Microfluidic Device Design Report

Raymond Chu[1], Alejandro Diaz[2], Dominic Diaz[1], Siting Liu[1], Izak Oltman[3],
Mentors: Hangjie Ji[1], Marcus Roper[1]

[1] *Department of Mathematics, University of California, Los Angeles*

[2] *Department of Mathematics, University of Maryland, College Park*

[3] *Department of Mathematics, University of Wisconsin, Madison*

August 17, 2018

Abstract

Properties of inertial flow in microfluidic devices are exploited to aid in many biological and manufacturing problems. However, there is a lack of theory to predict the focusing position of a particle in these microfluidic devices. To aid in this, we derive two linear differential equations to approximate the migration velocity of the particle flowing in fluid in a pipe of arbitrary cross-section under the assumption that the particle radius is asymptotically small. In our equations we replace the particle by a stresslet or a singular force. To aid our finite element method software in accurately solving these equations, we introduce regularization terms and show that these regularizations converge to well known results on an infinite plate. We then extend our governing equations to a square channel and show good agreement with prior results. To demonstrate flexibility of our method, we consider a pipe whose cross-section is a scalene triangle. Now consider a rectangular cross-section with inertial fluid flow passing through. Due to having inertial flow in the channel, the flow lacks fore-aft symmetry about obstacles placed in the channel. Thus, some initial cross sectional portion of dyed fluid will deform to a new shape once it passes a pillar in the channel. We focused our efforts on developing a method that computes the optimal pillar configuration to reproduce a desired output shape. We used the discrete Frećhet distance to compare our desired output shape to the output shape after some pillar configuration. We then used this distance metric in a greedy algorithm, which was efficient but not accurate enough at finding the best pillar configuration. Due to the difficulty of converging to a global minimum, sweeping through all possible pillar configurations of an intelligently sub-sampled data set proved very accurate but had a computational complexity that increased exponentially with the number of pillars in the channel.

Keywords: Inertial Focusing Position, Mircofluidics, Migration Velocity, Microfluidic Pillars, Pillar Sculpting

1. Introduction

The importance of flow with finite Reynolds number ($1 \leq Re \leq 100$) in micro-channels has recently come into light. Unlike Stokes flow ($Re \ll 1$), suspended particles can cross streamlines

Email addresses: raychu92@ucla.edu (Raymond Chu[1]), diaza5252@gmail.com (Alejandro Diaz[2]), domonickdiaz@gmail.com (Dominic Diaz[1]), siting6@math.ucla.edu (Siting Liu[1]), ioltman@wisc.edu (Izak Oltman[3])

resulting in net deformation of the flow. With $Re \ll 2300$, inertial flow is not operating within the realm of turbulent flow, so our flow is deterministic. These properties have been shown to have potential to aid in solving many biological and manufacturing problems.

One such application was discovered by Amini *et al.* [2]. They considered inertial flow in micro-channels with cylindrical pillars placed within the channel to deform the flow. Due to the deterministic mapping of fluid elements from upstream to downstream, Amini *et al.* [2] were able to control the shape of fluid flow in a micro-channel using different pillar configurations. In particular, they noticed that if the pillars were spread out enough ($\sim$ 6-10 times the diameter of the pillar), then the impact of each pillar became independent of one another. This process, known as pillar sculpting, has become increasingly important in biological and manufacturing fields [1]. Paulsen *et al.* [11] showed that fluid sculpting of a UV sensitive inert fluid can be used to create 3-D objects at higher resolution than layer-by-layer 3-D printing.

Another important application was discovered by Sollier *et al* [13]. They constructed geometries of micro-channels that would trap cancer cells in limit cycles which could significantly benefit cancer detection. However, currently the main methods for constructing such a geometry is through trial and error. This could be improved on with better theoretical results of the inertial focusing positions of particles in an arbitrary geometry. In the mathematical modeling section we approximate the particle's affect on the fluid as a singularity called a stresslet. We used a finite element method (COMSOL, Los Angeles) to obtain the migration velocity of our particle. As our finite element method does not have an extended finite element option, we followed Cortez and Varela's [6] formulation of a regularized stresslet with a small positive parameter ϵ. We also considered an equivalent PDE which replaces the stresslet with the forcing term in Stokes flow that gives rise to the stresslet and a regularized Gaussian that converges to the desired force as $\sigma \to 0$, where σ is the standard deviation for the Gaussian. Then we will show that as $\sigma \to 0$ and $\epsilon \to 0$ for both methods, that our migration velocity converges to already known results for an infinite plate by Schonberg and Hinch [12]. Then we compare our predicted inertial focusing position to known results for the square channel. We then show the generality of our method by predicting the inertial focusing position for a geometry that has not been analytically solved: a scalene right triangle with length 1 and width 0.5. Our methods are both acurrate and converge quickly (i.e. for the plane it took roughly 7 minutes per point and in the triangle it only took 3 minutes per point).

Machine learning and optimization have also provided us with useful methods to describe microfluidic flow past an arbitrary sequence of micropillars. Previously, in order to determine the complex flow transformations, computationally expensive calculations were required. More recently, it was discovered that by assuming that each individual pillar is sufficiently far away from the next one that their effects do not interact so that we can treat each pillar as a state transition matrix [16]. By pre-computing these matrices (through computing the advection of fluid particles around a specific pillar and creating the transition matrix that describes these advections) for each pillar diameter and computing the matrix multiplication, complex pillar configurations can be rapidly simulated. The next problem to be optimally solved is the inverse problem of finding the optimal micropillar sequence that creates a specified flow pattern. Unoptimized approaches to solving this problem required user insight and interaction but as more complex flow fields are desired, user interaction becomes less practical. A genetic algorithm applied to this inverse problem has successfully found novel flow designs and has even been able to decrease the number of pillars needed to sculpt a specific flow field [14, 16]. More recent approaches to solving this problem have shown that deep learning is much faster at solving this inverse problem than user interaction [15]. In terms of deep learning, it has been shown that more accurate methods can be created by intelligently sampling the training data [15]. Ap-

plying intelligent sampling techniques to deep learning approaches has been shown to provide competitive prediction accuracy while drastically reducing computation time when compared with the genetic algorithm [10]. In the machine learning section, we used a greedy algorithm to approach the inverse problem. We also applied Principal Component Analysis (PCA) and K-Means Clustering to the advection data to reduce the dimension of the data set and extract characteristic features. We then used this information to intelligently sub-sample our set of pillars, and examine how accurately a complete sweep of our sub-sampled data set could predict a pillar configuration that outputs a shape close to some target shape.

2. Machine Learning

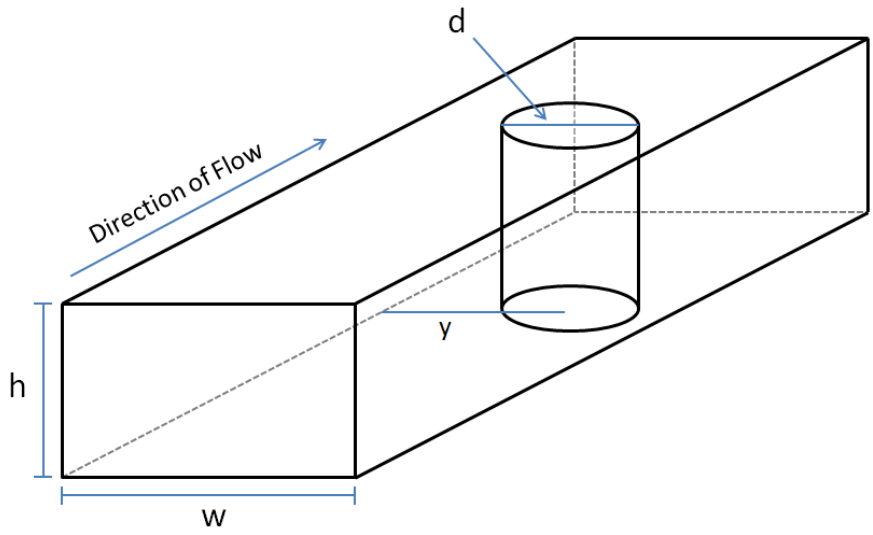


Figure 1: Schematic diagram of a microfluidic channel containing a single pillar.

2.1. Data

At the beginning of this project, we received a large data set with data describing the flow past a single pillar[1]. The data set included flow past 3224 different pillars for a fixed channel aspect ratio (the height divided by the width of the channel, h/w) and varying Reynolds number (Re), pillar diameter aspect ratio (the pillar diameter divided by the width of the channel, d/w), and pillar location (the pillar location divided by the width of the channel, y/w). See figure (1) for a schematic diagram describing each of these different parameters. Each data point provided us with the necessary vectors that described how flow from behind a specific pillar in the channel would deform to a new shape after the pillar. Plotting a single data point allows us to visualize a quiver plot with deformation vectors that describes how a shape changes. See figure (2) for an example quiver plot of a data point with pre and post pillar shape of dyed fluid. We were able to extend the data set to 6448 pillars since the original data only included pillars on the right side of the channel (by symmetry, we were able to extend our data set to include the pillars on the left side of the channel)

[1]The advection data that we analyzed was given to us from the Di Carlo lab at the University of California, Los Angeles and the Baskar group at Iowa State University.

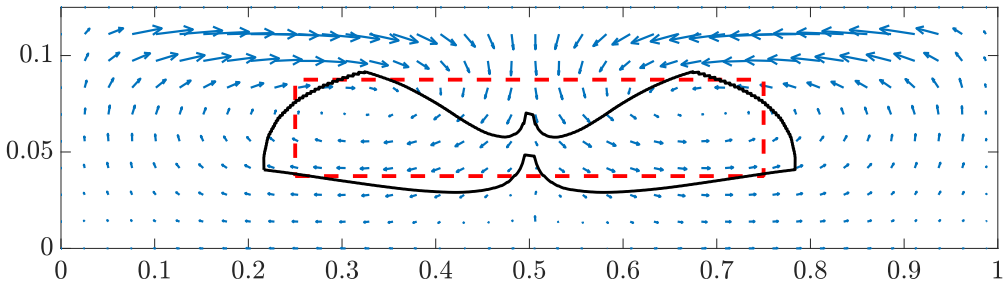


Figure 2: Quiver plot with original shape before the pillar and after the pillar for pillar described by the parameters $h/w = 0.25$, $d/w = 0.76$, $y/w = 0$, $Re = 40$. The blue arrows describe how deformation vectors from before to after the pillar. The red dashed rectangle defines the outline of a shape of dyed fluid before the pillar. The black shape defines the outline of the deformed shape after the pillar. Note that we are only plotting the top half of the quiver plots since the data is symmetric about the top and bottom of the channel.

2.2. Dimensionality Reduction: PCA

Being that each data point from the data set was described by a vector with 161802 entries, we employed a common machine learning technique commonly used for dimensionality reduction and feature extraction of our data set. We used Principal Component Analysis (PCA) to visualize our data in a reasonable dimension and learn more about the data set. PCA is a techinque that computes the eigenvectors of the covariance matrix of the data set. The first few eigenvectors correspond to the vectors that describe the most variance in the data (we call those vectors the Principal Components of the data). Once we find the smallest number n of Principal Components that describe a certain threshold percentage of the variance of the data, we compute the projection of each data point onto the new space defined by the n Principal Components (called the coefficients in the 1st through nth P.C.'s).

2.3. Discrete Fréchet Distance

To reconstruct a desired flow shape by predicting a specific pillar sequence, we needed a distance function to compare two flow shapes. The distance function that we used is the *discrete Fréchet distance*, which is a metric on the space of polygonal curves. We applied this metric by taking the boundaries of two different flow shapes and computing their discrete Fréchet distance with the algorithm provided by Eiter and Mannila [7]. It should be noted that the discrete Fréchet distance is sensitive to scaling and rigid motions of polygonal curves (i.e. translations, rotations, and reflections). For some applications this sensitivity may be desirable, but for others, the user may want the distance to be invariant with respect to scaling or rigid motions. In this case, the user must normalize the polygonal curves they want to compare before computing their distance. Ultimately it is up to the user to decide which predicted flow shapes are acceptable. For this project, we chose to not renormalize the flow shapes to account for scaling and rigid motions.

2.4. Greedy Algorithm

We approached the problem of determining a possible pillar configuration for a given desired flow shape by implementing a greedy algorithm. Greedy algorithms complete a given task by choosing the locally optimal choice at each stage. In our algorithm, we started with a target outlet flow shape and some connected inlet flow shape, then deformed the inlet flow once for each of the possible pillars in our data set. Note that based on validation of theory using numerical results we were able to determine that mass-flow is conserved when it deforms around some

obstacle in a microfluidic channel. Thus, for a specified desired flow shape, we simply computed the mass-flow of the desired shape and computed inlet flow of rectangular shape that span the whole height of the the channel[2] with the same mass-flow for our Greedy Algorithm and also for our sub-sampled sweep of the data set in Section 4.3. Then we calculated the discrete Fréchet distance from our desired shape to each of the possible deformed shapes and chose the pillar that corresponded to the minimum calculated distance. We then updated the inlet flow shape to the deformed shape after the optimal pillar, then iterated the algorithm. We let the algorithm run until either the distance from the predicted shape to the desired shape was less than some $\varepsilon > 0$, or the number of predicted pillars was equal to some $m \in \mathbb{N}$.

The problem with using a greedy algorithm is that it will not necessarily find a global optimum. To account for this, we added a random component to our algorithm: if the distance between the desired shape and predicted shape after k pillars increased after adding the $(k+1)th$ pillar, then we replaced a random number of the first k pillars by randomly chosen pillars. The idea here is that the random pillars would be able to push our predicted pillar configuration towards a global optimum.

2.5. K Means

Considering that the number of possible pillars we can use is large (6448), we decided to intelligently sub-sample from the data set in order to reduce the necessary computation and the run time. We coupled a machine learning technique called K Means with the results of our PCA to intelligently sample from the data. The K means algorithm is an iterative process that allows us to separate our data set into k different means based on their location in the PCA space.

3. Mathematical Modeling

3.1. Statement of Problem

Consider an infinitely long pipe with an arbitrary cross section of characteristic length ℓ. In this pipe flows a fluid governed by Poiseuille flow. We then introduce a spherical particle of radius a (much smaller than ℓ) into our channel, which induces a disturbed flow in our geometry. Additionally, we choose a lab reference frame such that the fluid flows in the z direction. In addition, assume that the Channel Reynolds Number $R_c = \frac{U_m \ell}{\nu}$ is finite, where U_m is the maximum background flow velocity, and ν is the kinematic viscosity of our fluid. As our Channel Reynolds Number is too large to be approximated by 0, we see that our flow is not operating within Stokes flow. Hence, we do not have the mirror symmetry and time reversibility properties of Stokes flow, which forbids particles from crossing stream lines.

[2]The only inlet flow shape that engineers can currently create is a rectangle that spans the whole height of the channel due to the small scale of microfluidic channels.

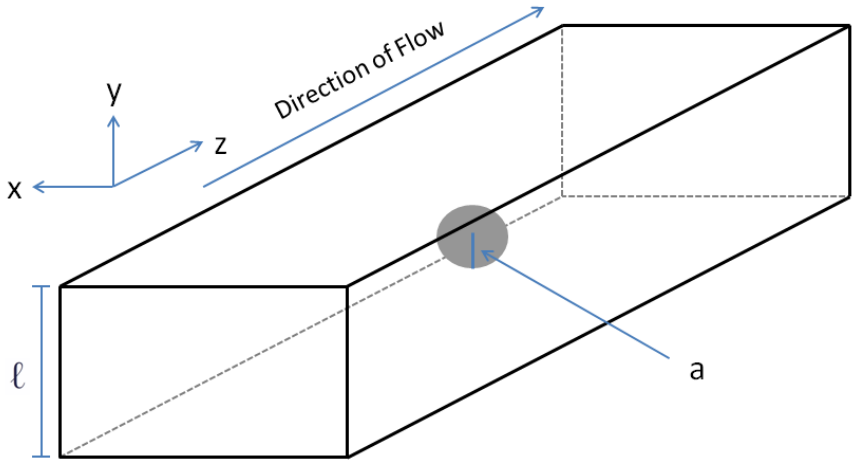


Figure 3: An example of our reference frame in a rectangular cross section.

As our flow is not in Stokes flow, we see that particles can move across stream lines. We define the migration velocity of a particle to be the velocity of the particle in the x and y directions. We present a theory of approximating the leading order term of the migration velocity of a particle with the assumption that the particle Reynolds number $R_p = (\frac{a}{\ell})^2 R_c$ is asymptotically small for an arbitrary geometry governed by the above assumptions. From these assumptions, we derive a linearized Navier–Stokes equation that governs the migration velocity. Then as our geometry is time dependent due to the movement of the particle, we introduce two methods of replacing the particle with a stresslet or a singular force, so that we can have a fixed geometry. We will show that this equation gives good results in comparison to well known results for an infinite plate and a square channel. Note that this method can be extended to solve for the inertial focusing position of a particle by calculating migration velocities on a grid of points and determining stable equilibrium.

3.2. Governing Equations

Let our lab frame have Cartesian coordinates such that the fluid flows in the z direction and the cross section is the span of the x and y directions. Let $\mathbf{u}$ be the total velocity of the channel, ρ the density of the fluid, μ the fluid viscosity, and p the pressure. We assume that μ and ρ are constants and that our flow is incompressible ($\nabla \cdot \mathbf{u} = 0$). If our sphere of radius a travels with velocity $\widetilde{\mathbf{u}_s}$ and rotational velocity Ω, then from the incompressible Navier–Stokes equations, we have the following governing equations:

$$\rho\Big(\frac{\partial}{\partial t}\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u}\Big) = -\nabla p + \mu\nabla^2\mathbf{u} \tag{1a}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{1b}$$

$$\mathbf{u} = 0 \text{ on walls} \tag{1c}$$

$$\mathbf{u} = \widetilde{\mathbf{u}_s} + \Omega\times a \text{ on particle surface} \tag{1d}$$

Notice that due to the final boundary condition, our geometry is time dependent. We then decompose the total fluid velocity as

$$\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}',$$

where $\overline{\mathbf{u}}$ is the background Poiseuille flow, and $\mathbf{u}'$ is the disturbed flow caused by the particle. With this decomposition, our momentum equation (1a) becomes

$$\rho\Big(\frac{\partial}{\partial t}(\overline{\mathbf{u}} + \mathbf{u}') + (\overline{\mathbf{u}} + \mathbf{u}') \cdot \nabla(\overline{\mathbf{u}} + \mathbf{u}')\Big) = -\nabla(p' + \overline{p}) + \mu\nabla^2(\overline{\mathbf{u}} + \mathbf{u}') \tag{2}$$

As our background flow ($\overline{\mathbf{u}}$) solves Navier-Stokes equation, we can reduce (2) to

$$\rho\Big(\frac{\partial}{\partial t}\mathbf{u}' + \overline{\mathbf{u}} \cdot \nabla\mathbf{u}' + \mathbf{u}' \cdot \nabla\overline{\mathbf{u}} + \mathbf{u}' \cdot \nabla\mathbf{u}'\Big) = -\nabla p' + \mu\nabla^2\mathbf{u}'. \tag{3}$$

We now transform our coordinates to a moving reference frame traveling in the z direction with the particle, with speed $\widetilde{\mathbf{u}_s} \cdot \mathbf{e}_z \equiv u_s$. Explicitly, if (x, y, z, t) are the lab coordinates, and (x', y', z', t') are the new coordinates, then $x = x', y = y', t = t'$ and $z' = z - \int_0^t u_s(t'')dt''$. Our derivatives will then change:

$$\begin{aligned} \nabla_{\mathbf{x}} &\to \nabla_{\mathbf{x}'} \\ \frac{\partial}{\partial t} &\to \frac{\partial}{\partial t'} - \mathbf{u}_s \cdot \nabla_{\mathbf{x}'}, \end{aligned}$$

where $\mathbf{u}_s = u_s\mathbf{e}_z$. Note that while we are in a moving frame, we are measuring velocities with respect to the lab frame. With this our momentum equation becomes:

$$\rho\Big(\frac{\partial}{\partial t}\mathbf{u}' - \mathbf{u}_s \cdot \nabla\mathbf{u}' + \overline{\mathbf{u}} \cdot \nabla\mathbf{u}' + \mathbf{u}' \cdot \nabla\overline{\mathbf{u}} + \mathbf{u}' \cdot \nabla\mathbf{u}'\Big) = -\nabla p' + \mu\nabla^2\mathbf{u}' \tag{4}$$

Next, let $\overline{\mathbf{u}_s}$ be the Poiseuille flow evaluated at the center of the sphere, and $\overline{\mathbf{u}'} = \overline{\mathbf{u}} - \overline{\mathbf{u}_s}$ be the background flow velocity measured with respect to the moving particle. By decoupling the background flow, we can write (4) as

$$\rho\Big(\frac{\partial}{\partial t}\mathbf{u}' + (\overline{\mathbf{u}_s} - \mathbf{u}_s) \cdot \nabla\mathbf{u}' + \overline{\mathbf{u}'} \cdot \nabla\mathbf{u}' + \mathbf{u}' \cdot \nabla\overline{\mathbf{u}} + \mathbf{u}' \cdot \nabla\mathbf{u}'\Big) = -\nabla p' + \mu\nabla^2\mathbf{u}' \tag{5}$$

The next step is to nondimensionlize this equation, expand variables in terms of a, and approximate to obtain a linear equation. Consider the nondimensional variables

$$\begin{aligned} r^* &= \frac{r}{l} & t^* &= \frac{U_m t R_p}{l} \\ \mathbf{u}^* &= \frac{\mathbf{u}}{U_m} & p^* &= \frac{pl}{\mu U_m}. \end{aligned}$$

Note that the time is scaled by R_p because the time scale at which the disturbed velocity changes in (5) should be of order R_p. This is because the disturbed velocity will only change over time if the particle migrates, and if the radius of the particle goes to zero, we should see no migration. Our derivatives will change again:

$$\nabla_x \to \frac{1}{l}\nabla_{x^*} \qquad \frac{\partial}{\partial t} \to \frac{U_m R_p}{l}\frac{\partial}{\partial t^*}$$

Replacing (5) with these nondimensional variables, replacing all velocities $\mathbf{u}$ with $U_m\mathbf{u}^*$ and getting rid of stars, we have

$$\left(R_p\frac{\partial}{\partial t}\mathbf{u}' + (\overline{\mathbf{u}_s} - \mathbf{u}_s)\cdot\nabla\mathbf{u}' + \overline{\mathbf{u}'}\cdot\nabla\mathbf{u}' + \mathbf{u'}\cdot\nabla\overline{\mathbf{u}} + \mathbf{u}'\cdot\nabla\mathbf{u}'\right) = -\nabla p' + \nabla^2\mathbf{u}'. \tag{6}$$

Following [12] we can approximate the unknown velocities and pressures in terms of α and R_p. We assume:

$$\mathbf{u}' \sim \alpha R_p\mathbf{u}_0' \qquad\qquad (\overline{\mathbf{u}_s} - \mathbf{u}_s) \sim \alpha R_p(\overline{\mathbf{u}_s} - \mathbf{u}_s)_0$$
$$p' \sim \alpha R_p p_0'$$

Plugging in these expansion into (6), we have that (6) becomes

$$\alpha^2 R_p^2\left(\frac{1}{\alpha}\frac{\partial}{\partial t}\mathbf{u}_0' + (\overline{\mathbf{u}_s} - \mathbf{u}_s)_0\cdot\nabla\mathbf{u}_0' + \frac{1}{\alpha R_p}(\overline{\mathbf{u}'}_0\cdot\nabla\mathbf{u}_0' + \mathbf{u}_0'\cdot\nabla\overline{\mathbf{u}}_0) + \mathbf{u}_0'\cdot\nabla\mathbf{u}_0'\right)$$
$$= \alpha R_p(-\nabla p' + \nabla^2\mathbf{u}')$$

Expanding this out and only considering first order terms, we arrive at

$$\left(\overline{\mathbf{u}'}_0\cdot\nabla\mathbf{u}_0' + \mathbf{u}_0'\cdot\nabla\overline{\mathbf{u}}_0\right) = -\nabla p' + \nabla^2\mathbf{u}'.$$

We then redimensionalize this to get the following dimensional linear equation

$$\rho\left(\overline{\mathbf{u}'}\cdot\nabla\mathbf{u}' + \mathbf{u}'\cdot\nabla\overline{\mathbf{u}}\right) = -\nabla p' + \mu\nabla^2\mathbf{u}' \tag{7}$$

We also have the following boundary and incompressability conditions (ignoring particle rotation):

$$\begin{gathered}\mathbf{u}' = \widetilde{\mathbf{u}_s} + \mathbf{u}_m - \bar{\mathbf{u}} \text{ on the boundary of the particle} \\ \mathbf{u}' = 0 \text{ on channel walls and as } r\to\infty \\ \nabla\cdot\mathbf{u}' = 0\end{gathered} \tag{8}$$

Our next step is to get rid of the boundary condition on the surface of the particle (8). Batchelor [3] presents an approximation for the disturbed velocity field due to a sphere in shear flow that is force and torque free. We will denote this approximation as $\mathbf{u}_{str}$ This velocity satisfies the following Stokes equation with a singularity:

$$\nabla p_{str} = \mu\nabla^2\mathbf{u}_{str} + \mathbf{f}_{str}, \tag{9}$$

where $\mathbf{f}_{str}$ is singular force acting at the location of the particle. By following Chwang and Wu [4], we see that $\mathbf{f}_{str}$ is the symmetric portion of the derivative of the Dirac delta function.

We use this to approximate the boundary condition on the particle (8) by the condition that

$$\mathbf{u}' \sim \mathbf{u}_{str} \text{ for } a \ll r \ll \ell \tag{10}$$

Next we get rid of this boundary condition by adding the stresslet forcing term $\mathbf{f}_{str}$ to the momentum equation (7). So that our final governing equations become:

$$\rho\Big(\overline{\mathbf{u}'}\cdot\nabla\mathbf{u}'+\mathbf{u}'\cdot\nabla\overline{\mathbf{u}}\Big)=-\nabla p'+\mu\nabla^2\mathbf{u}'+\mathbf{f}_{str} \tag{11}$$
$$\mathbf{u}'=0 \text{ on channel walls and as } r\to\infty$$
$$\nabla\cdot\mathbf{u}'=0$$

To implement these equations into COMSOL we used two different techniques. In the first method, we decouple our disturbed velocity field into a regular and singular portion and solve for the regular part. The second method directly applies the singular force in equation (11) into COMSOL. While implementing these singular terms into COMSOL, we found that our results were not very accurate. We fix this discrepancy by introducing a regularized stresslet and by approximating our singular force by a Gaussian. We will show that our approximations converge for well known results for infinite plate channels.

3.3. The stresslet method

For this method, we decouple the disturbed velocity $\mathbf{u}'$ as

$$\mathbf{u}'=\mathbf{u}_{str}+\mathbf{v},$$

where $\mathbf{u}_{str}$ is the stresslet velocity and $\mathbf{v}$ is the regular part of the disturbed velocity. Our momentum equation and boundary conditions become:

$$\rho\Big(\overline{\mathbf{u}'}\cdot\nabla\mathbf{u}'+\mathbf{u}'\cdot\nabla\overline{\mathbf{u}}\Big)=-\nabla(p_{str}+p_v)+\mu\nabla^2(\mathbf{u}_{str}+\mathbf{v})+\mathbf{f}_{str} \tag{12}$$
$$\mathbf{v} \text{ nonsingular as } r\to 0 \tag{13}$$
$$\mathbf{v}=-\mathbf{u}_{str} \text{ on walls}$$

We can get rid of the stresslet terms on the right side of (12) by using the definition of of the stresslet in (9). So our final momentum equation becomes

$$\rho\Big(\overline{\mathbf{u}'}\cdot\nabla(\mathbf{u}_{str}+\mathbf{v})+(\mathbf{u}_{str}+\mathbf{v})\cdot\nabla\overline{\mathbf{u}}\Big)=-\nabla p_v+\mu\nabla^2\mathbf{v}. \tag{14}$$

By Chwang and Wu [4] we have an explicit formula for the velocity field of our stresslet, which is given by

$$\mathbf{u}_{str}=\frac{-5a^3}{2r^5}(\gamma_x xz\mathbf{r}+\gamma_y yz\mathbf{r}),$$

where $\gamma_x=\partial_x\overline{\mathbf{u}}$, $\gamma_y=\partial_y\overline{\mathbf{u}}$, and $r=\sqrt{x^2+y^2+z^2}$. We can now solve for the regular part of the disturbed flow $\mathbf{v}$. By Schonberg and Hinch [12], $\mathbf{v}$ evaluated at center of the sphere is the migration velocity.

We found significant discrepancies between our results compared to Schonberg and Hinch [12] when we implemented equation (12) into our finite element solver. We hypothesized that this was due to the singularity of our stresslet. To fix this we used a regularization method as described in [5].

This regularization method approximates the Dirac delta in the singular forcing term by a

blob function. We follow Cortex and Varela [6] to approximate our forcing term by:

$$\phi_\varepsilon(r) = \frac{15\varepsilon^4}{8\pi(r^2+\varepsilon^2)^{7/2}}$$

According to Cortex and Varela [6] our regularized stresslet velocity field's i^{th} component is

$$\mathbf{u}_i = \frac{\mathbf{H}_2'}{r}\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k\mathbf{Z}_{jk} + (\mathbf{H}_2 + \frac{\mathbf{H}_1'}{r})\mathbf{x}_k\mathbf{Z}_{ik},$$

where

$$\nabla^2\mathbf{G} = \phi_\varepsilon \qquad \nabla^2\mathbf{B} = \mathbf{G}$$
$$\mathbf{H}_1(r) = r^{-1}\mathbf{B}' - \mathbf{G} \qquad \mathbf{H}_2(r) = r^{-3}(r\mathbf{B}'' - \mathbf{B}')$$

and

$$Z = \frac{10a^3\pi}{3}\begin{pmatrix} 0 & 0 & \gamma_x \\ 0 & 0 & \gamma_y \\ \gamma_x & \gamma_y & 0 \end{pmatrix}.$$

We solved for $\mathbf{B}$ and $\mathbf{G}$ in $\mathbb{R}^3$ by noticing that our blob function is radially symmetric, which reduces the Laplace equation to an ordinary differential equation.

Hence, our regularized stresslet is

$$\mathbf{u}_{str} = \frac{-5a^3}{4(\varepsilon^2+r^2)^{5/2}}\begin{pmatrix} 2xz(\gamma_x x + \gamma_y y) + \varepsilon^2\gamma_x z \\ 2yz(\gamma_x x + \gamma_y y) + \varepsilon^2\gamma_y z \\ 2z^2(\gamma_x x + \gamma_y y) + \varepsilon^2(\gamma_x x + \gamma_y y) \end{pmatrix}.$$

Note that when $\varepsilon = 0$ this coincides with the non-regularized stresslet velocity field. In our results section, we show that as ε goes to zero, our results for the infinite plan converge to well known results.

3.4. Dirac Delta function method

If we will modify the Navier–Stokes equation the same way as before (12), but we replace the boundary condition with a singular force. Instead of looking at the decomposition of the disturbed velocity, we calculate the disturbed velocity directly.

$$R_c\Big(\bar{\mathbf{u}}'\cdot\nabla\mathbf{u}' + \mathbf{u}'\cdot\nabla\bar{\mathbf{u}}'\Big) = -\nabla p' + \nabla^2\mathbf{u}' + \mathbf{F} \tag{15}$$
$$\nabla\cdot\mathbf{u}' = 0$$
$$\mathbf{u}' = 0 \text{ on the boundary}$$

with

$$\mathbf{F} = -\frac{10\pi\gamma}{3}(\partial_z\delta(x-x_s)\mathbf{e}_x + \partial_x\delta(x-x_s)\mathbf{e}_y), \tag{16}$$

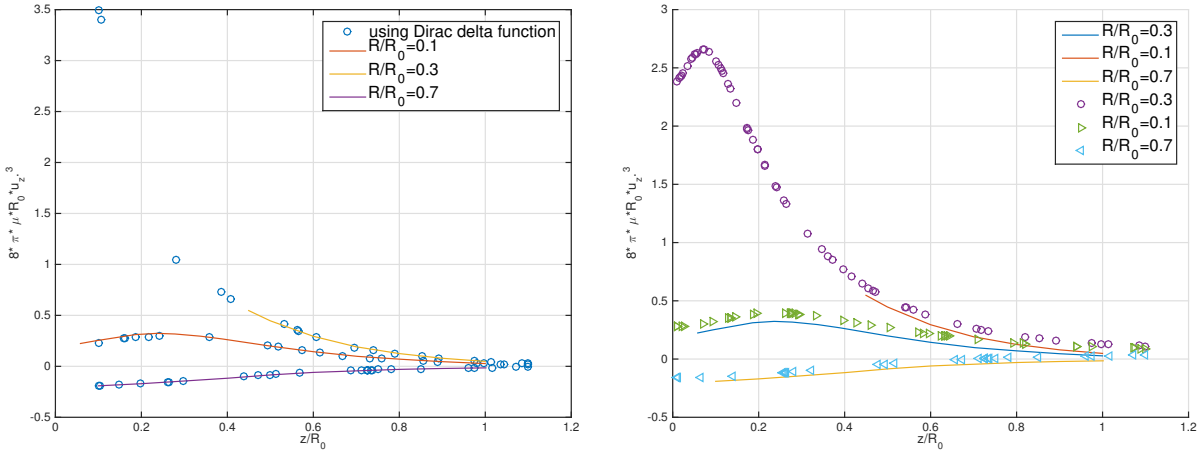
where δ is the Dirac delta function. We would like to know if we can determine the migration velocity using this governing equation directly.

In order to see how COMSOL Multiphysics deals with singularities, we first test it by solving

a 3-Dimensional Stokes equation with a pointwise force at the center of the cylinder pipe:

$$-\nabla p = \mu \nabla^2 \mathbf{u} + \mathbf{F}, \tag{17}$$

where the singular force is of the form $\mathbf{F} = \delta(x - x_s)\mathbf{e}_z$, located at $x = x_s$ with direction along the streamline. In COMSOL, the pointwise weak contribution reads as $test(w)$ at the origin. We set no-slip boundary condition on the wall of cylinder pipe, and periodic boundary condition on both ends of the pipe to model an infinite pipe. A pointwise pressure constraint is imposed on one point at the end of pipe. By comparing the velocity field in the y direction along three cut lines in the cylinder with results from [9], we can see from that the numerical results matches well with the analytic results far away from the singularity. (see figure (4)). However, as we approach the singularity, our results becomes inaccurate and very dependent on the mesh. In general, COMSOL Multiphysics deals with Dirac Delta Function very well when we are sufficiently far away from the singularity.



(a) Using Dirac Delta function (b) Using Gaussian approximation

Figure 4: Velocity field along different cut line in the cylinder pipe of stokes flow due to stokeslet in a pipe. The lines represent data from [9], while circles and triangles are results from our numerical experiments

Next, we set up a numerical experiment based on the singular force is of the form (16). The forcing term $\mathbf{F}$, when implemented in COMSOL, reads as a point-wise weak contribution $\frac{-10\pi\gamma}{3}(test(uz) + test(wx))$. We set up extremely fine mesh near the particle's location with finite element size of 1×10^{-5} in order to get a better resolution of the singularity. To evaluate the the migration velocity, we have two approaches. One way is to evaluate $\mathbf{u}$ at the location of the particle. Another way to approximate the migration velocity $\mathbf{u}$ is to implement the volume average of velocity over a sphere centred at the particle's center with some radius $r = r_s$. This average should approach the migration velocity since the stresslet ($\mathbf{u}_{str}$) is an odd function so when integrating $\mathbf{u} = \mathbf{v} + \mathbf{u}_{str}$, everything but the regular part drops out. However, due to finite mesh and the way that COMSOL deals with derivatives, we can't get enough points to resolve the singularity; neither direct evaluation of $\mathbf{u}$ nor the volume average of $\mathbf{u}$ gives a good approximation. Both depend on the mesh near the singularity and r_s. For example, in Figure 5, it shows the x-direction disturbance velocity along a cutline through the particle's center along the streamline direction. It is symmetric and we expected the integration to cancel out the $\mathbf{u}_{str}$ portion and give an approximation of $\mathbf{u}$ at $x = x_s$. But near the singularity there are only a few points, $\mathbf{u}$ is not resolved and the numerical errors are significant. Therefore, we need to

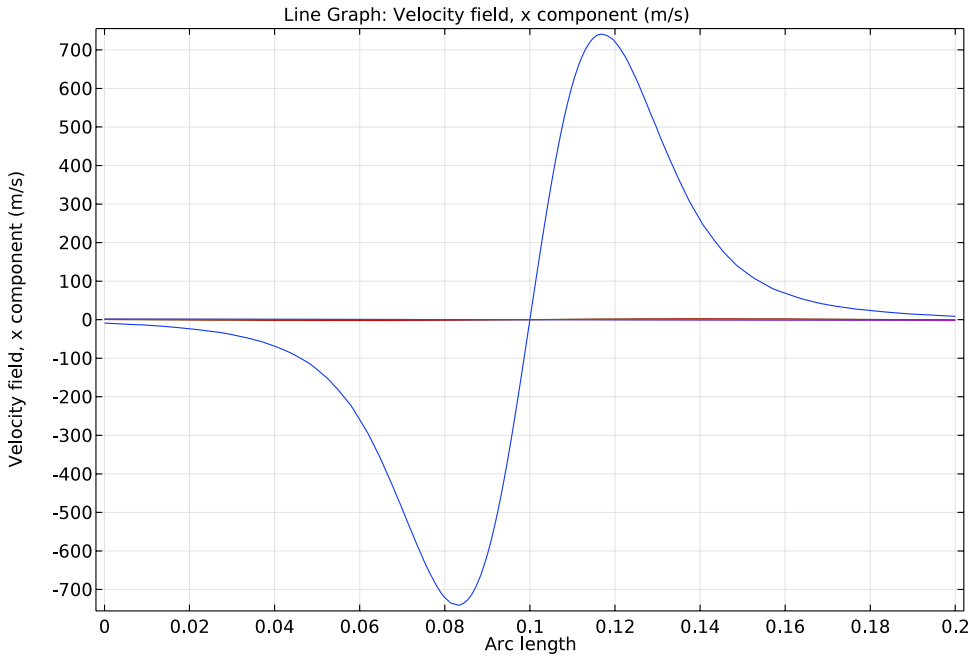
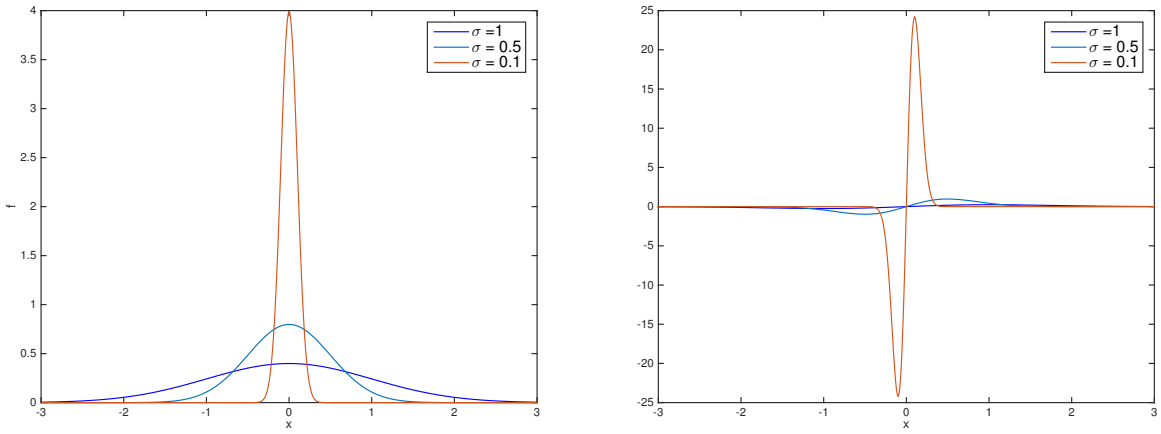


Figure 5: X-direction Velocity along the cutline

regularize the forcing term to get COMSOL work. A direct approach to approximate the Dirac delta function is to use the Gaussian distribution and send its variance $\sigma \to 0$.

$$\delta(x - x_s) = \lim_{\sigma \to 0} f_\sigma(x - x_s) = \lim_{\sigma \to 0} \frac{1}{2\sigma\pi^{1/2}} \exp\left(-(\frac{x - x_s}{2\sigma})^2\right) \tag{18}$$



(a) Gaussian distribution (b) Derivative of Gaussian distribution

Figure 6: Gaussian distribution and its derivative for different value of deviation σ

The derivative of Dirac delta function will be approximated by the derivative of the Gaussian (see figure (6)). We also implement the Gaussian approximation to replace the Dirac delta function in the cylinder pipe, and the result agree with the existing data (see figure (4)). Therefore, instead of having a pointwise weak contribution, the approximation expressed as the derivative

of the Gaussian distribution will act as a volume force in the system. The force is of form

$$\overline{\mathbf{F}} = -\frac{10\pi\gamma}{3}(\partial_z f_\sigma(x - x_s)\mathbf{e}_x + \partial_x f_\sigma(x - x_s)\mathbf{e}_y). \tag{19}$$

We will present numerical results via Gaussian approximation in later section.

4. Machine Learning Results

4.1. PCA

By applying PCA on the whole data set, we find that 97.02 percent of the data can be described by the first three Principal components. We plot the coefficients of the data set in the space of the first two Principal Components for visualization in Figure 7. The PCA is able to pick up varying pillar diameter (larger pillars tend to be further from the origin) and varying pillar location (similar pillar locations tend to be along the same polar angle from the origin) but does not pick up varying Reynolds number since there generally does not tend to be any sort of kernel function that can separate data for different Reynolds numbers. Being that a large percentage of the data was described by the first two Principal Components (94.64 percent), we examined the first few Principal Components by plotting them each as a quiver plot in a cross section in the channel. The first four Principal Components are plotted in Figure 8. The first four Principal Components of our data set were used as motivation for the K Means Clustering that we describe in section 2.5. The main motivation comes from the fact that the PCA picked up the most variation in the data with the first two Principal Components (see Fig. 8a and Fig. 8b) which contained a single node in the center of the channel and two side-by-side nodes with opposite rotations, respectively.

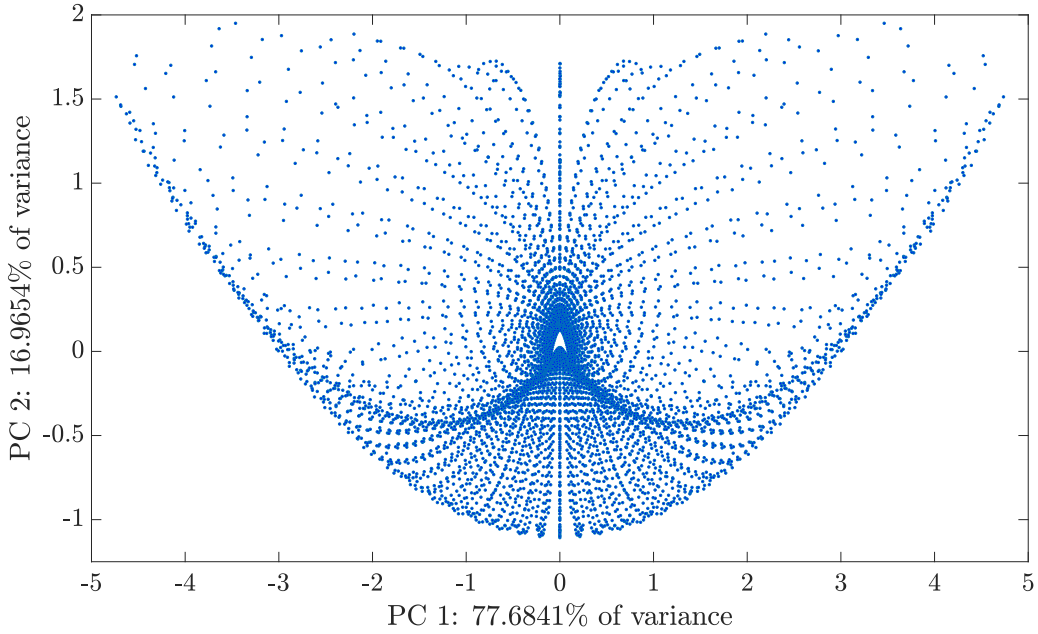
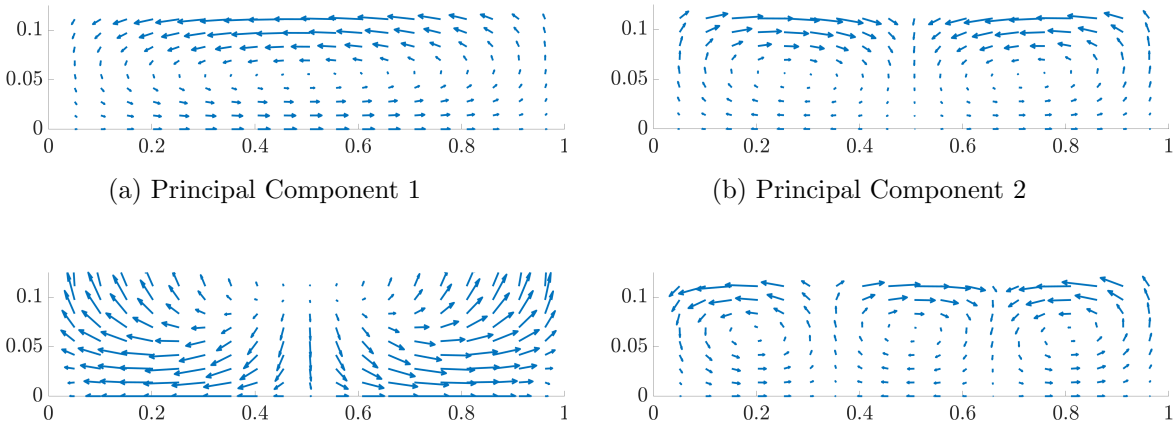


Figure 7: Coefficients of all 6448 data points in the space of the first two Principal Components. Each axis label describes how much variance in the data set is described by each Principal Components, respectively.



(a) Principal Component 1

(b) Principal Component 2

(c) Principal Component 3

(d) Principal Component 4

Figure 8: Plots of the first 4 Principal Components as quiver plots in a channel cross section.

4.2. Greedy Algorithm

To test our greedy algorithm, we first wanted to check the assumption that minimizing the discrete Fréchet distance from the deformed flow shape after each pillar to the desired shape will produce a predicted shape that is close enough to the desired shape. We checked this by first taking all pillars of a fixed Reynold's number, resulting in a sample of 1612 pillars, randomly subsampling the number of pillars from 1612 to 50, and then producing a flow shape from a random sequence of three pillars. We then found the deformed flow shapes from all combinations of three pillars from our 50-pillar data set (resulting in 50^3 flow shapes) and calculated the distances from each of these shapes to our initial randomly generated shape.

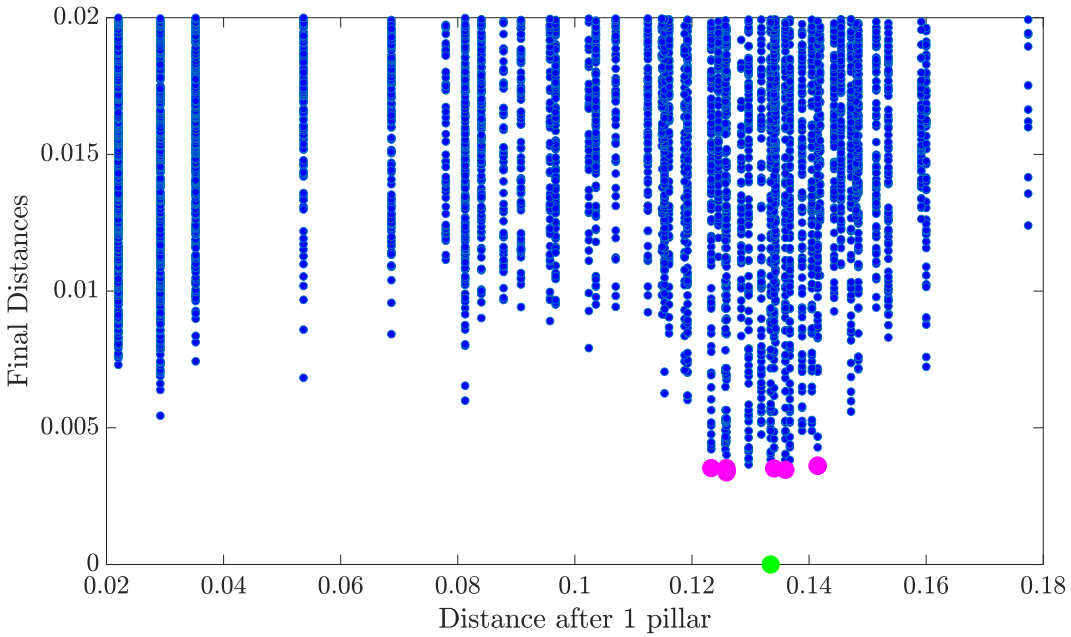


Figure 9: Discrete Fréchet distance after one pillar vs distance after three pillars. The green dot corresponds to the smallest distance and the pink dots correspond to the smallest 10 distances.

From figure 9 we see that choosing the pillar with the smallest initial distance after the first

pillar will not always converge to a global optimum. As stated in section 2.4, to "push" the predicted pillar configuration towards a global optimum, we introduced a random component to our greedy algorithm. We did this in two ways. First we calculated the distance from the predicted flow shape after each pillar to the desired flow shape, and if the distance from the kth pillar to the $(k+1)th$ pillar increased, then we randomly replaced a random number of any of the first k predicted pillars. For our second method, instead of replacing any of the previous k pillars, we only allowed for random replacement of the pillars in the sequence that cause the largest change in distance. We tested both of these replacement schemes to recreate a 'U' shape[3].

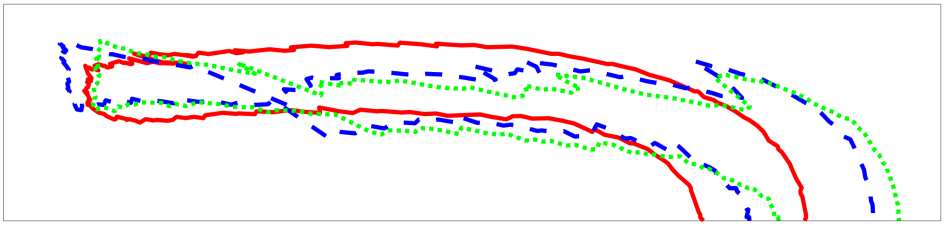


Figure 10: Plot of (the right halves of) the target 'U' shape and 'U' shapes predicted by the greedy algorithm. The solid red shape is the target 'U' shape (the shape we want to recreate from our algorithm), the blue dashed shape is the one obtained by randomly replacing any pillars, and the green dotted shape is obtained by randomly replacing the pillars that cause the largest change in distance. The discrete Fréchet distance between the blue dashed shape and the red solid shape is 0.0514 whereas the distance from the green dotted shape to the red solid shape is 0.0704.

From our comparison of the different random replacement schemes in our greedy algorithm, we saw that the method of replacing any of the previous pillars was slightly better than only replacing the pillars that caused the largest change in distance. However, the 'U' shape obtained from this method was still not close enough to the target shape.

4.3. Using K Means to reduce the data set

The results of our greedy algorithm imply that attempting to find the best combination of pillars to produce a specified desired shape cannot be based on minimizing the the distance after each pillar. This is further emphasized by Figure 9 since we see that the minimum distance after 3 pillars does not correlate with the minimum distance after the first pillar. Even randomly replacing 100 different pillars may not help us get from a local minimum to a global minimum. Consequently, we investigated other techniques to include in our pillar predicting algorithm that satisfy two criteria: (1) Faster than a genetic algorithm utilized in [14] and (2) Predict pillar configurations of the same or better accuracy than the genetic algorithm. Better accuracy in our sense is defined as having an output shape that has a smaller distance than the genetic algorithm and/or requires less pillars to sculpt the same shape.

To satisfy the two above criteria, we decided to intelligently sub-sample from the data set and simply run a sweep of all possible pillar combinations (with a max number of pillars N) of the sub-sampled set to find the best pillar configuration. Note that simply applying the K Means Algorithm described in Section 2.5 on Figure 7 for any number of means does not accurately cluster the data. The reason for this is because the data points must preliminary

[3]The pillar configuration used to create the 'U' shape was obtained from the Di Carlo group at UCLA using 8 pillars.

be separated into 4 classes and then compute K Means on each of those classes. The classes are as follows: one clockwise node, one counterclockwise node (as in Fig. 8a), two nodes with a clockwise one on the left and a counterclockwise node on the right (as in Fig. 8b), and two nodes with a counterclockwise one the the left and a clockwise node on the right. For a fixed Reynolds number ($Re = 30$), we plot the four classes in the space of the first two Principal Components in Figure 11. Note that the advantage of computing K Means on the space of the first two Principal Components is that the PCA tended to have similar quiver plots in spatially similar locations. By considering these four classes as four distinct data sets and computing the K Means Algorithm on each of them individually, we were able to cluster the data more accurately. For each of the Reynolds numbers in our data set ($Re = 10, 20, 30, 40$) we could scale the number of pillars to use in our algorithm from 1612 different pillars to around 60 characteristic pillars (including 30 large and 30 small diameter pillars) that describe most of the data set.

Due to the fact that we want to sweep through all possible combinations of a sequence of m pillars from our reduced data set (consisting of N pillars), the computation scales with N^m (thus, decreasing N significantly decreases computation time). Additionally, considering the fact that the larger pillars are the ones that have the most impact on deforming our initial shape, we can simply sweep through the 30 characteristic pillars with large diameter. As a second attempt to recreate the 'U' shape in Figure 10, running a simple sweep of all combinations of only two pillars from the sub-sampled data set produced the results in Figure 12. By comparing the results in Figure 10 from the Greedy Algorithm and in Figure 12 from a sweep of the sub-sampled data set both visually and quantitatively (using the Frećhet distance), we see that the sweep does a much better job of predicting the optimal pillar configuration.

The advantage of sweeping through the reduced data set is that it allows us to find the best pillar configurations using less pillars. For instance, the Greedy Algorithm's best solutions (plotted in Figure 10) took 8 pillars to find a reasonable shape whereas a Genetic Algorithm also used 8 pillars to make the 'U' shape. Sweeping through the sub-sampled data set of large pillars produced the 'U' shape in Figure 12b using only two pillars. Using less pillars minimizes the amount of diffusion that the dyed portion of fluid experiences and the cost of engineering a channel with the necessary channels.

One main drawback of using this parameter sweep method to find optimal pillar configurations is that the computation time scales exponentially with the max number of pillars in the sequence. For more complex shapes, such as the letter 'A' or 'M', this approach does not efficiently predict optimal pillar configurations since we would ideally want to use more large pillars than is reasonably computable using this method.

Overall, doing a sub-sampled sweep of possible pillar configurations is efficient for nice shapes such as the letter 'U' but not for more complex shapes. Due to the fact that the greedy algorithm scales linearly while the sub-sampled sweep scales exponentially with the number of pillars in a sequence, a potential area of future work would be to combine these two methods to find a method that has lower computational complexity than the sweeping method but is more accurate than the greedy algorithm. There is also potential in adding pillars of smaller diameter into our analysis.

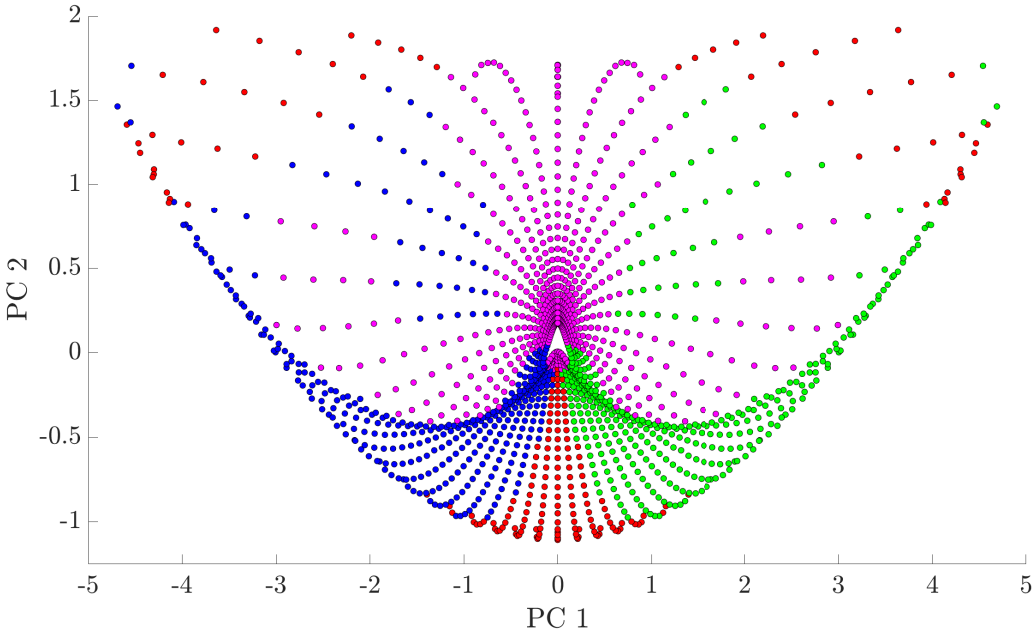
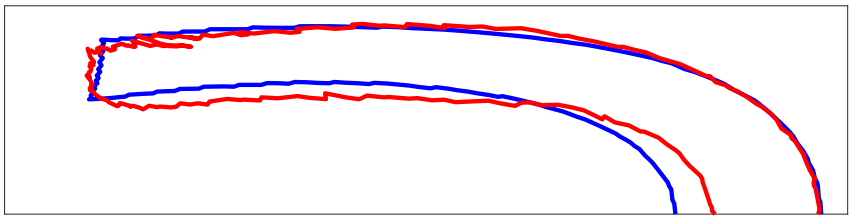
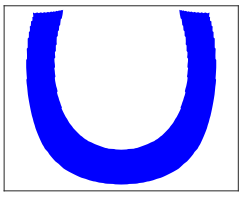


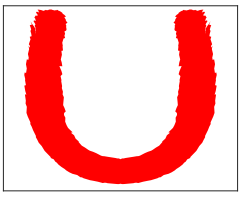
Figure 11: Plot of data for Reynolds number 30 on the space of the first two Principal Components. The four colors correspond to a different class of the data. Red points corresponds to having a counterclockwise node on the left and a clockwise node on the right. Blue points correspond to having only a clockwise node in the center of the channel. Pink points correspond to having a clockwise node on the left and a counterclockwise node on the right. Green points correspond to having only a counterclockwise node in the center of the channel.



(a) Visual comparison between our target shape (blue) and the shape produced by the Greedy Algorithm (red).



(b) Predicted 'U' shape



(c) Target 'U' shape

Figure 12: Comparisons between our target 'U' shape and the 'U' shape predicted by a sweep of all combinations of two pillars from our sub-sampled data set of 30 pillars. The red shape is the target 'U' shape while the blue shape is the best shape using only two large pillars. The Frećhet distance between the two shapes is 0.292. The image in (a) contains only the boundary of the two shapes superimposed, for visual comparison. The images in (b) and (c) contain the same shapes extended by symmetry, filled in and rotated by 90 degrees clockwise.

5. Mathematical Modeling Results

5.1. Two Parallel Infinite Plates

We consider two infinitely long parallerl plates separated by length ℓ. For our case, we nondimensionalized so that $\ell = 1$. As it's not possible to set up two infinitely long plates in COMSOL, we instead constructed an extruded rectangle of dimensions $1 \times 4 \times 10$. We chose to have our particle fixed at $y = 2, z = 5$, and varying the x position. Due to symmetry, our migration velocity in the y direction should be 0. Then we chose periodic boundary conditions on two sides of the wall to replicate an infinitely long plate. In addition, we also tested open boundary and symmetry (slip) boundary conditions as a replacement for periodic boundary conditions and saw only slight differences between the three boundary conditions. As our finite element method uses a quadratic basis, we see that it is not equipped to deal with singularities. To deal with this, we regularized both our stresslet and forcing term. In section 5.1.1 we will show that without a regularization term, our answers are inaccurate. In section 5.1.2 we find that our regularization terms significantly increase the accuracy of our results. Note that the figures below are describing the migration velocities in the x direction because the migration velocity in the y direction should be 0 due to the symmetry of our geometry. In addition, after adding a regularization term, we see that the y component of our migration velocity was always less than 3 percent of our x component, which arises from numerical error.

5.1.1. The Stresslet Method Without Regularization

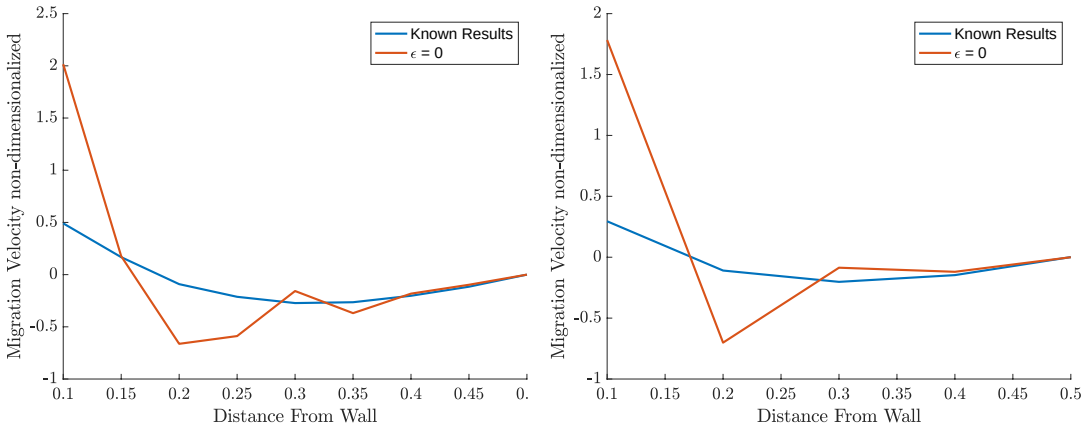
For our migration velocity to be comparable to Schonberg and Hinch's [12] results for an infinite plate, we had to non-dimensionalize our migration velocity $\mathbf{v}$ by dividing it by $a^3 R_c$ (equivalently αR_p). Note that we chose our geometry and non-dimensionalization such that $\ell = 1$, which means $\alpha = a$.

We show the results of implementing the stresslet method without a regularization term on Figures 13a and 13b. On COMSOL, we set our geometry as a $1 \times 4 \times 10$ rectangle with periodic boundary conditions to simulate two infinitely long plates. For both simulations, we set up a $1 \times 4 \times 10$ rectangle in the x, y and z direction respectively. Note that we only consider half the domain due to the symmetry of our geometry.

For figure 13a we sampled 10 points on the channel with y and z being fixed at $y = 2$ and $z = 5$. We varied x from .1 to .5 in increments of .05. While in figure 13b, we sampled it exactly the same as figure 13a except we only varied x from .1 to .5 in increments of .1.

Note that in figure 13a, the channel Reynolds number is 1, while in figure 13b, the channel Reynolds number is 75. Notice that the closer we were to the wall, the more inaccurate our results are when compared to Schonberg and Hinch [12]. However, as we got further away from the wall, our solutions become a better approximations of the migration velocities. We will show in section 5.1.2 that this discrepancy is due to COMSOL's inability to deal with points near singularities. We noticed that for the non-regularized stresslet method, our migration velocities in the y direction were large ($\sim$ 10 percent of the x migration velocity). These problems will be remedied in the following section by the introduction of regularization into our governing equations.

In addition, we chose $a = .05$ for the entire simulation. It does not matter what value of a is inputted into the equation as it does not effect the results because our governing equation is linear and $\ell = 1$, which allows us to show that our migration velocity is independent of a. But note to obtain our governing equation, we had to assume that a was asymptotically small in order to neglect terms.



(a) $R_c = 1$ with no regularization

(b) $R_c = 75$ with no regularization

Figure 13: Migration Velocity in an infinite plate with no regularization

5.1.2. The Stresslet method With Regularization

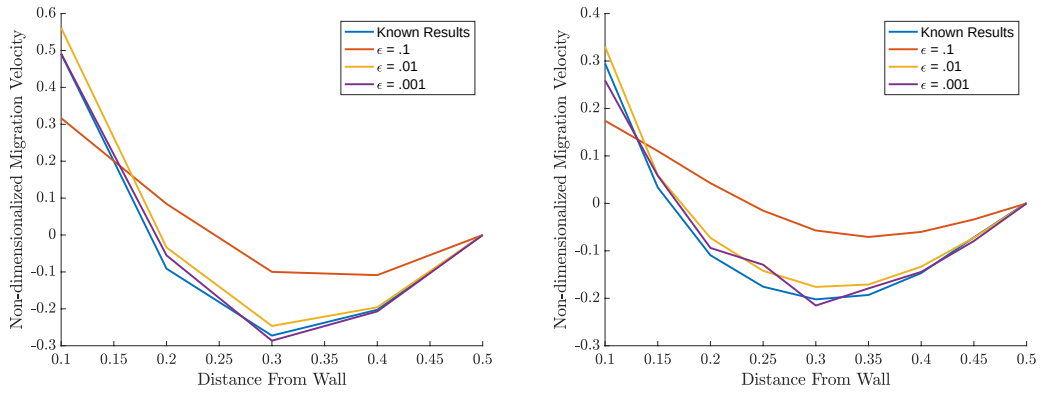
Now we introduce regularization terms into our stresslet and $\mathbf{f}_{str}$. In particular, we replaced our stresslet with the regularized stresslet and our $\mathbf{f}_{str}$ with a derivative of a Gaussian. These regularization terms tremendously improved the accuracy of our migration velocity compared to Schonberg and Hinch's results [12]. We tested $\varepsilon, \sigma = .1, .01, .001, .0001$, but found that our regularization term had no effect for when our small parameter was smaller than .001.

As seen in Figure 14a and figure 14b, we see that the implementation of a regularization term significantly increases the accuracy of our migration velocity compared to Schonberg and Hinch's values [12]. We can even see that as $\epsilon \to 0$, our solution converges to the known results.

For Figure 14b we used the regularized stresslet method, and we sampled points with y, z being fixed at $2, 5$ respectively and allowing x to vary from .1 to .5 in increments of .05. While in figure 14a, we followed figure 14b except we varied x from .1 to .5 by increments of .1

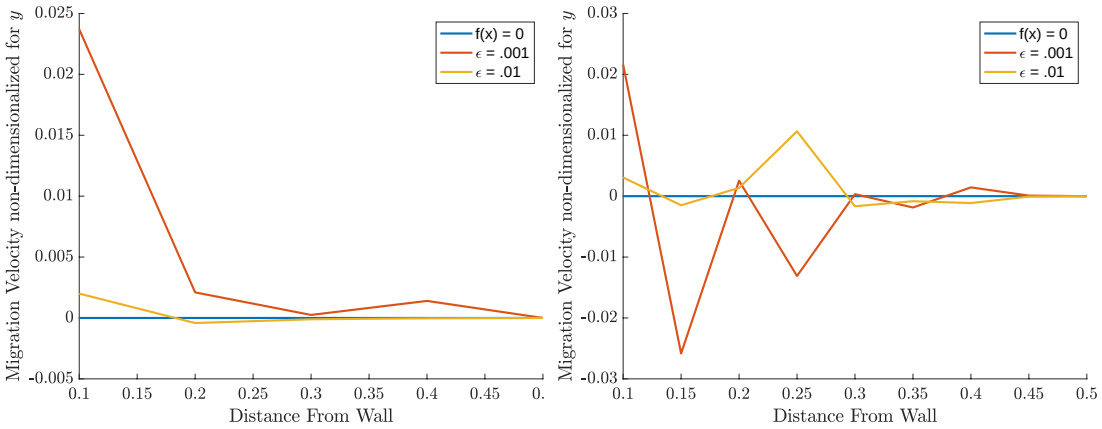
Note that as we kept y fixed at 5, by the symmetry of the geometry, we should have the migration velocity in the y direction be 0. Instead in our simulation, our y component of the migration velocity was almost always less than 3 percent of our x velocity, which we chalked up to numerical error. Consider figure 15a and 15b to see our migration velocities in the y direction with singularity regularization. We see that in these figures the y migration values are negligible.

Our numerical solver took roughly 7 minutes per point to converge on an infinite plate.



(a) $R_c = 1$, $\varepsilon = .1, .01, .001$ (b) $R_c = 75$, $\varepsilon = 0$

Figure 14: Migration Velocity in an infinite plate with regularization



(a) $R_c = 1$, y migration velocity (b) $R_c = 75$, y migration velocity

Figure 15: Migration Velocity in the y direction only

5.1.3. The Dirac Delta method

By implementing the Gaussian approximation and comparing it with existing results in [12], we are able to see that they are consistent. In figure (16), we justify that the volume average velocity makes sense. According to the figure, as $r_s \to 0$, the volume average of velocity over a sphere centred at the $x = x_s$ approaches the direct evaluation of $\mathbf{u}$ at the location of the particle. Therefore, we will simply use direct evaluation of $\mathbf{u}$ to calculate the migration velocity.

We also conduct numerical experiments to see how σ helps capture the migration velocity effectively. From figure (17a), we can see that when σ gets smaller, the volume force approximates the exact singular force better, hence numerical result agrees with [12] result better. For the sake of computational cost, we use $\sigma = 0.01$ in the following computation.

We also compare the numerical result for $Re = 75$ case, with the existing result as well as the stresslet approach, and see figure (17b) they all match well.

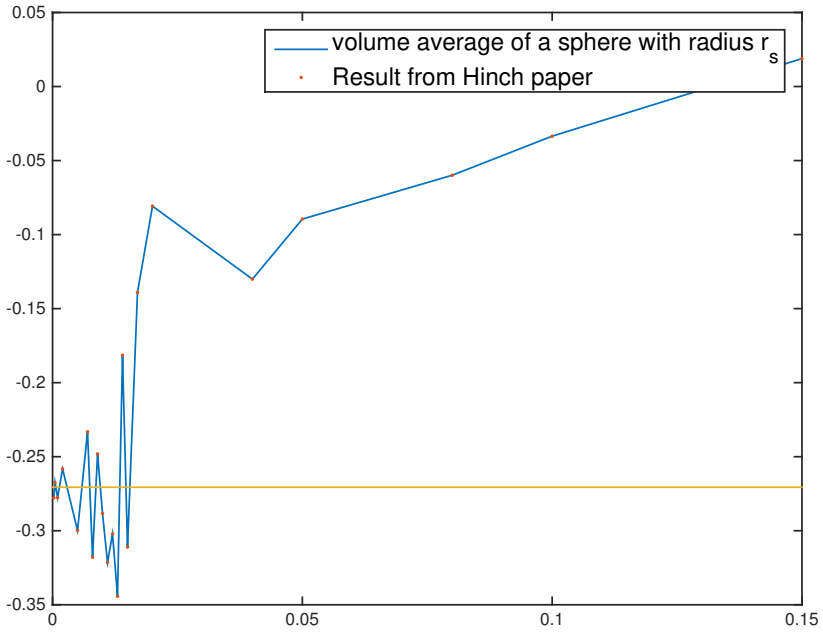
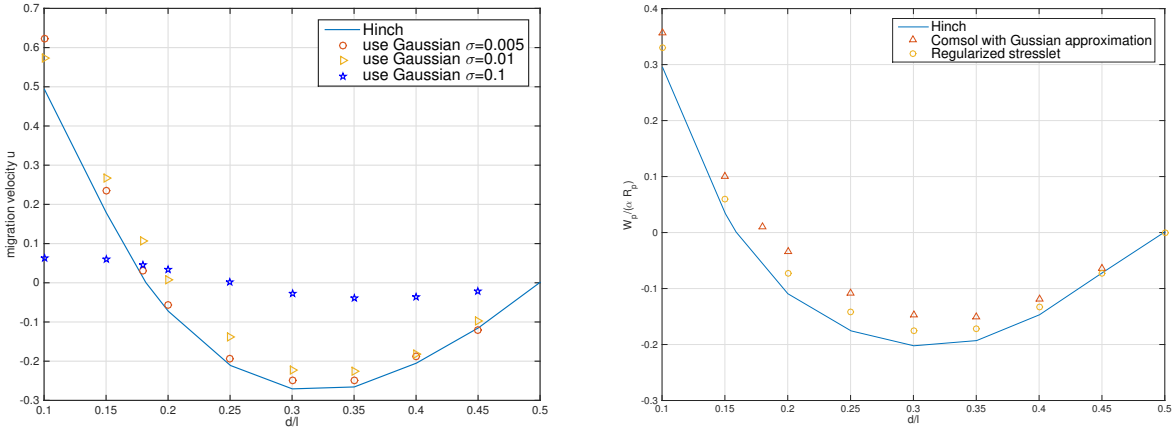


Figure 16: Volume average of sphere with different radius



(a) Migration velocity for different values of σ, Re=1 (b) Migration velocity Comparison, Re=75

Figure 17: Migration Velocity Plot

5.2. Square Channel

5.2.1. The Stresslet Method

For a square channel, we did not have results we could compare our migration velocities to, so instead we used the migration velocity to predict inertial focusing positions with the assumption that $\alpha \ll 1$. For our simulation, we choose $R_c = 80$, and $\varepsilon = 0.01$. We chose $\varepsilon = 0.01$ because based on figures 14b and 14a, $\varepsilon = 0.01$ is a good approximation to the true migration velocity. In addition, we choose $R_c = 80$ to compare our inertial focusing positions with Hood *et al* [8] where they had $a = .11, R_c = 1$. Our square channel was a $1 \times 1 \times 10$ rectangular box in the x, y and z direction respectively.

We chose to run a grid from $x, y = .1$ to $x, y = .5$ with z being fixed at $z = 5$. In particular, due to the symmetry of our channel, we only had to consider one quadrant of the cross section. We then plotted a velocity field of the data on figure 19. Then we used the symmetry of the square channel to obtain the magnitude of the migration velocity for $x, y \in [.1, .9] \times [.1, .9]$. We used this information to create figure 18, which is a $2D$ grid that has the migration magnitude as the color.

Notice that figure 19 and 18 predict an unstable equilibrium position at roughly $(.15, .15)$, $(.5, .5)$, $(.15, .85)$, $(.85, .15)$ and $(.85, .85)$. In addition, there appears to be a stable equilibrium position at roughly $(.15, .5)$,$(.5, .15)$,$(.85, .5)$ and $(.5, .85)$. These inertial focusing position agree with the results found by Hood *et al* [8] for $R_c = 80$ and $a = .11$. In addition, on figure 20 we plotted the contour plot of the migration velocity magnitude.

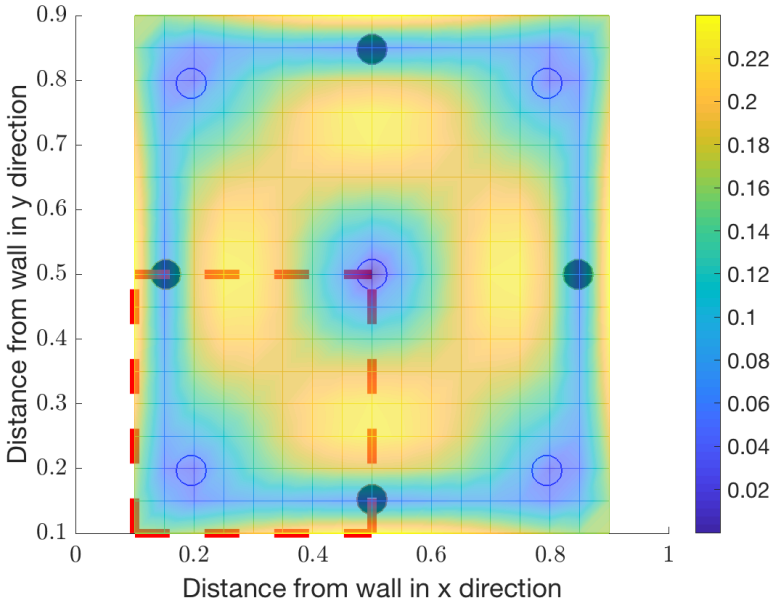


Figure 18: Migration magnitude of square channel with $R_c = 80$ Filled in circles represent stable equilibrium, while circles represent unstable equilibrium. The red dashes represent the location of the quiver plot in the figure below.

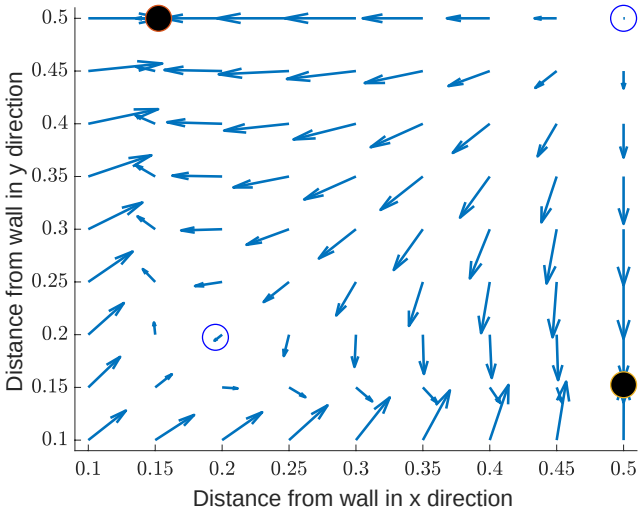


Figure 19: velocity field for a $1 \times 1 \times 10$ Square Channel for $x, y \in [.1, .5] \times [.1, .5]$ with $R_c = 80$ Filled in circles represent stable equilibrium position, while open circles represent unstable equilibrium

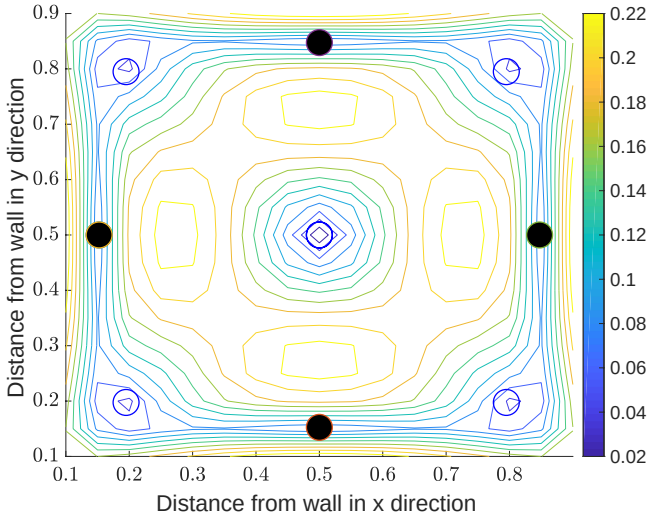


Figure 20: Contour of Migration Magnitude with $R_c = 80$. Filled in circles represent stable equilibrium points, while open circles represent unstable equilibrium points.

5.2.2. The Dirac Delta method

We implement the same experiment for an infinitely long square channel, and calculate the migration velocity in arbitrary positions of the cross section. The results are consistent with those using the stresslet method. See figure (21), the x, y-axis represents the location of a $[-0.5, 0.5] \times [-0.5, 0.5]$ channel. From figure (21), we are also able to predict the equilibrium position in the cross section.

5.3. Right Triangle with length 1 and width 0.5 Channel

To show case the generality of our method, we use our governing equations with COMSOL to predict the inertial focusing position of a non-symmetric geometry, which has not been studied before. In particular, we considered a right triangular channel with dimensions $1 \times 1/2 \times 10$ in

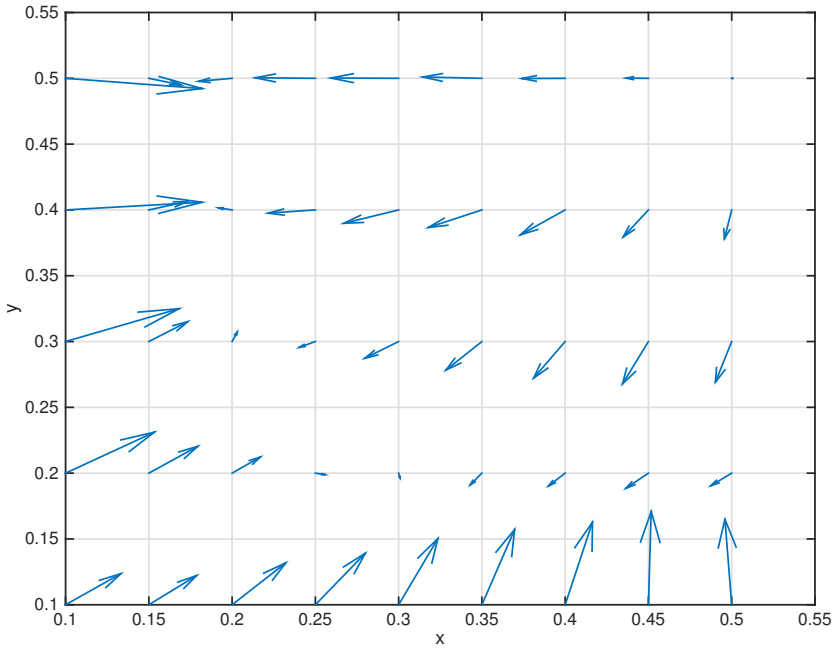


Figure 21: Migration velocity Comparison, Re=75

the x, y and z component respectively. As usual, we fix the z position of our particle at the center-line at $z = 5$. We set our Reynolds channel number to 1 with $\varepsilon = 0.01$.

We swept through points the x, y coordinates for $x \in (0.0375, .4)$ and $y \in (0.05, .8)$ in increments of .05 until we were .05 away from the boundary for y and similarly for x except we also ran x at $.0375, .075, .125, .175$,and $.225$. Each point took COMSOL roughly 3 minutes to solve, which overall took roughly 5 hours to obtain all the data. We plotted the quiver plot of these points on Figure 22. The red lines indicate where the triangle is.

Based on the quiver plot in Fig. 22, there appears to be stable equilibrium points in a neighborhood of $(.2, .07)$ and $(.1, .3)$, with an unstable one at $(.185, .225)$. Due to our lack of resolution, we could not pinpoint the exact location of the equilibrium points. In addition, as this geometry has not been studied before, we have no solutions to compare our results to. This shows the flexibility of our governing equations to deal with arbitrary geometries.

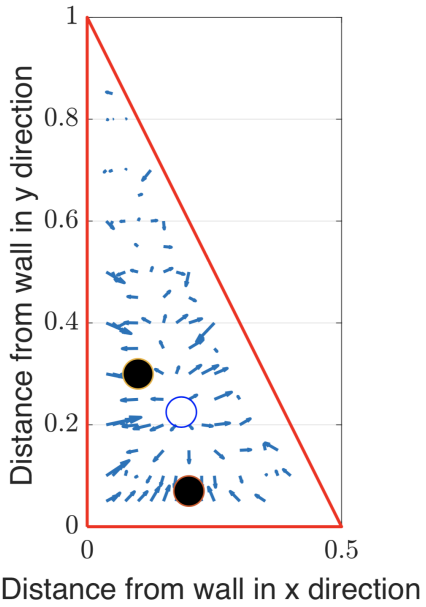


Figure 22: Quiver Plot of $1 \times \frac{1}{2} \times 10$ triangular channel with $R_c = 1$. Filled in circles represent stable equilibrium points, while circles represented unstable equilibrium points.

6. Conclusion

We derived a governing equation for the migration velocity of a particle in an arbitrary geometry with the assumption that $R_p \ll 1$. The time dependance of our boundary conditions owing to a moving particle makes this computation very expensive. We remedied this by two methods: one is by introducing a stresslet, and another is by introducing a singular point force into our equation. However, these methods appeared inaccurate when implemented into COMSOL. We fixed this by regularizing our stresslet and singular force. After regularization, we show that our regularized governing equations approximate well-known results and that they converge as our small positive regularization parameters go to 0. Then we extend this method to a channel that has not been studied before. The main advantages of our method is that our governing equation with the regularized methods are linear PDEs with a fixed geometry that are non-singular. This allows numerical solvers to quickly and accurately solve the resulting PDE. Indeed, our simulations on the infinite plate and the square channel took 7 minute per point, while the triangle took 3 minutes per point.

In the machine learning section, we used the discrete Fréchet distance in a greedy algorithm to solve the inverse problem of finding a pillar configuration to reproduce some target flow shape. We saw that this does not always find a global optimum. To address this issue, we randomly swapped pillars in the predicted pillar configuration in the hopes of decreasing the distance from our predicted shape to our target shape. We found that randomly swapping any of the previous pillars does better than only replacing the pillars that cause the largest change in distance. To further improve our solution to the inverse problem, we implemented Principal Component Analysis and K-Means clustering to intelligently subsample our set of pillars to a set of approximately 60 characteristic pillars. We then did a complete sweep of all possible configurations of m pillars, and found which of these pillar configurations minimized the distance from our predicted shape to our target shape. We found that using this method significantly improves the accuracy of our predicted shapes, while taking a relatively short amount of computation time. However, this method is only viable for small values of m. Hence

for future research, it will be valuable to test a combination of the greedy algorithm and a complete sweep of the subsampled pillar set to solve for target flow shapes that require a large number of pillars.

Appendix A. COMSOL implementation

Our code for comsol was created on COMSOL 5.3a under the Computational Fluid Dynamics (CFD) Module. In COMSOL we created a $1 \times 4 \times 10$ domain corresponding to x, y, and z dimensions respectively. We then placed a point at the location $(x_s, y_s, 5)$ to represent our particle (which was used to probe values and refine the mesh).

We chose a coarse meshing for the rectangle, while we refined the mesh around the point as

Maximum Element Size	0.00187
Minimum Element Size	2.88×10^{-5}
Maximum Element Growth	1.05
Curvature Factor	.2

The analytic normalized Poiseuille Flow in an infinitely long plane separated by one unit is given by $\overline{\mathbf{u}} = \mathbf{e}_z 4x(1-x)$. For more complicated cross sections, we used a Poisson solver to determine the background flow.

We set the fluid properties as $\mu = 1$ and $\rho = R_c$. In addition, we have for our initial values $\mathbf{u} = 0$ and $\mathbf{p} = 0$. We set the two 1×4 rectangles as outlets with a neutral boundary condition $\mathbf{p} = 0$. We set the two 4×10 rectangles as inlets with $\mathbf{u} = -\mathbf{u}_{stresslet}$. We imposed periodic boundary conditions on the final two walls to model infinitely long plates. In addition, we changed our solver to P2-P1. This extends the FEM to a quadratic basis for the velocity, while keeping a linear basis for the pressure. This step severely increases the accuracy of our result. We also changed the convective term in COMSOL's weak form into the one stated in deriving the governing equation. Written compactly, we changed the following:

$$-\rho(\mathbf{u} \cdot \nabla \mathbf{u} \cdot \phi) \rightarrow -\rho(\overline{\mathbf{u}'} \cdot \nabla \mathbf{u}' + \mathbf{u}' \cdot \nabla \overline{\mathbf{u}}) \cdot \phi,$$

where ϕ is the test function with three components, and we replaced $\mathbf{u}'$ with $\mathbf{u}_{str} + \mathbf{u}$. We ran a parametric sweep with the particle being centered at $y = 2$, while x was being varied. In addition, we also took the spherical average around the point and saw that it did not significantly affect the results. For our results to be consistent with Schonberg and Hinch's result on an infinitely long plate, we had to divide our velocities by $a^3 R_c$ where a is the radius of the particle, and R_c is the channel Reynolds Number.

[1] Hamed Amini, Wonhee Lee, and Dino Di Carlo, *Inertial microfluidic physics*, Lab on a Chip **14** (2014), no. 15, 2739–2761.

[2] Hamed Amini, Elodie Sollier, Mahdokht Masaeli, Yu Xie, Baskar Ganapathysubramanian, Howard A Stone, and Dino Di Carlo, *Engineering fluid flow using sequenced microstructures*, Nature communications **4** (2013), 1826.

[3] George Keith Batchelor, *An introduction to fluid dynamics*, Cambridge university press, 2000.

[4] Allen T Chwang and T Yao-Tsu Wu, *Hydromechanics of low-reynolds-number flow. part 2. singularity method for stokes flows*, Journal of Fluid Mechanics **67** (1975), no. 4, 787–815.

[5] Ricardo Cortez, *The method of regularized stokeslets*, SIAM Journal on Scientific Computing **23** (2001), no. 4, 1204–1225.

[6] Ricardo Cortez and Douglas Varela, *A general system of images for regularized stokeslets and other elements near a plane wall*, Journal of Computational Physics **285** (2015), 41–54.

[7] Thomas Eiter and Heikki Mannila, *Computing discrete fréchet distance*, Tech. report, Citeseer, 1994.

[8] Kaitlyn Hood, Sungyon Lee, and Marcus Roper, *Inertial migration of a rigid sphere in three-dimensional Poiseuille flow*, Journal of Fluid Mechanics **765** (2015), 452–479.

[9] N Liron and R Shahar, *Stokes flow due to a stokeslet in a pipe*, Journal of Fluid Mechanics **86** (1978), no. 4, 727–744.

[10] Kin Gwn Lore, Daniel Stoecklein, Michael Davies, Baskar Ganapathysubramanian, and Soumik Sarkar, *Hierarchical feature extraction for efficient design of microfluidic flow patterns*, Feature Extraction: Modern Questions and Challenges, 2015, pp. 213–225.

[11] Kevin S Paulsen, Dino Di Carlo, and Aram J Chung, *Optofluidic fabrication for 3d-shaped particles*, Nature communications **6** (2015), 6976.

[12] Jeffrey A Schonberg and EJ Hinch, *Inertial migration of a sphere in Poiseuille flow*, Journal of Fluid Mechanics **203** (1989), 517–524.

[13] Elodie Sollier, Derek E Go, James Che, Daniel R Gossett, Sean O'Byrne, Westbrook M Weaver, Nicolas Kummer, Matthew Rettig, Jonathan Goldman, Nicholas Nickols, et al., *Size-selective collection of circulating tumor cells using vortex technology*, Lab on a Chip **14** (2014), no. 1, 63–77.

[14] Daniel Stoecklein, Michael Davies, Nadab Wubshet, Jonathan Le, and Baskar Ganapathysubramanian, *Automated design for microfluid flow sculpting: multiresolution approaches, efficient encoding, and CUDA implementation*, Journal of Fluids Engineering **139** (2017), no. 3, 031402.

[15] Daniel Stoecklein, Kin Gwn Lore, Michael Davies, Soumik Sarkar, and Baskar Ganapathysubramanian, *Deep learning for flow sculpting: Insights into efficient learning using scientific simulation data*, Scientific reports **7** (2017), 46368.

[16] Daniel Stoecklein, Chueh-Yu Wu, Donghyuk Kim, Dino Di Carlo, and Baskar Ganapathysubramanian, *Optimization of micropillar sequences for fluid flow sculpting*, Physics of Fluids **28** (2016), no. 1, 012003.