the Lagrangian particle tracking procedure described in the previous chapter and the lagged MSD algorithm described here. The figure shows that we indeed recover the scaling that

$$\frac{D^*}{D_m} - 1 \propto \text{Pe}^2 \tag{3.16}$$

from Taylor Dispersion Theory. This validates that our differentiable implementation of Lagrangian dynamics matches the expected physics. In the following sections, we use this differentiable Lagrangian toolkit to solve optimization problems.

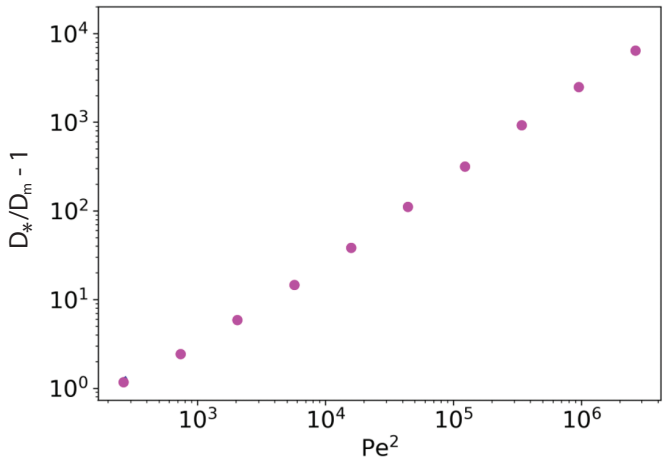

Figure 3.4: The ratio of the dispersion coefficient inferred from the channel flow simulations to the molecular diffusivities set in the simulation vs. the square of the Péclet number

A coding tutorial for the differentiable simulation of Taylor Dispersion described here is available on the JAX-IB GitHub repository.*

*Taylor Dispersion Tutorial: https://github.com/hashimmg/jax_ib/blob/main/jax_ib/notebooks/taylor_dispersion_demo.ipynb

4

Porous Media

Porous media are solid materials with a complex system of interconnected cavities, or pores, through which fluids can travel. The specific structure and geometry of porous media greatly affect their transport properties. There is an extensive theoretical, computational, and experimental literature on transport, dispersion, and mixing in porous media. While a wide variety of studies have modeled flow through porous media, very few have addressed

the inverse problem of designing porous media to achieve desired transport properties. At the time of writing, the only study known to us that has attempted to optimize the geometries of porous media and taken a similar approach to the work in this section is Nair et al. [2023]. They used genetic algorithms to optimize the relative positions of cylinders in a flow to control the wake and drag characteristics for a coastal defense application. In their work, after running their optimization to find optimal porous media designs, they simulate Lagrangian tracer particles moving through the designed porous media to study how these configurations affect particle sedimentation and erosion, the properties they specifically care about. Importantly, however, they cannot optimize for these specific Lagrangian characteristics directly, as we do here.

The lack of established methodologies for controlling and designing porous media is striking given that flows through porous media are ubiquitous in the natural world and are relevant to a wide range of science and engineering problems. For example, understanding porous media flows is not only important for studying the transport of water through soil and sedimentary rocks, which is essential to the process of aquifer recharge [Cattaneo et al., 2016], but also for designing industrial water filtration technologies. In the life sciences, research into porous media has enhanced our understanding of flows through the interstitial structures of the lung [Miguel, 2012] and of transport phenomena in biological tissues [Khaled and Vafai, 2003]. Models of porous media flows, in turn, are now being used to improve tissue bioengineering [German and Madihally, 2016] and to develop protocols for targeted drug delivery [Khanafer and Vafai, 2006]. Porous media have also gained attention for their potential usage in the process of carbon capture and sequestration [Farmahini et al., 2020].

4.1 Dispersion and Mixing in Porous Media

Dispersion in porous media has been studied extensively in the literature [Zuzovsky et al., 1983; Sahimi et al., 1986; Koch et al., 1989; Amaral Souto and Moyne, 1997]. In pressure-driven flow through porous media, mixing is mediated by molecular diffusion and mechanical dispersion, specifically the stretching and folding of the fluid around the solid obstacles in the flow [Le Borgne et al., 2015; Villermaux, 2019]. In porous media, the velocity gradients arising from the no-slip conditions at the fluid-solid interfaces contribute to overall dispersion in an analogous way to how the velocity gradients from channel walls mediate the Taylor Dispersion effect described in the previous section. The tortuosity of the porous medium also acts to divide and reunite flow paths, which further influences mixing in the flow.

Inducing mixing is especially challenging at low Reynolds number, where flows are laminar and viscosity dominates over inertia, and at high Péclet numbers, where there is little molecular diffusion and jumping across streamlines. Theoretically and experimentally, it has been shown that chaotic dynamics can be generated in two-dimensional unsteady flows and three-dimensional steady flows through porous media [Aref, 1984; Aref et al., 2017; Dentz et al., 2023]. In steady flows through two-dimensional porous media, however, it is impossible to generate chaotic dynamics. Thus, at low Reynolds numbers, mixing in these systems can neither rely on turbulence nor chaoticity; rather, it relies primarily on the mechanical dispersion properties of the porous medium geometry (e.g. how well it stretches and folds the fluid). Designing such porous media, also referred to as laminar static mixers in industrial applications, is thus an interesting and important passive flow control prob-

lem.

Manipulating the topology of porous media to control the transport properties is challenging. Full porous media simulations can be computationally expensive, and the flows are governed by many parameters (e.g. Reynolds number, Péclet number, number of obstacles, shape of obstacles, size of obstacles, and position of obstacles). A number of papers have argued that mixing in fluids is best studied from the lens of dynamical systems theory with the trajectory of particles in the fluid representing the dynamical system [Metcalfe et al., 2022; Heyman et al., 2021]. While no papers, to our knowledge, have optimized porous media to achieve specific dynamical system properties, any attempt would require optimizing over particle trajectory data. Other objective functions for quantifying mixing are statistical in nature (e.g. the dispersion coefficient). When simulations include the effects of molecular diffusion, the particle trajectories become inherently stochastic, which introduces noise into any objective function calculated from the trajectory data and makes the optimization problem more complex. These challenges were some of the motivations for the development of the fully differentiable Lagrangian fluid dynamics framework presented here. In this chapter, we demonstrate that with fully-differentiable Lagrangian simulations of flow through porous media, we are able to use gradient-based optimization techniques to design porous media topologies that optimize dispersion and mixing. The codebase developed here, however, is robust for optimizing arbitrary Lagrangian loss functions.

4.2 Simulations

Here, we simulate and optimize the geometry of bidisperse periodic porous media. Our simulations have three design parameters — the cylinder diameter ratio λ where $0 \leq \lambda \leq 1$,

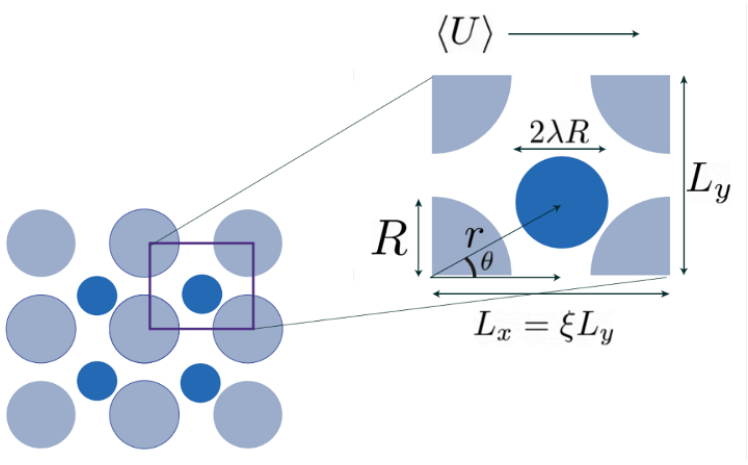

Figure 4.1: Diagram of the flow set-up for the periodic porous media

and the position (x, y) of the interior cylinder. The Reynolds number of the flow in this system is given by

$$\mathrm{Re}_M = \langle U \rangle D_M / \nu. \tag{4.1}$$

We set the Reynolds number of our simulations be 10^{-4} to model laminar static mixing. A diagram of the flow set-up is shown in Figure 4.1.

The velocity field as a function of the three flow parameters is calculated using a differentiable implementation of the Brinkman Penalty Method built on top of JAX-CFD [Alhashim and Brenner]. A sample velocity field generated with this code is shown in Figure 4.2

The differentiable simulation that we optimize over consists of both simulating the fluid velocity field and using Langevin dynamics (Equation 2.1) to simulate the trajectories of

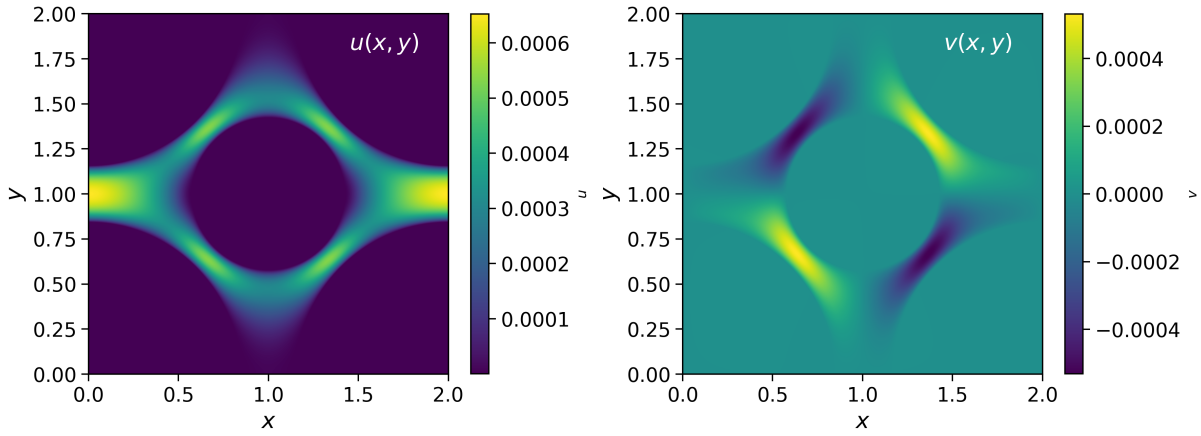

Figure 4.2: The fluid velocity field for flow through a periodic porous media solved via JAX-CFD: (a) $u(x, y)$ and (b) $v(x, y)$

diffusing particles in the flow. To reinforce the impermeability of the cylindrical posts in the MD simulation, we use a procedure similar to the wall modeling in the channel flow examples in Chapter 2 and Chapter 3. Here, each cylindrical post is modeled using the repulsive side of the Morse Potential (Equation 2.4) with r_0 equal to the radius of each cylinder. The repulsive MD particles used to represent the cylindrical posts are made immobile in the simulation. The molecular diffusivity of each simulation is set based on the desired Péclet number. We initialize the particles in the inlet between the corner cylindrical posts in a vertical line at $x = 0$.

One of the goals of this optimization problem is to show that we can compute meaningful gradients through stochastic particle trajectories. In Figure 4.3, which shows the trajectories of particles flowing through the same porous media at different Péclet numbers, the noise in a $\mathrm{Pe} = 1$ simulation can be compared to the noise in the $\mathrm{Pe} = 1000$ simulation through the same porous medium. In the following section, we report the optimization of dispersion coefficients in porous media with a Péclet number of 1, and we report the opti-

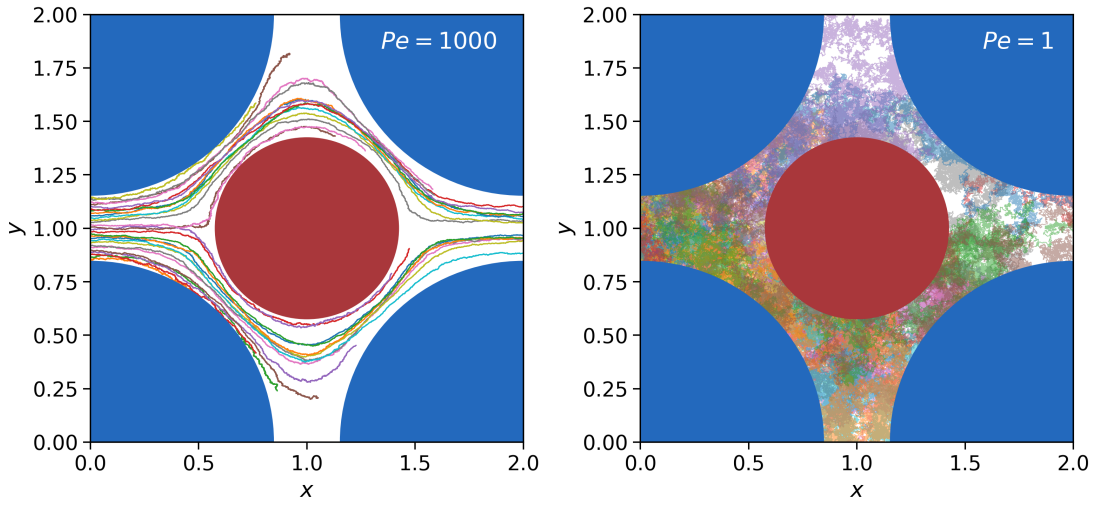

Figure 4.3: (a) Sample particle trajectories in a flow through a periodic porous medium at $\mathrm{Pe} = 1000$, (b) sample particle trajectories in a flow through a periodic porous medium at $\mathrm{Pe} = 1$

mization of short-time mixing indicators at higher Péclet numbers.

4.3 Optimizations

All of the optimizations performed here make use of IPOPT, an interior point optimization library, which implements gradient-based non-linear optimization algorithms [Wächter and Biegler, 2006].

4.3.1 Dispersion

To optimize dispersion in porous media at a Péclet number of 1, we fix the Péclet number across the different porous media topologies explored during the optimization procedure by computing the mean velocity from the CFD simulation and scaling the molecular diffu-

sivity to match the desired Péclet number. Since each simulation has a different molecular diffusivity, the objective function we use is

$$\mathcal{L} = D^*/D_m, \tag{4.2}$$

which is designed so that the optimizer maximizes the ratio of the dispersion coefficient to the molecular diffusion. We heretofore refer to $\mathcal{L}$ as the scaled dispersion coefficient. We calculate the dispersion coefficient using the differentiable lagged MSD algorithm described in Section 3.3.

The time it takes for particles to reach the diffusive regime, where we can calculate the dispersion coefficient, scales with Pe^2. As such, directly optimizing dispersion through full Lagrangian particle simulations, requires us to simulate extremely long particle trajectories. We optimize through simulations of 1,200 tracer particles traveling through the porous media for 1.5 million time steps with a time step size of $dt = 0.001$. Since dispersion is a statistical quantity, increasing the number of tracer particles and the number of time steps both reduce noise in the dispersion coefficient calculation.

Figure 4.4 shows the results of the optimization for a volume fraction of 0.7 with the images showing the behavior of the tracer particles at different optimization iterations. Through the iterations of optimization, we increase the dispersion coefficient by over 40% from the initial guess configuration. The optimization largely agrees with what we expect should be the optimal solution, although there may be some degeneracy. In the plot of the porous medium geometry in Step 1 of the optimization, the channel is blocked by the interior cylindrical obstacle so that the particles are confined. This explains why the scaled dispersion coefficient is less than 1 at this step: The confinement of the tracer particles

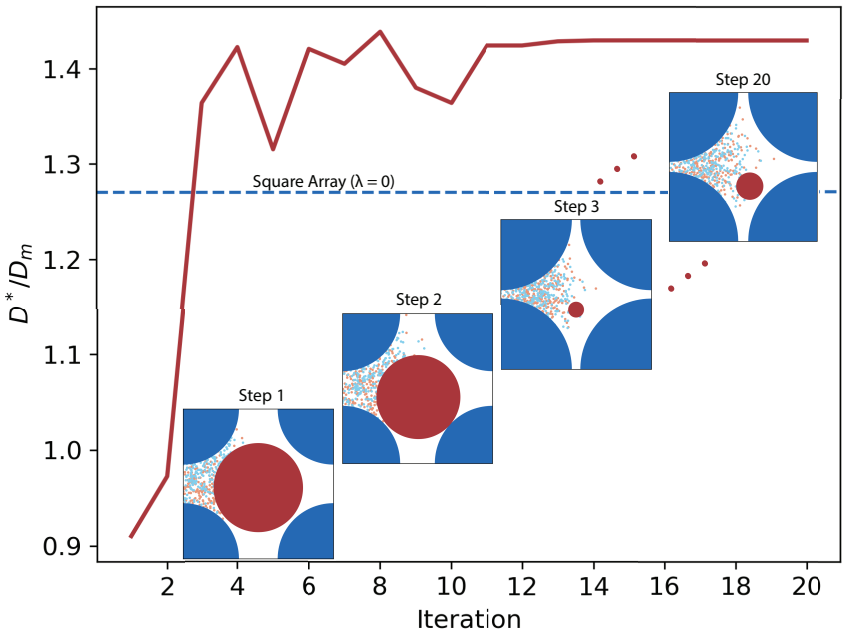

Figure 4.4: Optimization trajectory of the scaled dispersion coefficient for porous media with a volume fraction of 0.7 and a Péclet number of 1. The dashed line represents the scaled dispersion coefficient for the monodisperse square array.

makes their dispersion coefficient less than their molecular diffusivity. As the optimization continues, the configuration changes so that the inlet is no longer blocked and so that the interior cylinder does not block the pathway from the inlet to the outlet. The dashed line in Figure 4.4 shows the scaled dispersion coefficient for a monodisperse square array at the same volume fraction of 0.7. We see that the optimal configuration we find for the bidisperse periodic porous media also has a greater dispersion coefficent than that of the monodisperse square array.

The results here show that we are able to optimize through the full Lagrangian fluid sim-

ulations (of over 1 million time steps) required to exactly calculate dispersion in porous media. We do so for porous media at a Péclet number of 1, which is a very noisy system (see Figure 4.3), and this therefore demonstrates the robustness of the gradients. In general, mixing is harder to induce at higher Péclet numbers when advection dominates diffusion, so it would be of greater interest to optimize dispersion in higher Péclet number simulations. However, the length of the simulations (the number of time steps for the particles to reach the diffusive regime) has to be scaled with the Péclet number. We find that the gradient computation becomes increasingly unstable over these long simulations ($\mathrm{Pe} > 5$), which limited the range of Péclet numbers we could consider for the dispersion optimization. The problem of gradient explosion through stochastic but non-chaotic physical simulations has been studied by Metz et al. [2022] and remains not well-understood.

4.3.2 Short-Time Mixing Indicators

The optimization of the dispersion coefficient itself is quite computationally intensive, and we are unable to optimize the dispersion coefficient through high Péclet number simulations due to the instability of the gradient over long simulations. However, Figure 4.3, which compares a $\mathrm{Pe} = 1000$ simulation to a $\mathrm{Pe} = 1$ simulation through the same porous media reveals an important complementarity. At low Péclet numbers, the simulation is so noisy from the contributions of molecular diffusion that analyses of very long trajectories are required to disentangle the roles of mixing via diffusion and mixing via mechanical dispersion in the porous medium. That is, any short-time indicators of mixing will be too noisy in low Péclet number flows to be useful in identifying porous medium designs that optimize mixing. However, we have shown that we can increase mixing significantly at low

Péclet numbers using the long-time dispersion coefficient. At high Péclet numbers, porous media topologies greatly affect the mixing properties of flows, and this is visible even at short times. Therefore, to optimize mixing at high Péclet numbers, we can optimize short-time mixing indicators rather than the long-time dispersion coefficient.

This section investigates the use of short-time mixing indicators as a proxy for the dispersion coefficient. We demonstrate how we can optimize these mixing indicators in high Péclet number flows. In the literature, several short-time Lagrangian methods for quantifying the degree of mixing based on particle trajectories have been proposed. For example, Stone and Stone [2005] quantify mixing in rotating droplets by using the average distance between two groups of particles initialized apart from each other. Other methods look at residence time distributions that quantify the amount of time Lagrangian particles spend in stagnant regimes of the flow [Baléo and Cloirec, 2000].

In this work, we consider two short-time mixing indicators. We simulate two colors of passive tracer particles — the first group initialized in the top half of the unit cell and the second group initialized on the bottom half of the unit cell. The first mixing indicator that we minimize is the distance between the center of mass of the two species of particles. The second mixing indicator we minimize is the average distance between each particle and the particles of the other species [Stone and Stone, 2005]. Figure 4.5 shows an example of different time steps in a simulation where the mixing of the two species of particles can be visualized. The particles are initialized along the center line of the flow with the half of the particles colored green and the other half colored blue. Over time, we see that due to the convergence of streamlines, the particles begin to mix.

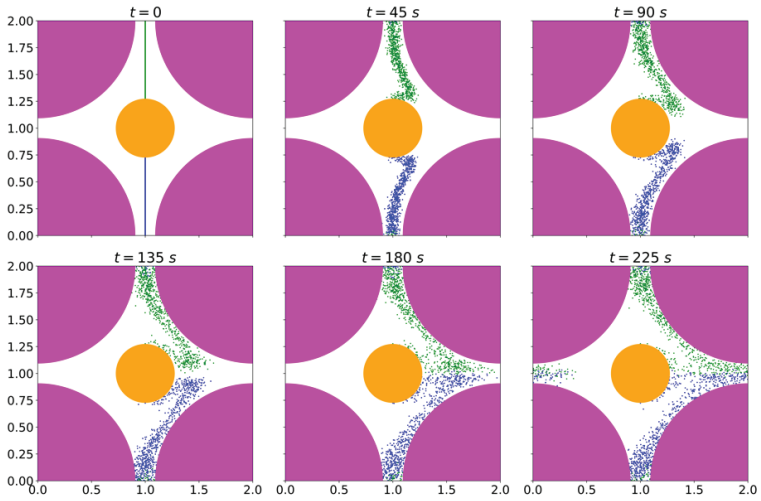

Figure 4.5: Sample simulation of two groups of passive tracer particles initialized separately mixing in a periodic porous media over time

We optimize over both the center of mass loss function and the average distance loss function using the gradient-based optimizer from IPOPT. The simulations we optimize over are significantly shorter in length than in the dispersion case, which had over 1 million time steps. Here, we simulate 5000 tracer particles (2500 in each group) for 5000 time steps with a time step size of 0.01. Figure 4.6 shows the results of minimizing the center of mass loss function for porous media at a fixed Péclet number of 100. We see that we are able to reduce the center of mass loss function by over a factor of two through the optimization. Although it is too computationally intensive to optimize over the scaled dispersion coefficient directly, we can run forward simulations to calculate how the scaled dispersion coefficient for the porous media changes as the geometry is optimized. The scaled dispersion coefficient corresponding to the porous medium geometry at every optimization iteration is plotted in Figure 4.6. We see that as the center of mass distance objective function is minimized, the scaled dispersion coefficient is increased by over 20% from the initial guess.

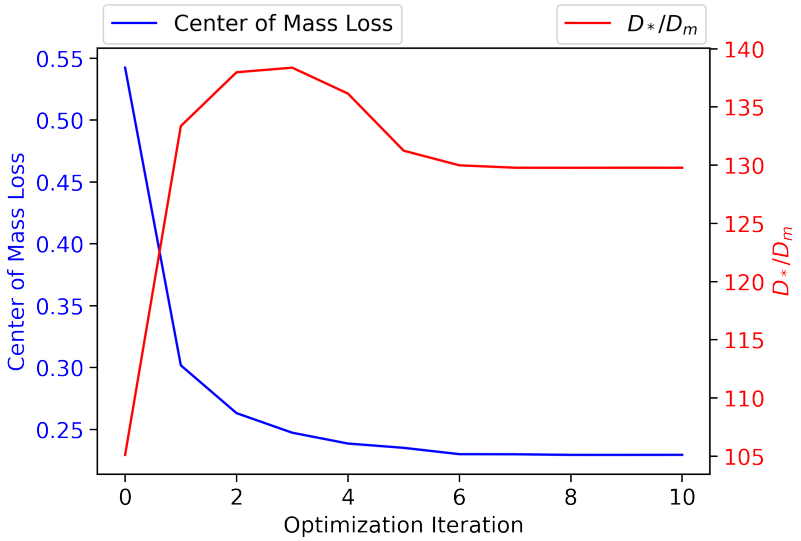

Figure 4.6: Optimization of the center of mass distance mixing indicator at a Péclet number of 100; the scaled dispersion coefficient, calculated via forward simulations but not optimized directly, is plotted over the average distance loss indicator

Although from the plot, we can see that the center of mass loss function is not an exact proxy for dispersion, minimizing this short-time mixing indicator does seem to increase the long-time scaled dispersion coefficient.

We perform the same optimization, now, using the average distance indicator (the average distance between each particle in one group from each particle in the other group, normalized by their initial average distance at $t = 0$). The results of this optimization are shown in Figure 4.7. We find that through the optimization iterations, the average distance indicator decreases by almost a factor of two. However, in this case, although minimizing the average distance indicator does increase the absolute dispersion coefficient, D_*, as calculated through forward simulations, it does not increase the scaled dispersion coefficient

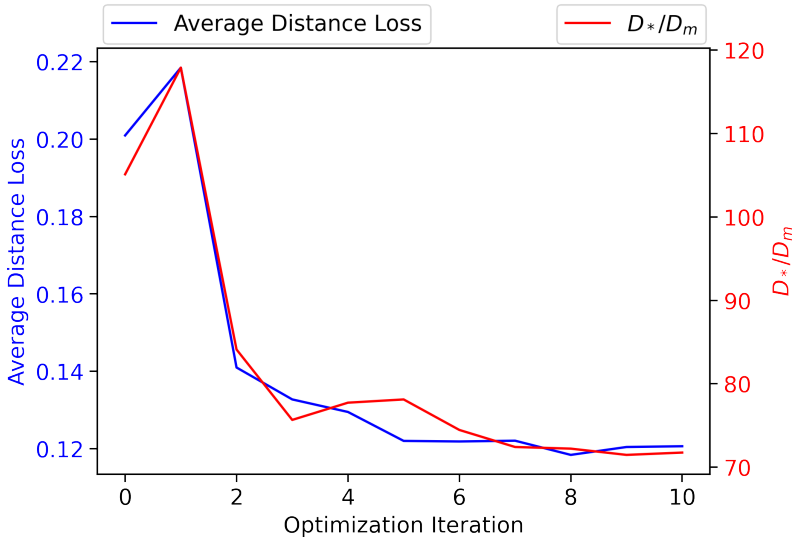

Figure 4.7: Optimization of the average distance mixing indicator at a Péclet number of 100; the scaled dispersion coefficient, calculated via forward simulations but not optimized directly, is plotted over the average distance loss indicator

D_*/D_m, which is plotted in Figure 4.7. Here, the optimizer decreases the average distance loss function by designing porous medium geometries that confine the two species of particles together. While introducing confinement does reduce the distance between particles of the two species types over time, it also has the effect of decreasing the scaled dispersion coefficient, which is what we see in the plot.

These two optimization examples demonstrate that our framework for designing porous media is robust. We experimented with a number of other short-time mixing indicators not reported here, and saw that we can basically define arbitrary Lagrangian loss functions, and when this loss function is not too noisy, gradient-based tools can be used to optimize it. The codebase developed here can thus be used to experiment with designing porous media for a wide variety of applications.

In the direct dispersion coefficient optimization, we were limited to working with low Péclet numbers. Here, we see that while short-time mixing indicators would be too noisy to optimize at low Péclet numbers, the short-time mixing indicator optimizations are well-behaved at high Péclet numbers. We find that while minimizing the average distance mixing indicator reduces the scaled dispersion coefficient, minimizing the center of mass distance mixing indicator seems to be a proxy for increasing the scaled dispersion coefficient. This provides us with two approaches for optimizing mixing in porous media across a range of Péclet numbers. In low Péclet number cases, we can optimize long-time dispersion directly, and in high Péclet number cases, we can optimize short-time proxies for dispersion. Since very little work has been done previously on directly optimizing mixing in fluids, little is known about the best mixing indicators for solving optimization problems or the mixing indicators that best serve as proxies for dispersion. In future work with this codebase, we aim to comprehensively study and compare different short-time mixing indicators.

5

Mixing in Journal Bearing

The previous chapter focused on optimizing mixing in steady, laminar flows through porous media. To demonstrate the versatility of the differentiable Lagrangian framework developed here, we apply it to the problem of mixing in journal bearing, which involves an unsteady velocity field. Optimizing over the characteristics and controlling time-dependent flows is an even more complex problem with little precedent in the literature. Journal bear-

ings consist of a freely rotating shaft, called a journal, inside of a sleeve, with fluid occupying the space between the journal and the sleeve. In fluid mechanics, the flow generated in journal bearing is known as eccentric Taylor-Couette flow. This flow is utilized across various industrial processes to facilitate mixing, such as in rotating annular chemical reactors and biorectors as well as in certain membrane separation techniques to enhance filtration [Vedantam et al., 2006; Nemri et al., 2016; Schrimpf et al., 2021].

5.1 Theory

Taylor-Couette flow describes the fluid flow between two coaxial cylinders rotating about each other, and eccentric Taylor-Couette flow describes the same system without the constraint that the cylinders have to be coaxial i.e. the centers of the inner and outer cylinder can be offset from each other. Eccentric Taylor-Couette flow nominally has five parameters — the radius of the interior cylinder R_1, the angular velocity of the interior cylinder $\dot{\theta}_1$, the offset between the center of the interior cylinder and the center of the exterior cylinder d, the radius of the exterior cylinder R_2 and the angular velocity of the external cylinder $\dot{\theta}_2$. A diagram of the Taylor-Couette flow setup is shown in Figure 5.1.

This system can take on more parameters, however, if the angular velocities $\dot{\theta}_1$ and $\dot{\theta}_2$ are not constant in time and instead have their own parametrizations.

As was discussed in the previous chapter on laminar static mixers, one of the major challenges in industrial mixing processes is to achieve efficient mixing with low Reynolds number flows. Previous studies of eccentric Taylor-Couette flow have demonstrated that if the

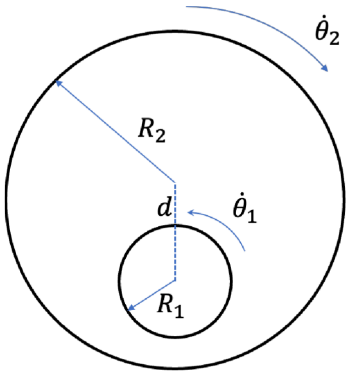

Figure 5.1: Flow parameters for the mixing in journal bearing optimization

generated flows are time-dependent, this set-up can be used to induce chaotic mixing at low Reynolds numbers [Swanson, 1991]. While there has been research on mixing protocols that generate chaotic flows [Aref, 1984; Aref et al., 2017], little work has been done on the inverse problem of designing journal bearing optimal configurations and rotation protocols to enhance mixing.

5.2 Simulations

To simulate particle trajectories moving through the unsteady velocity field in eccentric Taylor-Couette flow, a different approach is required from the one used in the previous chapter in steady flow through porous media. There, we are able to simulate particle trajectories based on a single fully-developed steady velocity field from the last step of the CFD simulation. Here, the velocity field that Lagrangian particles are advected according to must be updated at each time step of the simulation. Due to memory constraints involved in storing every time step of the CFD simulation and looping through the velocity field at each time step when running the MD simulation, we modify to run the CFD steps and

MD steps in tandem. The CFD simulation of eccentric Taylor-Couette flow used a differentiable implementation of the immersed boundary method developed by Dr. Mohammed Alhashim.

The goal of the simulations is to design the configuration and cylindrical rotation protocol to optimize mixing in the system. The configuration of the journal bearing set-up is defined using the following two parameters: the eccentricity, η, defined as

$$\eta = \frac{d}{R_2} \tag{5.1}$$

and the ratio between the cylindrical radii

$$\gamma = \frac{R_1}{R_2}. \tag{5.2}$$

We parametrize the time-dependent cylindrical rotation protocol using N Fourier series coefficients. Here, the angular rotation of the i_{th} cylinder is described by

$$\theta_i(t) = \theta_0 + \sum_{n=0}^{N} a_{in} \cos(2\pi f t + \varphi_i) + b_{in} \sin\left(2\pi f t + \varphi_i\right). \tag{5.3}$$

The oscillation of the inner and outer cylinder creates a time-dependent flow with an unsteady velocity field. For the simulations here, we include Fourier modes up to $N = 5$ for both cylinders. In total, the simulation has 25 design parameters that we optimize over.

The circular wall of the outer cylinder that confines the fluid is generated with static particles that interact repulsively with the tracer particles, following the method used for the wall in Taylor Dispersion in Chapter 3. The impermeability of the inner cylinder is rein-

forced in the MD simulation by modeling the cylinder using the repulsive side of the Morse potential (Equation 2.4) with a radius $r_0 = R_1$, following to the method used to model the impermeability of the cylindrical posts in the porous media simulations in Chapter 4. In the simulations here, we do not consider the effects of the molecular diffusion of the tracer particles. Particles are only advected according to the velocity field at each time step (Equation 2.2). Since mixing is not very hard to induce in this system through advection alone, molecular diffusion would just add additional noise to the mixing indicators.

The ability to optimize over Lagrangian objective functions of arbitrary complexity creates an enormous range of fluid behaviors that objective functions can be customized to target. Here, we consider a mixing indicator based on the average distance between two different groups of passive tracer particles, one group initialized in the upper half of the sleeve and the other group initialized in the lower half of the sleeve. This follows the approach used by Stone and Stone [2005] to quantify mixing in rotating droplets. Figure 5.2 shows an example simulation of Taylor-Couette flow with the differentiable code. The two groups of passive tracer particles are marked in orange and blue, and as the simulation evolves in time, we see the distributions of the orange and blue particles become more homogenized in the flow.

5.3 Optimizations

Since the simulation of Lagrangian particle dynamics in eccentric Taylor-Couette flow is fully differentiable, we are able to use automatic differentiation to efficiently compute the gradient of the mixing indicator, the average distance between the two groups of tracer par-

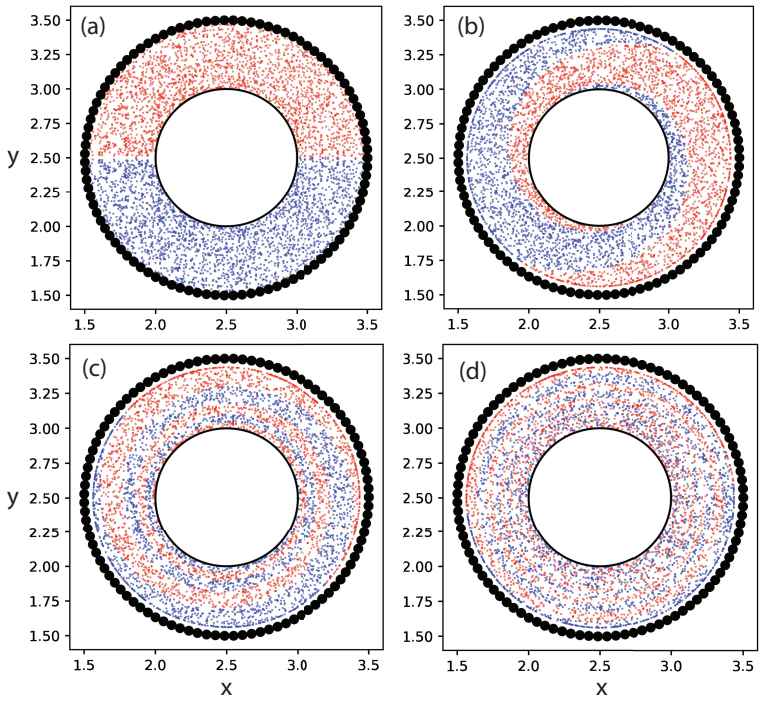

Figure 5.2: The evolution of Lagrangian tracer particles in time from (a) to (b) to (c) to (d) in a simulation of Taylor-Couette flow

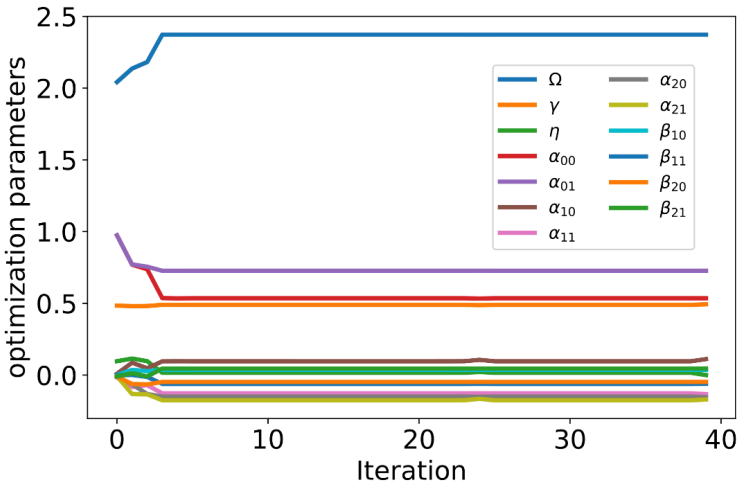

Figure 5.3: The trajectory of the design parameters during the mixing efficiency optimization in eccentric Taylor-Couette flow

ticles, with respect to all 25 design parameters. We then use the gradient-based optimization algorithm from IPOPT to optimize the design parameters. We measure the mixing indicator after two rotation periods, so in effect, we optimize for the design with the greatest mixing efficiency. Figure 5.3 shows the trajectory of a selection of the design parameters (only Fourier coefficients up to $N = 2$ are plotted) during the 40 optimization steps. From the plot, it is clear that an optimum is reached after only a few iterations.

In Figure 5.4, we normalize the mixing indicator by the initial average distance between the two groups of particles and plot how this normalized mixing indicator decreases throughout the simulation. This is plotted for the initial guess parameters and the optimized design parameters. The overlaid images show the mixing of tracer particles at different time steps in the simulations for the initial guess protocol and the optimized protocol. The degree of mixing can be assessed visually from the distribution of the two groups of particles in these

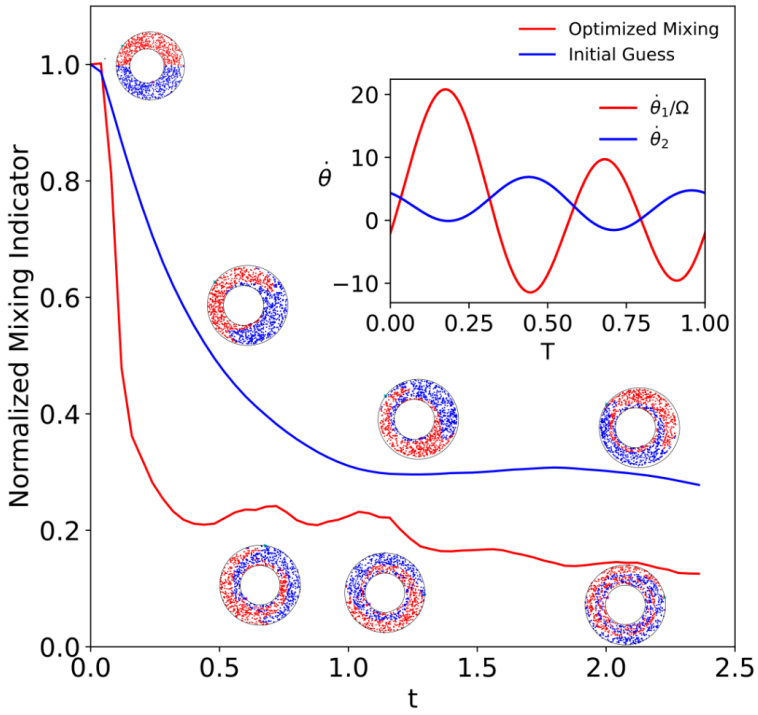

Figure 5.4: The normalized mixing indicator as a function of time for the initial guess parameters and the optimized parameters with images of the tracer particles shown at different points of the simulation; the inset plot shows the angular velocities corresponding to the optimal rotation protocols for the inner and outer cylinders

images. The inset figure shows the angular velocities corresponding to the optimized rotation protocols for the inner and outer cylinders with $T = \tau f$ where f is the frequency of the rotation.

First, the optimal rotation protocols found here suggest that the cylinders do not need to be counter-rotating in order to achieve optimal mixing and that a phase offset between the cylinders' rotation periods is enough to achieve optimal mixing. Although only one example of an optimization is shown in Figures 5.3 and 5.4, we tested the optimization with a

variety of different initial guesses. Remarkably, each optimization increased the mixing efficiency to a similar level, albeit with very different design parameters. This suggests that our technique for optimizing the configuration and kinematics of the journal bearing works well but that there is a great deal of parameter degeneracy in the problem. The methodology here can be used to design journal bearings that optimize other Lagrangian objective functions. An interesting example would be to optimize for chaoticity in the flow by designing an objective function based on the finite time Lyapunov exponent of the dynamical system defined by the particle trajectories.

6

Flow Sculpting

In previous chapters, we optimize full Lagrangian fluid particle simulations over up to 25 design parameters. To showcase the ability of this methodology to solve extremely high-dimensional flow optimization problems (with greater than 500 parameters) in order to finely control fluid properties, we consider a passive flow control problem known as flow sculpting. In flow sculpting, the position, size, and shape of obstacles are placed in a flow

to "sculpt" tracer particles flowing past them into highly specific configurations [Stoecklein et al., 2017, 2019]. This is also a flow control problem that to solve through the use of automatic differentiation, requires differentiable Lagrangian particle simulations, which is the methodology developed here.

The technique of flow sculpting has, in the literature, been tested and demonstrated by designing flows such that if dye is injected at specific locations, the dye will form into specific letters of the alphabet [Stoecklein et al., 2017, 2019]. However, the ability to finely control flows has important applications beyond forming letters with fluid dye. For example, the ability to passively direct the flow of particles and orient those particles simply by placing obstacles in a flow can be leveraged to design more effective biosensors and on-chip flow cytometry diagnostic tools or to develop methods to contactlessly manipulate and assemble sensitive materials. General methods for solving inverse problems of this complexity and scale were, until lately, completely unimaginable. In 2008, in a paper about the engineering heuristics that can be used to design effective biosensors, Squires et al. [2008] motivate their work by writing, "Computers and algorithms may be getting faster, cheaper and easier, but they simply cannot handle a blind search through the eight-or-more-dimensional parameter space." In our work, we hope to demonstrate that we can handle a search over even more parameters than that in a fully physics-based optimization by leveraging the fact that with automatic differentiation, the computational cost of computing gradients of objective functions is independent of the number of design parameters.

Beyond the applications of flow sculpting, this problem is interesting because it is an inverse problem that, to our knowledge, has never been solved in a fully physics-based way and therefore provides a point of comparison between different methods. In two studies

Stoecklein et al. [2017] and Stoecklein et al. [2019] compute a set of fluid deformations around pillars of various sizes at various locations in the flow. They then use deep learning in Stoecklein et al. [2017] and genetic algorithms in Stoecklein et al. [2019] to piece together a sequence of pillars to induce a net deformation in the flow field that causes passive tracer particles to form into a specific configuration. Note that in order to use these computational approaches, they require pillars to be placed very far apart from each other so they can model flows around pillars independently with the assumption that there is no cross-talk between them. In our approach, we aim to be more general by (1) putting no constraints on the distance between obstacles, (2) considering obstacles of different shapes, and (3) being more precise by solving the inverse problem through full fluid simulations that include all the obstacles and can model the interactions between them.

6.1 Optimizations

In flow sculpting the objective is to inversely design the initial distribution of tracer particles as well as the positions and sizes of obstacles such that passive tracer particles traveling through the flow will configure into specific target shapes. Figure 6.1 shows schematically the set-up and desired output of flow sculpting problems.

In our framework, which enables us to optimize through arbitrary Lagrangian loss functions, we approach the problem by first choosing a target configuration, which can be any binary image file. Then, we convert the file into an array of particles at the positions of the 1s in the binary file. The density and the number of particles can be set to a specific value.

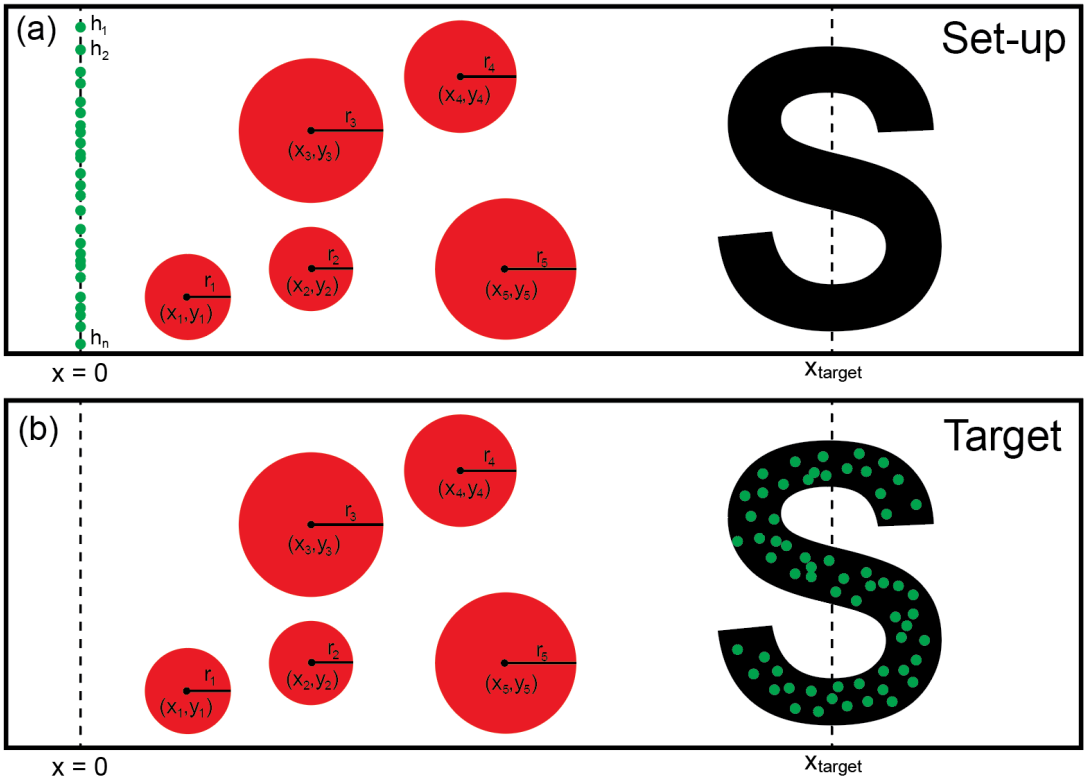

Figure 6.1: (a) Diagram of the set-up of flow sculpting problems with parameters including the initial distribution of the tracer particles, the sizes and positions of the obstacles, and the target configuration, (b) Diagram of the desired output of flow sculpting problems, which is an arrangement of obstacles such that the passive tracer particles form into the target configuration downstream from the tracer particles

With this array of particles that exist in the target configuration, we define our objective function to be the mean squared distance between the passive tracer particles in the flow at the end of each simulation and the array of target particle locations.

Following Stoecklein et al. [2017, 2019], to induce complex deformation patterns into the flow, we make the pressure drop across the channel a design parameter in the simulation and consider higher Reynolds number flows, on the order of $\mathrm{Re} = 10$. The other parameters are the initial y positions of the ~ 500 tracer particles in the simulation as well as the obstacle positions and sizes. Note that in contrast to the original three-dimensional flow sculpting work in Stoecklein et al. [2017] and Stoecklein et al. [2019], here we consider only two-dimensional flows, which significantly limits the expressivity of the flow for creating arbitrary shapes and designs. This will be seen in the optimization results.

6.2 Preliminary Results

In the first flow sculpting optimization, we attempt to sculpt out the letter "S" in the flow. Figure 6.2 shows the optimized flow and position of the tracer particles after 15 optimization iterations. The optimization started from a random configuration of obstacles and with the passive tracer particles distributed uniformly in a line from $y = 0$ to $y = 2$ at $x = 0$. Although after optimization, the passive tracer particles do not sculpt out the full "S", we see that it has sculpted the flow such that a portion of the letter "S" is made by the passive particles. Through these iterations, the loss function we design to calculate the distance between the tracer particles and the target shape decreases by two orders of magnitude. This suggests that the loss function and optimization are working well but that there are not enough degrees of freedom in the system to sculpt out arbitrarily complex curves

that invert the parabolic concentration profile of channel flows. The original flow sculpting works by Stoecklein et al. [2017] and Stoecklein et al. [2019] used three-dimensional flows with the image plane being orthogonal to the direction of the flow. Here, our image plane is in the direction of the flow, which makes this a more complex problem to solve.

Nevertheless, even with this very simple two-dimensional flow, we show that certain optimization problems can be solved well. For example, in Figure 6.3, we show the results of optimizing the obstacles in a flow to funnel tracer particles into a "tube," represented by the two blue barriers, at a specific location in the flow. In this case, there is much less complexity in the shape of the target distribution but still a high degree of specificity in where the tracer particles need to be directed in the flow (based on the position of the tube and the angle of the tube). In this case, our Lagrangian optimization framework works well. Three obstacles are placed strategically to funnel most of the tracer particles to a final location within the tube boundaries.

These two examples demonstrate some of the preliminary work we have done on flow sculpting using automatic differentiation. It is worth reiterating that these are ~ 500 parameter optimization problems solved through fully physics-based simulations that involve solving the Navier-Stokes equations with the Penalty Method and simulating tracer particles and the interactions between those tracer particles and the obstacles. The initial results from this work on flow sculpting are promising, as they suggest that if we introduce greater complexity into the flows we are optimizing over, we can sculpt flows based on pure

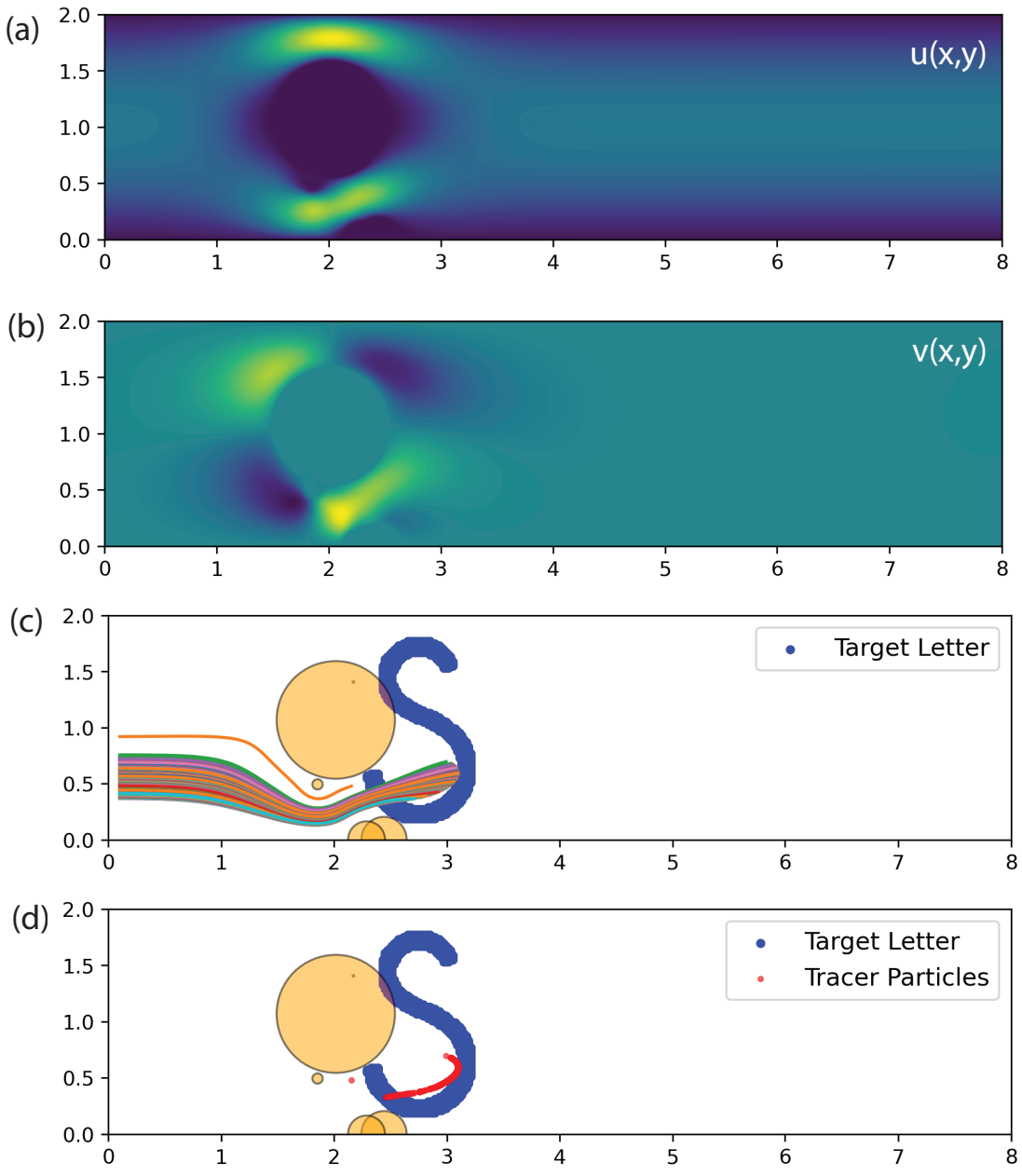

Figure 6.2: Optimizing letter design: (a) Optimized velocity field $u(x, y)$, (b) Optimized velocity field $v(x, y)$, (c) Trajectory of tracer particles moving through flow, (d) Final position of tracer particles in the flow

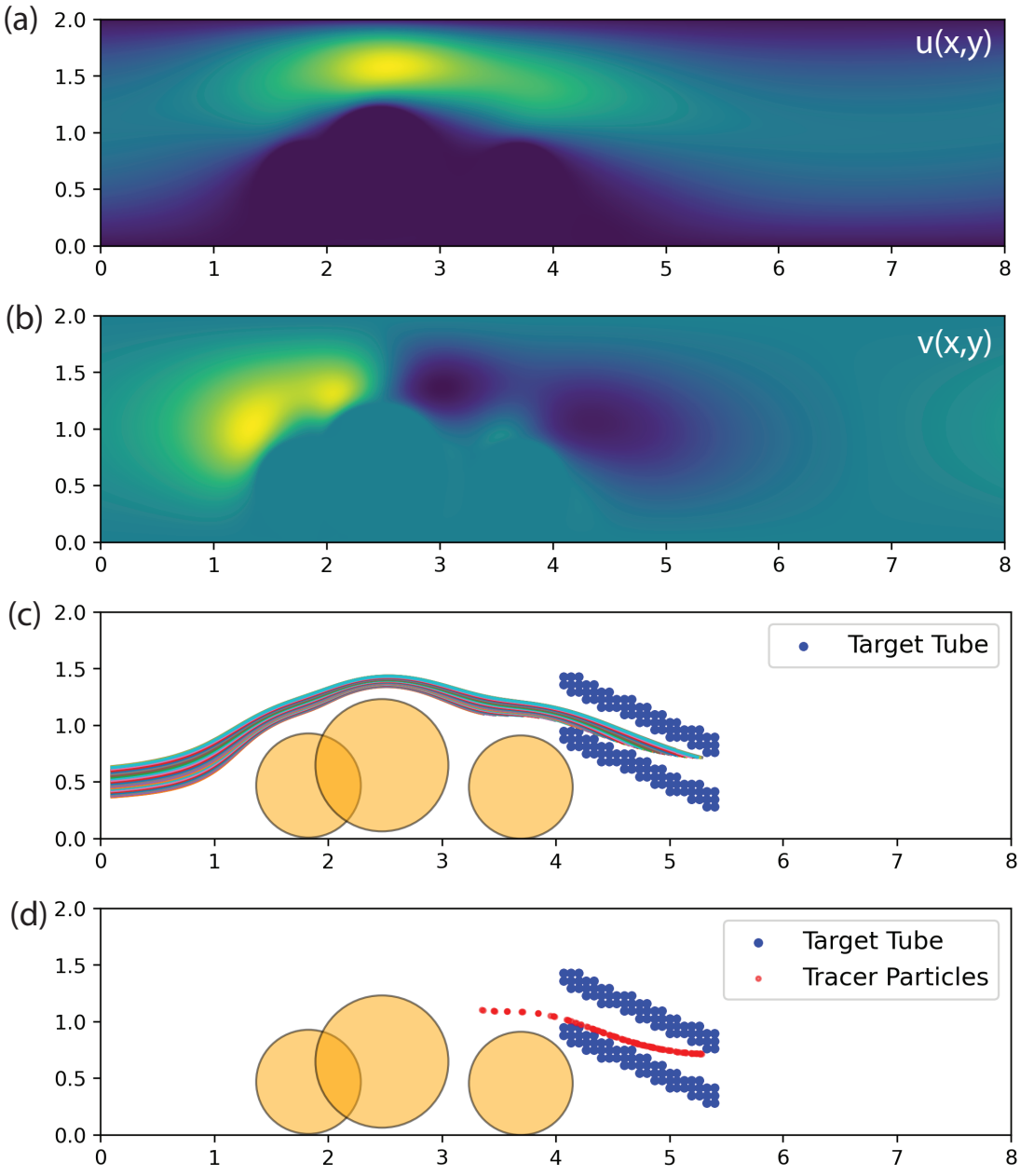

Figure 6.3: Optimizing particle positions: (a) Optimized velocity field $u(x, y)$, (b) Optimized velocity field $v(x, y)$, (c) Trajectory of tracer particles moving through flow, (d) Final position of tracer particles in the flow

physics, without the use of machine learning or the need to put constraints on the distance between obstacles. Future work on this problem will increase the complexity of the flow in two dimensions to enable the sculpting of more complex designs or will extend our simulations to three dimensions. Here, we have a constant pressure gradient across the entire unit cell. Introducing a more complex or time-dependent pressure gradient may give the flow more expressivity for sculpting out letters even in two dimensions. We can also add complexity by enabling tracer particles to be injected at different times in the simulation, and then we can optimize those injection times.

7

Catheter Design

Another example of a passive flow control problem with important biomedical applications is designing catheters geometries that reduce upstream bacterial swimming, which is one of the leading causes of health care-associated infections in patients [Shuman and Chenoweth, 2018]. One approach to mitigating upstream bacterial swimming that does not involve the use of antibiotics, which should be used sparingly because they can select for antibiotic re-

sistant strains, is to introduce obstacles in a flow that confine bacteria and affect their overall trajectories [Di Giacomo et al., 2017]. Designing the geometry of catheters is a complex problem that involves both modeling fluid flow in a catheter and modeling the transport of bacteria in that catheter.

In a recent paper, Zhou et al. [2024] approached this inverse problem using machine learning, specifically Fourier Neural Operators. In Zhou et al. [2024], they report that "[i]t took 30 min each to generate a training instance for a total of 1000 instances in parallel (on 50 GPUs for 10 hours), 20 min on 1 GPU to train the model, and 15s on 1 GPU for [the] trained AI model to generate the optimal design." Our objective is to demonstrate that we can solve the same inverse design problem with (1) fewer forward simulations than they require just to train the model, (2) using only 1 GPU, (3) with a methodology that is purely physics-based, using no machine learning, and (4) with more flexibility over the shapes and designs of obstacles.

Following Peng and Brady [2020], we model the dynamics of the bacteria in the catheter using

$$dq/dt = \left[1/2\omega + \sqrt{2/\tau_R}\eta(t)\right] \times q \tag{7.1}$$

where q represents the orientation of the bacteria and ω represents the local vorticity. To implement this, we output the fluid vorticity in addition to the velocity fields from the differentiable CFD simulation. Then, it is simple to modify the differentiable implementation of Langevin dynamics, used in the previous optimization problems, to instead simulate the dynamics of bacteria in the catheter. Figure 7.1 shows an example of an upstream bacterial swimming trajectory in a channel with no rotational or translational Brownian motion and no obstacles in the flow.

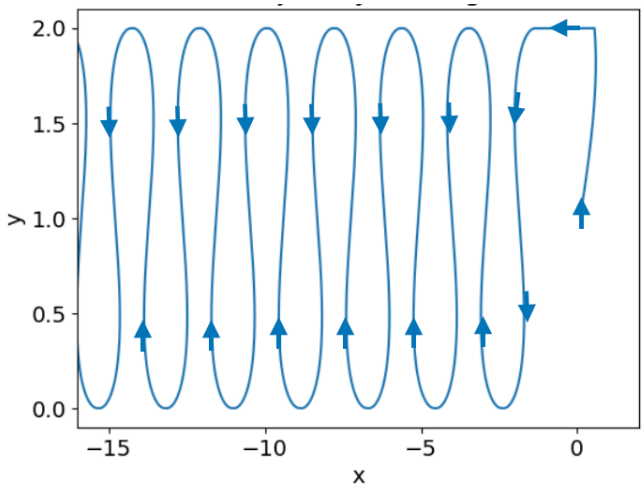

Figure 7.1: Trajectory of an upstream swimming bacteria in a channel in the absence of rotational and translational Brownian motion

With the bacterial dynamics implemented, the rest of the workflow of this problem closely follows the flow sculpting problem but with much fewer parameters. Here, the objective function we aim to maximize is the average x-position of the swimming bacteria. In our simulations, the flow is in the positive x-direction, so maximizing the average x-position of the swimming bacteria corresponds to minimizing upstream bacterial swimming. In order to replicate the results of Zhou et al. [2024], we use the Penalty Method to simulate flow through a channel with elliptical and triangular obstacles embedded. In Zhou et al. [2024], they use only triangular obstacles. We experiment with ellipses as a comparison to triangles to test the effect of the sharp triangular edge on the gradient behavior when we differentiate through the full catheter simulation. The flow fields around such obstacles, solved with the Penalty Method using JAX-CFD, are shown in Figure 7.2.

Enforcing the impermeability conditions at the obstacles in the MD simulation is more

complicated with shapes that are not circular. In Chapter 4, Chapter 5, and Chapter 6, the impermeability of the cylindrical posts is enforced by placing an immobile particle at the center of the cylindrical post and modeling that particle as interacting with the Lagrangian tracer particles with a repulsive potential whose effective radius corresponds to the radius of the cylindrical post. If we were to use a potential here to model the impermeability of the ellipses and triangles, the potential would not be radially symmetric. In our preliminary work, we model the impermeability of the ellipses and triangles by placing small repulsive particles along the boundary of the ellipse and triangle edges, following the approach used to model the wall in the Taylor Dispersion simulations in Chapter 3 and in the mixing in journal bearing problem in Chapter 5.

Thus far, we have used our differentiable code base to test forward simulations of the catheter design problem and to test the stability of the gradient computation. Figure 7.3 shows the trajectory of a single bacterium traveling in a catheter with two elliptical obstacles. We can see in this simulation how the presence of the obstacles already has the effect of mitigating upstream swimming. We see in this trajectory that the bacteria repeatedly gets trapped by the obstacles. This is the behavior we aim to optimize for with thousands of bacteria in the simulation.

Since this simulation is fully differentiable, in the next steps on this project, we will use gradient-based optimization to directly design catheters that are most effective at preventing the upstream swimming of bacteria. We aim to scale this work to the same level of

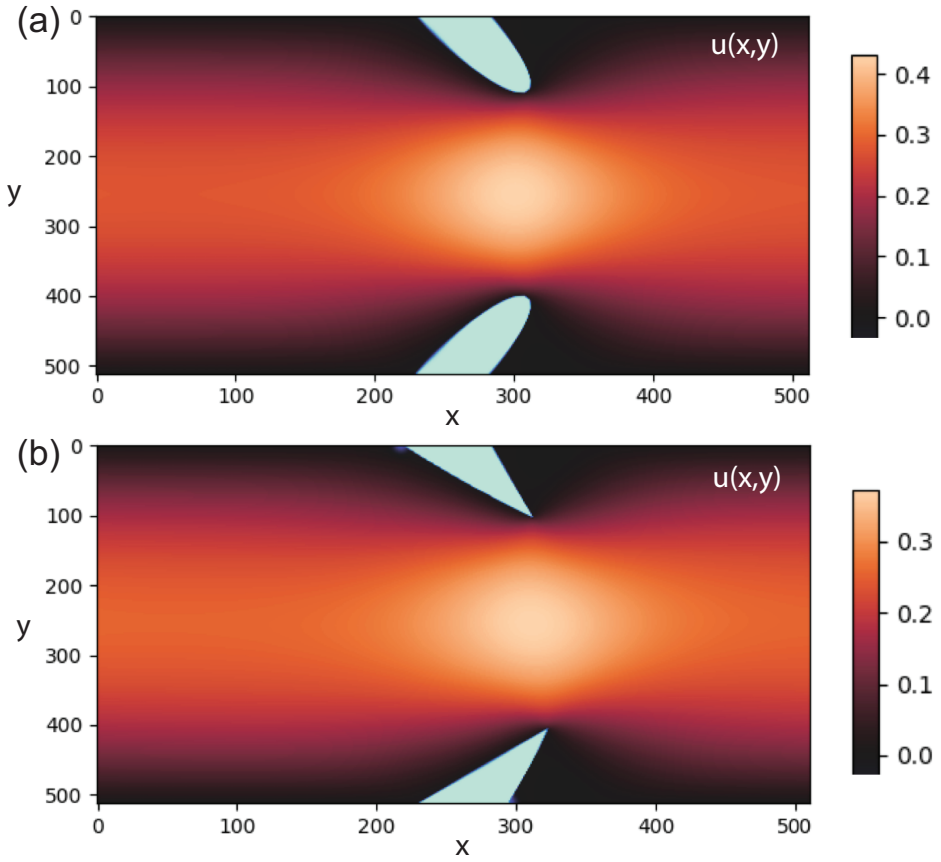

Figure 7.2: (a) Velocity field $u(x, y)$ for a channel flow with elliptical obstacles embedded in the flow (solved for with JAX-CFD), (b) Velocity field $u(x, y)$ for a channel flow with triangular obstacles embedded in the flow (solved for with JAX-CFD)

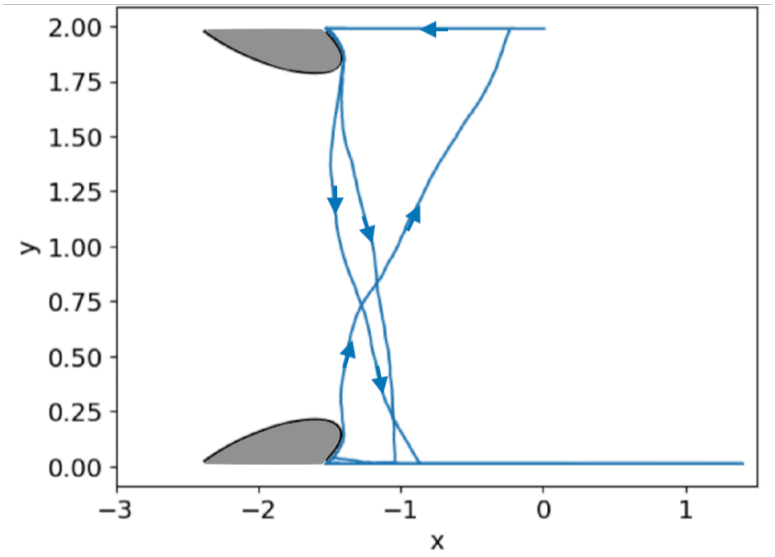

Figure 7.3: Trajectory of a swimming bacterium in a catheter with elliptical obstacles that mitigate upstream travel

complexity as Zhou et al. [2024] and replicate their results. This project also opens up interesting avenues for exploring shape optimization, an important problem in many flow control contexts. In future work, we may experiment with the use of neural networks to parametrize the potential around obstacles that are not circular. Note that this type of usage of machine learning is extremely distinct from the type of usage in Zhou et al. [2024]. Here, neural networks would be used as differentiable functions that take in the parameters defining the shapes and locations of obstacles and calculate the potential due to these obstacles at arbitrary locations in the flow. This use of neural networks to accelerate small steps in physics-based simulations is very different than using neural networks to emulate entire physics simulations. In light of the work in this thesis, which shows that we do not need to replace physics with neural networks to solve complex flow control inverse problems, we believe the role of machine learning in the field should be of this nature — using it to accelerate complicated steps like parameterizing shapes, not replacing fundamental physics.

8

Conclusion

In this work, we describe the development of a fully-differentiable framework for Lagrangian fluid simulations that has been integrated with a differentiable molecular dynamics engine. We start from a preexisting implementation of a differentiable Navier-Stokes solver, which enables us only to differentiate through Eulerian objective functions [Kochkov et al., 2021]. We expand the toolkit available for differentiable fluid mechanics by build-

ing a framework for fully-differentiable simulations of Lagrangian tracer particles that include the effects of molecular diffusion, intermolecular interactions, and active Brownian motion. The differentiable simulation tools developed in this thesis have greatly expanded the complexity of the flow control problems we can solve with automatic differentiation. Without it, we would not have been able to make progress on the inverse design problems described in this thesis (designing porous media to optimize mixing, optimizing mixing in journal bearing, sculpting flows, and designing catheters).

In Chapter 3, we validate this differentiable Lagrangian toolkit by showing that it can replicate the results of Taylor Dispersion Theory. While we did not solve any optimization problems involving flow through a channel in this chapter, the code developed to replicate Taylor Dispersion Theory is fully-differentiable and could easily be used to solve optimization problems, perhaps involving the shape of the channel or designing time-dependent flows that enhance dispersion.

In Chapter 4, we simulate and optimize over the dispersion coefficients of porous media. This is an extremely computationally intensive problem that we are able to solve without simplifying any of the physics. Specifically, we are able to compute meaningful gradients with respect to the design parameters through end-to-end simulations that involve solving the Navier-Stokes equations with the Penalty Method, simulating the motions of diffusing tracer particles in this flow for over 1 million time steps, and using an FFT-based algorithm to compute dispersion coefficients. In this work, we solved the optimization problem of designing porous media arrangements to maximize axial dispersion. This code could easily be modified to optimize for specific dispersion tensors. In this chapter, we also demonstrated our ability to optimize through short-time mixing indicators. Although mixing has been

studied extensively in the literature, since very little work has been done on mixing optimization in fluid mechanics, future work using our code could study and identify mixing indicators that are the most effective for solving optimization problems. Altogether, this work demonstrated that the framework developed here enables us to run very complex optimizations over stochastic particle trajectories.

After optimizing dispersion in porous media, we greatly scaled up the complexity of our optimization problems both in the number of design parameters and in the dynamics involved. In Chapter 5, we used gradient-based optimization to design a protocol to induce optimal mixing in journal bearing. This is an inverse design problem involving an unsteady flow field. In this work, we only considered one mixing indicator, and our optimizer was able to find many mixing protocols to maximize this mixing indicator. Future work could use the same code to optimize for different, more complicated mixing targets. For example, optimizing a mixing indicator based on the finite time Lyapunov exponent of the dynamical system defined by the particle trajectories would be a very interesting direction to take this project, as it would allow us to optimize directly for chaotic mixing.

In Chapter 6, we describe ongoing work in using the differentiable Lagrangian fluid simulation framework for flow sculpting — configuring passive tracer particles into arbitrary shapes by strategically placing obstacles in a flow. Our initial optimizations suggest that we need to introduce greater complexity into the flow to be able to sculpt out arbitrary curves but that the objective function and optimization procedure both work well. Future work on flow sculpting using automatic differentiation will focus on adding greater degrees of freedom to the flow, perhaps by making it time-dependent.

Finally, in Chapter 7 we describe preliminary work we have done to use the differentiable

Lagrangian codebase to design catheters that mitigate the upstream flow of bacteria that cause infections in patients. Thus far, our work has consisted of building differentiable forward simulations of bacteria swimming in a flow and simulating more complex obstacle shapes with the Penalty Method. In future work, we aim to optimize over the catheter design and compare our results and computation requirements to the work from Zhou et al. [2024], where this problem was approached using machine learning.

Ultimately, the framework developed here has greatly increased the complexity of the optimization problems we can solve in fluid mechanics. We aim to continue adapting our code to solve new optimization problems that have important engineering applications. Machine learning approaches to solving flow control problems require a large amount of data to be generated through physics-based flow simulations. Then, neural networks need to be trained on this data and tested before any optimization problems can be solved. With the code developed here, solving complex optimization problems is now as simple as designing a differentiable forward simulation. Here, no training procedure is needed — we are able to solve problems just based on the pure fluid simulations. With the flow sculpting and catheter design problems, we aim to show that our end-to-end optimization procedure requires significantly less computation than the researchers who worked on the ML approaches to these problems used just to develop their training datasets (before they solved any optimization problems).

Our ability to optimize for complex fluid behaviors without replacing any of the physics with ML models suggests a new path forward for solving optimization problems in fluid mechanics. In the past few years, there has been over-excitement about the use of ML in fluid mechanics. Our work raises questions about why we should be using ML models,

which are not interpretable and are susceptible to generalization errors, to solve inverse design problems in fluid mechanics at all. Instead, we argue that neural networks should be used as arbitrary differentiable functions to accelerate small tasks within physics-based simulations, but not to emulate the physics itself. Our codebase will continue to be made publicly available. It is designed to streamline the process of solving flow optimization problems by abstracting away much of the work that goes into making the code fully differentiable. We hope that the fluid mechanics community will find that by merely writing forward CFD simulations with our codebase, they can now easily compute gradients through full simulations and solve novel optimization problems. In our new and ongoing work, we are continuing to expand the limits of the differentiable simulation capabilities we have and the optimization problems we can solve with this methodology.

References

M. G. Alhashim and M. P. Brenner. Engineering of polydisperse porous media for enhanced fluid flows through systematic topology tuning via differentiable direct numerical simulation. *Physical Review Fluids*. doi: 10.1103/PhysRevFluids.00.004100.

K. Allen, T. Lopez-Guevara, K. L. Stachenfeld, A. Sanchez Gonzalez, P. Battaglia, J. B. Hamrick, and T. Pfaff. Inverse design for fluid-structure interactions using graph network simulators. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 13759–13774. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/59593615e358d52295578e0d8e94ec4a-Paper-Conference.pdf.

H. P. Amaral Souto and C. Moyne. Dispersion in two-dimensional periodic porous media. Part II. Dispersion tensor. *Physics of Fluids*, 9(8):2253–2263, 08 1997. ISSN 1070-6631. doi: 10.1063/1.869347. URL https://doi.org/10.1063/1.869347.

H. Aref. Stirring by chaotic advection. *Journal of Fluid Mechanics*, 143:1–21, Jan. 1984. doi: 10.1017/S0022112084001233.

H. Aref, J. R. Blake, M. Budišić, S. S. S. Cardoso, J. H. E. Cartwright, H. J. H. Clercx, K. El Omari, U. Feudel, R. Golestanian, E. Gouillart, G. F. van Heijst, T. S. Krasnopolskaya, Y. Le Guer, R. S. MacKay, V. V. Meleshko, G. Metcalfe, I. Mezić, A. P. S. de Moura, O. Piro, M. F. M. Speetjens, R. Sturman, J.-L. Thiffeault, and I. Tuval. Frontiers of chaotic advection. *Rev. Mod. Phys.*, 89:025007, Jun 2017. doi: 10.1103/RevModPhys.89.025007. URL https://link.aps.org/doi/10.1103/RevModPhys.89.025007.

J. Baléo and P. L. Cloirec. Numerical Simulation of the Spatial Distribution of Mean Residence Time in Complex Flows through Porous Media. *Progress of Theoretical Physics Supplement*, 138:690–695, Jan. 2000. doi: 10.1143/PTPS.138.690.

L. Balkenhol, C. Trendafilova, K. Benabed, and S. Galli. candl: Cosmic microwave background analysis with a differentiable likelihood, 2024.

D. A. Bezgin, A. B. Buhendwa, and N. A. Adams. Jax-fluids: A fully-differentiable high-order computational fluid dynamics solver for compressible two-phase flows. *Computer Physics Communications*, 282:108527, 2023. ISSN 0010-4655. doi: https://doi.org/10.1016/j.cpc.2022.108527. URL https://www.sciencedirect.com/science/article/pii/S0010465522002466.

D. A. Bezgin, A. B. Buhendwa, and N. A. Adams. JAX-Fluids 2.0: Towards HPC for Differentiable CFD of Compressible Two-phase Flows. *arXiv e-prints*, art. arXiv:2402.05193, Feb. 2024. doi: 10.48550/arXiv.2402.05193.

J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

M. P. Brenner and P. Koumoutsakos. Editorial: Machine learning and physical review fluids: An editorial perspective. *Phys. Rev. Fluids*, 6:070001, Jul 2021. doi: 10.1103/PhysRevFluids.6.070001. URL https://link.aps.org/doi/10.1103/PhysRevFluids.6.070001.

M. P. Brenner, J. D. Eldredge, and J. B. Freund. Perspective on machine learning for advancing fluid mechanics. *Phys. Rev. Fluids*, 4:100501, Oct 2019. doi: 10.1103/PhysRevFluids.4.100501. URL https://link.aps.org/doi/10.1103/PhysRevFluids.4.100501.

H. C. Brinkman. On the permeability of media consisting of closely packed porous particles. *Flow, Turbulence and Combustion*, 1(1):81–86, 1949a. doi: 10.1007/BF02120318. URL https://doi.org/10.1007/BF02120318.

H. C. Brinkman. A calculation of the viscous force exerted by a flowing fluid on a dense swarm of particles. *Flow, Turbulence and Combustion*, 1(1):27–34, 1949b. doi: 10.1007/BF02120313. URL https://doi.org/10.1007/BF02120313.

S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214. URL https://doi.org/10.1146/annurev-fluid-010719-060214.

V. Calandrini, E. Pellegrini, P. Calligari, K. Hinsen, and G. R. Kneller. nMoldyn - Interfacing spectroscopic experiments, molecular dynamics simulations and models for time correlation functions. *Collection SFN*, 12:201–232, 2011. doi: 10.1051/sfn/201112010. URL https://hal.science/hal-00720549.

J.-E. Campagne, F. Lanusse, J. Zuntz, A. Boucaud, S. Casas, M. Karamanis, D. Kirkby, D. Lanzieri, A. Peel, and Y. Li. JAX-COSMO: An End-to-End Differentiable and GPU Accelerated Cosmology Library. *The Open Journal of Astrophysics*, 6:15, Apr. 2023. doi: 10.21105/astro.2302.05163.

J. Carlson, A. Jaffe, A. Wiles, C. M. Institute, and A. M. Society. *The Millennium Prize Problems*. American Mathematical Society, 2006. ISBN 9780821836798. URL https://books.google.com/books?id=7wJIPJ80RdUC.

L. Cattaneo, A. Comunian, G. De Filippis, M. Giudici, and C. Vassena. Modeling groundwater flow in heterogeneous porous media with yagmod. *Computation*, 4 (1), 2016. ISSN 2079-3197. doi: 10.3390/computation4010002. URL https://www.mdpi.com/2079-3197/4/1/2.

A. J. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22(104):745–762, 1968. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2004575.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, Dec. 1989. doi: 10.1007/BF02551274. URL https://hal.science/hal-03753170.

M. Dentz, J. J. Hidalgo, and D. Lester. Mixing in porous media: Concepts and approaches across scales. *Transport in porous media*, 146(1-2):5–53, 2023. ISSN 0169-3913.

R. Di Giacomo, S. Kroedel, B. Maresca, P. Benzoni, R. Rusconi, R. Stocker, and C. Daraio. Deployable micro-traps to sequester motile bacteria. *Scientific reports*, 7(1): 45897–45897, 2017. ISSN 2045-2322.

M. C. Engel, J. A. Smith, and M. P. Brenner. Optimal control of nonequilibrium systems through automatic differentiation. *Phys. Rev. X*, 13:041032, Nov 2023. doi: 10.1103/PhysRevX.13.041032. URL https://link.aps.org/doi/10.1103/PhysRevX.13.041032.

A. H. Farmahini, S. Krishnamurthy, D. Friedrich, S. Brandani, and L. Sarkisov. Performance-based screening of porous materials for carbon capture. *arXiv e-prints*, art. arXiv:2009.12289, Sept. 2020. doi: 10.48550/arXiv.2009.12289.

C. L. German and S. V. Madihally. Applications of computational modelling and simulation of porous medium in tissue engineering. *Computation*, 4(1), 2016. ISSN 2079-3197. doi: 10.3390/computation4010007. URL https://www.mdpi.com/2079-3197/4/1/7.

C. P. Goodrich, E. M. King, S. S. Schoenholz, E. D. Cubuk, and M. P. Brenner. Designing self-assembling kinetics with differentiable statistical physics models. *Proceedings of the National Academy of Science*, 118(10):e2024083118, Mar. 2021. doi: 10.1073/pnas.2024083118.

A. Gunes Baydin, B. A. Pearlmutter, A. Andreyevich Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *arXiv e-prints*, art. arXiv:1502.05767, Feb. 2015. doi: 10.48550/arXiv.1502.05767.

Z. Hao, S. Liu, Y. Zhang, C. Ying, Y. Feng, H. Su, and J. Zhu. Physics-informed machine learning: A survey on problems, methods and applications, 2023.

Hecht-Nielsen. Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks*, pages 593–605 vol.1, 1989. doi: 10.1109/IJCNN.1989.118638.

J. Heyman, D. R. Lester, and T. Le Borgne. Scalar signatures of chaotic mixing in porous media. *Phys. Rev. Lett.*, 126:034505, Jan 2021. doi: 10.1103/PhysRevLett.126.034505. URL https://link.aps.org/doi/10.1103/PhysRevLett.126.034505.

P. Karnakov, S. Litvinov, and P. Koumoutsakos. Optimizing a DIscrete Loss (ODIL) to solve forward and inverse problems for partial differential equations using machine learning tools. *arXiv e-prints*, art. arXiv:2205.04611, May 2022. doi: 10.48550/arXiv.2205.04611.

P. Karnakov, S. Litvinov, and P. Koumoutsakos. Flow reconstruction by multiresolution optimization of a discrete loss with automatic differentiation. *The European physical journal. E, Soft matter and biological physics*, 46(7):59–59, 2023. ISSN 1292-8941.

K. Khadra, P. Angot, S. Parneix, and J.-P. Caltagirone. Fictitious domain approach for numerical modelling of Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 34(8):651–684, Dec. 2000. doi: 10.1002/1097-0363(20001230)34:8<651::AID-FLD61>3.3.CO;2-4.

A.-R. Khaled and K. Vafai. The role of porous media in modeling flow and heat transfer in biological tissues. *International Journal of Heat and Mass Transfer*, 46(26):4989–5003, 2003. ISSN 0017-9310. doi: https://doi.org/10.1016/S0017-9310(03)00301-6. URL https://www.sciencedirect.com/science/article/pii/S0017931003003016.

K. Khanafer and K. Vafai. The role of porous media in biomedical engineering as related to magnetic resonance imaging and drug delivery. *Heat and Mass Transfer*, 42(10):939–953, Aug. 2006. doi: 10.1007/s00231-006-0142-6.

E. M. King, C. X. Du, Q.-Z. Zhu, S. S. Schoenholz, and M. P. Brenner. Programmable patchy particles for materials design, 2023.

D. L. Koch, R. G. Cox, H. Brenner, and J. F. Brady. The effect of order on dispersion in porous media. *Journal of Fluid Mechanics*, 200:173–188, 1989. doi: 10.1017/S0022112089000613.

D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021. ISSN 0027-8424. doi: 10.1073/pnas.2101784118. URL https://www.pnas.org/content/118/21/e2101784118.

R. Krueger, E. King, and M. Brenner. Tuning colloidal reactions, 2023.

T. Le Borgne, M. Dentz, and E. Villermaux. The lamellar description of mixing in porous media. *Journal of fluid mechanics*, 770:458–498, 2015. ISSN 0022-1120.

D. Maclaurin, D. Duvenaud, and R. P. Adams. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, 2015. URL /bib/maclaurin/maclaurinautograd/automl-short.pdf,https://indico.lal.in2p3.fr/event/2914/session/1/contribution/6/3/material/paper/0.pdf,https://github.com/HIPS/autograd.

K. L. McIlhany and S. Wiggins. Eulerian indicators under continuously varying conditions. *Physics of Fluids*, 24(7):073601, 07 2012. ISSN 1070-6631. doi: 10.1063/1.4732152. URL https://doi.org/10.1063/1.4732152.

G. Metcalfe, D. Lester, and M. Trefry. A Primer on the Dynamical Systems Approach to Transport in Porous Media. *arXiv e-prints*, art. arXiv:2202.13053, Feb. 2022. doi: 10.48550/arXiv.2202.13053.

L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman. Gradients are not all you need, 2022.

A. F. Miguel. *Lungs as a Natural Porous Media: Architecture, Airflow Characteristics and Transport of Suspended Particles*, pages 115–137. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-21966-5. doi: 10.1007/978-3-642-21966-5_5. URL https://doi.org/10.1007/978-3-642-21966-5_5.

M. Minkov, I. A. D. Williamson, L. C. Andreani, D. Gerace, B. Lou, A. Y. Song, T. W. Hughes, and S. Fan. Inverse design of photonic crystals through automatic differentiation. *ACS Photonics*, 7(7):1729–1741, June 2020. ISSN 2330-4022. doi: 10.1021/acsphotonics.0c00327. URL http://dx.doi.org/10.1021/acsphotonics.0c00327.

T. Moghaddam and N. Banazadeh Neishabouri. On the Active and Passive Flow Separation Control Techniques over Airfoils. In *Materials Science and Engineering Conference Series*, volume 248 of *Materials Science and Engineering Conference Series*, page 012009, Oct. 2017. doi: 10.1088/1757-899X/248/1/012009.

A. Nair, A. Kazemi, O. Curet, and S. Verma. Porous cylinder arrays for optimal wake and drag characteristics. *Journal of Fluid Mechanics*, 961:A18, Apr. 2023. doi: 10.1017/jfm.2023.255.

M. Nemri, S. Charton, and E. Climent. Mixing and axial dispersion in taylor–couette flows: The effect of the flow regime. *Chemical engineering science*, 139:109–124, 2016. ISSN 0009-2509.

Y. Ogawa and M. Kawahara. Shape Optimization of Body Located in Incompressible Viscous Flow Based on Optimal Control Theory. *International Journal of Computational Fluid Dynamics*, 17(4):243–251, July 2003. doi: 10.1080/1061856031000135091.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.

Z. Peng and J. F. Brady. Upstream swimming and taylor dispersion of active brownian particles. *Phys. Rev. Fluids*, 5:073102, Jul 2020. doi: 10.1103/PhysRevFluids.5.073102. URL https://link.aps.org/doi/10.1103/PhysRevFluids.5.073102.

R.-E. Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 11 2006. ISSN 0956-540X. doi: 10.1111/j.1365-246X.2006.02978.x. URL https://doi.org/10.1111/j.1365-246X.2006.02978.x.

J. Rabault, M. Kuchta, A. Jensen, U. Réglade, and N. Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, Apr. 2019. doi: 10.1017/jfm.2019.62.

M. Roper, T. M. Squires, and M. P. Brenner. Symmetry unbreaking in the shapes of perfect projectiles. *Physics of Fluids*, 20(9):093606-093606-13, Sept. 2008. doi: 10.1063/1.2982500.

M. Sahimi, B. D. Hughes, L. Scriven, and H. Ted Davis. Dispersion in flow through porous media—i. one-phase flow. *Chemical Engineering Science*, 41(8):2103–2122, 1986. ISSN 0009-2509. doi: https://doi.org/10.1016/0009-2509(86)87128-7. URL https://www.sciencedirect.com/science/article/pii/0009250986871287.

S. S. Schoenholz and E. D. Cubuk. Jax, m.d. a framework for differentiable physics*. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124016, dec 2021. doi: 10.1088/1742-5468/ac3ae9. URL https://dx.doi.org/10.1088/1742-5468/ac3ae9.

M. Schrimpf, J. Esteban, H. Warmeling, T. Färber, A. Behr, and A. J. Vorholt. Taylor-couette reactor: Principles, design, and applications. *AIChE Journal*, 67(5):e17228, 2021. doi: https://doi.org/10.1002/aic.17228. URL https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/aic.17228.

T. A. Shams, S. I. A. Shah, A. Javed, and S. H. R. Hamdani. Airfoil selection procedure, wind tunnel experimentation and implementation of 6dof modeling on a flying wing micro aerial vehicle. *Micromachines*, 11(6), 2020. ISSN 2072-666X. doi: 10.3390/mi11060553. URL https://www.mdpi.com/2072-666X/11/6/553.

M. Shuman, Emily K. and M. Chenoweth, Carol E. Urinary catheter-associated infections. *Infectious disease clinics of North America*, 32(4):885–897, 2018. ISSN 0891-5520.

T. M. Squires, S. R. Manalis, and R. J. Messinger. Making it stick: convection, reaction and diffusion in surface-based biosensors. *Nature biotechnology*, 26(4):417–426, 2008. ISSN 1087-0156.

D. Stoecklein, K. G. Lore, M. Davies, S. Sarkar, and B. Ganapathysubramanian. Deep learning for flow sculpting: Insights into efficient learning using scientific simulation data. *Scientific reports*, 7(1):46368–46368, 2017. ISSN 2045-2322.

D. Stoecklein, M. Davies, J. M. de Rutte, C.-Y. Wu, D. Di Carlo, and B. Ganapathysubramanian. Flowsculpt: software for efficient design of inertial flow sculpting devices. *Lab on a chip*, 19(19):3277–3291, 2019. ISSN 1473-0197.

Z. B. Stone and H. A. Stone. Imaging and quantifying mixing in a model droplet micromixer. *Physics of Fluids*, 17(6):063103, 06 2005. ISSN 1070-6631. doi: 10.1063/1.1929547. URL https://doi.org/10.1063/1.1929547.

R. Sturman and S. Wiggins. Eulerian indicators for predicting and optimizing mixing quality. *New Journal of Physics*, 11(7):075031, July 2009. doi: 10.1088/1367-2630/11/7/075031.

P. D. Swanson. *Regular and chaotic mixing of viscous fluids in eccentric rotating cylinders*. PhD thesis, 1991.

A. P. Toshev, H. Ramachandran, J. A. Erbesdobler, G. Galletti, J. Brandstetter, and N. A. Adams. JAX-SPH: A Differentiable Smoothed Particle Hydrodynamics Framework. *arXiv e-prints*, art. arXiv:2403.04750, Mar. 2024. doi: 10.48550/arXiv.2403.04750.

S. Vedantam, J. B. Joshi, and S. B. Koganti. Three-dimensional cfd simulation of stratified two-fluid taylor-couette flow. *The Canadian Journal of Chemical Engineering*, 84(3):279–288, 2006. doi: https://doi.org/10.1002/cjce.5450840303. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cjce.5450840303.

E. Villermaux. Mixing versus stirring. *Annual Review of Fluid Mechanics*, 51 (Volume 51, 2019):245–273, 2019. ISSN 1545-4479. doi: https://doi.org/10.1146/annurev-fluid-010518-040306. URL https://www.annualreviews.org/content/journals/10.1146/annurev-fluid-010518-040306.

A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006. ISSN 0025-5610.

B. Yagiz, O. Kandil, and Y. V. Pehlivanoglu. Drag minimization using active and passive flow control techniques. *Aerospace Science and Technology*, 17(1):21–31, 2012. ISSN 1270-9638. doi: https://doi.org/10.1016/j.ast.2011.03.003. URL https://www.sciencedirect.com/science/article/pii/S1270963811000393.

T. Zhou, X. Wan, D. Z. Huang, Z. Li, Z. Peng, A. Anandkumar, J. F. Brady, P. W. Sternberg, and C. Daraio. Ai-aided geometric design of anti-infection catheters. *Science Advances*, 10(1):eadj1741, 2024. doi: 10.1126/sciadv.adj1741. URL https://www.science.org/doi/abs/10.1126/sciadv.adj1741.

M. Zuzovsky, P. M. Adler, and H. Brenner. Spatially periodic suspensions of convex particles in linear shear flows. III. Dilute arrays of spheres suspended in Newtonian fluids. *Physics of Fluids*, 26(7):1714–1723, July 1983. doi: 10.1063/1.864370.