Simple Filaments Swimming Model in Incompressible Viscous Fluids

Sang-Eun Lee and Louis J. Nass

May 8, 2021

Abstract

In this paper, we implement the immersed boundary method in two-dimensional spaces, simply using a 2D elastic filament in imcompressible fluids.

1 Introduction

In the last four to five decades, scientists have been interested in the locomotion of various microorganisms in fluids. Not only are we concerned with microorganisms from the kingdom of animalia, but the subject of research can be broadly include the kingdom of fungi, bacteria, protozoa, chromista, or even seeds of plants. Amongst all interesting topics of the investigation, a researcher group may cast a rosy hue on understanding the locomotion of the monoflagellum microbes such as a sperm. Specifically, sperm motility is highly related to infertility so as the investigation of the sperm motility is indispensable for helping the field of gynecology.

We propose a numerical method of simulating a simple filament immersed in a fluid structure. In fact, the previous researchers have done without inertia of the microbes due to the fact that the mass of microbes are proportionally negligible so that the Reynolds number is assumed zero. Thus, the Navier-Stokes equation can be written by the form of linear diffusion equation, which is called Stokes' equation. In the model we applied, instead of considering the Stokes' equation, we implement the full model of incompressible viscous fluids using the Navier-Stokes equations. This can also help understand the locomotion of immersed structure in a quite larger than the microscopic scale. To implement the fluid motion in the simulation, we employ modified Chorin's projection method [1], which will be discussed in Section 2.1.

For the immersed structure in the fluid, we employ the immersed boundary (IB) method modelled by Peskin [5]. The IB method enables us to observe various objects immersed in the fluid structure. The main idea of the method is considered that there is a fibre or a fibre bundle immersed in the fluid. Each fibre, described by Lagrangian coordinate, can be discretised into several nodes. The forces are then generated by each of nodes in the immersed structure and will interact with the fluid grid described by an Eulerian coordinate. We are able to decompose the force from the immersed structure into tensile and bending forces, and spreading the force into the grid is employed by the discritised Dirac-delta function. Detailed description of the immersed structure formulation will be discussed in Section 2.2 and 2.3.

Since the nature of the delta function, a singularity incurred from the delta function is always troublesome. This is emphasised when the problem requires to find the Green's function. Relatively

recent work [2], Cortez removed the singularity of Stokeslet by changing the inhomogeneous function by regularised delta distribution. We are trying to employ this idea to regularise the delta function.

In Section 3, we display some features from the experiment conducted by various combinations of cases to validate our model. Basically, the experiment is to show the immersed boundary is properly swimming in the fluid structure. Prior to implementation of the immersed boundary, it is inevitable to validate whether or not each force implemented in the computation has any technical deficiency. To validate this, we begin with assuming that there exists only either tensile or bending force on the discretised points of a single and simple filament. Later, to make the model sophisticated, we will add up tensile and bending forces.

The goal of this project is to observe the synchronization of two simple filaments asymmetrically located at the initial step [3]. Once the preferred curvature is time dependence for each filament, the swimming model can be realised with the synchronization. In [3], the background fluid is Stokes flow, whereas our background force has nonzero Reynolds number so that the synchronization result can be different from [3].

2 Mathematical Formulation

In this section, we discuss a theoretical formulation of the immersed boundary method for solving the locomotion of single or multiple immersed structures in the viscous and incompressible fluids.

2.1 Projection Method for Incompressible Navier-Stokes Equations

Given that the incompressible Navier-Stokes equations in 2D:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \mu \Delta \mathbf{u} + \mathbf{F}, \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2}$$

The material derivative is indeed

$$\frac{D\mathbf{u}}{Dt} = \mathbf{u}_t + (\mathbf{u} \cdot \nabla)\mathbf{u}.$$

In this method, (2) is featured as a crucial role of determining the pressure [1]. To begin, we may neglect the pressure term.

2.1.1 Solution of Projected Velocity Field

Since we are in two dimensions, we write $\mathbf{u} = (u, v)$ and $\mathbf{F} = (f, g)$. Using the component-wise expression of the velocity field $\mathbf{u}$, the discretized version of (1) can be expressed, with the discretization

of advection term by $(\mathbf{u}^n \cdot \nabla)\mathbf{u}^*$,

$$\rho\left(\frac{u^*_{i,j}-u^n_{i,j}}{\Delta t}+u^n_{i,j}\frac{u^*_{i+1,j}-u^*_{i-1,j}}{2\Delta x}+v^n_{i,j}\frac{u^*_{i,j+1}-u^*_{i,j-1}}{2\Delta y}\right)=\mu\left(\frac{u^*_{i+1,j}-2u^*_{i,j}+u^*_{i-1,j}}{(\Delta x)^2}\right)+\mu\left(\frac{u^*_{i,j+1}-2u^*_{i,j}+u^*_{i,j-1}}{(\Delta y)^2}\right)+f^n_{i,j},$$

(x component)

$$\rho\left(\frac{v^*_{i,j}-v^n_{i,j}}{\Delta t}+u^n_{i,j}\frac{v^*_{i+1,j}-v^*_{i-1,j}}{2\Delta x}+v^n_{i,j}\frac{v^*_{i+1,j}-v^*_{i-1,j}}{2\Delta y}\right)=\mu\left(\frac{v^*_{i+1,j}-2v^*_{i,j}+v^*_{i-1,j}}{(\Delta x)^2}\right)+\mu\left(\frac{v^*_{i,j+1}-2v^*_{i,j}+v^*_{i,j-1}}{(\Delta y)^2}\right)+g^n_{i,j}.$$

(y component)

Here, $\mathbf{u}^*$ is an auxiliary velocity field. To solve these schemes, equations above need to use the Peaceman-Rachford method [4], it is often called generally alternating direction implicit method (ADI). For the x component, assume u^{**} by another auxiliary velocity vector. Then x component equation can be decomposed by two equations,

$$\rho\left(\frac{u^{**}_{i,j}-u^n_{i,j}}{\Delta t}+u^n_{i,j}\frac{u^{**}_{i+1,j}-u^{**}_{i-1,j}}{2\Delta x}+v^n_{i,j}\frac{u^n_{i,j+1}-u^n_{i,j-1}}{2\Delta y}\right)=\mu\left(\frac{u^{**}_{i+1,j}-2u^{**}_{i,j}+u^{**}_{i-1,j}}{(\Delta x)^2}\right)+\mu\left(\frac{u^n_{i,j+1}-2u^n_{i,j}+u^n_{i,j-1}}{(\Delta y)^2}\right)+f^n_{i,j}, \tag{3}$$

$$\rho\left(\frac{u^*_{i,j}-u^{**}_{i,j}}{\Delta t}+u^n_{i,j}\frac{u^{**}_{i+1,j}-u^{**}_{i-1,j}}{2\Delta x}+v^n_{i,j}\frac{u^*_{i,j+1}-u^*_{i,j-1}}{2\Delta y}\right)=\mu\left(\frac{u^{**}_{i+1,j}-2u^{**}_{i,j}+u^{**}_{i-1,j}}{(\Delta x)^2}\right)+\mu\left(\frac{u^*_{i,j+1}-2u^*_{i,j}+u^*_{i,j-1}}{(\Delta y)^2}\right). \tag{4}$$

Collecting each terms,

$$u^{**}_{i,j}+\frac{\Delta t u^n_{i,j}}{2\rho\Delta x}\left(u^{**}_{i+1,j}-u^{**}_{i-1,j}\right)-\frac{\mu\Delta t}{\rho(\Delta x)^2}\left(u^{**}_{i+1,j}-2u^{**}_{i,j}+u^{**}_{i-1,j}\right)=u^n_{i,j}+\frac{\Delta t}{\rho}f^n_{i,j},$$

$$u^*_{i,j}+\frac{\Delta t v^n_{i,j}}{2\rho\Delta y}\left(u^*_{i,j+1}-u^*_{i,j-1}\right)-\frac{\mu\Delta t}{\rho(\Delta y)^2}\left(u^*_{i,j+1}-2u^*_{i,j}+u^*_{i,j-1}\right)=u^{**}_{i,j}.$$

If we assume $\Delta x = \Delta y = h$ and $\Delta t = k$, then we obtain the following matrix equations:

$$\begin{bmatrix} d & m_1 & 0 & \cdots & 0 & l_1 \\ l_2 & d & m_2 & \ddots & & 0 \\ 0 & l_3 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & 0 \\ 0 & & & & & m_{N-1} \\ m_N & 0 & \cdots & 0 & l_N & d \end{bmatrix} \begin{bmatrix} u^{**}_{1,j} \\ \\ \vdots \\ \\ \\ u^{**}_{N,j} \end{bmatrix} = \begin{bmatrix} u^n_{1,j} + \frac{\Delta t}{\rho} f^n_{1,j} \\ \\ \vdots \\ \\ \\ u^n_{N,j} + \frac{\Delta t}{\rho} f^n_{N,j} \end{bmatrix},$$

$$\begin{bmatrix} d & q_1 & 0 & \cdots & 0 & r_1 \\ r_2 & d & q_2 & \ddots & & 0 \\ 0 & r_3 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & 0 \\ 0 & & & & & q_{N-1} \\ q_N & 0 & \cdots & 0 & r_N & d \end{bmatrix} \begin{bmatrix} u^{*}_{i,1} \\ \\ \vdots \\ \\ \\ u^{*}_{i,N} \end{bmatrix} = \begin{bmatrix} u^{**}_{i,1} \\ \\ \vdots \\ \\ \\ u^{**}_{i,N} \end{bmatrix},$$

where

$$\begin{aligned} d &= 1 + \frac{2\mu k}{\rho h^2}, \\ m_s &= \frac{k u^n_{s,j}}{2\rho h} - \frac{\mu k}{\rho h^2}, \quad l_s = -\frac{k u^n_{s,j}}{2\rho h} - \frac{\mu k}{\rho h^2}, \\ q_s &= \frac{k v^n_{i,s}}{2\rho h} - \frac{\mu k}{\rho h^2}, \quad r_s = -\frac{k v^n_{i,s}}{2\rho h} - \frac{\mu k}{\rho h^2}, \end{aligned}$$

for $s = 1, 2, \cdots, N$ throughout all off-diagonal elements. Similarly, for y components of (1) without pressure term with another auxiliary term v^{**}, we have two different equations alternating directions of summand:

$$\begin{aligned} \rho\left(\frac{v^{**}_{i,j} - v^n_{i,j}}{\Delta t} + u^n_{i,j}\frac{v^{**}_{i+1,j} - v^{**}_{i-1,j}}{2\Delta x} + v^n_{i,j}\frac{v^n_{i,j+1} - v^n_{i,j-1}}{2\Delta y}\right) =& \mu\left(\frac{v^{**}_{i+1,j} - 2v^{**}_{i,j} + v^{**}_{i-1,j}}{(\Delta x)^2}\right) \\ &+ \mu\left(\frac{v^n_{i,j+1} - 2v^n_{i,j} + v^n_{i,j-1}}{(\Delta y)^2}\right) + g^n_{i,j}, \end{aligned} \tag{5}$$

$$\begin{aligned} \rho\left(\frac{v^{*}_{i,j} - v^{**}_{i,j}}{\Delta t} + u^n_{i,j}\frac{v^{**}_{i+1,j} - v^{**}_{i-1,j}}{2\Delta x} + v^n_{i,j}\frac{v^{*}_{i,j+1} - v^{*}_{i,j-1}}{2\Delta y}\right) =& \mu\left(\frac{v^{**}_{i+1,j} - 2v^{**}_{i,j} + v^{**}_{i-1,j}}{(\Delta x)^2}\right) \\ &+ \mu\left(\frac{v^{*}_{i,j+1} - 2v^{*}_{i,j} + v^{*}_{i,j-1}}{(\Delta y)^2}\right). \end{aligned} \tag{6}$$

Collecting each terms,

$$\begin{aligned} v^{**}_{i,j} + \frac{\Delta t u^n_{i,j}}{2\rho\Delta x}\left(v^{**}_{i+1,j} - v^{**}_{i-1,j}\right) - \frac{\mu\Delta t}{\rho(\Delta x)^2}\left(v^{**}_{i+1,j} - 2v^{**}_{i,j} + v^{**}_{i-1,j}\right) &= v^n_{i,j} + \frac{\Delta t}{\rho} g^n_{i,j}, \\ v^{*}_{i,j} + \frac{\Delta t v^n_{i,j}}{2\rho\Delta y}\left(v^{*}_{i,j+1} - v^{*}_{i,j-1}\right) - \frac{\mu\Delta t}{\rho(\Delta y)^2}\left(v^{*}_{i,j+1} - 2v^{*}_{i,j} + v^{*}_{i,j-1}\right) &= v^{**}_{i,j}. \end{aligned}$$

We arrive the following matrix equations again:

$$\begin{bmatrix} d & m_1 & 0 & \cdots & 0 & l_1 \\ l_2 & d & m_2 & \ddots & & 0 \\ 0 & l_3 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & 0 \\ 0 & & & & & m_{N-1} \\ m_N & 0 & \cdots & 0 & l_N & d \end{bmatrix} \begin{bmatrix} v^{**}_{1,j} \\ \\ \vdots \\ \\ v^{**}_{N,j} \end{bmatrix} = \begin{bmatrix} v^{n}_{1,j} + \frac{\Delta t}{\rho} g^{n}_{1,j} \\ \\ \vdots \\ \\ v^{n}_{N,j} + \frac{\Delta t}{\rho} g^{n}_{N,j} \end{bmatrix},$$

$$\begin{bmatrix} d & q_1 & 0 & \cdots & 0 & r_1 \\ r_2 & d & q_2 & \ddots & & 0 \\ 0 & r_3 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & & & 0 \\ 0 & & & & & q_{N-1} \\ q_N & 0 & \cdots & 0 & r_N & d \end{bmatrix} \begin{bmatrix} v^{*}_{i,1} \\ \\ \vdots \\ \\ v^{*}_{i,N} \end{bmatrix} = \begin{bmatrix} v^{**}_{i,1} \\ \\ \vdots \\ \\ v^{**}_{i,N} \end{bmatrix}.$$

Notice that the quasi-tridiagonal system for both u and v are same, so we can save up the computational time using Thomas algorithm.

2.1.2 Solution for the Pressure Term

Now, we are ready to determine the pressure term. We propose two methods to solve the pressure, one is direct finite differences and the other is fast Fourier transformation. We explore Chorin's original projection method [1], but, the pressure of the fluid was uncovered by the iterative method after initializing pressures, exploiting the pressure has developed to apply either the method of fast Fourier transform or finite difference method in the latest decade.

Solution by the Finite Difference Method From [1], we have

$$\begin{aligned} \rho \frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} &= -\nabla p, \\ \mathbf{u}^{n+1} &= \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p. \end{aligned}$$

Take the divergence on both sides, we have

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}^* - \nabla \cdot \left(\frac{\Delta t}{\rho} \nabla p \right),$$

and because of (2), the LHS $\nabla \cdot \mathbf{u}^{n+1} = 0$. Note that we have $\nabla \cdot \mathbf{u}^* \neq 0$, since $\mathbf{u}^*$ is projected variable so that the divergence is nonzero value. Thus,

$$\nabla \cdot \nabla p = \frac{\rho}{\Delta t} (\nabla \cdot \mathbf{u}^*) = \frac{\rho}{\Delta t} (u^*_x + v^*_y). \tag{7}$$

Observe that we have a linear diffusion equation for p. After discretizing the equation, we have

$$\nabla \cdot \begin{pmatrix} \frac{p_{i+1,j}-p_{i-1,j}}{2\Delta x} \\ \frac{p_{i,j+1}-p_{i,j-1}}{2\Delta y} \end{pmatrix} = \nabla \cdot \left(\frac{\rho}{\Delta t} \mathbf{u}^*_{i,j} \right),$$

$$\frac{p_{i+2,j} - 2p_{i,j} + p_{i-2,j}}{(2\Delta x)^2} + \frac{p_{i,j+2} - 2p_{i,j} + p_{i,j-2}}{(2\Delta y)^2} = \frac{\rho(u^*_{i+1,j} - u^*_{i-1,j})}{2\Delta t \Delta x} + \frac{\rho(v_{i,j+1} - v_{i,j-1})}{2\Delta t \Delta y}.$$

Notice that the RHS is composed of all known values, whereas p's in the LHS is unknown. If we assume that we have periodic boundary condition for p as well, then we need to divide the recursive relation into four different cases: 1) i is even, j is even; 2) i is odd, j is even; 3) i is even, j is odd; and 4) i is odd, j is odd. Applying the compatibility condition for this system, we can obtain the pressure term.

Method of Fast Fourier Transformation Recognize that (7) is 2D Poisson equation, apply the Fourier transformation such that

$$\widehat{f}_{k,l} = \sum_{m=1}^{N} \sum_{j=1}^{N} f_{m,j} e^{2\pi i m k/N + 2\pi i j l/N}, \text{ for } k, l = 1, \ldots, N,$$

with the inversion

$$f_{m,j} = \frac{1}{N^2} \sum_{k=1}^{N} \sum_{l=1}^{N} \widehat{f}_{k,l} e^{-2\pi i k m/N - 2\pi i l j/N}, \text{ for } m, j = 1, \ldots, N.$$

Thus, the transformed recursion for (7) is

$$\widehat{p}_{k,l} = -\frac{1}{4\left(\sin^2 \frac{\pi k}{N} + \sin^2 \frac{\pi l}{N}\right)} \widehat{f}_{k,l}, \text{ for } k, l = 1, \ldots, N-1,$$

and $\widehat{p}_{N,N} = 0$, due to the compatibility condition. Since we obtain $\widehat{p}_{k,l}$, we can obtain with the inversion formula to find the pressure $p_{m,j}$.

2.2 Force from the Immersed Boundary

In the previous section, we have not specified what the external force $\mathbf{F}$ is. The external force can be any forces exerted on the fluid structure, such as gravity. In this section, we assume that there is an elastic filament immersed in the viscous and incompressible fluid. Ignoring the gravity, we assume that the external force $\mathbf{F}$ only comes from the immersed structure.

From now on, we denote that $\mathbf{X}(s,t)$ as the position vector of the filament written in the Lagrangian coordinate, and the force $\mathbf{f}(s,t)$ is the point force generated by the immersed boundary, where s is the Lagrangian or arc length parameter. Then, the force of the fluid $\mathbf{F}$ can be written as

$$\mathbf{F}(\mathbf{x},t) = \int_{\Gamma} \mathbf{f}(s,t) \delta(\mathbf{x} - \mathbf{X}(s,t)) \, ds, \tag{8}$$

where Γ is the curve along the filament. Notice that (8) converts the coordinates from the Lagrangian to the Eulerian. Once we have the force at each grid point on the fluid structure, we have the explicit form of the force $\mathbf{F}$ of (1).

2.3 Tensile and Bending Forces of the Immersed Structure

From (8), we do not know yet what the force $\mathbf{f}$ is. In this section, we investigate the forces from the immersed boundary using the energy formulation. Energy method enables us to derive the force density of the immersed structure since the kinetic energy of the immersed boundary along the fluid represents of the locomotion of the filament immersed in the fluid structure.

2.3.1 Tensile Force

Assume that there are two points from the immersed boundary, say $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$. Further assume that there is a spring between $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$, and the force at each $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ is

$$\begin{aligned}\mathbf{f}_k &= -S_T\left(\frac{\|\mathbf{X}_{k+1}-\mathbf{X}_k\|}{\Delta s}-1\right)\frac{\mathbf{X}_{k+1}-\mathbf{X}_k}{\|\mathbf{X}_{k+1}-\mathbf{X}_k\|},\\ \mathbf{f}_{k+1} &= -\mathbf{f}_k,\end{aligned}$$

where Δs is the rest length and S_T is the stiffness constant. The problem comes up with the complexity, since the more the immersed boundary points increase, the more the complexity of the force at each immersed boundary point increases. We employ the energy formulation for the sake of simplicity of the calculation. The energy of the spring between $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$ is

$$E(\mathbf{X}_k,\mathbf{X}_{k+1}) = \frac{S_T}{2}\left(\frac{\|\mathbf{X}_k-\mathbf{X}_{k+1}\|}{\Delta s}-1\right)^2\Delta s,$$

and note that the energy is non-negative constant. Since the tensile force model of the spring is connected, we need the total energy of the spring from $\mathbf{X}_1$ to $\mathbf{X}_N$. The total energy of the spring can be written in the continuous case by

$$E(\mathbf{X}_1,\ldots,\mathbf{X}_N) = \frac{S_T}{2}\int_\Gamma\left(\left\|\frac{\partial\mathbf{X}}{\partial s}\right\|-1\right)^2 ds.$$

2.3.2 Bending Force

Now that we have established tensile forces between two discrete points, we will also consider bending forces between two segmented connections in the filament. Assume that there are three sequential points from the immersed boundary, say $\mathbf{X}_{k-1}, \mathbf{X}_k$, and $\mathbf{X}_{k+1}$. Further assume that there are springs between points $\mathbf{X}_{k-1}$ and $\mathbf{X}_k$ and between points $\mathbf{X}_k$ and $\mathbf{X}_{k+1}$. Then, the bending force among the three points is

$$\begin{aligned}\mathbf{f}_k &= -\frac{2S_B}{\Delta s}\left(\frac{||\mathbf{X}_{k+1}-2\mathbf{X}_k+\mathbf{X}_{k-1}||}{\Delta s^2}-c(t,k)\right)\frac{\mathbf{X}_{k+1}-2\mathbf{X}_k+\mathbf{X}_{k-1}}{||\mathbf{X}_{k+1}-2\mathbf{X}_k+\mathbf{X}_{k-1}||}\\ \mathbf{f}_{k+1} &= -\frac{1}{2}\mathbf{f}_k\\ \mathbf{f}_{k-1} &= \mathbf{f}_{k+1},\end{aligned}$$

where Δs is the rest length, S_B is the bending constant, and $c(t,k)$ is the preferred bending curvature. We derive these formulas from the discrete definition of bending energy. Discrete bending energy between three points, $\mathbf{X}_{k-1}, \mathbf{X}_k$, and $\mathbf{X}_{k+1}$, is defined as

$$E(\mathbf{X}_{k-1},\mathbf{X}_k,\mathbf{X}_{k+1}) = \frac{S_B}{2}\left(\frac{||\mathbf{X}_{k+1}-2\mathbf{X}_k+\mathbf{X}_{k-1}||}{\Delta s^2}-c(t,k)\right)^2\Delta s,$$

where, again, we observe that the energy is a non-negative constant. Similar to our tensile forces, the fiber connects points $\mathbf{X}_1$ to $\mathbf{X}_N$. Thus, the total bending energy can be represented in the continuous case by

$$E(\mathbf{X}_1, \ldots, \mathbf{X}_N) = \frac{S_B}{2} \int_\Gamma \left(\left\| \frac{\partial^2 \mathbf{X}}{\partial s^2} \right\| - c(t, \mathbf{X}) \right)^2 \, ds.$$

3 Results in Two-dimensional Cases

In this section, we will run force tests on various immersed boundary structures. In each test, we will set the fluid viscosity $\mu = 1$, the fluid density $\rho = 1$, and the spacial grid for the Navier-Stokes solver $h = 1/129$.

3.1 Tensile force test

First, we will observe two different tensile tests. Our initial test will examine our tensile forces' ability to expand the distance between two points, then we will examine the forces' ability to contract the distance between two points.

3.1.1 Expanding test

In this test, we will set two points, $\mathbf{X}_1$ and $\mathbf{X}_2$, at an initial distance that is less than the rest distance. Here we will let $||\mathbf{X}_1 - \mathbf{X}_2|| = 0.2$, $\Delta s = 0.5$, and $S_T = 30$. We observe the following results in figure (1).

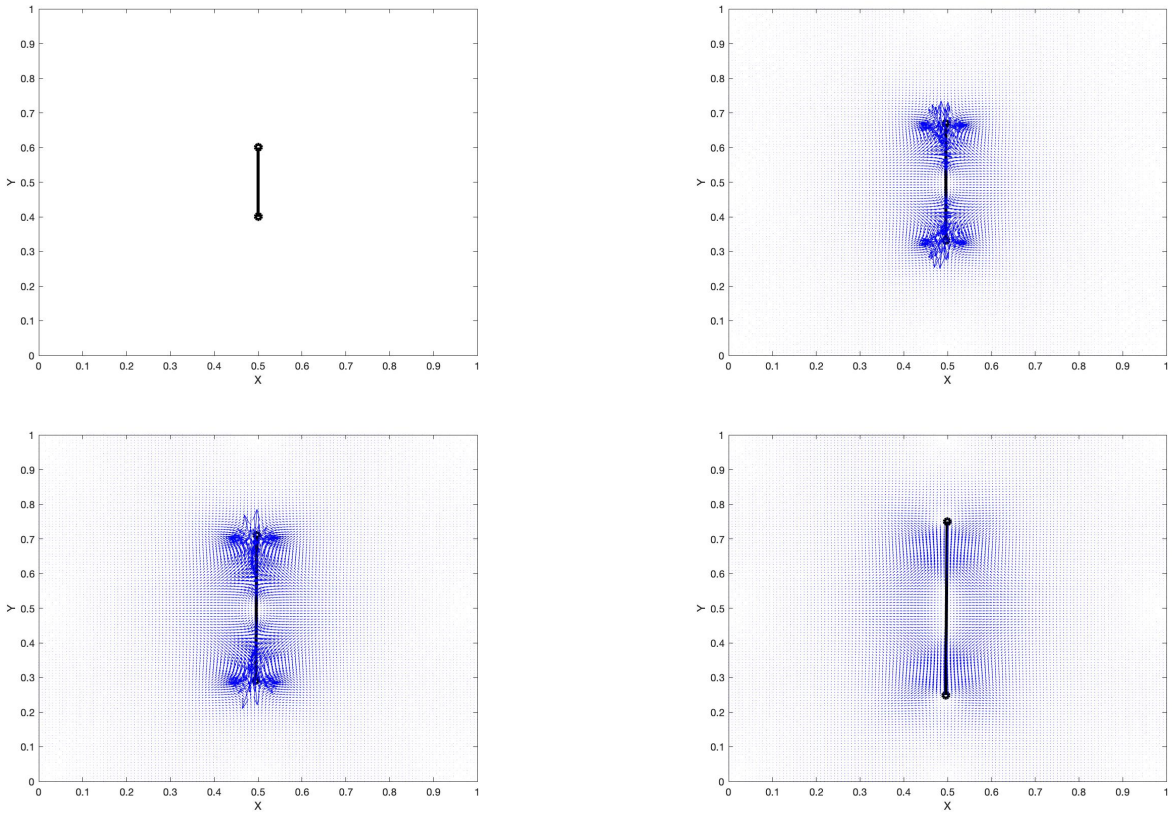

Figure 1: The expansion of the distance of two points after $0, 5, 10$, and 100 time-steps with the associated vector field.

Here, we recognize that the points do expand to the rest length. Then, after an extended period of time, we see that the forces are no longer enacting on the points once they reach the proper rest length.

3.1.2 Contracting test

In this test, we will set two points, $\mathbf{X}_1$ and $\mathbf{X}_2$, at an initial distance that is greater than the rest distance. Here we will let $||\mathbf{X}_1 - \mathbf{X}_2|| = 0.2$, $\Delta s = 0.05$, and $S_T = 30$. We observe the following results in figure (2).

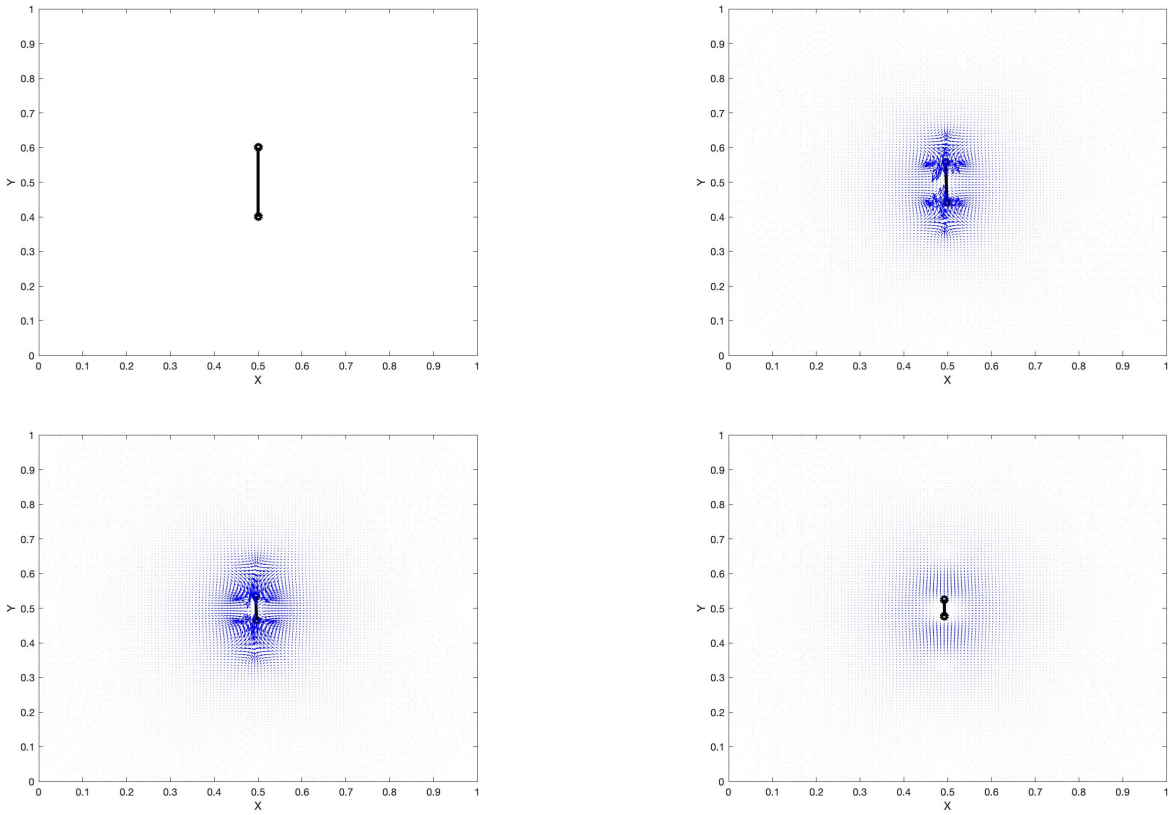

Figure 2: The contracting of the distance of two points after $0, 5, 10$, and 100 time-steps with the associated vector field.

Similar to the expansion test, we observe that the points contract until they reach the proper rest length. As the points approach the proper rest length, the forces enacting on the points weaken and eventually cease after an extended period of time.

3.2 Bending force test

Next, we will observe two different bending tests. Our initial test will examine our bending forces' ability to straighten an angled filament consisting of three points, then we will examine the forces' ability to angle a straight filament consisting of three points.

3.2.1 Straightening test

In this test, we will set three points, $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$ in a configuration where the points are not in a straight line. In order for our bending forces to properly straighten our points, we set the preferred curvature $c = 0$. Additionally, we set $\Delta s = 0.1$, $S_T = 30$, $S_B = 0.1$. We observe the following results in figure (3).

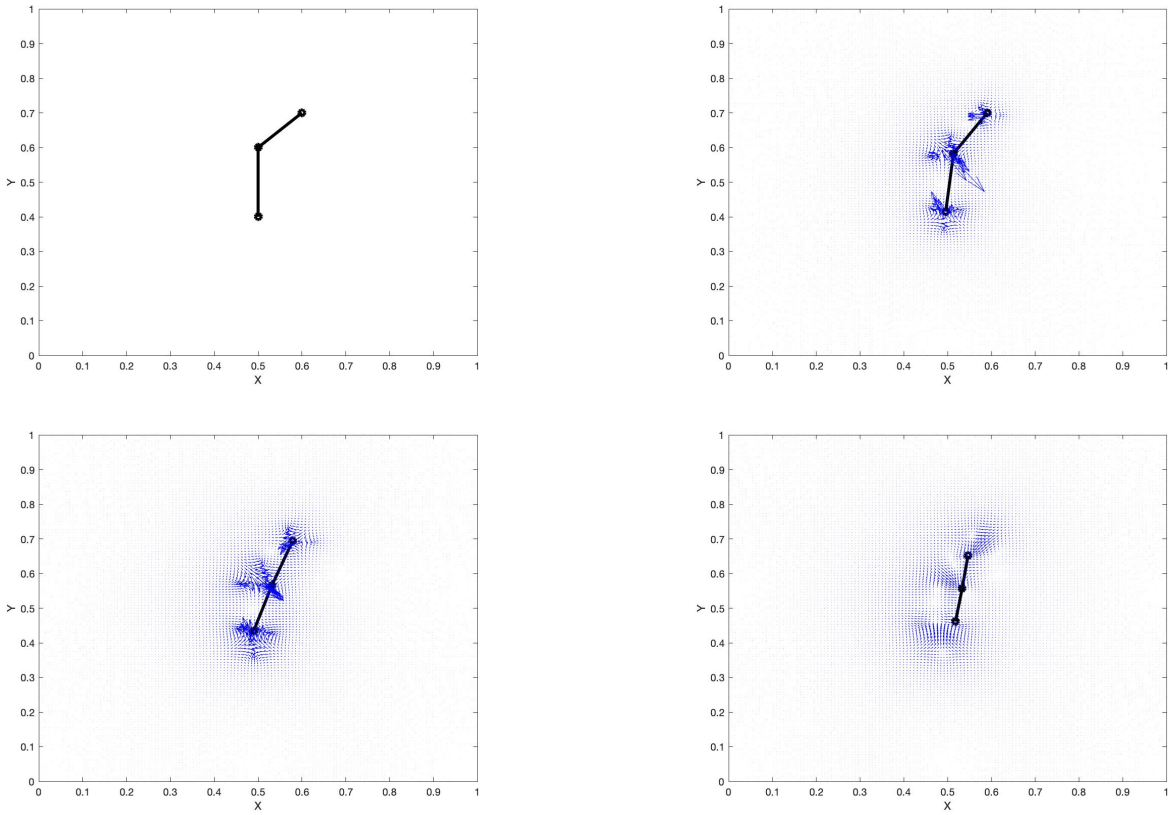

Figure 3: The straightening of three points after $0, 2, 5$, and 20 time-steps with the associated vector field.

Here, we observe that the points begin at an angle and straighten out over time. Then, once the filament is properly aligned, the forces weaken and the filament is at rest.

3.2.2 Bending test

In this test, we will set three points, $\mathbf{X}_1, \mathbf{X}_2$, and $\mathbf{X}_3$ in a configuration where the points are in a straight line. In order for our bending forces to properly angle our points, we set the preferred curvature $c = 15$. Additionally, we set $\Delta s = 0.1$, $S_T = 30$, $S_B = 0.1$. We observe the following results in figure (4).

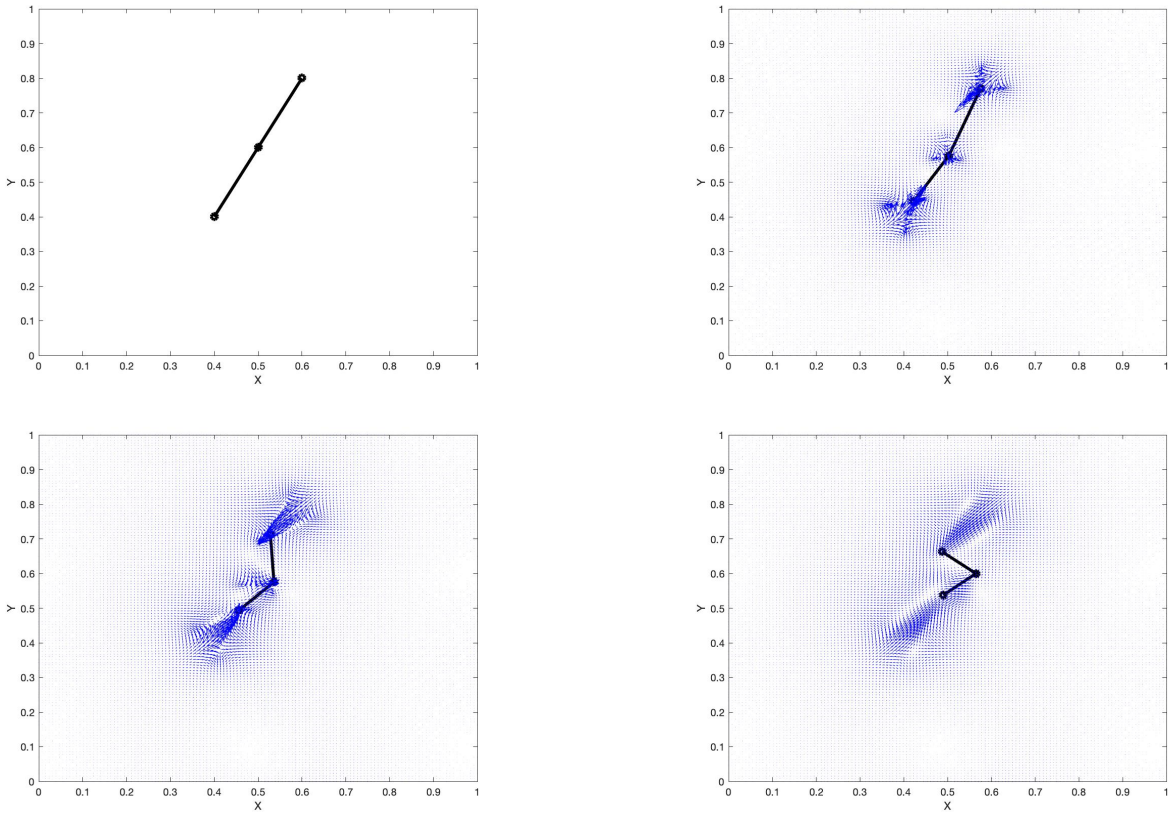

Figure 4: The bending of three points after $0, 5, 15$, and 40 time-steps with the associated vector field.

Here, we observe that the points begin at on a straight line and form an angle over time. Similar as above, once the filament is properly aligned, the forces weaken and the filament is at rest.

3.3 Straightening two filaments

Last, we will incorporate a second filament, where both immersed boundaries have unique shapes. We will initialize both filaments in the fluid and solve for their corresponding forces where their preferred curvature is a straight line. We will initialize our filaments as semi-circles then as shifted sinusoidal curves.

3.3.1 Straightening two semi-circles

In this test, we will set two filaments, each with ten points, in the shapes of semi-circles with a handle on the right-hand-side. The semi-circles have a radius of $r = 0.2$, $\Delta s = 0.05$, $S_T = 30$, $S_B = 0.1$, and $c = 0$. We observe the following results in figure (5).

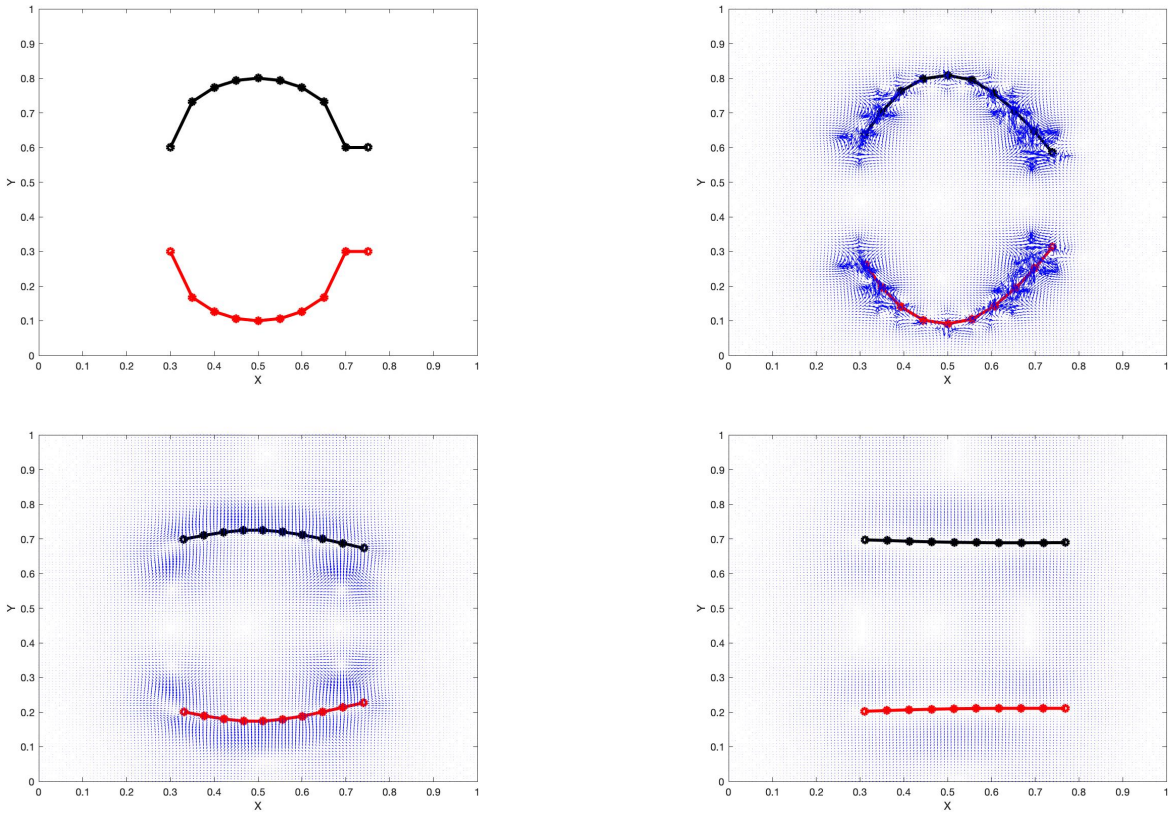

Figure 5: The straightening of two semi-circles with right-hand-side handles after $0, 5, 50$, and 200 time-steps with the associated vector field.

Similar to the examples above, we observe the filaments straightening to the preferred orientation. As they approach straight lines, the associated forces acting on the points continue to reduce. Once the filament reaches equilibrium, the enacting forces are negligible.

3.3.2 Straightening two sinusoidal curves

In this test, we will set two filaments made of twenty two points in phase shifted sinusoidal curves. This is inspired by *Hydrodynamic interactions of sheets vs filaments: Synchronization, attraction, and alignment* by Sarah Olsen and Lisa Fauci. In their paper, Olsen and Fauci explore simulations of sperm cells, where the flagella are initialized as phase shifted sinusoidal curves. Then, using time-dependent preferred curvature, they observe that the locomotion of the flagella of two cells synchronize after a long enough time period [3]. In our test, we will simply set our preferred curvature constant $c = 0$. Additionally, we set $\Delta s = 0.02$, $S_T = 30$, and $S_B = 0.01$. We observe the following results in figure (6).

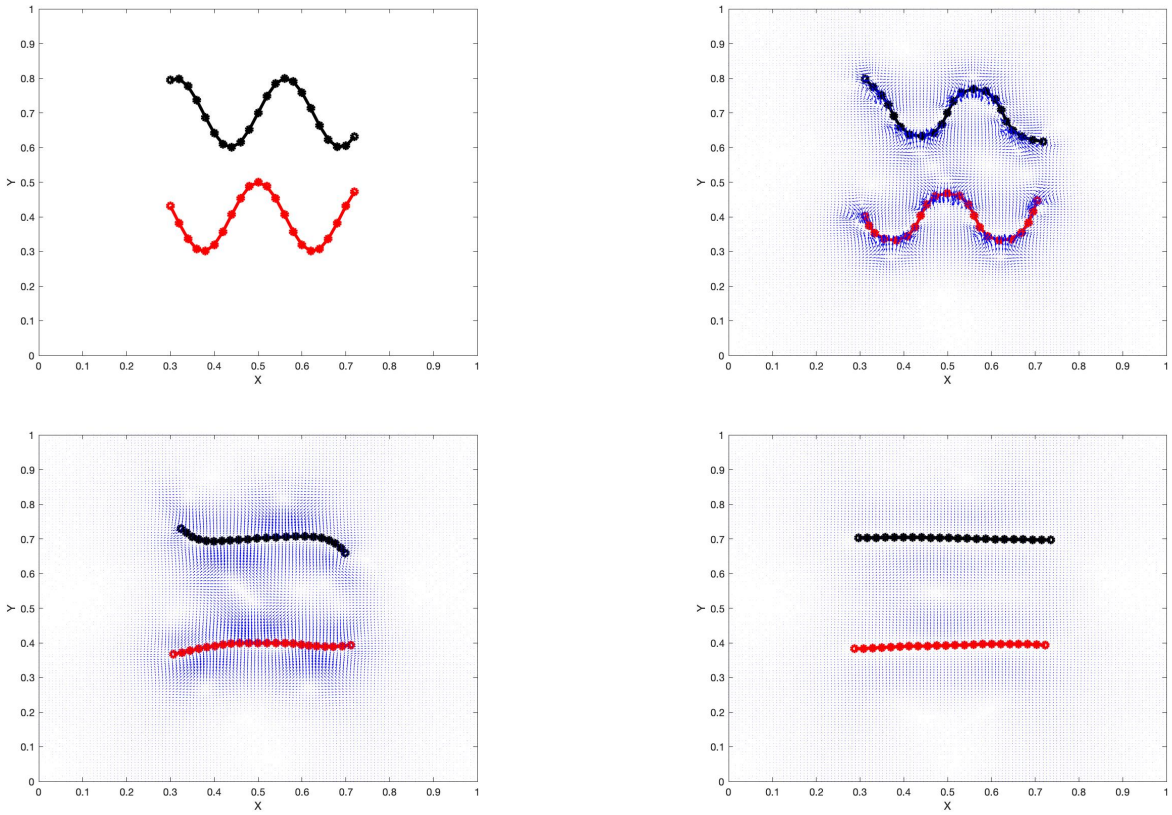

Figure 6: The straightening of two shifted sinusoidal curve after $0, 10, 50$, and 200 time-steps with the associated vector field.

Consistent with our previous results, we observe that the curved filaments indeed straighten out over time. Again, we observe that, after a long enough time period, the forces begin the shrink and vanish as the immersed boundaries approach the preferred orientation.

4 Discussion and Conclusion

From the previous section, we observe that the properly implemented tensile and bending forces allow us to orient our filaments in a preferred shape. In each of the examples, we initialize our filament in a shape that is not in equilibrium, observe the forces enacting on the points to properly orient, then the forces dissipate once the filament approaches equilibrium. This can be regarded as on overall success, where we can achieve preferred shapes from arbitrarily initialized filaments.

However, there are some areas in which our procedure may be improved. In each of our tests, we observe blue shadows from the vector fields that linger from where the filament passes. This is a result of residual forces that are spread on our fluid grid that are not updated. This occurs because our rest length Δs, in each case, is much greater than the spacial grid of the fluid solver, h. To combat this leakiness, we should update our rest length to be approximately one-half the spacial grid.

Additionally, in section 3.1, we observe slight tilts in our filaments once they reach equilibrium. This should not be the case. We believe this occurs, again due to the significantly larger rest length.

Because the rest length is so large, the forces that accumulate in a ball of radius $2h$ around each Lagrangian point come solely from the individual point. There is no interference from the other Lagrangian point, and as such, the points are subject to forces that may cause the point to drift in either direction. This also may occur due to an indexing error within the routine. From the figures above, we observe that the vector fields seem shifted around the filaments incorrectly. The vector fields should entirely surround each point, however there appears to be a white 'shadow' around each point. This suggests that the Lagrangian point is not centered properly when we spread and extrapolate forces from the point to the grid and vice versa. If the point is not properly centered then it will be subject to drifting forces.

The natural extension for this project is to incorporate time-dependent bending forces. Similar to Olsen and Fauci in [3], we aim to model the locomotion of the flagella of a sperm cell. This movement is achieved by defining our preferred curvature c as a temporally dependent function. Additionally, we may consider incorporating fixed boundaries within the fluid as well as background flows. In our simulations, the spacial domain is $\mathbb{R}^2$ with an initial fluid velocity $\mathbf{u} = \mathbf{0}$ everywhere. However, to accurately observe sperm cell movement, we must restrict our domain and include an initial flow representative of the cell's natural habitat. Once properly implemented, we then can observe the interacting effects of phase shifted filaments with identical time dependent preferred curvatures in a non-zero Reynold's number fluid.

A MATLAB Routine

```
clc
clear
close all

%We want to live in a unit square
X0=0;X1=1;
Y0=0;Y1=1;

%Want a 33x33 sized grid, will adjust to something more refined later
Ng=129;
h=1/(Ng-1);
x=set_grid(X0,X1,h);
y=set_grid(X0,X1,h);
[X,Y]=meshgrid(x,y);

%Initial fluid velocities, here we assume the fluid is at rest when we
%input our IB
u0=zeros(Ng);
v0=zeros(Ng);

%Time step, we define our initial time at 0 and our end time after Nt
%steps
dt=0.00001;
Nt=200;
T0=0;T1=Nt*dt;
t=set_grid(T0,T1,dt);

%Resting length for Lag points, we define this as h/2
ds=0.02;

%Values for liquid
mu=1;
rho=1;

%Defining a filament that looks like a semicircle of radius r centered at
%(centx,centy)
r=0.2;
centx=0.5;
centy1=0.6;
x=centx-r:ds:centx+r+ds;
N=length(x);
y1=0.1*sin(8*pi*x)+0.7;%real(sqrt(r^2-(x-centx).^2)+centy1);%
X_fil_prev1=[x;y1];%[0.4,0.5,0.6;0.4,0.6,0.8];%

centy2=0.3;
y2=0.1*sin(8*pi*x+pi/2)+0.4;%real(-sqrt(r^2-(x-centx).^2)+centy2);%
```

```
X_fil_prev2=[x;y2];


% %Movie stuff
% v=VideoWriter('Different_movie.avi');
% open(v);

%Plotting the filament to see what it looks like
 figure(1)
plot(X_fil_prev1(1,:),X_fil_prev1(2,:),'ko-','LineWidth',3);
hold on
plot(X_fil_prev2(1,:),X_fil_prev2(2,:),'ro-','LineWidth',3);
hold on
quiver(X,Y,u0',v0',0,'b');
hold off
xlim([X0,X1]);
ylim([Y0,Y1]);
xlabel('X');
ylabel('Y');
%  frame=getframe(gcf);
%  writeVideo(v,frame);


%Setting tensile and bending coefficents
St=30;
Sb=0.01;

%Reg parameter for delta function

m=zeros(Nt,1);
theta=zeros(Nt,1);
%Now consider for various time steps
for kk=1:Nt
    %Calculate forces of filaments
    f1=forces(X_fil_prev1,ds,St,Sb);
    f2=forces(X_fil_prev2,ds,St,Sb);

    %Spread forces on a grid
    [F1_fil1,F2_fil1]=spread_forces(Ng,X_fil_prev1,f1,h,ds);
    [F1_fil2,F2_fil2]=spread_forces(Ng,X_fil_prev2,f2,h,ds);

    %Total force
    F1=F1_fil1+F1_fil2;
    F2=F2_fil1+F2_fil2;

    %Now we need to solve NS for one time step

    %Remove redundant row and column from initial condition
```

```
    if kk==1
        uprev=u0(1:end-1,1:end-1);
        vprev=v0(1:end-1,1:end-1);
    else
        uprev=upres(1:end-1,1:end-1)';
        vprev=vpres(1:end-1,1:end-1)';
    end

    M=length(uprev);

    %Remove redundant row and column from force, recognize that our solver uses
    %change in x on the columns and change in y on the rows whereas F1 and F2
    %are opposite
    Fu=(F1(1:end-1,1:end-1))';
    Fv=(F2(1:end-1,1:end-1))';

    %RHS of first ADI step
    RHSu=uprev+(dt./rho).*Fu;
    RHSv=vprev+(dt./rho).*Fv;

    %Initialize empty matrices
    u_star_star=NaN(M);
    u_star=u_star_star;
    v_star_star=u_star_star;
    v_star=v_star_star;

    %ADI in the x-direction
    for l=1:M
        uprev_vec=uprev(:,l);
        %Constructing our Linear system for the first ADI step
        [Lu,Uu]=matrix_periodic(uprev_vec,h,dt,rho,mu);
        %Solving for u_star_star_vec and v_star_star_vec
        utemp1=Lu\RHSu(:,l);
        vtemp1=Lu\RHSv(:,l);

        u_star_star_vec=Uu\utemp1;
        v_star_star_vec=Uu\vtemp1;

        u_star_star(:,l)=u_star_star_vec;
        v_star_star(:,l)=v_star_star_vec;
    end

    %Transpose everything to repeat in the y-direciton
    u_star_star=u_star_star';
    v_star_star=v_star_star';
    vprev=vprev';
    for mm=1:M
        vprev_vec=vprev(:,mm);
```

```
        %Constructing our Linear system for second ADI step
        [Lv,Uv]=matrix_periodic(vprev_vec,h,dt,rho,mu);
        %Solving for u_star_vec and v_star_vec
        utemp2=Lv\u_star_star(:,mm);
        vtemp2=Lv\v_star_star(:,mm);

        u_star_vec=Uv\utemp2;
        v_star_vec=Uv\vtemp2;

        u_star(:,mm)=u_star_vec;
        v_star(:,mm)=v_star_vec;
    end
    %Transpose back so that our solutions are in the original form
    u_star=u_star';
    v_star=v_star';

    %Solving for pressure
    RHSp=pressure_right_hand_side(u_star,v_star,h,dt,rho);
    %2-D FFT of RHSp
    RHSp_hat=fft2(RHSp);
    %Solve for p_hat
    p_hat=NaN(M);
    for n=0:M-1
        for q=0:M-1
            if n==0 && q==0
                p_hat(n+1,q+1)=0;
            else
                p_hat(n+1,q+1)=(h^2).*RHSp_hat(n+1,q+1)./(-4.*((sin(pi.*n./M)).^2+(
                    sin(pi.*q./M)).^2));
            end
        end
    end
    %Inverse 2-D FFT
    p=real(ifft2(p_hat));
    %Need x-deriv and y-deriv of p for grad p, use center difference
    D0=diag(-1.*ones(M-1,1),-1)+diag(1.*ones(M-1,1),1);
    D0(1,end)=-1;
    D0(end,1)=1;

    %Estimating x-deriv of p
    Dp_x=(1/(2*h))*D0*p;
    %Estimating y-deriv of p
    Dp_y=(1/(2*h))*(D0*(p'))';
    %Solving for present time step
    upres=u_star-(dt/rho).*Dp_x;
    vpres=v_star-(dt/rho).*Dp_y;

    %Use periodicity to retrive present veolcities, note that we need to
```

```
    %transpose one final time to observe change in x on the rows and change in
    %y on the columns
    upres=[upres,upres(:,1)];
    upres=([upres;upres(1,:)])';
    vpres=[vpres,vpres(:,1)];
    vpres=([vpres;vpres(1,:)])';

    U1=combine_forces(Ng,X_fil_prev1,upres,vpres,h);
    U2=combine_forces(Ng,X_fil_prev2,upres,vpres,h);

    %Update our filaments
    X_fil_pres1=X_fil_prev1+dt.*U1;
    X_fil_prev1=X_fil_pres1;

    X_fil_pres2=X_fil_prev2+dt.*U2;
    X_fil_prev2=X_fil_pres2;
    if kk==10
        figure(2)
        plot(X_fil_prev1(1,:),X_fil_prev1(2,:),'ko-','LineWidth',3);
        hold on
        plot(X_fil_prev2(1,:),X_fil_prev2(2,:),'ro-','LineWidth',3);
        hold on
        quiver(X,Y,upres',vpres',0,'b');
        hold off
        xlim([X0,X1]);
        ylim([Y0,Y1]);
        xlabel('X');
        ylabel('Y');
    elseif kk==50
        figure(3)
        plot(X_fil_prev1(1,:),X_fil_prev1(2,:),'ko-','LineWidth',3);
        hold on
        plot(X_fil_prev2(1,:),X_fil_prev2(2,:),'ro-','LineWidth',3);
        hold on
        quiver(X,Y,upres',vpres',0,'b');
        hold off
        xlim([X0,X1]);
        ylim([Y0,Y1]);
        xlabel('X');
        ylabel('Y');
    elseif kk==200
        figure(4)
        plot(X_fil_prev1(1,:),X_fil_prev1(2,:),'ko-','LineWidth',3);
        hold on
        plot(X_fil_prev2(1,:),X_fil_prev2(2,:),'ro-','LineWidth',3);
        hold on
        quiver(X,Y,upres',vpres',0,'b');
        hold off
```

```
        xlim([X0,X1]);
        ylim([Y0,Y1]);
        xlabel('X');
        ylabel('Y');
    end
%         plot(X_fil_prev1(1,:),X_fil_prev1(2,:),'ko-','LineWidth',3);
%         hold on
%        plot(X_fil_prev2(1,:),X_fil_prev2(2,:),'ro-','LineWidth',3);
%         hold on
%         quiver(X,Y,upres',vpres',0,'b');
%         hold off
%         xlim([X0,X1]);
%         ylim([Y0,Y1]);
%         xlabel('X');
%         ylabel('Y');
%     frame=getframe(gcf);
%     writeVideo(v,frame);

%     v1=X_fil_pres1(:,1)-X_fil_pres1(:,2);
%     v2=X_fil_pres1(:,3)-X_fil_pres1(:,2);
%
%     theta(kk)=acos((v1'*v2)/(norm(v1)*norm(v2)));
%     m(kk)=norm(v1);
end

% close(v);


function x=set_grid(X0,X1,delta)
M=1+(X1-X0)/delta;
x=linspace(X0,X1,M)';
end

function f=forces(X_fil_prev,ds,St,Sb)
    %Solving for tensile forces
    N=length(X_fil_prev);
    ftk=zeros(size(X_fil_prev));
    ftkp1=ftk;
    fbk=ftk;
    fbkp1=fbk;
    fbkm1=fbk;
    for k=1:N-1
        v1=X_fil_prev(:,k+1)-X_fil_prev(:,k);
        ftk(:,k)=St*(norm(v1)/ds-1)*(v1/norm(v1));
        ftkp1(:,k+1)=-ftk(:,k);
    end


```

```
    %Solving for bending forces
    c=0;%sqrt(0.1);%sqrt(0.2);
    for k=2:N-1
            v2=X_fil_prev(:,k+1)-2*X_fil_prev(:,k)+X_fil_prev(:,k-1);
            if norm(v2)==0
                fbk(:,k)=0;
                fbkm1(:,k-1)=0;
                fbkp1(:,k+1)=0;
            else
                fbk(:,k)=(2*Sb/ds)*(norm(v2)/(ds^2)-c)*(v2/norm(v2));
                fbkm1(:,k-1)=(-Sb/ds)*(norm(v2)/(ds^2)-c)*(v2/norm(v2));
                fbkp1(:,k+1)=(-Sb/ds)*(norm(v2)/(ds^2)-c)*(v2/norm(v2));
            end
    end
    %Summation of forces
    f=ftk+ftkp1+fbk+fbkm1+fbkp1;
end

function [F1,F2]=spread_forces(Ng,X_fil_prev,f,h,ds)
        %Now we spread the forces
    N=length(X_fil_prev);
    %Initialize our force matrices
    F1 = zeros(Ng, Ng);
    F2 = zeros(Ng, Ng);

    %Redefine our filament as x-coordinates and y-coordinates
    x=X_fil_prev(1,:);
    y=X_fil_prev(2,:);

    %Redefining our force as force on x-coordinates and force on y-coordinates
    fx=f(1,:);
    fy=f(2,:);

    %Spreading force from filament to grid using cosine delta function approx
    for j=1:N
        ndelta=2;
        delfun=@(x) 1/4*(1 + cos(pi*x/2));
        nfx=floor(x(j)/h)+1;
        nfy=floor(y(j)/h)+1;
        Indx = nfx+(-ndelta+1:1:ndelta);
        Indy = nfy+(-ndelta+1:1:ndelta);
        [ptsx,ptsy] = meshgrid(Indx,Indy);
        dx = (x(j))/h - ptsx;
        dy = (y(j))/h - ptsy;
        dspread = delfun(dx).*delfun(dy);
        Indxmod = mod(Indx-1,Ng)+1;
        Indymod = mod(Indy-1,Ng)+1;
        F1(Indxmod, Indymod) = F1(Indxmod, Indymod) + dspread * fx(j) * ds / h^2;
```

```
        F2(Indxmod, Indymod) = F2(Indxmod, Indymod) + dspread * fy(j) * ds / h^2;
    end
end

function U=combine_forces(Ng,X_fil_prev,upres,vpres,h)
    N=length(X_fil_prev);
    x=X_fil_prev(1,:);
    y=X_fil_prev(2,:);
    %Now we need to go from the grid back to forces on the Lag points
    U=NaN(2,N);
    for j=1:N
        ndelta=2;
        delfun=@(x)1/4*(1 + cos(pi*x/2));
        nfx=floor(x(j)/h)+1;
        nfy=floor(y(j)/h)+1;
        Indx = nfx+(-ndelta+1:1:ndelta);
        Indy = nfy+(-ndelta+1:1:ndelta);
        [ptsx,ptsy] = meshgrid(Indx,Indy);
        dx = (x(j))/h - ptsx;
        dy = (y(j))/h - ptsy;
        dspread = delfun(dx).*delfun(dy);
        Indxmod = mod(Indx-1,Ng)+1;
        Indymod = mod(Indy-1,Ng)+1;
        u_grid_vel=upres(Indxmod,Indymod);
        v_grid_vel=vpres(Indxmod,Indymod);
        X_vel=(1/h^2).*u_grid_vel.*dspread;
        Y_vel=(1/h^2).*v_grid_vel.*dspread;
        U(:,j)=[sum(X_vel,'all');sum(Y_vel,'all')];
    end

end

function [L,U]=matrix_periodic(uprev,h,dt,rho,mu)
%Coefficient of lower diag
a=-(dt./(2.*h)).*uprev-((mu.*dt)./(rho.*h.^2)).*ones(length(uprev),1);

%Coefficient of middle diag
b=(1+((2.*mu.*dt)./(rho.*h.^2))).*ones(length(uprev),1);
%Coefficeint of upper diag
c=(dt./(2.*h)).*uprev-((mu.*dt)./(rho.*h.^2)).*ones(length(uprev),1);

A=diag(a(2:end),-1)+diag(b)+diag(c(1:end-1),1);
A(1,end)=a(1);
A(end,1)=c(end);
[L,U]=lu(A);
end

function RHSp=pressure_right_hand_side(ustar,vstar,h,dt,rho)
```

```
M=length(ustar);
%Establishing our periodic center difference matrix
D0=diag(-1.*ones(M-1,1),-1)+diag(1.*ones(M-1,1),1);
D0(1,end)=-1;
D0(end,1)=1;

%Estimating x-deriv in u component
Dx=D0*ustar;
%Estimating y-deriv in v component
Dy=(D0*(vstar'))';

%Initialize RHSp
RHSp=(rho./(2.*dt.*h)).*(Dx+Dy);
end
```

References

[1] Chorin, A. J. (1967). The Numerical Solution of the Navier-Stokes Equations for an Incompressible Fluid. *Bulletin of American Mathematical Society, 73*, pp. 928-931. https://doi.org/10.1090/S0002-9904-1967-11853-6

[2] Cortez, R. (2001). The Method of Regularized Stokeslets. *SIAM J. Sci. Comput., 23(4)*, pp. 1204-1225. https://doi.org/10.1137/S106482750038146X

[3] Olsen, S. D., & Fauci, L. J. (2015). Hydrodynamic interactions of sheets vs filaments: Synchronization, attraction, and alignment. *Physics of Fluids, 27* http://dx.doi.org/10.1063/1.4936967

[4] Peaceman, D. W., & Rachford, Jr, H. H. (1955). The Numerical Solution of Parabolic and Elliptic Differential Equations. *Journal of the Society for Industrial and Applied Mathematics, 3(1)*, pp. 28-41. https://doi.org/10.1137/0103003

[5] Peskin, C. (2002). The Immersed Boundary Method. *Acta Numerica, 11*, pp. 479-517. https://doi.org/10.1017/S0962492902000077