**B.Tech Project - CP302**

# Understanding Behaviour of Elastic Particles in Fluid


*Submitted by*

**Akshat Chauhan (2021meb1265)**

**Kayitha Saran Yadav (2021meb1290)**

**Korupolu Deekshitha (2021meb1292)**

**Patel Mahant (2021meb1306)**

**Ujjawal Kumar (2021meb1331)**

*Under the supervision of*

**Dr. Navaneeth K. Marath**




**Department of Mechanical Engineering**

**Indian Institute of Technology, Ropar**


**Year of Submission: 2024**

# Abstract

An integrated approach to fluid-structure interaction and particle separation in microfluidic devices offers a versatile solution to address complex microscale problems. By combining insights from the Immersed Boundary Method (IBM) and advances in microscale fluid dynamics, researchers can accurately simulate the interaction between elastic structures and fluids while simultaneously developing systems that efficiently separate particles in microfluidic environments. The IBM uses a dual-grid framework—an Eulerian grid for the fluid and a Lagrangian mesh for the structure—to simulate the dynamic interaction between moving, flexible structures and surrounding fluids. This method has been particularly effective in modeling biological systems, such as blood flow through heart valves or the motion of insect wings, where both the deformation of the structure and the flow of fluid must be accurately captured in real-time. At the same time, developments in microfluidics have revolutionized particle separation techniques, enabling the continuous sorting of cells and particles based on physical properties such as size, deformability, and electrical charge. For example, microfluidic devices can sort cancer cells from healthy cells in a blood sample or isolate pathogens with high efficiency, making them crucial in healthcare and disease monitoring.

The integration of the Immersed Boundary Method with microfluidic separation techniques creates a powerful, multi-purpose approach that extends the capabilities of both methods. IBM allows for the accurate modeling of fluid-structure interactions within microfluidic channels, helping researchers design devices that can simulate and manipulate flexible biological structures, such as tissues or synthetic membranes. This combined approach is especially useful in the design of lab-on-chip or organ-on-chip systems, where the precise behavior of biological structures in fluid environments is essential for success. In industrial applications, the use of IBM can help optimize processes like chemical separation, where different particles or compounds need to be efficiently sorted based on their behavior in fluid flows.

By merging these computational and experimental techniques, researchers can develop more efficient, scalable, and adaptive solutions for a wide range of applications. Whether improving diagnostic tools, advancing research in fluid mechanics, or enhancing industrial processes, the combination of IBM simulations and microfluidic particle separation opens new avenues for innovation. This integrated framework provides a flexible and powerful approach to solving real-world problems at the microscale, offering significant potential for future developments in both biological and industrial systems.

# Contents

# List of Figures

# Nomenclature

| | |
|---|---|
| $\mu$ | Dynamic Viscosity |
| $\rho$ | Density |
| $E$ | Elastic Potential Energy |
| $F$ | Force per unit area |
| $f$ | Force Density on Eulerian Grid |
| $k$ | Spring stiffness |
| $M$ | Mass Density |
| $P$ | Pressure |
| $Re$ | Reynolds Number |
| $X_M$ | Coordinates of master node |
| $X_{SL}$ | Coordinates of slave node |

# 1 Introduction[3]

In the modern world, challenges arise at the micro-scale level, requiring to be analyzed and solved in the same dimensions. From understanding biological processes within cells to improving diagnostics and medical devices, micro-scale problems demand precise and innovative approaches. The computational fluid dynamic method like the Immersed boundary method helps to a greater extent by computationally analyzing how elastic structures interact with fluid environments, making it essential for simulating real-world systems.

The **Immersed Boundary Method (IBM)**is a technique used to simulate fluid-structure interactions, particularly in systems where an elastic or flexible structure is immersed in a fluid. Originally introduced by Dr.Charles Peskin in 1972 to model the dynamics of heart valves, IBM has since evolved into a powerful tool for studying complex systems involving fluid-structure interactions across various fields such as biology, engineering, and biomechanics.

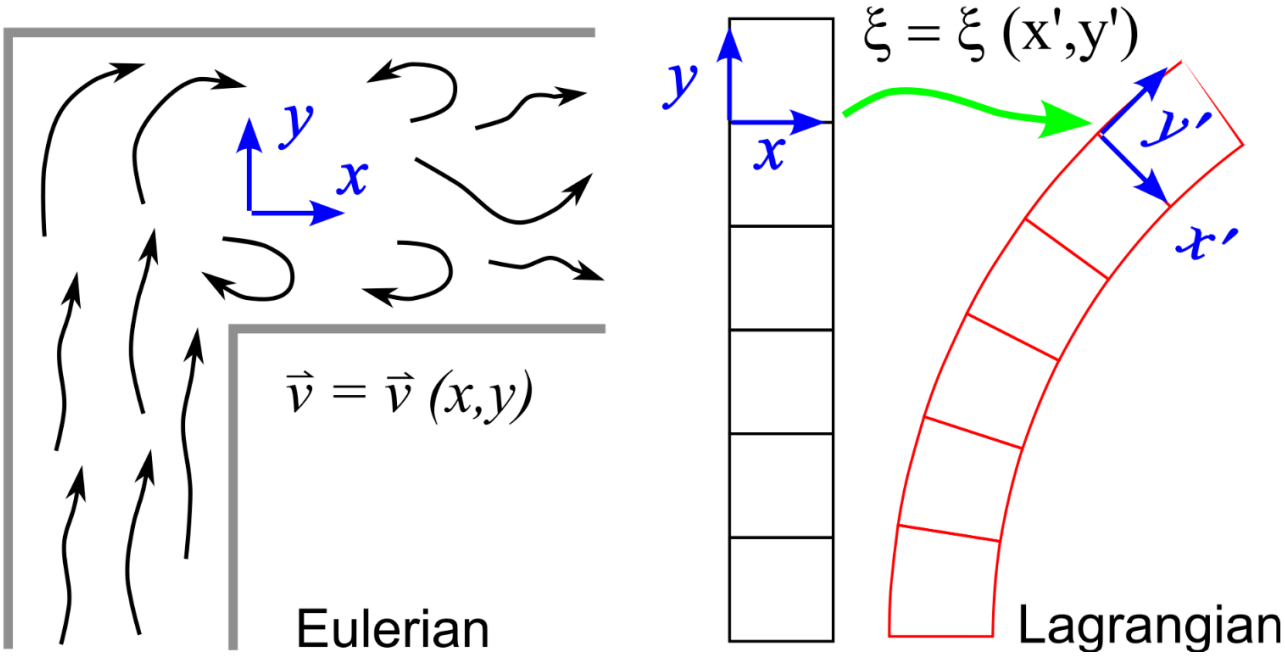
IBM operates on a unique Eulerian-Lagrangian framework, where the fluid is represented on a fixed Eulerian grid, while the immersed structures (such as elastic membranes or fibers) are modeled on a Lagrangian grid that can move and deform. This method allows for the coupling of fluid dynamics and structural mechanics, enabling accurate simulations of how fluids and structures influence each other. The IBM is particularly advantageous when dealing with complex, moving geometries, as it can handle large deformations of the immersed boundaries without requiring grid remeshing.

The IB2d implementation, as discussed in the first paper, brings the power of IBM to two-dimensional simulations. It is developed as an open-source software in Python and MATLAB, offering an accessible platform for researchers to model systems where fluids interact with flexible boundaries or immersed structures. By simulating the behavior of these structures in fluids, IBM is widely used in various applications that require modeling of fluid-structure interactions in biological, environmental, and industrial contexts.

# 2 Literature Review

## 2.1 Background and Significance of the IB Method

Fluid-structure interaction (FSI) problems are crucial in both engineering and biological systems, where the interaction between fluids and elastic structures leads to complex, fully coupled dynamics. The immersed boundary method, pioneered by Dr.Charles Peskin, is particularly well-suited to model such systems, with applications ranging from cardiovascular dynamics to aquatic locomotion and insect flight. The strength of the IB method lies in its ability to handle complex, time-dependent geometries using a fixed Eulerian grid for the fluid and a Lagrangian mesh for the elastic structures.




## 2.2 Existing Implementations and Challenges

There are several high-performance implementations of the IB method, such as IBAMR (which supports adaptive mesh refinement and parallelization in C++) and IBIS (implemented in FORTRAN). However, these require a deep understanding of lower-level programming languages and are computationally intensive, making them inaccessible to many researchers. Moreover, the installation and usage of these software packages can be complex due to dependencies on various libraries and high-performance computing requirements.

While there are other open-source IB codes, like matIB and pyIBM, they do not offer the range of fiber models, examples, or efficiency found in IB2d. Additionally, these earlier tools were often confined to specific problem domains or lacked broad applicability in biomechanics.

## 2.3 IB2d: Addressing Accessibility and Versatility

The IB2d package is positioned as a highly accessible tool for both experienced and novice users in the scientific community, providing full implementations in Python and MATLAB. IB2d simplifies the workflow for solving FSI problems and allows users to model a wide range of biomechanics problems, such as plant biomechanics, insect flight, muscle-fluid interactions, and physiological processes.

One of the key contributions of IB2d is its variety of fiber models that grant flexibility in simulating different material properties of immersed structures. These include:

**Hookean and Non-Hookean Springs:** For simulating elastic stretching.

**Torsional Springs:** For simulating resistance to bending.

**Target Points:** For imposing motion or fixation of certain points.

**Mass Points:** For adding inertia or gravity effects.

**Porous Models:** To simulate the permeability of biological tissues.

**Muscle Models:** Representing muscle dynamics using force-velocity and length-tension relationships.

Additionally, IB2d includes models for electrophysiology, such as the FitzHugh-Nagumo model, enabling simulations of action potential propagation in excitable tissues.

# 3 Analytical and Simulation Details [2]

In this section, we discuss the details of the simulation. We understood the simulation of a massive cylinder in a vertical channel with an external flow against gravity. We will first discuss the fiber models used and how to calculate the force on the Eulerian grid for different fiber models. Then we will discuss the workflow of the simulation and different functions and files used in the simulation.

## 3.1 What are Fiber Models?

The governing equation for fluid flow in the immersed boundary in eulerian form is given as:

$$\rho\left(\frac{\partial \mathbf{u}(\mathbf{x},\mathbf{t})}{\partial \mathbf{t}} + \mathbf{u}(\mathbf{x},\mathbf{t}) \cdot \nabla \mathbf{u}(\mathbf{x},\mathbf{t})\right) = -\nabla \mathbf{P}(\mathbf{x},\mathbf{t}) + \mu \nabla^2 \mathbf{u}(\mathbf{x},\mathbf{t}) + \mathbf{f}(\mathbf{x},\mathbf{t}) \tag{1}$$

where, u(x,t) is the fluid velocity, P(x,t) is the pressure field, and f(x,t) is the force applied on the Eulerian grid by the immersed boundary. The immersed boundary method assumes periodic boundary conditions and a square fluid domain. The interaction equations between the fluid and the immersed structure are given by:

$$\mathbf{f}(\mathbf{x},\mathbf{t}) = \int \mathbf{F}(\mathbf{r},\mathbf{t})\delta(\mathbf{x} - \mathbf{X}(\mathbf{r},\mathbf{t}))dr \tag{2}$$

$$\mathbf{U}(\mathbf{X}(\mathbf{r},\mathbf{t}),\mathbf{t}) = \frac{\partial \mathbf{X}(\mathbf{r},\mathbf{t})}{\partial \mathbf{t}} = \int \mathbf{u}(\mathbf{x},\mathbf{t})\delta(\mathbf{x} - \mathbf{X}(\mathbf{r},\mathbf{t}))dx \tag{3}$$

where, X(r,t) provides the Cartesian with coordinates at time t of the material point labeled by Lagrangian parameter r, F(r,t) is the force per unit area imposed onto the fluid by elastic deformations in the immersed structure as a function of the Lagrangian position, r, and time, t. The force density, F(r,t) is a function of the current immersed boundary's configuration. Various fiber models calculate the force density F(r,t) [1].

## 3.2 Fiber Models Used in Simulation

To calculate the force density F(r,t), we implement various fiber models to the Eulerian and Lagrangian points. The following types of fiber models are used in the simulation:

1. Springs (Hookean)
2. Torsional Springs(Beams)
3. Target Points

4. Mass Points (with gravity)

5. Background Flow Profiles

We calculate the force density based on the potential energy due to the deformation. Once the total deformation energy has been calculated,

$$\mathbf{E}(\mathbf{X}(\mathbf{r},\mathbf{t}),\mathbf{t}) = \sum_{k=1}^{M} \mathbf{E_k}(\mathbf{X_{k,1}}, \mathbf{X_{k,2}}, ...., \mathbf{X_{k,M}}) \tag{4}$$

then the corresponding elastic forces are calculated by the derivative of the deformation energy, where the elastic deformation force at point c of fiber model k is calculated as

$$\mathbf{F_{k,c}}(\mathbf{X}(\mathbf{r},\mathbf{t}),\mathbf{t}) = -\frac{\partial \mathbf{E}(X(r,t),t)}{\partial \mathbf{X_{k,c}}} \tag{5}$$

Note that X contains all immersed boundary points coordinates, M is the number of fiber structures in the system, and M is the number of immersed boundary points in the fiber structure.

### 3.2.1 Springs(Hookean)

Resistance to stretching between successive Lagrangian points can be achieved by modeling the connections with Hookean springs of resting length $R_L$ and spring stiffness. If the virtual spring displacement is below or beyond, the model will drive the system back towards a lower energy state. The elastic potential energy for a Hookean spring is given by:

$$\mathbf{E_{spring}} = \frac{1}{2} k_s (||X_{SL} - X_M|| - R_L)^2 \tag{6}$$

where $X_M$ and $X_{SL}$ are master and slave node coordinates respectively. The corresponding deformation forces is given by differentiation of the elastic energy:

$$\mathbf{F_{spring}} = k_s \left(1 - \frac{R_L}{||X_{SL} - X_M||}\right) \cdot \begin{pmatrix} x_{SL} - x_M \\ y_{SL} - y_M \end{pmatrix} \tag{7}$$

### 3.2.2 Torsional Springs

Resistance to bending between three successive Lagrangian points is modeled using a torsional spring connecting the three nodes. The model assumes a desired angle $\theta$, a prescribed curvature $\mathbf{C}$ between the three Lagrangian points, with corresponding bending stiffness $k_b$. The corresponding bending energy is given as:

$$\mathbf{E_{bend}} = \frac{1}{2} k_b (\hat{\mathbf{z}} \cdot (X_R - X_M) \times (X_M - X_L) - \mathbf{C})^2 \tag{8}$$

where $X_R$, $X_M$ , $X_L$ , are right, left, and master/middle Lagrangian nodal coordinates. The penalty force is designed to drive any deviations in the angle between these links back toward a lower energy state given by:

$$\mathbf{F_{bend}} = k_b \left[ (x_R - x_M)(y_M - y_L) - (y_R - y_M)(x_M - x_L) - \mathbf{C} \right] \cdot \begin{pmatrix} (y_M - y_L) + (y_R - y_M) \\ -(x_R - x_M) - (x_M - x_L) \end{pmatrix} \tag{9}$$

### 3.2.3 Target Points

Target points are used to describe the motion of Lagrangian points. Each Lagrangian point is associated with a virtual or target point. Target points are Lagrangian points that are anchored or have specific target positions. they both are connected by a spring with zero resting length. and the Elastic Energy associated is given by:

$$\mathbf{E_T}(\mathbf{X_M}) = \frac{1}{2} k_T (||X_M - X_M^T||)^2 \tag{10}$$

where $k_T$ is the target point stiffness and $X_M$ and $X_M^T$ are the coordinates of the physical Lagrangian and virtual target points, respectively. The corresponding deformation force is given by the derivative of the elastic energy.

$$\mathbf{F_T} = -k_T \begin{pmatrix} x_M - x_M^T \\ y_M - y_M^T \end{pmatrix} \tag{11}$$

### 3.2.4 Massive Points

Massive points are artificial points and they do not interact with the fluid directly and they are virtual points. Y(r,t) gives the coordinates of the massive points with mass density M(r) that do not interact with the fluid. X(r,t) gives the coordinates of the massless points that interact with the fluid. The massive and massless points are connected by a stiff virtual spring. As the fluid moves it exerts some force on the massless point, so this point will move to a new location, but since the massless point is connected by a stiff spring i.e. $k_M >> 1$ , which takes the massive point with it. The force acting on the massless point is transmitted to the massive point and it is given by:

$$\mathbf{F_M} = -k_M \left(Y(r,t) - X(r,t)\right) \tag{12}$$

This force will drive the position of the massive point according to the equation:

$$\mathbf{M(r)}\frac{\partial^2(Y(r,t)}{\partial t^2} = -F_M - M(r)g\hat{e_2} \tag{13}$$

### 3.2.5 Background Flow Profiles

Although the computational domain is assumed to have periodic boundary conditions, we can induce a desired background flow profile by artificially adding a force term in the Naiver Stokes Equation 1. Essentially, the additional force will be a penalty-type term, which exerts a force onto a desired subset of the fluid grid, if the fluid velocity does not match the desired flow profile. Such a forcing term can take the form:

$$\mathbf{F_{arb}} = k_{arb}\left(u(r,t) - u_{flow}(r,t)\right) \tag{14}$$

where $k_{arb}$ is the penalty strength coefficient and $u_{flow}$ is the desired background velocity profile. In this simulation, a parabolic profile into the channel was generated in the form:

$$\mathbf{u_{flow}(x,t)} = U_{max}\left(1 - \left(\frac{0.5 - x}{R}\right)^2\right) \tag{15}$$

The y-component of the desired velocity was assumed to be 0.

## 3.3 Files Used in Simulation

In this subsection, we will describe the simulation workflow. We will be looking at the MATLAB files and Python codes which we will be using for the simulation. At present, we are analyzing the open-source codes available for IBM from [2].

### 3.3.1 Main2d and Input2d

For every IBM simulation, there is a main2d and input2d file associated with it. The input2d is a file where the user needs to select the parameters for the simulation i.e. the fluid parameters, temporal information, grid parameters, fiber model construction, how to save the data, etc. Once the user has selected the desired parameters and necessary flags in input2d, the simulation is then started by calling the main2d file. This script reads all the information from the input2d file and passes it to the IBM-Driver script. We will come back to the IBM Driver later.

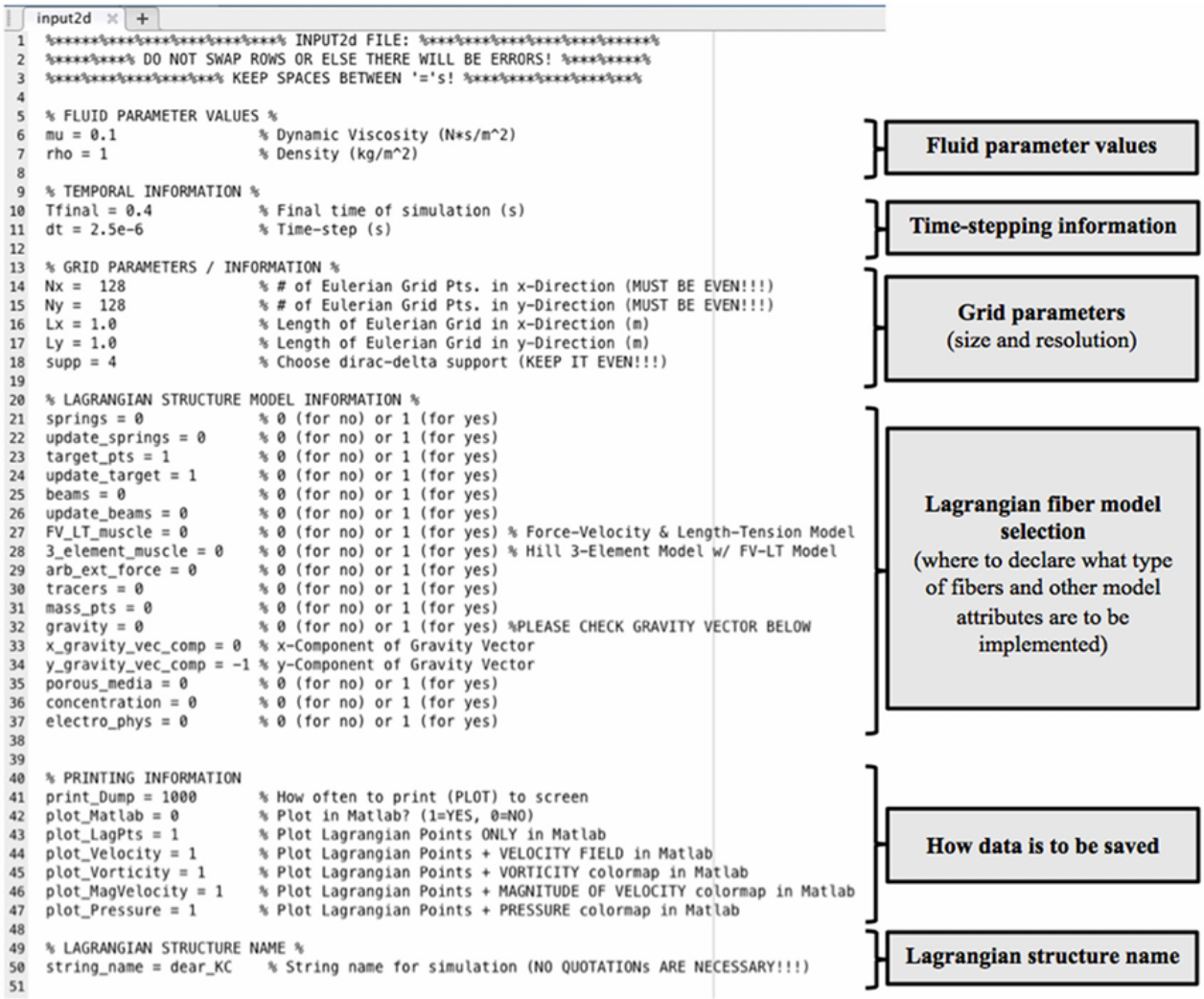



Figure 1: Input2d File Format

For our simulation of a cylinder in a vertical channel with an external flow against gravity, the fiber models which was selected were springs, target points, beams, external background forcing, and mass points with gravity in the negative y-direction. The fluid density $\rho = 1\frac{kg}{m^3}$ and viscosity $\mu = 0.01\frac{N.s}{m^2}$ were chosen. We will be observing the velocity contours and the motion of the Lagrangian points in the simulation.

### 3.3.2 Geometry Files

The first thing to do is to create a geometry for the simulation. The geometry file will read in the input parameters like grid spacing, and dimensions of the computational domain. From that, we define the dimensions of other structures used in the simulation. It will also read the fiber models that we have selected in the input2d file and according to that, it will create a *.vertex*, *.spring*, *.beam*, *.target*, *and .mass* files associated with the geometry of the simulation. These files represent the coordinates of the respective fiber model points and the stiffness of the spring used in the respective model.
**The .vertex file** lists all the initial Lagrangian coordinates in the domain.

| | |
|---|---|
| 1 | $N_{Lag}$ |
| 2 | x-Vertices y-Vertices |
| ⋮ | ⋮ ⋮ |

Figure 2: Vertex File Format

**The .spring file** lists all the master and slave nodes for each linear spring, including their stiffness and resting length. If non-linear springs are used, it also shows the degree of non-linearity.

| | | | | | |
|---|---|---|---|---|---|
| 1 | $N_{springs}$ | | | | |
| 2 | Lag. $ID_{master}$ | Lag. $ID_{slave}$ | Spring Stiffness, $k_{spr}$ | Resting Length, $R_L$ | Deg. of Nonlinearity, $\beta$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Figure 3: Spring File Format

**The .beam file** lists the right, middle, and left Lagrangian indices of each beam and their associated stiffness and curvature.

| | | | | | |
|---|---|---|---|---|---|
| 1 | $N_{beams}$ | | | | |
| 2 | Lag. $\text{ID}_L$ | Lag. $\text{ID}_M$ | Lag. $\text{ID}_R$ | Beam Stiffness, $k_b$ | Curvature, $C$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

Figure 4: Beam File Format

**The .target file** lists all target point indices with the target point stiffness.

| | | | |
|---|---|---|---|
| 1 | $N_{target}$ | | |
| 2 | Lag. ID | Target Pt. Stiffness, $k_T$ | |
| ⋮ | ⋮ | ⋮ | |

Figure 5: Target Points File Format

**The .mass file** lists the Lagrangian mass point indices with their associated mass and spring stiffness.

| | | | | |
|---|---|---|---|---|
| 1 | $N_{mass}$ | | | |
| 2 | Lag. ID | Mass Pt. Stiffness, $k_M$ | Mass (kg) | |
| ⋮ | ⋮ | ⋮ | ⋮ | |

Figure 6: Mass Points File Format

The computational domain in our case is 1m x 1m in dimension with 3 identical channels with a massive cylinder at the center with different masses. 64 grid points were chosen in each direction. The final geometry after running the geometry file was obtained as shown:

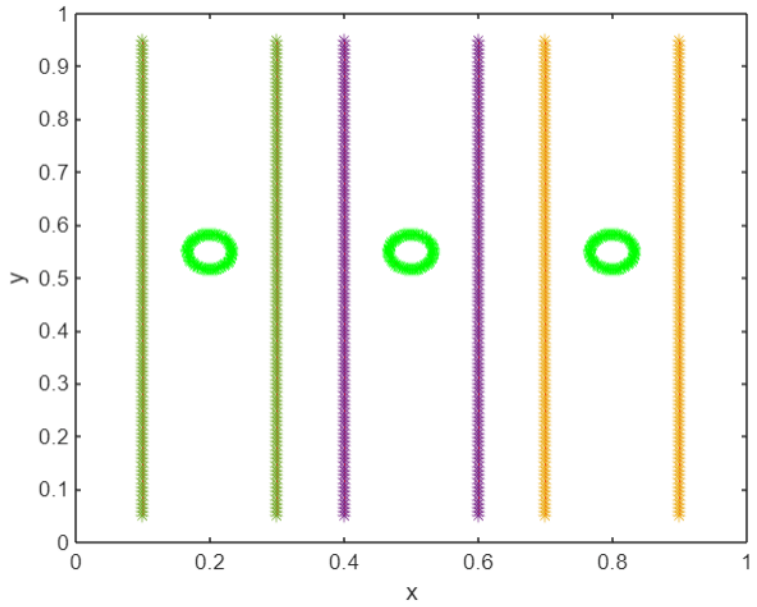



Figure 7: Geometry

### 3.3.3 IBM Driver

The IBM Driver file is common for all the simulations. It takes the input from the input2d file and runs the selected functions for the fiber models the user has chosen in the input2d file. The IBM driver is like the core of all the simulations.

It will read in the geometry files according to the fiber models selected. After reading the geometry files, it will initialize the simulation by creating the zero vectors of velocity, vorticity, pressure, and the forces on the Lagrangian grid. After initializing the simulation, the time stepping gets started. It will update the position of the Lagrangian points and find the corresponding forces that will act on the Eulerian grid due to the fluid-structure interaction.

After finding the forces on the Eulerian grid, the fluid solver will update the fluid velocity, pressure, and forces with the iterations. The Naiver-Stokes equation is solved using spectral methods in this simulation. After getting the final velocity, pressure, vorticity, and the positions of the Lagrangian points, the IBM driver will check which results the user wants to plot. Accordingly, it will be plotted in MATLAB.

The IBM Driver uses many other functions that are defined specifically for a purpose like reading the geometry files, initializing the simulation, updating the velocity by solving Naiver-Stokes, updating the position of Lagrangian points, etc. These are also common for all the simulations.

# 4 Results and discussions

After calling the main2d file which will pass the input to the IBM driver and run the simulation, we got the following results:

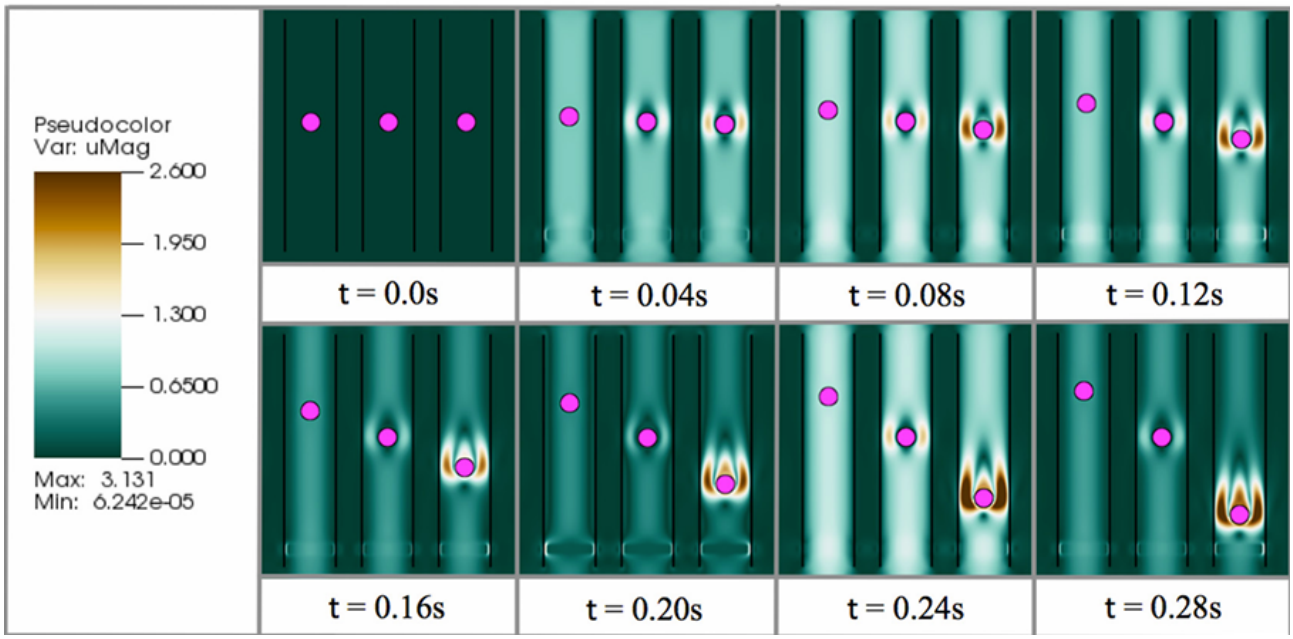



Figure 8: Results

In the simulation, the masses were arranged so that one mass would be influenced by an external force, another would remain in equilibrium with the flow and gravity, and the final mass would have a significant body force compared to the external flow. As observed, one of the masses gets carried away by the flow, one remains in the original position, and the last moves downward in the direction of gravity. The external flow in the simulation is in the laminar regime with a Reynolds Number $Re$ of approximately 500.

By analyzing this simulation, we can now analyze the behavior of an elastic particle in a fluid. We will need to adjust the spring stiffness of the massive points to account for the amount of elasticity of the particle. In all of our simulations, we will be working in the laminar regime since the application of this project is to separate the diseased cells from the blood, and the blood flow inside the veins is assumed to be laminar.

# 5 Future Work

Till now studied the background of IBM and how it is implemented in MATLAB. We were trying to understand the code that needed to be run for a simulation so that we could write our code as per our needs in this project. We analyzed the fluid-structure interaction(FSI) of a single rigid circular particle in a 2d model. Later we are going to analyze the FSI of a single elliptic particle, a group of circular particles interacting with the fluid, and a group of elliptic particles interacting with the fluid. Also, we will try to model the Fluid-structure interaction of particles in a curvilinear 2d channel. For this to happen, additionally we need to be aware of secondary flows in the channel(induced due to curvature). Further, we will analyze the FSI of a rotating particle. The applications of these simulations will be in separating the diseased cell which will have high elasticity as compared with the normal cells.

# References


[1] Nicholas A Battista, W Christopher Strickland, and Laura A Miller. Ib2d: a python and matlab implementation of the immersed boundary method. *Bioinspiration & biomimetics*, 12(3):036003, 2017.

[2] nickabattista. Ib2d, 2024. Accessed: 2024-09-18.

[3] Jun Zhang, Sheng Yan, Dan Yuan, Gursel Alici, Nam-Trung Nguyen, Majid Ebrahimi Warkiani, and Weihua Li. Fundamentals and applications of inertial microfluidics: A review. *Lab on a Chip*, 16(1):10–34, 2016.